paramlink: An R package for parametric linkage analysis

Magnus Dehli Vigeland

March 20, 2011

This document gives an introduction to the R package `paramlink`, which provides various functions for SNP-based parametric linkage analysis, including LOD score calculations, power analysis, pedigree manipulation and exploratory methods. Likelihoods are calculated using the Elston-Stewart algorithm.

1 An example session

We begin by running through a very simple linkage analysis, in which we load a pedigree, plot it, set the disease model, and calculate the LOD score of a single marker.

To get started, we load the `paramlink` package and the toy pedigree given in the `toyped` dataset:

```
> library(paramlink)

> data(toyped)
> toyped

  ID FID MID SEX AFF M_A1 M_A2
1  1   0   0   1   2    1    0
2  2   0   0   2   1    1    1
3  3   1   2   1   2    1    2
4  4   1   2   1   2    1    2
```

The pedigree is a data frame whose columns are individual ID, father ID, mother ID, sex (male=1, female=2), affection status (healthy=1, affected=2, unknown=0), and and two columns containing the alleles of a single SNP marker. Individual 2 is homozygous for the '1' allele, 3 and 4 are heterozygous, and individual 1 has a missing allele.

This is the standard LINKAGE format for pedigrees, but without a "family ID" column. (Files with family ID column cause no trouble, see Section 6.)

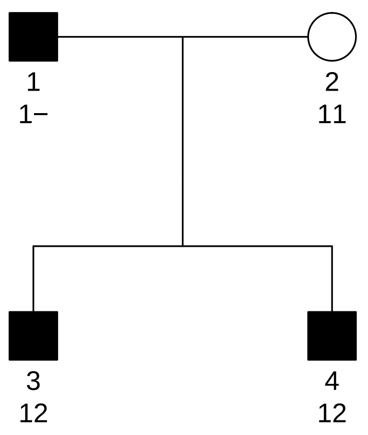
Most of the functions in `paramlink` take as input not data frames but objects of class `linkdat`. A `linkdat` object is basically a list containing various information about the pedigree, markers and the disease model. We transform the 'toy' data frame to a `linkdat` object as follows:

```
> x = linkdat(toyped)
```

To plot the pedigree with genotypes, use the `plot` method[1]:

```
> plot(x, marker = 1)
```

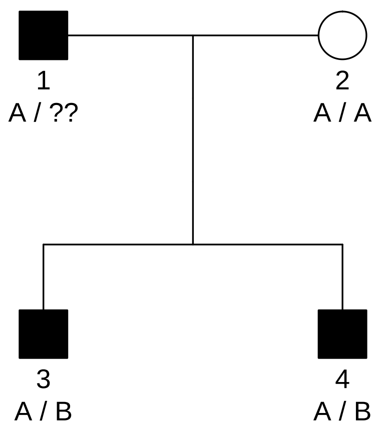
[1]The `plot.linkdat` function this is a wrapper for `plot.pedigree` in the `kinship` package.



The symbols are standard for medical pedigrees: female = circle; male = square; affected by disease = filled, non-affected = open.

The genotype labels can be changed using optional arguments to the `plot` function. The following example should be fairly self explanatory; see `?plot.linkdat` for more information:

```
> plot(x, marker = 1, alleles = c("A", "B"), missing = "??", sep = " / ")
```



Setting the model

To perform parametric linkage analysis, a *disease model* has to be described. The parameters include whether the disease is autosomal or X-linked, penetrance values, and allele frequencies for the disease and marker loci. In `paramlink` we set the model using the function `setModel`, whose argument `model` takes an integer value (1-4) with the following meaning:

1. Autosomal dominant
2. Autosomal recessive
3. X-linked dominant
4. X-linked recessive

If nothing else is indicated, the other parameters are given default values: Full penetrance, no phenocopies, disease allele frequency=0.00001, and equifrequent SNP markers (both alleles have frequency 0.5).

For our toy example we want an autosomal dominant model:

```
> x = setModel(x, model = 1)
```

This is a good moment to take a look at the `summary` of the `linkdat` object, to check that everything is as we expect:

```
> summary(x)
```

```
Pedigree:
4 individuals
2 founders, 2 nonfounders
1 nuclear subfamily

Marker data:
1 markers in total
0 individuals with no available genotypes
12.5% missing genotypes

Marker simulation:
No simulation indicated

Model parameters:
Autosomal inheritance with penetrances: (f0, f1, f2) = (0, 1, 1)
Disease allele frequency: 1e-05
Marker allele frequencies: 0.5 0.5
```

Note in particular, the `(f0, f1, f2)` parameters, which refer to the standard notation for penetrance values:

$$f_i = P(\text{affected} \mid i \text{ copies of the disease allele}).$$

See Section 2 for how to set model parameters other than the default values.

Calculating the LOD score

To calculate the LOD score of the marker in our toy pedigree (with the disease model we just set), simply write

```
> lod(x)
```

```
Computing singlepoint LOD scores at each marker
for the following recombination values:
  t = 0
  t = 0.1
  t = 0.2
  t = 0.5
Max LOD score: 0.3010257
Achieved at marker(s): M1
```

By default, the function calculates the LOD scores for each marker for a set of recombination fractions[2], and reports the maximum score (0.3 in our example).

We can specify the recombination fraction(s) we want using the `t` argument. Note also how to suppress the verbose output and instead inspect the LOD scores as a matrix:

[2]Similarly to the MLINK program of the LINKAGE suite.

```
> lods = lod(x, t = c(0, 0.01, 0.05), silent = T)
> lods

            M1
0    0.3010257
0.01 0.2923404
0.05 0.2576747
```

2 More about setting the disease model

As explained above, the `model` argument of `setModel` offers a very simple way to set one of the four standard models for rare, fully penetrant monogenic diseases. To indicate parameter values other than the defaults, these can be supplied using other arguments of `setModel`, the most important of which are `chrom` ("autosomal" or "X"), `penetrances`, `dfreq` (disease allele frequency) and `afreq` (marker allele frequencies). For example, the shortcut we used above for an autosomal dominant model,

```
> x = setModel(x, model = 1)
```

is equivalent to the command

```
> x = setModel(x, chrom = "autosomal", penetrances = c(0, 1, 1),
+     dfreq = 1e-05, afreq = c(0.5, 0.5))
```

If `x` already has a model, `setModel` uses the existing parameter values for any missing arguments. This makes it easy to change one parameter while keeping everything else as before. For example, the following command alters the penetrances (but nothing else) of `x`'s model, to give 1% phenocopy rate and 90% penetrance:

```
> x = setModel(x, penetrances = c(0.01, 0.9, 0.9))
> x$model

Model parameters:
Autosomal inheritance with penetrances: (f0, f1, f2) = (0.01, 0.9, 0.9)
Disease allele frequency: 1e-05
Marker allele frequencies: 0.5 0.5
```

For autosomal models, the `penetrances` argument should always be a vector of length 3, (f_0, f_1, f_2), whose values are assumed to hold for both males and females. In X-linked cases f_2 is meaningless for males (having only one X chromosome), and `penetrances` should be a list of two vectors, of the form `list(male = c(f0_m, f1_m), female = c(f0_f, f1_f, f2_f))`. For example, the default for X-linked recessive models (model 4) is `penetrances = list(male = c(0, 1), female = c(0, 0, 1))`.

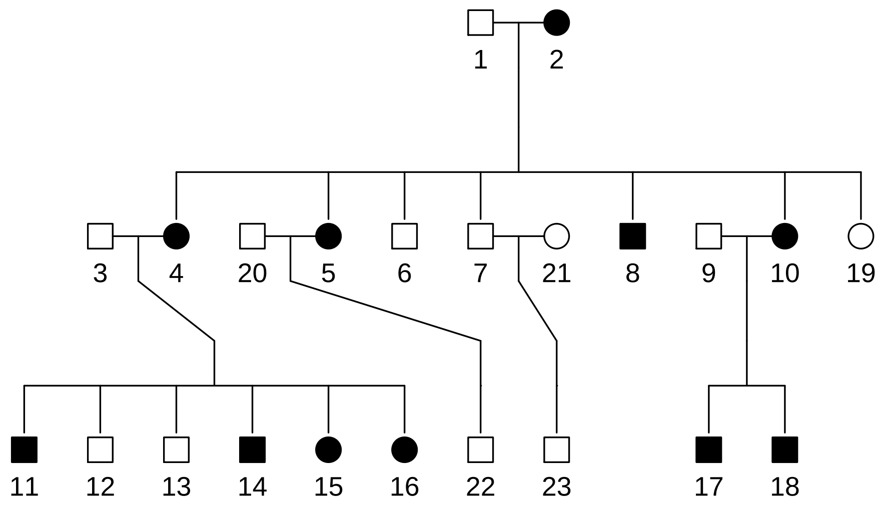
3 Computing and plotting LOD scores

We move on to a more interesting pedigree, contained in `largefam` dataset.

```
> data(largefam)
> y = linkdat(largefam, model = 1)
```

Notice that the disease model can be set already in the `linkdat` command, instead of using `setModel` as an additional step. We set the model to autosomal dominant, which is consistant with the pattern shown in the pedigree:

```
> plot(y)
```



Typing `summary(y)` will show that most of the family members are genotyped with 650 markers. We will compute single-point LOD scores for each marker and plot the results.

As we saw in the first section, the `lod` function by default computes LOD scores for each marker at the recombination fractions 0, 0.1, 0.2 and 0.5. In this example we will start by overriding this, testing only for complete linkage (recombination fraction equal to 0):

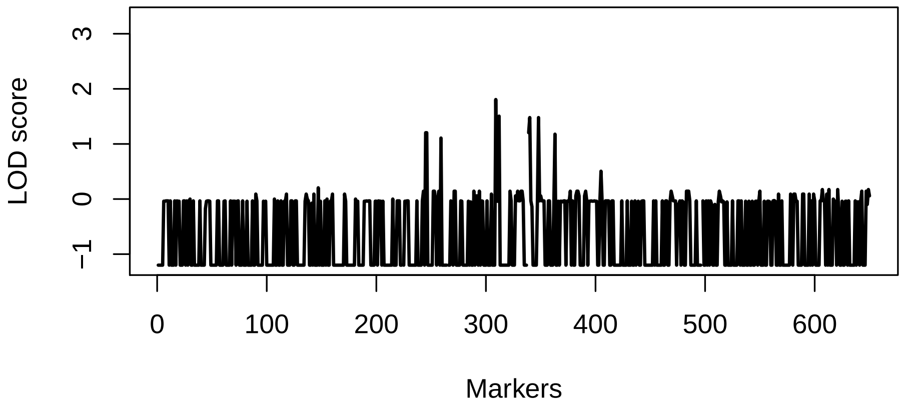
```
> lods0 = lod(y, t = 0)
```

```
Computing singlepoint LOD scores at each marker
for the following recombination value:
  t = 0
Max LOD score: 1.80618
Achieved at marker(s): M309
```

The results show a maximum LOD score of 1.8, obtained by the 309'th marker.

The output of any call to `lod` is an object of class `linkres`, for which a plot method has been written. Hence to visualize the LOD scores it suffices to write:

```
> plot(lods0)
```



Note: The genomic positions of the 650 markers is not part of the `largefam` dataset. Hence the x-axis shows only the numbers 1-650.

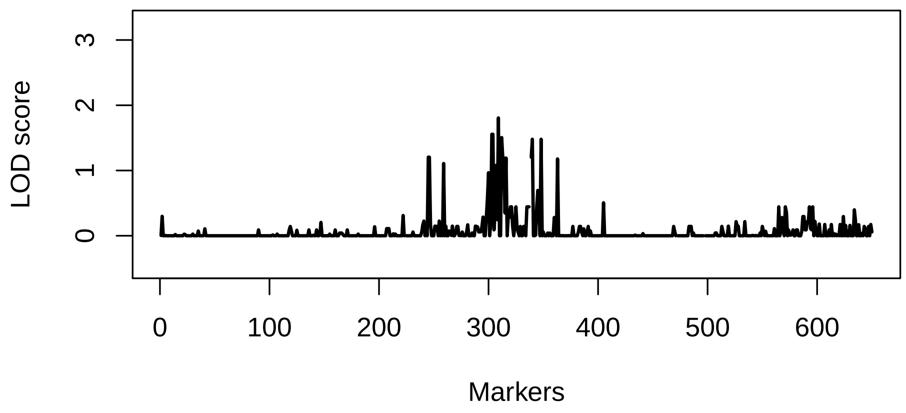
Instead of specifying a set of fixed recombination fractions, we can let the program find the optimal recombination fraction for each marker. This is implemented by first computing a maximum likelihood estimate of `t` (using R's own `optimize`), given the pedigree and the marker genotypes, and then computing the LOD score for this recombination fraction[3]. This behavior is achieved by setting `t="max"`, and is somewhat more time consuming than the approach above.

```
> lods.max = lod(y, t = "max")

Computing singlepoint LOD scores for each marker,
maximizing over all recombination values.

Max LOD score: 1.80618
Achieved at the following marker(s):
         M309
LOD   1.80618
t_max 0.00000

> plot(lods.max)
```



We see that the linkage peak is the same as before, but the curve looks rather different: While many LOD scores for $t = 0$ are negative (in fact $-\infty$ if the genotypes are incompatible with zero recombinations), the `t="max"` approach always gives $LOD \geq 0$ (since $t = 0.5$ implies $LOD = 0$).

Finally, note that `lod` takes an optional argument `markers` where the user can specify a subset of markers. LOD scores are then computed for these markers only. For example, let's zoom in on the markers surrounding number 309:

```
> lods.peak = lod(y, t = "max", markers = 306:312)

Computing singlepoint LOD scores for each marker,
maximizing over all recombination values.

Max LOD score: 1.80618
Achieved at the following marker(s):
         M309
LOD   1.80618
t_max 0.00000

> lods.peak
```

[3]This is similar to the ILINK program in the LINKAGE suite.

```
          M306       M307      M308    M309 M310 M311    M312
LOD   0.3485592 1.08009484 0.2474003 1.80618  0.0  0.0 1.50515
t_max 0.2316881 0.09236886 0.1444397 0.00000  0.5  0.5 0.00000
```

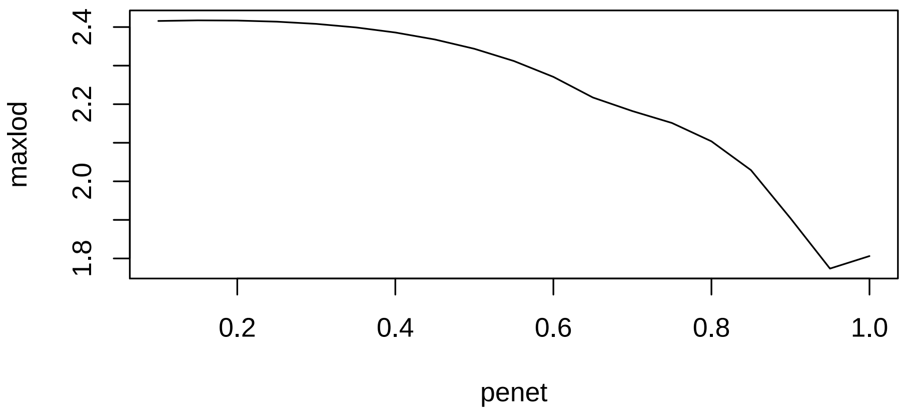
A more advanced example: Varying the penetrance

To main strength of `paramlink` is of course the environment within which it exists: R itself. In this section we give a quick example of combining `paramlink` with basic R functionality.

We address the following question: For the `largefam` dataset we analysed above – what happens to the LOD score peak if we vary the model parameters? In particular we will look at allowing reduced penetrance.

Reducing the penetrance is done by setting `penetrances=c(0, f, f)`, where `f` is some number less than 1. Instead of doing this one value at a time, we take advantage of R's `sapply`. The code below assumes `y` is as in the previous section. We use `t=0` to save time; `t="max"` would of course also work. The argument `max.only=T` forces `lod` to return only the maximum LOD score, instead of all the scores collected in a `linkres` object.

```
> penet = seq(from = 0.1, to = 1, by = 0.05)
> maxlod = sapply(penet, function(f) {
+     z = setModel(y, penetrances = c(0, f, f))
+     lod(z, t = 0, max.only = T)
+ })
> plot(penet, maxlod, type = "l")
```



The result is quite interesting: The LOD score peak gets higher when we reduce the penetrance. This suggests that something is wrong: Either the model is incorrect (i.e. disease is not autosomal dominant) or perhaps there are mistakes in the pedigree. We will come back to this in Section 5.

4 Genotype probability distributions

A novel feature of `paramlink` is the possibility of computing genotype probability distributions for specified individuals, conditional on pedigree/disease/partial marker data.

For an example we go back to the toy pedigree and make up a different marker. Note how `setMarkers` works: Its input is an existing `linkdat` object and a matrix containing the alleles (one row for each individual; two columns), and outputs a `linkdat` object with the given genotypes. The default symbol for missing alleles is 0.

```
> toy = linkdat(toyped, model = 1)
```

```
Pedigree read, 4 individuals
No simulation column found
Peeling order set, 1 maximal nuclear subfamily.
1 markers read.
```

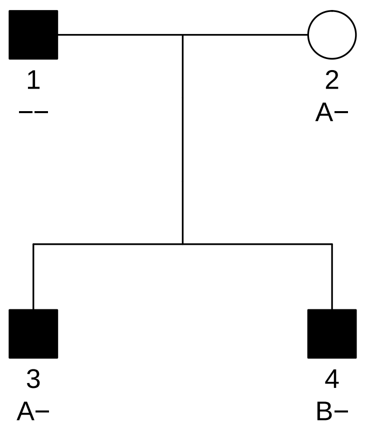
```
> new.marker = rbind(c(0, 0), c("A", 0), c("A", 0), c("B", 0))
> new.marker
```

```
     [,1] [,2]
[1,] "0"  "0"
[2,] "A"  "0"
[3,] "A"  "0"
[4,] "B"  "0"
```

```
> toy = setMarkers(toy, new.marker)
```

After setting a new marker it's a good idea to check the result by plotting the pedigree:

```
> plot(toy, marker = 1)
```



What is the probability that – independently of the disease – individual 3 is homozygous for the A allele? The genoDistr function gives the answer:

```
> genoDistr(toy, id = 3, partialmarker = 1)
```

```
Computing genotype probabilities for individual 3
conditional on the following existing genotypes:
  ID M1
1  1 --
2  2 A-
3  3 A-
4  4 B-

Using default recombination t=0.5, i.e. marker segregates independently of disease.

Results:
  AA   AB   BB
0.28 0.72 0.00
```

Hence the probability that individual 3 is homozygous is 28%. The default behavior of genoDistr is to assume the marker is unlinked to the disease, i.e. recombination fraction $t = 0.5$. However, the user can specify any recombination fraction. For example, given that the marker we created is completely linked to the disease ($t = 0$), the genotype probability distribution of individual 3 is:

```
> genoDistr(toy, id = 3, partialmarker = 1, t = 0)

Computing genotype probabilities for individual 3
conditional on the following existing genotypes:
  ID M1
1  1 --
2  2 A-
3  3 A-
4  4 B-

Results:
       AA        AB        BB
0.1666690 0.8333310 0.0000000
```

5 Power calculations

The questions usually addressed in power analyses for linkage projects are

1. What is the *maximum* LOD score obtainable in a given pedigree for a marker completely linked to the disease?
2. What is the *expected* LOD score (the ELOD) for a completely linked marker?

For non-trivial cases a simulation based approach is normally used to answer these. In `paramlink` this is implemented in the `linkage.power` function, which simulates a specified number of SNPs conditional on the pedigree, disease and model data[4], and summarizes the LOD scores for these markers.

As an example, let us return to the `largefam` pedigree, and run a power analysis for this pedigree:

```
> linkage.power(y)

Simulating genotypes for the following individuals:
   1, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18
100 markers simulated
Highest singlepoint LOD score for simulated markers: 3.61236
Markers obtaining maximum score: 5 %
ELOD: 1.300426
```

By default, only those individuals that were originally genotyped, are simulated. We could change this by adding an argument `all=TRUE`, or by setting `y`'s simulation vector `y$sim` (see next section).

The default number of simulated markers is 100. To specify a different number, use the `N` argument. For a family of this size it is advisable to simulate a lot more than 100 markers.

Also we could inquire how many of the markers exceed 3 (a commonly used significance threshold for autosomal disease models) or any other threshold. This is done by including the argument `threshold=3` in the function call.

Finally, to generate reproducible output it can be a good idea to set the seed for the random number generator manually, using the `seed` argument.

To sum up, a typical call to `linkage.power` would be something like this (but with a much larger value of N):

```
> simLODs = linkage.power(y, N = 5, threshold = 3, seed = 1111)

Simulating genotypes for the following individuals:
   1, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18
5 markers simulated
Highest singlepoint LOD score for simulated markers: 3.61236
Markers obtaining maximum score: 20 %
ELOD: 1.451180

Threshold: 3
Markers with score above threshold: 20 %
```

[4]Using a similar algorithm as the SLINK program in the LINKAGE/FASTLINK suite.

```
> simLODs

        M1       M2        M3         M4        M5
0 1.028029 3.61236 1.177791 0.3912066 1.046512
```

Comparing the result of the power analyses above with the LOD scores in Sect. 3 we see that none of the real markers came close to the maximum of 3.6. As it turns out, two of the family members (6 and 12) were wrongly diagnosed; they should be marked as affected. We fix this using the `modifyPedigree` command (see Sect. 7)

```
> y = modifyPedigree(y, c(6, 12), "AFF")
```

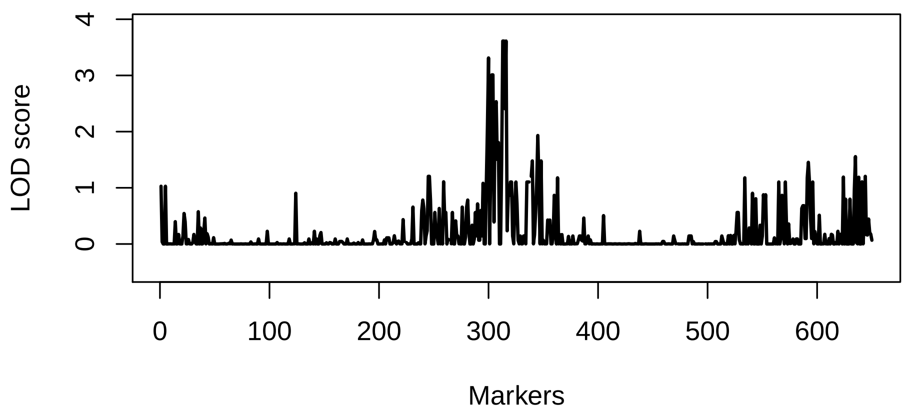
Computing the LOD scores using the modified pedigree gives more promising results:

```
> lods = lod(y, t = "max")

Computing singlepoint LOD scores for each marker,
maximizing over all recombination values.

Max LOD score: 3.61236
Achieved at the following marker(s):
         M313    M314    M316
LOD   3.61236 3.61236 3.61236
t_max 0.00000 0.00000 0.00000

> plot(lods)
```

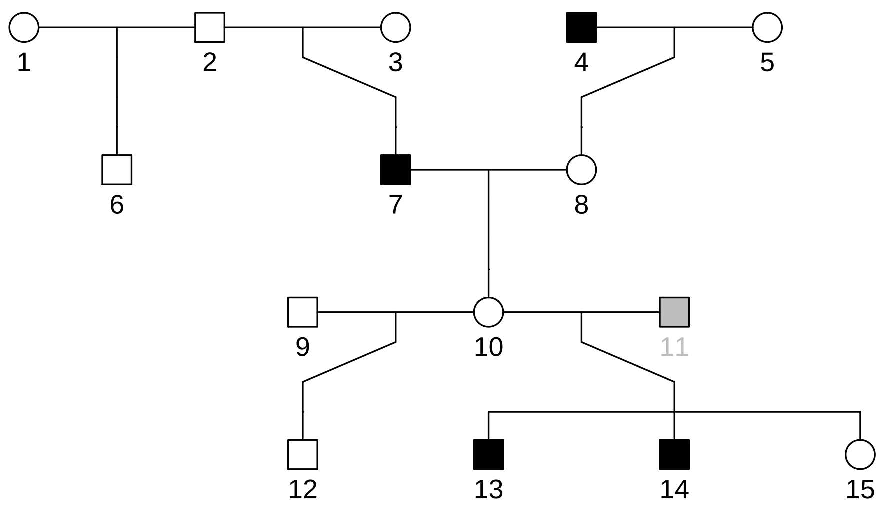


5.1 An example with X-linked disease

In this section we perform a power analysis of the `Xped` pedigree included in the package. This is a complex pedigree with several half-sibships. The disease pattern is consistent with an X-linked recessive inheritance mode, so we indicate this as our model (cf. Section 1):

```
> data(Xped)
> z = linkdat(Xped, model = 4, verbose = F)

> plot(z)
```



Let's check the maximum LOD score:

```
> linkage.power(z)

Simulating genotypes for the following individuals:
   1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
100 markers simulated
Highest singlepoint LOD score for simulated markers: 1.204122
Markers obtaining maximum score: 20 %
ELOD: 0.4816481
```

What would the power be if only the boys 12-14 were available for genotyping? By default, genotypes for all individuals are simulated, but we can change this by setting the `sim` vector of `z`. This vector should consist of 2's and 0's, meaning *included* and *not included* respectively.

```
> z$sim = rep(0, 15)
> z$sim[12:14] = 2
> linkage.power(z)

Simulating genotypes for the following individuals:
   12, 13, 14
100 markers simulated
Highest singlepoint LOD score for simulated markers: 0.60206
Markers obtaining maximum score: 52 %
ELOD: 0.2665544
```

As we see, the maximum LOD score using only the boys is 0.6, only half of what we can achieve with everyone included. (Of course some members of the family do not contribute any linkage information, e.g. male founders without daughters, so including *everyone* would be a waste of time and resources.)

6 Reading and writing files

The package includes the functions `read.linkdat` and `write.linkdat` for reading and writing pedigree files in LINKAGE format. These functions are basically wrappers for `read.table` and `write.table`.

As an example we will write the `toyped` pedigree to a file called "test.ped":

```
> data(toyped)
> x = linkdat(toyped)
> write.linkdat(x, file = "test.ped")
```

By default the resulting pedfile is in accordance with traditional (pre-makeped) LINKAGE format: No column headers, and family ID as the first column. (This column will always be all 1's, since `linkdat` objects contain only one family.) These actions can be controlled by using the optional arguments `col.names` and `famid`:

```
> write.linkdat(x, file = "test2.ped", famid = F, col.names = T)
```

Reading pedigree files is equally simple. To read the files "test.ped" and "test2.ped" we just created (the resulting `linkdat` objects will be equal), the appropriate commands are

```
> x1 = read.linkdat(file = "test.ped")
> x2 = read.linkdat(file = "test2.ped", header = T)
```

If the input file contains a family ID column, this will be detected automatically. If the file contains a column indicating simulation status (this should be in SLINK format: 0=not included, 2=included), the user must specify the column number using the `simcol` argument of `read.linkdat`. See help files for details.

7 Pedigree creation and manipulation

As an alternative to writing a ped-file describing your pedigree in LINKAGE format, it is often convenient to let `paramlink` create the pedigree for you. There are also several functions for modifying both the pedigree and the marker alleles of existing `linkdat` objects. All of these should be fairly easy to understand from their help pages, so we will just give a few simple examples.

The basic functions for *creating* pedigrees are `nuclearPed` and `cousinPed`. The former makes a nuclear family with a specified number of male and female offspring, while the latter makes a pedigree linking two cousins of the specified degree. The output of both functions are `linkdat` objects with no model set and with all individuals non-affected.

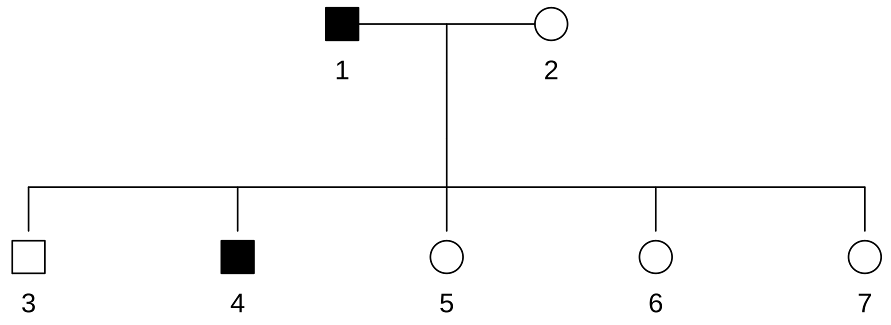
Functions for *manipulating* pedigrees include `addChildren` (which adds a specified number of offspring to the indicated parents), `modifyPedigree` (for changing either the sex or affection status of the specified individuals), `removeIndiv` (removes an individual and all its descendants) and `subset.linkdat` (for extracting a sub-pedigree and/or a subset of the markers).

For manipulating the marker data of a `linkdat` object, use `modifyMarker` (to modify the alleles of a single marker) and `setMarkers` (to reset all the marker info).

It should be noted that all of these functions – like `setModel` – return new `linkdat` objects, i.e., they do not do in-place modifications. Hence their output should always be stored in some variable. We conclude this section with an example to give a rough idea of how to use some of these functions. First, let's create a pedigree of a nuclear family with two affected boys and three non-affected girls:

```
> a = nuclearPed(boys = 2, girls = 3)
> a = modifyPedigree(a, id = c(1, 4), "AFF")

> plot(a)
```



Now let's do a few changes:
Give individual 4 two affected children, one boy and one girl. Note: `mother=0` creates a new founder.

```
> a = addChildren(a, father = 4, mother = 0, children = 2, sex = c(1,
+     2), aff = 2)
```

Create a marker for which everyone is homozygous AA:

```
> a = setMarkers(a, matrix("A", nrow = 10, ncol = 2))
```

Making individuals 1,4 and 10 heterozygous, and giving 9 a missing allele:

```
> a = modifyMarker(a, id = c(1, 4, 10), alleles = c("A", "B"))
> a = modifyMarker(a, id = 9, alleles = c("A", 0))
```

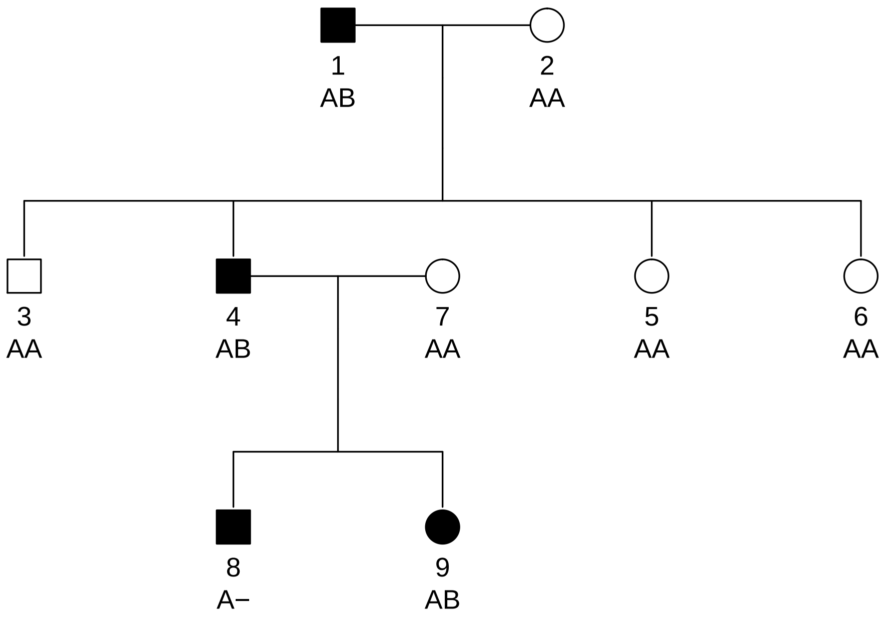
Removing individual 7:

```
> a = removeIndiv(a, 7)
```

```
Pedigree read, 9 individuals
Relabeled individuals:
 1 -> 1
 2 -> 2
 3 -> 3
 4 -> 4
 5 -> 5
 6 -> 6
 8 -> 7
 9 -> 8
 10 -> 9
No simulation column found
Peeling order set, 2 maximal nuclear subfamilies.
1 markers read.
```

The new pedigree looks like this:

```
> plot(a, marker = 1)
```


1
AB
2
AA
3
AA
4
AB
7
AA
5
AA
6
AA
8
A–
9
AB