The ASPEX package: affected sib-pair exclusion mapping

David Hinds and Neil Risch version 1.94, 2003/09/18 18:19:28

Contents

1 Introduction

ASPEX is a set of programs for performing multipoint exclusion mapping of affected sibling pair data for discrete traits. There are five main analysis programs: `sib_ibd`, `sib_phase`, `sib_map`, `sib_tdt`, and `sib_kin`. The `sib_ibd` and `sib_phase` programs are for exclusion mapping. The `sib_ibd` program is faster, tailored for data sets where both parents are typed, and makes no assumptions about missing information. The `sib_phase` program is slower, uses allele frequencies to reconstruct missing information, and is tailored for data sets where parents are missing, but additional typed children may be used to reconstruct and phase the parents. The `sib_map` program is for multipoint marker distance estimation from sib pair data. The `sib_tdt` program is for transmission disequilibrium testing. And the `sib_kin` program is for verifying the degree of relatedness of individuals within a family.

The programs are all written in ANSI C and should be easy to port to any platform with an ANSI standard C compiler. They have a UNIX “feel”, and are designed to produce simple output that can be easily processed by other programs.

In addition to the main programs, we include a few useful utilities for manipulating input and output files, and some sample data and command files.

Source code and documentation for the ASPEX package is available from `<http://aspex.sourceforge.net/>`. This site also provides a web-based discussion forum and bug tracking system. Papers using ASPEX should include a citation like:

- Hinds, D.A., Risch, N. (1996) The ASPEX package: affected sib-pair exclusion mapping. http://aspex.sourceforge.net/.

2 Compiling the programs under UNIX

Here is a short summary of the installation process:

- Install the TCL library if necessary.
- Unpack the "tar" file in a convenient place.
- Customize the `Makefile` for your system, if necessary
- Run `make` to build the programs.
- Install the programs with "`make install`".

All of the main programs in this package use the "TCL" library to parse parameter settings. This is a separate public-domain package available on the Internet. If it not already installed on your system, you will need to install it before compiling these programs. Information about TCL is available at `<http://www.tcl.tk>`.

These programs are supplied as a compressed "tar" file. To unpack these files, execute:

```
zcat aspex-2.0.tar.Z | tar xf -
```

This will create a new directory, `aspex-2.0`, containing all the source code and documentation for the programs.

In `aspex-2.0/Makefile`, there are several variables that may need to be set up for your system. The `TCL` variable needs to point to the place where TCL is installed. The TCL library needs to be in `$(TCL)/lib/libtcl.a`, and the TCL header file needs to be `$(TCL)/include/tcl.h`. The `BIN` variable specifies where the executables should be installed by "`make install`".

You may also need to customize how the C compiler is invoked, via the `CC`, `CFLAGS`, and `EXTRA` variables. The `Makefile` has defaults for several common operating systems set up as special targets: "solaris", "linux", "osf", "sunos", and "sgi". To use these targets to compile the ASPEX programs for a particular platform, just type (for example) "`make solaris`". Then, to install the executables in the `BIN` directory, run "`make install`".

3 Basic syntax and parameter specifications

The ASPEX programs use the "TCL" library for reading and parsing parameter files. TCL is a simple, flexible scripting language that is in the public domain, and is available for many different systems. The same parameter file can

generally be used for all of the ASPEX programs, with each program extracting just the information it needs and ignoring the rest.

All the ASPEX programs accept the following command-line parameters:

`-V`

Reports the ASPEX release number for this program.

`-v`

Selects more verbose output. Multiple `-v` options can be specified for increasing levels of verbosity.

`-q`

Selects "quiet" output: suppresses warnings about Mendelian incompatibilities in marker data files.

`-c` ***cmd***

Executes the specified TCL command.

`-f` ***file***

Executes TCL commands from the specified file.

The programs read parameters either from the command line or from separate parameter files. Multiple `-c` and `-f` options will be evaluated in order from left to right.

Here is a sample TCL parameter file:

```
# The number of marker loci
set nloc 5
# Marker names
set loc { l1 l2 l3 l4 l5 }
# Distances between markers
set dist { 0.1 0.1 0.1 0.1 0.1 0.1 }
# Sibling recurrence risk ratio
set risk 2.0
# Identity by descent probabilities: additive model
set z "[expr 0.25/$risk] 0.50 [expr 0.50-0.25/$risk]"
```

Parameters are specified by name (as in, "`set nloc 5`"). Comments are preceded by the "#" character. The TCL interpreter can perform basic math, as shown in the last line.

Lists of values can be grouped using braces or quotes. List elements are separated by blanks, not commas. Lists can also be split across several lines, for example:

```
set dist {
    0.1 0.1 0.1 0.1 0.1
}
```

The ASPEX programs will also use default parameter files if present. If they exist, the `aspexrc1` file in the current directory will be processed before any command line parameters, and the `aspexrc2` file will be processed after the command line. The names and paths for these parameter files can be changed by setting the `ASPEXRC1` and `ASPEXRC2` system environment variables.

The following parameters are common to all the ASPEX programs:

nloc

An integer: the number of marker loci. If not specified explicitly, it will be determined from the `loc` parameter.

loc

A list of `nloc` strings identifying the markers. The markers should be listed in map order along the chromosome.

omit

A list of strings identifying families or individuals whose data should be left out of analyses. Specify an individual using a string of the form "X.Y" where X is the family ID and Y is the person's ID. To omit an entire family, specify "X.*" where X is the family ID.

blank

A string: the allele identifier used for missing data. The default is "0".

fid_width

An integer: the width of the family identifiers, in characters, to use when formatting tabular output. The default is 2.

pid_width

An integer: the width of the person identifiers, in characters, to use when formatting tabular output. The default is 2.

allele_width

An integer: the width of the allele names, in characters, to use when formatting tabular output. The default is 2.

loc_width

An integer: the width of marker names, in characters, to use when formatting tabular output. The default is 10.

discard_partial

A boolean value: indicates if partially typed loci (one allele known, one blank) should be counted, or whether they should be treated as both-blank. The default is 1 or true. The `sib_tdt` program always discards partial genotypes.

sex_linked

A boolean value: indicates if the allele data is for the sex chromosomes, as opposed to autosomes. The default is false (autosomal).

sick_set

A string: the list of character values for the affected status field that are interpreted as "affected". The default is "YyTt2".

well_set

A string: the list of character values for the affected status field that are interpreted as "unaffected". The default is "NnFf1".

unk_set

A string: the list of character values for the affected status field that are interpreted as "unknown". The default is "Uu?0".

Programs that generate exclusion maps (`sib_ibd` and `sib_phase`) use the following additional parameters:

dist

A list of `nloc+1` map distances between all the markers, including distances from the end markers to the corresponding telomere. All map distances are specified in Morgans.

dist_xx, dist_xy

These parameters are similar to the `dist` parameter, but specify sex-specific recombination maps.

mapping

Specifies the mapping function for recombination fractions. Valid values are "Kosambi" or "Haldane". The default is "Kosambi".

z

A list of three numbers: the probabilities of two siblings being identical by descent for 0, 1, or 2 alleles, given that they are both affected. For sex-linked data, the list should consist of only two numbers, since identity by descent is only calculated for the maternal alleles. These values are only used when `most_likely` is false.

max_step

The maximum gap to leave between data points interpolated between markers in the lod map, in units of map distance. The default is 0.01 Morgans

fix_step

A flag indicating if the gap size between map points should be fixed at `max_step`, or whether it should be allowed to vary between markers. The default is false, which guarantees that there will be a data point at every marker position.

most_likely

A flag indicating if a maximum likelihood calculation of the sharing at each locus should be done. The default is false (don't do the calculation). When this flag is set, for each marker position or point along the map, the programs will determine the set of Z values that give the highest LOD score at that position. The corresponding % sharing and maximized LOD scores are reported.

linear_model

A flag indicating if the maximum likelihood calculation should fit to a linear model, or to a two-parameter model over all possible Z values. The default is true (i.e., use just a linear model). At present, the two-parameter model does not use a "possible triangle" constraint.

no_Dv

A flag indicating if maximum likelihood calculations with a linear model should use a model with no dominance variance. The default is false. When false, maximum likelihood calculations assume the following "multiplicative" model for `z` values:

```
z[2] = y^2
z[1] = 2*y*(1-y)
z[0] = (1-y)^2
```

where `y` is the sharing at this locus. If `no_Dv` is true, then this model is replaced by an additive model, where `z[1]` is fixed at 0.5:

```
z[2] = y-0.25
z[1] = 0.5
z[0] = 0.75-y
```

In terms of the sibling recurrence risk ratio, `lambda`, the multiplicative model has the form:

```
z[2] = 1 + 0.25/lambda - 1/sqrt(lambda)
z[1] = 1/sqrt(lamda) - 0.5/lamdba
z[0] = 0.25/lambda
```

and the additive model has the form:

```
z[2] = 0.5 - 0.25/lambda
z[1] = 0.5
z[0] = 0.25/lambda
```

truncate_sharing

If `most_likely` is turned on, then this flag indicates if sharing should be required to be at least 50% for the likelihood maximization. The effect is that positive LOD scores will only be indicated for positions with greater than expected sharing. For affected sib pairs, this is sensible because a predicted sharing of less than 50% would be inconsistent with any simple genetic model. If `count_discordant` is enabled, then the direction of truncation is reversed: sharing is required to be no more than 50%. The default is 1 or true.

exclusion_level

A floating point value, in LOD score units. If set, then instead of finding a maximum likelihood model, the programs will find the model farthest from the null hypothesis that has a LOD score no higher than the specified value. Thus, this finds an upper bound on the effect of a putative gene at a given position, for exclusion at this level. The default (0.0) disables the exclusion calculation. This value should never be positive.

count_once

A boolean value: indicates if only strictly independent sib pairs should be counted. Normally, for families with more than two sibs, all pairwise combinations are scored. If this flag is set, then only pairs with the first affected sib will be counted. The default is 0 or false.

first_pair

A boolean value: indicates if only the first appropriate sib pair in each family should be counted, as opposed to all pairs, or all pairs including the first sib, as indicated by `count_once`. The default is 0 or false.

count_unaffected

A boolean value: indicates if sib pairs should be counted where the first sib is unaffected. The default is 0 or false, i.e., count pairs with affected sibs.

count_discordant

A boolean value: indicates if the disease status for the second sib in a pair should be discordant with the first sib. The default is 0 or false, i.e., count pairs that are concordant for disease status.

error_freq

A floating-point number: this specifies the probability of a typing error at an arbitrary marker position. The `sib_ibd`, `sib_phase`, and `sib_map` programs use this to identify marker data that is likely to represent typing errors. The method is based on detection of unlikely recombination patterns, so it is only effective in

regions that are densely typed. When an error is detected, all marker data at that position for that family will be excluded from subsequent calculations. The default error frequency is 0 (meaning that all data is assumed to be correct). Reasonable values are on the order of 0.01.

Be careful when using `count_once` in conjunction with `count_discordant`. The disease status of the first member of each pair is always determined by `count_unaffected`. When `count_once` is enabled, the number of discordant sib pairs counted will depend on whether `count_unaffected` is set or not. If `count_unaffected` is false, then within each family, pairs of the first affected sib with all unaffected sibs will be counted. If `count_unaffected` is true, then pairs of the first unaffected sib with all affected sibs will be counted.

4 Marker data format

A marker input data file normally consists of a single header line, followed by multiple marker data records, with one line of marker data per person. The ASPEX programs handle several file format variants by automatically determining the format of the input data. There are two basic data formats: the "basic" or "Risch" format, with pedigree information implied by line ordering, and pre-makeped LINKAGE format, with explicit pedigree information. The Risch format is supported mainly for historical reasons, and we discourage its use because the extra information in a LINKAGE file can help catch some kinds of errors.

The header line consists of a space-separated list of marker names for which this file contains data.

Each record of the basic Risch marker data format looks like:

```
[fid] [pid] [1a] [1b] [2a] [2b] [3a] [3b] ...
```

or if a disease status field is available:

```
[fid] [pid] [sick] [1a] [1b] [2a] [2b] [3a] [3b] ...
```

where `[fid]` is a unique identifier for this family, `[pid]` is an identifier for this person, `[sick]` indicates if the person is sick or healthy, and the rest of the fields are pairs of allele identifiers. Family, person, and allele identifiers can be arbitrary numbers or strings of up to 15 characters. The family and person identifiers may be separated by a period. Other fields must be separated by blank space (either spaces or tabs).

For `[sick]`, values of 'Y', 'y', 'T', 't', and 2 indicate the person is affected, and 'N', 'n', 'F', 'f', and 1 indicate unaffected. If the value matches '?', 'U', 'u', or the `blank` allele, the person will be considered to have unknown disease status. These mappings can be changed using the `sick_set`, `well_set`, and `unk_set` parameters. Siblings with unknown disease status will never be included in a sib pair, but will be used for reconstructing missing parents.

Within a group of records for a complete family, the first record should describe the father, followed by the mother, followed by the children. Blank lines are ignored, and spacing within a line is not important.

Here is a sample marker data file for a single family, with four marker positions, with a column indicating disease status:

```
mark1 mark2 mark3 mark4
001 1  0  1 1  1 2  1 3  2 2
001 2  0  0 0  3 4  2 3  2 4
001 3  2  1 2  1 4  2 3  2 2
001 4  2  1 3  1 3  3 3  2 4
```

For sex-linked data, all alleles on a "Y" chromosome should be coded as "Y" in the marker data. Here is a sample sex-linked marker data file:

```
mark1 mark2 mark3 mark4
001 1  0  1 Y  1 Y  3 Y  2 Y
001 2  0  0 0  3 4  2 3  2 4
001 3  2  1 Y  4 Y  2 Y  2 Y
001 4  2  1 3  1 3  3 3  2 4
```

To simplify the use of marker data files prepared for other programs, the ASPEX programs can also read LINKAGE marker data files, with or without the header line describing which markers are present. In this case, the marker data should have the form:

```
[fid] [pid] [dad] [mom] [sex] [1a] [1b] [2a] [2b] ...
```

or if a disease status field is present:

```
[fid] [pid] [dad] [mom] [sex] [sick] [1a] [1b] [2a] [2b] ...
```

In this format, three new columns are used to explicitly identify the parents and gender of each individual. The ID's of parents who are not present should be the same as the `blank` allele specifier. The `[sex]` field should be either 'M', 'm', or '1' for males; or 'F', 'f', or '2' for females. Family structures other than simple nuclear families will generate error messages.

If a LINKAGE format file has a header line listing allele names, the ASPEX programs will use it. However, the header line is optional if a single LINKAGE format file contains data for all `nloc` markers in the order they are specified in the `loc` parameter.

Here is the a sample marker data file in linkage format:

```
mark1 mark2 mark3 mark4
001 1  0 0 m  0  1 1  1 2  1 3  2 2
001 2  0 0 f  0  0 0  3 4  2 3  2 4
001 3  1 2 m  2  1 2  1 4  2 3  2 2
001 4  1 2 m  2  1 3  1 3  3 3  2 4
```

For sex-linked data in LINKAGE format, males can be listed either as homozygotes at all loci, or with one allele coded as 'Y' at each locus. Here is a sample sex-linked data file, with males coded as homozygotes:

```
mark1 mark2 mark3 mark4
001 1  0 0 m  0  1 1  1 1  3 3  2 2
001 2  0 0 f  0  0 0  3 4  2 3  2 4
001 3  1 2 m  2  1 1  4 4  2 2  2 2
001 4  1 2 f  2  1 3  1 3  3 3  2 4
```

Finally, as a special case, a data file may contain only the pedigree, gender, and affected-status columns, and no genotype data. This is useful if several definitions of affected status are used, because the status information can be kept separate from the genotyping data. In this case, the header line should contain just the single word "status", like:

```
status
001 1  0 0 m  0
001 2  0 0 f  0
001 3  1 2 m  2
001 4  1 2 f  2
```

5 The sib_ibd program

This program reads a file of genotype data for nuclear families consisting of two parents and two or more "affected" siblings. It reports identity-by-descent information for all affected sibling pairs.

The program works by first trying to identify the parental origin of each sibling allele. At a given locus, for each parent, the program compares the corresponding alleles for each member of an affected sibling pair. If both alleles can be uniquely identified, the pair is scored as either identical or non-identical by descent from that parent at that position. If the match is ambiguous, the position is scored as uninformative.

If marker data from additional siblings is available, it will be used to reconstruct missing parents. The reconstruction will depend on the settings of the `count_discordant`, `count_unaffected`, and `count_once` parameters. When scoring a given sib pair, the first sib in that pair is used for reconstruction. Also, any sibs that are not part of any counted pair will be used. In the common case of concordant affected sib pairs, this means that one affected and all unaffected sibs will be used for reconstruction. Be aware that the IBD results for different types of sib pairs (affected, unaffected, discordant) cannot be simply added to get results for all pairs, due to the different reconstruction criteria used in each case.

A side effect of the use of the first sib from each pair in parent reconstruction is that the the choice of that sib, and thus, the reconstruction and sharing results, will depend on the order of sibs in the input files. The `limit_build` parameter can be used to prevent this, at the cost of losing some information for small families with untyped parents. Alternatively, the `best_order` parameter specifies that sibs should be sorted by number of typed loci. The "first sib" will be the one with fewest untyped loci, which may improve reconstructions if many subjects have incomplete data. This also affects the `count_once` option: all pairs will be constructed using the most completely typed sib.

The lod score at a particular probe position is determined by finding the nearest informative markers flanking the probe on either side, for both the maternal and paternal alleles. The probability of the given marker data assuming the probe is in a particular IBD state is calculated from the known IBD states of these flanking alleles, and the recombination distances to the probe. These probabilities are then used to determine the odds of the marker data, given a disease gene of known penetrance at the probe locus, versus the odds of the marker data if the probe has zero penetrance.

In a family with more than two affected siblings, each sibling's marker data will contribute to several sib pairs, as each pair is counted independently. While this does not bias the observed sharing results, it does mean that data from large families will carry more weight than data from small families. This effect can be controlled with the `count_once` parameter.

The `sib_ibd` program can use gender-specific recombination maps for calculation of LOD scores, if available. In this case, distances should be specified with the `dist_xx` and `dist_xy` parameters. It is also possible to request separate LOD scores for maternal and paternal sharing contributions, with the `sex_split` parameter. These LOD scores are only independent (and additive) if a "multiplicative" disease model is used. If `most_likely` is specified along with `sex_split`, then the maternal and paternal scores are reported using the overall maximum-likelihood model.

5.1 Command syntax

`sib_ibd` [`-v`] [`-c` *cmd*] [`-f` *file*] [*marker_data ...*] [> *ibd_data*]

One or more marker data files can be listed on the command line. If no files are specified, marker data will be read from standard input. The ibd listing will be sent to standard output.

5.2 TCL parameters

The following parameters should be specified using TCL commands via either the `-c` or `-f` mechanisms:

limit_build

This parameter restricts parent reconstructions to only use sibs that are not part of counted pairs. The default is false, meaning that the first sib in a pair will also be used for reconstructions.

best_order

This indicates that sibs should be sorted using the number of typed loci. The internal ordering of sibs within a family determines how parent reconstructions are done, and determines the choice of pairs if `count_once` is used. The default is false, meaning the order of sibs in the input data is preserved.

sex_split

This parameter specifies if separate LOD scores should be calculated for paternal and maternal sharing.

show_pairs

A boolean value: if set, then posterior probabilities of IBD=0,1,2 for each location are reported for each sib pair.

Here is a sample parameter file for `sib_ibd`:

```
set nloc 4
set loc { "l1" "l2" "l3" "l4" }
set dist { 0.1 0.1 0.1 0.1 0.1 }
set blank "00"
set fid_width 6
set discard_partial true
set risk 2.0
set z "[expr 0.25/$risk] 0.50 [expr 0.50-0.25/$risk]"
set mapping Haldane
```

In this example, the identical-by-descent probabilities are derived from a simple sibling recurrence risk ratio assuming an additive model.

The `sib_ibd` program will automatically reorder markers based on the map in the `loc` parameter. The order in which markers are given in the input file (or files) is not important. Families do not need to be present in all input files, however, if a family is present, all family members must be present and in the same order in each case.

5.3 Output

The default output is a summary of sharing at each locus, broken down by parent. The markers in the output file will be sorted into their proper order along the chromosome, as specified in the `loc` parameter. A chi-squared statistic for the observed sharing, with one degree of freedom, is calculated for each marker, along with a multipoint likelihood.

In some cases, `sib_ibd` can determine that for a particular sib pair, one allele is shared and one is not, but there is no way of knowing if it is the maternal allele or the paternal allele. These cases will be included in the "Combined" sharing table.

If `most_likely` is enabled, then `sib_ibd` will calculate a maximum likelihood estimate of the sharing at each marker position In this case, the table will also include maximum likelihood estimates of the z0, z1, and z2 values (for the specified model); the corresponding percent sharing; and the multipoint LOD score, for each marker position.

If verbose (`-v`) output is selected, a multipoint exclusion map will be generated, with lines like:

```
[offset] [lod score]
```

where `[offset]` is a genetic recombination distance, and `[lod score]` is the log likelihood ratio of the marker data for a disease gene at this position with the given sibling recurrence risk ratio, versus a sibling risk ratio of 1.

If `most_likely` is enabled, the verbose output is like:

```
[offset] [sharing] [z0] [z1] [z2] [mlod score]
```

where `[sharing]` and `[mlod score]` are the percent sharing and lod score obtained by the maximum likelihood calculation, and `z0`, `z1`, and `z2` are the maximum likelihood model parameters.

If `sex_split` is enabled, then instead of reporting a single LOD score at each marker or map position, three LOD scores are reported: the partial paternal and maternal scores, as well as the regular combined score. If separate paternal and maternal maps are available, then both map positions will be included in the map output, like:

```
[male map] [female map] [sharing] [z0] [z1] [z2] [mlod]
```

or if `sex_split` is set:

```
[male map] [female map] [sharing] [z0] [z1] [z2] [pat] [mat] [mlod]
```

If very verbose (`-vv`) output is requested, then haplotype data and the raw IBD results will be displayed. For each sibling in a family, there will be a line like:

```
[fid] [pid] [a1f] [a1m] [a2f] [a2m] ...
```

where `[fid]` is the family ID, `[pid]` is the person ID of a child, `[a1f]` is the allele at position 1 inherited from the father, and `[a1m]` is the allele inherited from the mother. The alleles are given as either "ND" for not determined, "NC" for not consistent, or the allele name from the input data.

Following the genotype data for a family will be IBD results for each sib pair, of the form:

```
[fid] [pid1] [pid2] [ibd1f] [ibd1m] [ibd2f] [ibd2m] ...
```

where [fid] is the family ID, [pid1] and [pid2] are the person ID's of two siblings, [ibd1f] is the ibd result for marker 1 through the father, and [ibd1m] is the same through the mother. The ibd result is either "0" for not informative, "-" for no match, or "+" for a match. When sib_ibd determines that one allele is shared (but cannot determine which it is), the ibd result will be listed as "?".

6 The sib_tdt program

This program reads a file of genotype data for nuclear families consisting of two parents and one or more "affected" siblings. It reports frequencies for all alleles in the parents, and allele transmission frequencies for transmission disequilibrium (TDT) tests.

This version of sib_tdt will calculate empirical probabilities for chi-squared statistics, which will accurately reflect association independent of linkage within families. This calculation is done by permuting parent alleles while fixing the IBD status of sibs within a family.

There are two run-time parameters that affect the accuracy of p-values calculated using the permutation procedure: min_reps and max_reps. For each replicate dataset, a TDT score is calculated and compared with the score for the actual data, and the empirical p-value equals the number of replicate scores that exceed the actual score, divided by the total number of replicates. The algorithm will generate new replicates until the numerator exceeds min_reps, or until the denominator exceeds max_reps, whichever happens first.

6.1 Command syntax

sib_tdt [-v] [-c *cmd*] [-f *file*] [*marker_data ...*] [> *tdt_data*]

One or more marker data files can be listed on the command line. If no files are specified, marker data will be read from standard input. The TDT listing will be sent to standard output.

6.2 TCL parameters

The following parameters should be specified using TCL commands via either the -c or -f mechanisms:

one_sib

A boolean value: indicates that transmissions to just the first sib in each family should be scored. The default is false.

min_reps

During p-value estimation, stop generating replicates when at least this many have scores larger than the observed TDT score. This determines the accuracy of large p-values. The default is 1000.

max_reps

During p-value estimation, never generate more than this number of sample replicates. This determines the accuracy of small p-values. The default is 40000.

Here is a sample parameter file for sib_tdt:

```
set loc { "l1" "l2" "l3" "l4" }
set blank "00"
set discard_partial true
```

6.3 Output

The default non-verbose output is a table summarizing sample coverage and chi-squared statistics for each marker. The table lists, for each position, the number of distinct alleles, the heterozygosity based on just typed parents, and the overall percentage of typed individuals. The TDT results are summarized by the sum of chi-squared statistics for transmission of all alleles, and the maximum chi-squared obtained for any one allele, and the corresponding estimated p-values.

If `sex_split` is true, then separate chi-squared scores and p-values are reported for maternal and paternal transmissions.

For the default `min_reps` and `max_reps` settings, p-values are accurate to within about 5%, but gradually become less accurate for $p<0.01$.

The sum statistic and maximum statistic are generally similar, but the sum statistic is more sensitive to cases where multiple alleles are in disequilibrium.

If verbose (`-v`) output is selected, the summary table will be replaced by allele transmission tables for each marker, with the form:

```
[al] [n] [%]  [ft] [fn] [fc]  [mt] [mn] [mc]  [st] [sn] [sc]
```

where `[al]` is the allele name, `[n]` is the number of times it is seen in the parents, and `[%]` is the percent frequency. `[ft]` is the number of times the allele was transmitted through the father, `[fn]` is the number of times it was not transmitted, and `[fc]` is the chi-squared score for this outcome. `[mt]`, `[mn]`, and `[mc]` are the same, for the mother. Likewise, `[st]`, `[sn]`, and `[sc]` combine the results for both parents.

The combined counts may be larger than the sum of the counts for the two parents. If two parents and a child are all heterozygous with the same genotype at a given position, one copy of each of the child's alleles was transmitted and the other was not. These cases are added into the "combined" totals.

If only one parent is typed at a particular marker, transmission through that parent will be scored in cases where it is not biased by allele frequencies. This reduces to cases where the parent and child are both heterozygous, but have only one allele in common (i.e., parent AB, child AC). This differs from treatment in previous versions: prior to version 1.11, single-parent transmissions were scored (incorrectly) even in situations that were biased, and from version 1.11 through 1.16, single-parent cases were never scored.

If very verbose (`-vv`) output is selected, the TDT results are replaced by a detailed listing of allele transmissions for each child. The listing indicates which allele was inherited from which parent, for all children. In the special case described above where `sib_tdt` cannot assign the transmitted alleles to specific parents, the allele pair is enclosed in square brackets.

7 The sib_phase program

The `sib_phase` program generates a multipoint exclusion map from marker data for affected sibling pairs. While `sib_phase` still calculates separate likelihoods for each affected pair assuming they are independent, it will use all

the available marker data from a family when calculating likelihoods. It will use allele frequencies to reconstruct missing parents, and will also phase the parents across multiple markers. When parent marker data is available, for two-sib families, `sib_phase` is equivalent to (but slower than) `sib_ibd`.

The execution time of `sib_phase` depends sharply on the amount of missing marker information. The calculation can become very time consuming for large families if there are adjacent loci for which the available marker data still allows a large number of possible inheritance states. One example of such a case is when two adjacent loci are untyped for most but not all children. Another case is when at least one parent is untyped, and all typed individuals are homozygous for the same allele. When `sib_phase` encounters a family that it expects will take a long time to evaluate, it will print a warning message with an estimated "cost" for the family, which is roughly on the order of the calculation time, in seconds. The `sib_phase` program also has a strict upper limit of 16 sibs per family.

The memory requirements of `sib_phase` are an exponential function of sibship size. The worst-case requirements scale as 8*2^(2N-2) for a sibship of size N. However, for the common situation in which a sib is either untyped, or typed at nearly all markers, memory requirements should actually scale as 8*2^(N-1), a much more manageable number for all reasonable sibship sizes. In practice, memory requirements should rarely ever be an issue, even with large sibships.

There are two analysis modes in `sib_phase`, depending on the value of the `use_allele_freq` parameter. Allele frequencies are estimated from all the marker data, including parents and children. If `use_allele_freq` is disabled, `sib_phase` should give the same results as `sib_ibd`, with the exception that it can reconstruct parents when both are untyped. If `use_allele_freq` is enabled, then when parents cannot be reconstructed, the allele frequencies are used to estimate the probabilities of each IBD state.

Like `sib_ibd`, `sib_phase` will use gender-specific recombination maps if specified with the `dist_xx` and `dist_xy` parameters. Separate maternal and paternal LOD scores will be reported if the `sex_split` parameter is used. With `sib_phase`, the parental LOD scores will generally not be additive, unless all parents are typed. The reconstruction of missing parental genotypes using allele frequencies is done for both parents together, so the LOD score for one parent is "contaminated" with information derived from the other parent that is difficult to remove. While the resulting parental LOD scores are not independent, they may still be useful for qualitatively assessing how excess sharing is distributed between the two parents. If `most_likely` is specified along with `sex_split`, then the maternal and paternal scores are reported using the overall maximum-likelihood model.

7.1 Command syntax

`sib_phase` [`-v`] [`-c` *cmd*] [`-f` *file*] [*marker_data ...*] [> *map_data*]

One or more marker data files can be listed on the command line. If no files are specified, marker data will be read from standard input. The exclusion map will be sent to standard output.

7.2 TCL parameters

The following parameters should be specified using TCL commands via either the `-c` or `-f` mechanisms:

use_allele_freq

A boolean value: indicates if allele frequencies should be used to estimate IBD probabilities when parents are missing. The default is true.

fixed_freq

A boolean value: if false, indicates that allele frequencies should be estimated from the given marker data. If true, then frequencies should be specified in the parameter file. The default is false.

sex_split

This parameter specifies if separate LOD scores should be calculated for paternal and maternal sharing.

show_freqs

A boolean value: if set, then allele frequencies are reported.

show_pairs

A boolean value: if set, then posterior probabilities of IBD=0,1,2 for each location are reported for each sib pair.

There is a new TCL command, `freq`, for specifying fixed allele frequencies in the parameter file, when `fixed_freq` is enabled. The syntax is:

```
freq [loc] { [a1] [f1] [a2] [f2] ... }
```

where `[loc]` is the marker name, `[a1]` and `[f1]` are an allele name and its frequency, and so on.

Here is a sample parameter file for `sib_phase`:

```
set nloc 4
set loc { "l1" "l2" "l3" "l4" }
set blank "0"
set discard_partial true
set dist { 0.1 0.1 0.1 0.1 0.1 }
set risk 2.0
set z "[expr 0.25/$risk] 0.50 [expr 0.50-0.25/$risk]"
set mapping Haldane
set most_likely true
set no_Dv true
set fixed_freq true
freq "l1" { "1" 0.3 "2" 0.7 }
freq "l2" { "1" 0.5 "2" 0.5 }
freq "l3" { "1" 0.9 "2" 0.1 }
freq "l4" { "1" 0.5 "2" 0.5 }
```

In this example, the identical-by-descent probabilities are derived from a simple sibling recurrence risk ratio assuming an additive model.

7.3 Output

The default output of `sib_phase` is a table summarizing the actual and expected identity by state at each marker position. A chi-squared value for the IBS totals is generated for each marker, and the multipoint LOD score is also shown. If `most_likely` is enabled, then `sib_phase` will instead calculate a maximum likelihood estimate of the sharing at each marker position (or point along the map). As with `sib_ibd`, the table will list the percent sharing, model parameters, and lod score at each marker position.

For autosomal data, the identity by state estimate is given by:

```
IBS = 0.5 + (0.75 * S2) - (0.5 * S3) + (0.25 * S4)
```

where S2 is the sum of squared allele frequencies at this position, S3 is the sum of cubed frequencies, and S4 is the sum of fourth powers of the frequencies.

For sex-linked data, the identity by state estimate is given by:

```
IBS = 0.5 + ((1.0 - SAME/2.0) * S2) - ((0.5 - SAME/2.0) * S3)
```

where S2 and S3 are as for an autosomal marker, and SAME is the fraction of sib pairs in the sample that are same-sex.

If verbose (-v) output is selected, sib_phase will generate a multipoint exclusion map. The format is the same as for sib_ibd.

8 The sib_map program

The sib_map program generates two-point and multipoint maximum likelihood estimates of the map distances between markers, based on marker data from nuclear families. It will also determine support intervals. Allele frequencies will be used to reconstruct missing parents. All sibling marker data is used to determine likelihoods, without ever being broken down into sib pairs.

The sib_map program uses the same likelihood engine used in sib_phase, so it also has a strict limit of 16 siblings per family.

The multipoint maximization algorithm determines the complete set of distances that gives the global maximum likelihood for the marker data across all markers. The two-point algorithm considers each pair of adjacent markers separately, ignoring information from more distant markers.

The support intervals calculated for multipoint distance estimates assume that all other distances are fixed at their maximum likelihood positions while one target interval is varied. In a future version, we may implement a method which will optimize the other distances as the target interval is varied.

If error_freq is non-zero, then sib_map will attempt to identify likely typing errors using provisional distance maps using the multipoint algorithm. The map is recalculated iteratively until no new errors are discovered.

By default, sib_map calculates a single sex-averaged map. If sex_split is enabled, then separate maps are calculated based on the observed paternal and maternal recombination rates.

In a second mode of operation, sib_map calculates three-point distances for one marker against all other pairs of adjacent markers along a map. This data can be used to verify map orders, or to position new markers on an already determined map. This mode always uses sex-averaged distances.

8.1 Command syntax

sib_map [-v] [-c *cmd*] [-f *file*] [*marker_data* ...] [> *map_data*]

One or more marker data files can be listed on the command line. If no files are specified, marker data will be read from standard input. The map(s) will be written to standard output.

8.2 TCL parameters

The following additional parameters can be specified using TCL commands via either the `-c` or `-f` mechanisms:

support

Specifies the LOD score cutoff for support intervals. The default is 1.0 log units, which means that support intervals will cover the range of distances that give LOD scores within 1.0 of the LOD score of the most likely distance. The default is 0.6 LOD units.

epsilon

Specifies the convergence criterion for the iterative distance refinement routines. The default is 0.00001 Morgans.

do_shuffle

A boolean value: specifies if `sib_map` should perform the map shuffling function, to verify map order and/or place new markers on an existing map. The default is false.

sex_split

A boolean value: if set, then separate paternal and maternal recombination maps will be calculated. The default is false.

show_freqs

A boolean value: if set, then allele frequencies are reported.

Here is a sample parameter file for `sib_map`:

```
set nloc 4
set loc { "M1" "M2" "M3" "M4" }
set blank "0"
set mapping Kosambi
```

8.3 Output

The normal output of `sib_map` consists of the two-point and multipoint distances between each pair of adjacent markers, and the corresponding support intervals. If two markers are determined to be unlinked, their distance will be reported as "`[inf]`" (for "infinity"). In this case, the support interval will give a lower bound on the most likely distance between the pair. A LOD score is reported for each interval, giving the likelihood for the most likely distance, compared to the likelihood of the two markers being unlinked.

From simulations, we estimate that for support levels of 0.2, 0.6, and 0.8 LOD units, the true distance should be within the support interval about 70%, 90%, and 95% of the time, respectively. These are not strict confidence intervals, however, so these probabilities should be used only as rough guidelines.

If verbose (`-v`) output is selected, then tables of LOD scores versus distance for each marker pair will also be generated.

If `do_shuffle` is true, then the output is, for each marker, a table giving three-point distance estimates for that marker with every other pair of adjacent markers along the map. The total distance spanning the three markers, and

the corresponding LOD score, is generated for all possible orders of markers (XAB, AXB, ABX). Thus, a comparison of the LOD scores indicates where the test marker is likely to be in relation to the pair.

Following the distances and LOD scores, `sib_map` will print one of several symbols based on a comparison of the LOD scores. If the test marker appears to be to the left of the specified pair, then "<" or "<<" will be printed: the number of arrows indicates that the LOD score difference exceeds that number times the value of `support`. Similarly, ">" or ">>" will be printed if the marker is to the right of the pair. "+" or "++" will be printed if the marker is most likely to be between the specified pair. If the map order is correct and well supported by the data, the symbols for each marker should show a pattern of ">" rows, then one "+" row, then "<" rows, as the marker is shuffled through its true position. To use the output to position new loci on a map, in the parameter file for `sib_map`, list the new loci first in the map, followed by all the already-mapped loci in their proper order.

9 The sib_kin program

The `sib_kin` program estimates the likelihood of an observed set of linkage data, for pairs of individuals, given different assumptions about their degree of relatedness. It generates likelihoods under the (generally false) assumption that all markers are independent, so the actual lod scores should be interpreted with caution. It can be used to verify family structures and/or check for some kinds of sample identification errors. It works by comparing the observed identity by state statistics for each pair with what would be expected by chance, given the observed allele frequencies.

The power of `sib_kin` is proportional to the number of informative and unlinked loci in the input data. For best results, the input data should consist of reasonably evenly spaced markers spanning multiple chromosomes. Typical genome scan data is ideal.

It should be noted that `sib_kin` impicitly assumes that the sample is in Hardy-Weinberg equilibrium, and wild deviations from this will lead to uninterpretable results. For example, data with alleles renumbered within families should not be used with this program.

9.1 Command syntax

`sib_kin` [`-v`] [`-c` *cmd*] [`-f` *file*] [*marker_data* ...] [> *ibs_data*]

One or more marker data files can be listed on the command line. If no files are specified, marker data will be read from standard input. Identity by state and likelihood data for relative pairs will be written to standard output.

9.2 TCL parameters

The following additional parameters can be specified using TCL commands via either the `-c` or `-f` mechanisms:

count_parents

A boolean value: specifies that IBS and likelihood information should be reported for relative pairs with putative parents, as well as between sibs. The default is false.

four_way

A boolean value: specifies that a four-way comparison of the likelihoods of each pair being parent-sib, full sibs, half sibs, or unrelated should be performed. The default is false.

Here is a sample parameter file for `sib_kin`:

```
set nloc 100
set loc {
  D1S100 D1S200 D1S300
  ...
  D22S100 D22S200 D22S300
}
set blank "0"
set most_likely true
```

A `sib_kin` parameter file should either specify `most_likely`, or `four_way`, or give a specific set of `z` parameters to identify what relationship(s) are to be tested. Since `sib_kin` assumes that all markers are unlinked, the marker list may span multiple chromosomes, and markers need not be in any particular order. For best power, the markers should span all the autosomes.

9.3 Output

The output of `sib_kin` is a table with one line of output per relative pair, grouped by family. Non-verbose output leaves out families where all pairwise relationships are consistent with the family structure given in the data files; verbose output includes results for all families. Observed IBS statistics are reported, along with one or more log-likelihoods. The null hypothesis for each likelihood is that the pair is unrelated. If `most_likely` is true, then the maximum likelihood set of `z` values and corresponding LOD score are reported. If `four_way` is true, then log-likelihoods for a standard set of four relationships will be reported. Otherwise, the likelihoods for a pre-specified set of `z` parameters will be reported.

For the four-way analysis, in addition to the LOD scores for the four hypothetical relationships (where the score for the pair being unrelated is 0.0), `sib_kin` will identify which relationship was most likely, as either “U”, “I”, “P”, “S”, or “H”, (for unrelated, identical, parent, sibling, or half-sib) if that relationship is at least 3 LOD units more likely than any of the other possibilities. If no relationship meets the cutoff, then “?” will be indicated.

10 The sib_clean program

The `sib_clean` program is a stripped down version of `sib_phase`, which simply checks its input data for Mendelian inconsistencies and/or likely typing errors. Its output is a new linkage file that is error-free. In each case where an error is found, the genotype data for all members of the affected family at that marker will be deleted.

10.1 Command syntax

`sib_clean` [`-v`] [`-c` *cmd*] [`-f` *file*] [*marker_data ...*] [> *clean_data*]

One or more marker data files can be listed on the command line. If no files are specified, marker data will be read from standard input. The updated, clean linkage data is written to standard output.

10.2 TCL parameters

The following parameters should be specified using TCL commands via either the `-c` or `-f` mechanisms:

fixed_freq

A boolean value: if false, indicates that allele frequencies should be estimated from the given marker data. If true, then frequencies should be specified in the parameter file. The default is false.

show_freqs

A boolean value: if set, then allele frequencies are reported.

There is a TCL command, `freq`, for specifying fixed allele frequencies in the parameter file, when `fixed_freq` is enabled. The syntax is:

```
freq [loc] { [a1] [f1] [a2] [f2] ... }
```

where `[loc]` is the marker name, `[a1]` and `[f1]` are an allele name and its frequency, and so on.

Here is a sample parameter file for `sib_clean`:

```
set loc { "l1" "l2" "l3" "l4" }
set blank "0"
set fid_width 3
set pid_width 3
set allele_width 3
set error_freq 0.01
set dist { 0.1 0.1 0.1 0.1 0.1 }
set fixed_freq true
freq "l1" { "1" 0.3 "2" 0.7 }
freq "l2" { "1" 0.5 "2" 0.5 }
freq "l3" { "1" 0.9 "2" 0.1 }
freq "l4" { "1" 0.5 "2" 0.5 }
```

A distance map only needs to be specified if `error_freq` is non-zero.

10.3 Output

The output is the input data, in LINKAGE format, with inconsistent data and likely typing errors replaced by blank alleles. As with the other ASPEX programs, all the inconsistencies and likely typing errors are also reported.

Reanalysis of a dataset after cleaning with `sib_clean` may yield different results in some cases. When estimating allele frequencies from the input data, the ASPEX programs use the raw input data without removing inconsistencies or likely errors, so the cleaned data will yield slightly different allele frequency estimates. The differences in analysis results will generally be very small.

11 Other useful add-ons

These utilities are all written in “Perl”, a widely available public domain interpreted programming language. Some of them work by running one of the other programs in this package and digesting the output one way or another. Others are useful for manipulating genetic marker data files.

For more information about Perl, or to find out how to get it, see `<http://www.perl.com>`.

11.1 The ligate filter

UNIX command syntax:

`ligate` [`-r`] [`-a`] [`-n`] [`-f`] [`-p`] [`-b` *blank*] [`-c` *markers*] [`-d` *markers*] [*marker_data ...*]

The `ligate` filter can be used to cut and paste linkage data files together, and to convert between common file format variants. Any number of files can be specified, in either Risch or LINKAGE format, in any combination. The default output is a LINKAGE format file formed by merging all the input data.

If a linkage file is missing the ASPEX-style header line listing the marker names, then marker names "M1.1", "M1.2", etc will be generated automatically. The next such file would get marker names "M2.1", "M2.2", and so on.

Command line parameters:

`-r`

Specifies that the data should be written in Risch format, as opposed to LINKAGE format.

`-a`

Specifies "alphabetic" codes for gender and blanks. Numeric gender codes (1, 2) will be translated to "m" and "f", and blank parents and alleles will be coded as "x".

`-n`

Specifies "numeric" codes for gender and blanks. Alphabetic gender codes ("m", "f") will be translated to the corresponding numbers, and blanks will be coded as "0".

`-b` *blank*

Specifies the code for the blank allele. The default is "0".

`-f` *blank*

Specifies that new rows (with all blank genotype data) should be created in the output file for untyped parents. ASPEX does not require that an untyped parent be listed in the data file, but some linkage programs do.

`-p` *blank*

Specifies that the original allele codes should be "packed" and recoded as 1, 2, ... for each marker. Some linkage programs require that alleles be coded this way, rather than, say, as raw allele sizes.

`-c` *markers*

The list of markers will be included in the output file, in the order given, and all other markers will be omitted.

`-d` *markers*

The specified list of markers will be deleted from the output file.

11.2 The restrict filter

UNIX command syntax:

`restrict` [`-b` *blank*] [`-kids` *[+-]n*] [`-sick` *[+-]n*] [`-well` *[+-]n*] [`-parents` *[+-]n*] [`-onlysick`] [`-pos` *m[-n]*] [`-sib` *m[-,]n*] [`-keep` *id...*] [`-remove` *id...*] [*marker_data*]

The `restrict` filter selects families from a linkage data file based on family structure. It is structured somewhat like the UNIX `find` command. This filter replaces the "*_kids", "*_parent", and "only_sick" filters in previous ASPEX releases.

Command line parameters:

`-b` ***blank***

Specifies the code for the blank allele.

`-parents` ***[+-]n***

Only families with the specified number of typed parents will be accepted. A parameter of the form "+n" selects for families with at least "n" typed parents; "-n" selects for no more than "n"; and a plain number selects for an exact match.

`-kids` ***[+-]n***

Selects families based on the total number of children.

`-sick` ***[+-]n***

Selects families based on the number of affected children.

`-well` ***[+-]n***

Selects families based on the number of unaffected of children.

`-onlysick`

Specifies that only affected children should be included in the output.

`-pos` ***m[-n]***

For families meeting the other criteria, this selects based on family order in the input file. Either a single family number or a range of family numbers can be specified.

`-sib` ***m[-,]n***

For families meeting the other criteria, this selects specific siblings from each family. Sibs are numbered starting at 1 in their order in the input file. Either ranges ("m-n") or specific pairs ("m,n") can be selected.

`-keep` ***id...*****]**

Only include families with the specified ID's. The ID list may be space- or comma-delimited; if space-delimited, it should be enclosed in quotes. An ID can either be just a family ID, or a family ID and person ID separated by a period.

`-remove` ***id...*****]**

Remove individuals with the specified ID's, with the same format rules as `-keep`.

11.3 The list_untyped filter

UNIX command syntax:

`list_untyped` [*blank_allele*] [< *marker_data*]

DOS command syntax:

`perl list_unt` [*blank_allele*] [< *marker_data*]

This filter scans a file of marker data in `sib_ibd` format, and extracts any individual that has an untyped or partially typed marker. It accepts one argument:

blank_allele

Specifies the allele that indicates missing data. The default is '0'.

11.4 The list_incompat filter

UNIX command syntax:

`list_incompat` [`-c` *cmd*] [`-f` *file*] [< *marker_data*]

DOS command syntax:

`perl list_inc` [`-c` *cmd*] [`-f` *file*] [< *marker_data*]

The `list_incompat` filter has exactly the same syntax as the `sib_ibd` program, except that the `-v` option is not used. It runs the `sib_ibd` program, and filters the allele inheritance data to produce a listing of all siblings whose genotypes are incompatible with their parents.

11.5 The rec_dist filter

UNIX command syntax:

`rec_dist` [`-v`] [`-c` *cmd*] [`-f` *file*] [< *marker_data*]

The `rec_dist` filter has the same syntax as `sib_ibd`. It processes the output of `sib_ibd` and derives estimates of the recombination fractions between all pairs of markers. For compactness, the default output is a matrix showing the recombination fraction multiplied by 1000.

If `-v` is specified, then a table is generated showing, for each marker pair, the number of times the IBD state was known to be the same at those two positions, and the number of times it differed, with the corresponding recombination fraction.

11.6 The xmgr_map script

UNIX command syntax:

`xmgr_map` *file*

DOS command syntax:

`perl xmgr_map` *file*

The `xmgr_map` script is used with the `xmgr` plotting program to generate nicely formatted multipoint exclusion maps. It reads a `sib_ibd` parameter file, and outputs a set of `xmgr` commands to label the X axis with the marker names at the appropriate positions. The output of `xmgr_map` should be appended to the output of `sib_ibd` to create a complete `xmgr` input file.

The `xmgr` program is a completely separate public domain graphing program for Unix systems. Current information about `xmgr` can be found at `<http://plasma-gate.weizmann.ac.il/Xmgr/>`.

12 Tips and Tricks

Here are a few random bits of useful information that may be helpful when using ASPEX. Many of the tricks depend on features of the TCL command language embedded in all the ASPEX tools. A discussion of TCL syntax is beyond the scope of this document, but this section at least gives an idea of the sorts of things that can be done with TCL. As you will see, TCL syntax, in particular the meaning of different quoting and bracketing constructs, is somewhat obscure. If you want to learn more about TCL, a very good reference is “Tcl and the Tk Toolkit” by John Ousterhout.

12.1 Data organization

I typically organize files for a project so that ASPEX parameter files are in the main directory, and data files are in a subdirectory (maybe organized by chromosome, depending on how the files were generated). I also keep map files (containing just the `loc` and `dist` parameters) in a separate subdirectory. Since ASPEX input file handling is very flexible, I try to leave the raw data as close to its original form as possible: if the data is organized one marker per file, or one chromosome per file, or one gel per file, I keep that organization, and let ASPEX take care of extracting the right information for a particular analysis.

The `omit` parameter also helps here, because it allows subsetting of data without having to touch the raw genotype files. In my limited experience, editing raw data files in the course of a study (say, if an individual is excluded due to a nonpaternity, or a family is excluded due to a misdiagnosis) is best kept to a minimum: it is nice to be able to leave the data alone, and just have a command file that describes (with comments) who is being included and excluded, and why.

Use the `aspexrc1` and `aspexrc2` files to store all the defaults for a project (things like field widths, the blank allele identifier, and the setting of `count_once`). They can also be used to select default analysis options. The order of evaluation of these files can be important: set something in `aspexrc1` if you might want to override it with a command-line option, but if you want to evaluate something depending on a command-line option, it needs to go in `aspexrc2`. Options that will probably never change can be put in either place.

For example, here could be a reasonable `aspexrc1` file:

```
# These won't ever change
set blank "N"
set fid_width 3
set pid_width 3
set allele_width 3
# Defaults we may want to override
set count_once true
set truncate_sharing false
set sex_linked false
set no_Dv false
```

And here is something that could go in `aspexrc2`. This initializes `z` based on the current setting of `no_Dv` if a sibling recurrence risk ratio, `risk`, is available. Otherwise, it selects a maximum likelihood calculation.

```
# This needs to execute after the command-line options
if [ info exists risk ] then
    if $sex_linked {
        set z "[expr 0.5/$risk] [expr 1.0-0.5/$risk]"
```

```
        } else {
            set z1 [expr $no_Dv ? 0.50 : 1/sqrt($risk) - 0.50/$risk]
            set z "[expr 0.25/$risk] $z1 [expr 1.0 - 0.25/$risk - $z1]"
        }
    } else {
        set most_likely true
    }
```

Another data organization trick that is useful when working with ASPEX is to keep affected-status information in a separate file that does not include any genotype data. This is particularly useful for analyses with more than one definition of disease status, because the same set of raw genotype data files can be analyzed with several separate disease-status files.

12.2 More command file tricks

Here is another useful bit of TCL code for `aspexrc1`: it allows a marker map to be given in absolute rather than relative positions.

```
proc map {a} {
    global loc dist
    set n [ expr [llength $a] - 1 ]
    set d1 [lindex $a 0]
    for {set i 1} {$i < $n} {incr i} {
        set d2 [lindex $a $i]
        lappend dist [ expr $d2 - $d1 ]
        set d1 $d2
        incr i
        lappend loc [lindex $a $i]
    }
    lappend dist [ expr [lindex $a $n] - $d1 ]
}
```

So... the following normal aspex map description:

```
set loc { A B C }
set dist { 0.1 0.2 0.2 0.1 }
```

could instead be specified as:

```
map {
  0.0
  0.1 A
  0.3 B
  0.5 C
  0.6
}
```

If you prefer specifying distances in centimorgans, put the following in `aspexrc2`:

```
# Convert the dist array to centimorgans
set d ""
set n [ expr [llength $dist] ]
for {set i 0} {$i < $n} {incr i} {
    lappend d [ expr 0.01 * [lindex $dist $i] ]
}
set dist $d
```

A parameter file can also be structured to contain different sets of options for different ASPEX programs. The name of the invoking program is available as a TCL variable, `argv0`. For example:

```
# Omit families with half sibs from regular analysis
if { $argv0 != "sib_kin" } {
    set omit { 101060.* 200442.* }
}
```

12.3 Shortcuts for simple analyses

Sometimes, it is convenient to specify lists of markers on the command line rather than in a parameter file. For instance, to calculate the recombination distance between two markers:

```
sib_map -q -c "set loc { mark1 mark2 }" files ...
```

or to calculate a TDT or sharing statistics at a single marker:

```
sib_tdt -q -v -c "set loc mark1" files ...
sib_ibd -q -c "set loc mark1" files ...
```

These shortcuts work best if you have already put all your default parameter settings in `aspexrc1` and `aspexrc2`.

12.4 The example subdirectory

The files in the `example` subdirectory demonstrate several ways of organizing linkage data for use with ASPEX.

We generated simulated data for 50 three-sib families, assuming a gene with a sibling risk recurrence ratio of 10 and a succeptibility allele frequency of 0.10, with a population frequency of disease of 0.02. The example files include:

- `aspexrc1` and `aspexrc2`: sample ASPEX startup files, with lots of comments.
- `c1.map`: a marker map for the simulated data, again with comments.
- `c1_all.dat`: affected status and genotype data for the 12 markers described in `c1.map`.
- `c1_1.dat`, `c1_2.dat`, `status.dat`: the same data as `c1_all.dat`, divided into several genotype data files and a file containing just the affected status information.

Here are some examples of simple analysis commands to try with these files, to familiarize yourself with ASPEX syntax:

```
# Basic multipoint IBD analysis with typed parents
sib_ibd -f c1.map c1_all.dat
# Suppress warnings about Mendelian incompatibilities
sib_ibd -q -f c1.map c1_all.dat
# The same, but arranging the input data differently
sib_ibd -q -f c1.map status.dat c1_1.dat c1_2.dat

# Multipoint using allele frequencies to fill in missing parents
sib_phase -f c1.map c1_all.dat
# The same, but for discordant pairs instead of affected pairs
sib_phase -f c1.map -c "set count_discordant true" c1_all.dat
# Score affected pairs for a specific relative risk ratio
sib_phase -f c1.map -c "set risk 5.0" c1_all.dat
# Generate a detailed multipoint map for plotting
sib_phase -q -v -f c1.map -c "set risk 5.0" c1_all.dat

# Remove data inconsistencies, then re-run analysis
sib_clean -f c1.map c1_all.dat > clean.dat
sib_phase -f c1.map clean.dat

# TDT with multi-allele score statistics for each marker
sib_tdt -f c1.map c1_all.dat
# Detailed TDT scores for each allele
sib_tdt -v -f c1.map c1_all.dat
# Shorthand for scoring just one marker
sib_tdt -v -c "set loc M7" c1_all.dat

# Generate a distance map from the sib data
sib_map -f c1.map c1_all.dat
# Shorthand for recombination distance between two markers
sib_map -q -c "set loc { M7 M8 }" c1_all.dat
```

13 Diagnostic messages

Most of the ASPEX diagnostics relate to errors in the marker data and/or parameter files, and all the programs generate similar messages:

? inconsistent data in family 'x': marker(s) 'x', 'x'...

The marker data at the specified position(s) contains a Mendelian inconsistency, possibly due to a typing error, nonpaternity, or a new mutation. The marker data at these positions will be left out of the IBD calculations for this family.

? likely typing error in family 'x' marker 'x': q = x

Based on sharing at flanking markers, and the value of the `error_freq` parameter, it is more likely that the data at this marker is a typing error, than for the data to be correct. The `q` value specifies the increase in likelihood if the data is treated as an error, in log units.

? map degeneracy detected in family %s

If a map contains multiple markers at the same location, and a family has an apparent recombination between colocalized markers, then probability calculations for that family become indeterminate, and the family will be dropped from all likelihood calculations. This message should only be generated when error detection is disabled.

? marker 'x' not defined: data will be ignored.

A data file contains data for the specified marker, however, this name was not defined in the `loc` array. Marker data for this locus will be skipped.

? invalid disease data for family 'x' person 'x' at file 'x' line x.

The disease status column for this individual does not contain a valid value. The disease status of parents is ignored, so errors will not be reported.

? invalid gender data for family 'x' person 'x' at file 'x' line x.

The gender column does not contain a valid value.

? invalid pedigree data for family 'x' person 'x' at file 'x' line x.

This error can only happen when reading linkage format data. It indicates a formatting error in the father, mother, and gender columns in the input data.

? family 'x' is not a nuclear family at file 'x' line x.

This error can only happen when reading linkage format data. It indicates that either a parent is missing, or the family spans more than two generations.

? inconsistent pedigree structure for family 'x' person 'x' at file 'x' line x.

This indicates that the structure of this family is not the same in different marker data files. For instance, a parent in one file is a sibling in another.

? inconsistent disease data for family 'x' person 'x' at file 'x' line x.

The disease status for this person disagrees with a value read from an earlier data file.

? inconsistent gender data for family 'x' person 'x' at file 'x' line x.

The gender of this person disagrees with a value read from an earlier data file.

? inconsistent data for marker 'x' for family 'x' person 'x' at file 'x' line x.

Two input files both contain allele data for the specified marker for this person, and the data does not agree.

? not enough data at file 'x' line x.

The specified line did not have the expected number of allele identifiers. Verify that the number of alleles specified in the header line agrees with the number of columns of marker data.

? too much data at file 'x' line x.

The specified line was processed successfully, but after all expected data had been read, there were some non-blank characters left over. Verify that the number of alleles specified in the header line agrees with the number of columns of marker data.

? family 'x' person 'x' is male but has two alleles at 'x'.

This error can only happen when reading sex-linked data. Males should either be coded as homozygous at all loci (in linkage format), or using the special 'Y' allele to indicate the Y chromosome (either format).