OneMap Tutorial

Software for constructing genetic maps in outcrossing species

Gabriel Rodrigues Alves Margarido[1]
Marcelo Mollinari[1]
Antonio Augusto Franco Garcia[1*]

[1]Department of Genetics
Escola Superior de Agricultura "Luiz de Queiroz" (ESALQ), Universidade de São Paulo (USP)
Av. Pádua Dias, 11 - Caixa Postal 83
CEP: 13400-970 - Piracicaba - São Paulo - Brazil
Tel: +55 19 34294125
Fax: +55 19 34336706
E-mail: aafgarci@esalq.usp.br

[*]corresponding author

http://www.r-project.org
April 28, 2009

Overview

OneMap is an environment for constructing linkage maps in outcrossing plant species, using full-sib families derived from two outbred (non-homozygous) parents. It is implemented as a package to be used under the freely distributed R software, which is a language and environment for statistical computing (www.r-project.org).

Wu et al. (2002a) proposed a methodology to construct genetic maps in outcrossing species, which allows the analysis of a mixed set of different marker types containing various segregation patterns. Also, it allows the simultaneous estimation of linkage and linkage phases between markers, and it was successfully applied in the analysis of sugarcane data sets (Garcia et al., 2006; Oliveira et al., 2007). Actually, the analysis of these sugarcane data sets motivated the implementation of *OneMap*.

Despite those results, the construction of linkage maps could be greatly enhanced with the use of multipoint likelihood through Hidden Markov Models (HMM). Jiang and Zeng (1997) explained in detail this methodology, emphasizing its advantages and limitations, for populations derived from inbred lines. Merging the ideas of Wu et al. (2002a) and the HMM framework, as done by Wu et al. (2002b), led us to develop this new version of *OneMap*.

This new version (1.0-0) handles two-point analysis between markers and performs the grouping step in the same way in the previous version (0.1-1), following Wu et al. (2002a)

approach. However, marker ordering in a linkage group can be done not only using Rapid Chain Delineation - RCD (Doerge, 1996), but also through HMM-based algorithms, in the same way as implemented in MAPMAKER/EXP (Lander et al., 1987). The three-point test is maintained as defunct in version 1.0-0 only for historical reasons.

OneMap is available as source code for Windows™ and Unix. It is released under the GNU General Public License, is open-source and the code can be changed freely. It comes with no warranty.

To download the free software R, please visit the Comprehensive R Archive Network (cran.r-project.org). The user of *OneMap* is supposed to have some experience with R, since the analysis is done using the command line. R comes with a 'getting started manual' and various others useful documents can be found on CRAN and by searching the web. This tutorial will not discuss basic R usage and assumes that the user has some experience on it.

After installing R, *OneMap* can be installed by opening R and issueing the command

```
> install.packages("onemap")
```

OneMap can also be installed by downloading the appropriate files directly at the CRAN web site and following the instructions given in the section "6.3 Installing Packages" of the "R Installation and Administration" manual (http://cran.r-project.org/doc/manuals/R-admin.pdf).

Citation

Margarido, G.R.A., Souza, A.P. and Garcia, A.A.F. OneMap: software for genetic mapping in outcrossing species. ***Hereditas*** 144: 78-79, 2007.

Introduction to OneMap

OneMap is comprised by a small set of functions (listed below). There are other functions used internally by the software. However, you do not need to use them directly.

Getting started

The following example is intended to show the usage of all *OneMap* functions. With basic knowledge of R syntax, one should have no big problems using it. It is assumed that the user is running Windows™. Hopefully, these examples will be clear enough to help any user to understand its functionality and start using it.

1. Start R by double-clicking its icon.

Function type	Function name	Function description
Input	read.outcross	Read data from an outcross
Data manipulation	make.seq	Creates a sequence of markers based on objects of other types
	marker.type	Informs the segregation type of a genetic marker
	add.marker	Adds markers to a sequence
	drop.marker	Drops markers from a sequence
Genetic mapping	rf.2pts	Estimates recombination fractions (two points)
	group	Assigns markers to linkage groups
	rcd	Orders markers in a sequence using RCD algorithm
	compare	Compares all possible orders of markers in a sequence
	try.seq	Tries to map a marker into a given linkage group
	order.seq	Automates map construction through "compare" and "try.seq" functions
	ripple	Compares alternative orders for a map and displays the plausible ones
	map	Constructs a multipoint linkage map for a sequence in a given order
	rf.graph.table	Plots a pairwise recombination fraction matrix using a color scale.
Defunct	def.rf.3pts	Estimates recombination fractions (three points method)

2. Load *OneMap* (after installing it):

   ```
   > library(onemap)
   ```

3. To save your project anytime, type:

   ```
   > save.image("C:/.../yourfile.RData")
   ```

   or access the toolbar File → Save Workspace.

Creating the data file

This step might be quite difficult, since the data file is not very simple and many errors can occur while reading it. The input file format is similar to that used by MAPMAKER/EXP (Lander et al., 1987), so experienced users of genetic analysis software should be familiarized with it.

Basically, the input file is a text file with a first line indicating the number of individuals and the number of markers. Then, the genotype information is included separately for each marker.

The character "*" indicates the beginning of information input for a new marker, followed by the marker name. Next, there is a code indicating the marker type, according to Wu's et al. (2002a) notation. In short, Wu et al. (2002a) classified marker types in the following way:

			Parent						Offspring	
		crosstype	Cross			Observed bands			Observed bands	Segregation
A		1	ab	×	cd	ab	×	cd	ac, ad, bc, bd	1:1:1:1
		2	ab	×	ac	ab	×	ac	a, ac, ba, bc	1:1:1:1
		3	ab	×	co	ab	×	c	ac, a, bc, b	1:1:1:1
		4	ao	×	bo	a	×	b	ab, a, b, o	1:1:1:1
B	B_1	5	ab	×	ao	ab	×	a	$ab, 2a, b$	1:2:1
	B_2	6	ao	×	ab	a	×	ab	$ab, 2a, b$	1:2:1
	B_3	7	ab	×	ab	ab	×	ab	$a, 2ab, b$	1:2:1
C		8	ao	×	ao	a	×	a	$3a, o$	3:1
D	D_1	9	ab	×	cc	ab	×	c	ac, bc	1:1
		10	ab	×	aa	ab	×	a	a, ab	1:1
		11	ab	×	oo	ab	×	o	a, b	1:1
		12	bo	×	aa	b	×	a	ab, a	1:1
		13	ao	×	oo	a	×	o	a, o	1:1
	D_2	14	cc	×	ab	c	×	ab	ac, bc	1:1
		15	aa	×	ab	a	×	ab	a, ab	1:1
		16	oo	×	ab	o	×	ab	a, b	1:1
		17	aa	×	bo	a	×	b	ab, a	1:1
		18	oo	×	ao	o	×	a	a, o	1:1

Actually, it is strongly recommended to check Wu et al. (2002a) paper before using *OneMap*. Marker types must be one of the following: *A.1, A.2, A.3, A.4, B1.5, B2.6, B3.7, C.8, D1.9, D1.10, D1.11, D1.12, D1.13, D2.14, D2.15, D2.16, D2.17* or *D2.18*. The letter and the number before the dot indicate the segregation type (i.e., 1:1:1:1, 1:2:1, 3:1 or 1:1), while the number after the dot indicates the offspring observed bands. The paper cited above gives details with respect to marker types; we will not discuss them here.

Finally, after each marker name, comes the genotype data for the segregating population. The coding for marker genotypes used by *OneMap* is also the same one proposed by Wu et al. (2002a) and the possible values vary according to the marker type. Missing data are indicated with the character "-" and a comma separates the information for each individual.

Here is an example of such file for 10 individuals and 5 markers:

```
10 5
*M1 B3.7        ab,ab,-,ab,b,ab,ab,-,ab,b
*M2 D2.18       o,-,a,a,-,o,a,-,o,o
*M3 D1.13       o,a,a,o,o,-,a,o,a,o
*M4 A.4         ab,b,-,ab,a,b,ab,b,-,a
*M5 D2.18       a,a,o,-,o,o,a,o,o,o
```

The input file must be saved in text format, with extensions like ".txt". It is a good idea to open the text file called "example_out.txt", available with *OneMap*, and saved on the directory you installed it, to see how your file should be.

Importing data

1. Once the input file is created, data can be loaded into R. The function used to import data is named `read.outcross`. Its usage is quite simple:

   ```
   > example_out <- read.outcross("C:/workingdirectory", "example_out.txt")
   ```

   The first argument is the directory where the input file is located and the second one is the file name.

2. You can change the working directory in R using function `setwd()` or in the toolbar clicking File → Change dir. If you set your working directory to the one containing the input file, you can just type:

   ```
   > example_out <- read.outcross(file = "example_out.txt")
   ```

   If no error has occurred, a message will display some basic information about the data, such as number of individuals and number of markers:

3. Because this particular data set is distributed along with the package, you can load it simply typing

   ```
   > data(example_out)
   ```

4. Loading the data creates an object of class `outcross`, which will further be used in the analysis. R command `print` recognizes objects of this class. Thus, if you type

   ```
   > example_out
   ```

   you will see some information about the object.

Estimating two-point recombination fractions

1. To start the mapping analysis, the first step is estimating the recombination fraction between all pairs of markers, using two-point tests:

   ```
   > twopts <- rf.2pts(example_out)
   ```

This command uses as default values of LOD-Score 3 and maximum recombination fraction 0.35.

2. Different values for the criteria can be chosen using:

```
> twopts <- rf.2pts(example_out, LOD = 3, max.rf = 0.4)
```

3. Although the two-point test was implemented in C language, which is much faster than R, this step can take quite some time, depending on the number of markers involved and their segregation type, since all combinations will be estimated and tested. Besides, the results use a lot of memory and a rather powerful computer is needed. For example, the analysis of a real data set with 1741 markers (segregating 3:1 and 1:1) took 2.8 hours, running under Windows™ on a Pentium® 4 CPU 3.00 GHz with 1 GB RAM memory.

4. When the two-point analysis is finished, an object of class `rf.2pts` is created. Typing

```
> twopts
```

will show a message with the criteria used in the analysis and information about printing details of this object:

5. If you want to see the results for given markers, say `M1` and `M3`, the command is:

```
> print(twopts, "M1", "M3")
```

Each line indicates a possible linkage phase. 1 denotes coupling phase in both parents (CC), 2 and 3 denote coupling phase in parent 1 and 2, respectively, and repulsion in the other (CR and RC), and 4 denotes repulsion phase in both parents (RR).

Assigning markers to linkage groups

1. Once the recombination fractions and linkage phases for all pairs of markers have been estimated and tested, markers can be assigned to linkage groups. To do this, first use the function `make.seq` to create a sequence with all markers:

```
> mark.all <- make.seq(twopts, "all")
```

The function `make.seq` is used to create sequences from objects of several kinds, as will be seen along this tutorial. Here, the object is of class `rf.2pts` and the second argument specifies which markers one want to use. In this example, `"all"` indicates that all markers will be analyzed. If one wants to use only markers one and two, for example, the option will be `c(1,2)`. These numbers refer to the lines where the markers are located on the data file. Function `marker.type` could be helpful (`M1` and `M2` in this case).

2. The grouping step is very simple and can be done by using the function `group`:

   ```
   > LGs <- group(mark.all)
   ```

   For this function, optional arguments are `LOD` and `max.rf`, which define thresholds to be used when assigning markers to linkage groups. If none provided (default), criteria previously defined for the object `twopts` are used.

3. The previous command generates an object of class `group` and the command `print` for such object has two options. If you simply type:

   ```
   > LGs
   ```

   you will get detailed information about the groups, i.e., all linkage groups will be printed, displaying the names of markers in each one of them.

   However, in case you just want to see some basic information (such as the number of groups, number of linked markers and more), the command is:

   ```
   > print(LGs, detailed = FALSE)
   ```

4. You can notice that all markers are linked to some linkage group. If the LOD-Score threshold is changed to a higher value, some markers are kept unassigned:

   ```
   > LGs <- group(mark.all, LOD = 6)
   > LGs
   ```

5. Changing back to the previous criteria, now setting the maximum recombination fraction to 0.40:

   ```
   > LGs <- group(mark.all, LOD = 3, max.rf = 0.4)
   > LGs
   ```

Genetic mapping of linkage group 3

1. When marker assignment to linkage groups is finished, the mapping step can take place. First of all, you must set the mapping function that should be used to display the genetic map through the analysis. You can choose between Kosambi or Haldane mapping function. To use Haldane, type

   ```
   > set.map.fun(type = "haldane")
   ```

   To use Kosambi

```
> set.map.fun(type = "kosambi")
```

Now, you must define which linkage group will be mapped. In other words, a linkage group must be "extracted" from the object of class `group`, in order to be mapped. For simplicity, we will start here with the smallest one, which is linkage group 3. This can be easily done using the following code:

```
> LG3 <- make.seq(LGs, 3)
```

The first argument (`LGs`) is an object of class `group` and the second is a number indicating which linkage group will be extracted, according to the results present in object `LGs`. The object `LG3`, generated by function `make.seq`, is of class `sequence`, showing that this function can be used with several types of objects.

2. If you type

   ```
   > LG3
   ```

   you will see which markers are comprised in the sequence, and also that no parameters are estimated.

3. To obtain a preliminary order of these markers, the `rcd` function can be used:

   ```
   > LG3.rcd <- rcd(LG3)
   ```

4. To order the markers in linkage group 3 (`LG3`), by evaluating all possible orders, the function `compare` can be used:

   ```
   > LG3.comp <- compare(LG3)
   ```

   This step takes some time, because this sequence contains one marker of type D1 and one of type D2, besides one marker segregating in 3:1 fashion (type C). Thus, although the number of possible orders is relatively small (60), for each order there are various possible combinations of linkage phases. Also, the convergence of the EM algorithm takes considerably more time, since markers of type C are not very informative.

   The first argument is an object of class `sequence` (the extracted group `LG3`) and the object generated by this function is of class `compare`.

5. To see the results of the previous step, type

   ```
   > LG3.comp
   ```

By default, the software stores 50 orders, which may or may not be unique.

In this example, note that there are nine possible unique orders. The value of `LOD` refers to the overall LOD-Score, considering all orders tested. `Nested LOD` refers to LOD-Scores *within* a given order, i.e., scores for different combinations of linkage phases for the same order.

For example, order 1 has the largest value of log-likelihood and, therefore, its LOD-Score is zero for some combination of linkage phases. For this same order and other linkage phases, LOD-Score is -2.43. g Analyzing the results for order 2, notice that its highest LOD-Score is very close to zero, indicating that this order is also quite plausible. Notice also that `Nested LOD` will always contain at least one zero value, corresponding to the best combination of phases for markers in a given order.

6. Since it is a good idea to choose the order with the highest likelihood, the final map can be obtained with the command

   ```
   > LG3.final <- make.seq(LG3.comp, 1, 1)
   ```

   The first argument is the object of class `compare`. The second argument indicates which order is chosen: 1 is for the order with highest likelihood, 2 is for the second best, and so on. The third argument indicates which combination of phases is chosen for this specific order: 1 also means the combination with highest likelihood.

   For simplicity, these values are defaults, so typing

   ```
   > LG3.final <- make.seq(LG3.comp)
   ```

   will have the same effect.

7. To see the final map, simply type

   ```
   > LG3.final
   ```

   At the leftmost position, marker names are displayed. `Position` shows the cumulative distance using the Kosambi mapping function, as defined above. Finally, `Parent 1` and `Parent 2` show the diplotypes of both parents, that is, the manner in which alleles are arranged in the chromosomes. Notation is the same as that used by Wu et al. (2002a).

Genetic mapping of linkage group 2

1. Now let us map the markers in linkage group number 2. Again, "extract" that group from the object `LGs`:

```
> LG2 <- make.seq(LGs, 2)
> LG2
```

Note that there are 10 markers in this group, so it is unfeasible to use the `compare` function with all of them since it will take a long time to proceed.

2. First, use `rcd` to get a first order estimate:

```
> LG2.rcd <- rcd(LG2)
> LG2.rcd
```

3. Use the `marker.type` function to check the segregation types of all markers in this group:

```
> marker.type(LG2)
```

4. Based on their segregation types and distribution on the preliminary map, markers M4, M9, M19, M20 and M24 are the most informative ones (type A is the better, followed by type B). So, let us create a framework of markers using `compare`:

```
> LG2.init <- make.seq(twopts, c(4, 23, 19, 20, 24))
> LG2.comp <- compare(LG2.init)
> LG2.comp
```

Now, the first argument to `make.seq` is an object of class `rf.2pts`, and the second argument is a vector of integers, specifying which molecular markers will constitute the sequence.

5. Select the best order:

```
> LG2.frame <- make.seq(LG2.comp)
```

6. Next, try to map the remaining markers. Since there are more markers of type D1 than D2, the latter will be tried in the end:

```
> LG2.extend <- try.seq(LG2.frame, 9)
> LG2.extend
```

Based on the LOD-Scores, marker M9 is probably located between markers M23 and M24. However, the "*" symbol indicates that more than one linkage phase is possible. Detailed results can be seen with:

```
> print(LG2.extend, 5)
```

where the second argument indicates the position where the marker was placed. Note that the first allele arrangement is most likely.

7. The best order can be obtained with:

```
> LG2.frame <- make.seq(LG2.extend, 5, 1)
```

When using `make.seq` with an object of class `try`, the second argument is the position on the map (according to the scale to the right in the output) and the last argument indicates the combination of phases. The same can be done with (omitting the combination of phases):

```
> LG2.frame <- make.seq(LG2.extend, 5)
> LG2.frame
```

Continuing with other markers, one by one:

```
> LG2.extend <- try.seq(LG2.frame, 29)
> LG2.extend
> LG2.frame <- make.seq(LG2.extend, 7)
> LG2.frame
> LG2.extend <- try.seq(LG2.frame, 27)
> LG2.extend
> LG2.frame <- make.seq(LG2.extend, 1)
> LG2.frame
> LG2.extend <- try.seq(LG2.frame, 16)
> LG2.extend
> LG2.frame <- make.seq(LG2.extend, 2)
> LG2.frame
> LG2.extend <- try.seq(LG2.frame, 21)
> LG2.extend
> LG2.final <- make.seq(LG2.extend, 6)
```

8. Compare the order obtained through `rcd` and that obtained via HMM:

```
> LG2.rcd
> LG2.final
```

Although `rcd` can result in a few different orders for this linkage group, the likelihood of the HMM-based order is always higher in this case.

9. This process of adding markers sequentially can be automated with the use of function `order.seq`.

   ```
   > LG2.ord <- order.seq(LG2, n.init = 5, THRES = 3)
   ```

   In the syntax above, `n.init = 5` means that five markers (the most informative ones) will be used in the `compare` step; `THRES = 3` indicates that the `try.seq` step will only add markers to the sequence which can be mapped with LOD-Score higher than 3. NOTE: Although very useful, this function can be misleading, specially if there are not many fully informative markers, so use it carefully.

10. Check the final order:

    ```
    > LG2.ord
    ```

    and note that markers `16`, `21` and `29` could not be safely mapped to a single position (`LOD-Score > THRES` in absolute value). The output displays the "safe" order and the most likely positions for unmapped markers, where "***" indicates the most likely position and "*" corresponds to other plausible positions.

11. To get the safe order, use

    ```
    > LG2.safe <- make.seq(LG2.ord, "safe")
    > LG2.safe
    ```

    and to get the order with all markers, use

    ```
    > LG2.all <- make.seq(LG2.ord, "force")
    > LG2.all
    ```

    Notice that, for this linkage group, the "forced" map obtained with `order.seq` is the same as that constructed before with `compare` plus `try.seq`, but *this is not always the case.*

12. The `order.seq` function can perform two rounds of the `try.seq` step, first using `THRES` and then `THRES - 1` as threshold. This generally results in safe orders with more markers mapped, but takes longer to run. To do this,type:

    ```
    > LG2.ord <- order.seq(LG2, n.init = 5, THRES = 3, touchdown = TRUE)
    > LG2.ord
    ```

    For this particular sequence, the `touchdown` step could not map any additional marker, but this will not happen all time.

13. Finally, to check for alternative orders, use the `ripple` function:

```
> ripple(LG2.final, ws = 4, LOD = 3)
```

The second argument, `ws = 4`, means that subsets (windows) of four markers will be permutated sequentially (4! orders for each window), to search for other plausible orders. The `LOD` argument means that only orders with LOD-Score smaller than 3 will be printed.

14. The `ripple` command showed that the final order obtained is indeed the best for this linkage group. The map can than be obtained using

```
> LG2.all
```

Genetic mapping of linkage group 1

1. Finally, linkage group 1 (the largest one) will be mapped. Extract and map it using `rcd`:

```
> LG1 <- make.seq(LGs, 1)
> LG1.rcd <- rcd(LG1)
> LG1.rcd
```

2. Construct the linkage map:

```
> LG1.ord <- order.seq(LG1, n.init = 6, touchdown = TRUE)
> LG1.ord
```

Notice that the second round of `try.seq` added markers `M11` and `M25`.

3. Now, get the order with all markers:

```
> LG1.final <- make.seq(LG1.ord, "force")
> LG1.final
```

Compared to the `rcd` order, this final map has markers `M12` and `M30` in different positions resulting in higher log-likelihood.

4. Check the final map:

```
> ripple(LG1.final)
```

5. Print it

```
> LG1.final
```

Map estimation for a given order

1. If, for any reason, one wants to estimate parameters for a given linkage map, it is possible to define a sequence and use the `map` function. For example, for markers `M30`, `M12`, `M3`, `M14` and `M2`, in this order, use

```
> any.seq <- make.seq(twopts, c(30, 12, 3, 14, 2))
> any.seq.map <- map(any.seq)
```

   This is a subset of the first linkage group. When used like this, `map` function searches for the best combination of phases between markers.

2. Furthermore, a sequence can also have user-defined linkage phases. The next example shows incorrect phases for the same order of markers:

```
> any.seq <- make.seq(twopts, c(30, 12, 3, 14, 2), phase = c(4,
+     1, 4, 3))
> any.seq.map <- map(any.seq)
```

3. If one needs to add or drop markers from a predefined sequence, use the functions `add.marker` and `drop.marker`. Adding markers 4 to 8 in `any.seq` can be done using

```
> any.seq <- add.marker(any.seq, 4:8)
> any.seq
```

   Removing markers 3, 4, 5, 12 and 30 from `any.seq`:

```
> any.seq <- drop.marker(any.seq, c(3, 4, 5, 12, 30))
> any.seq
```

Plotting the Recombination Fraction Matrix

1. For a given sequence, it is possible to plot the recombination fraction matrix based on a color scale using the function `rf.graph.table`. This matrix can be useful to make some diagnostics about the map. For example, using the function `group` with `LOD=2.5`:

```
> LGs <- group(mark.all, LOD = 2.5)
> LGs
```

   This value of `LOD` wrongly results in linkage groups with markers from distinct ones (`LG2` and `LG3`) in the same group. Ordering markers will provide

```
> LG.err <- make.seq(LGs, 2)
> LG.err.ord <- order.seq(LG.err)
> LG.err.ord
```

2. The map with option "force" is:

```
> LG.err.map <- make.seq(LG.err.ord, "force")
```

3. However, plotting the recombination fraction matrix will result in:

```
> rf.graph.table(LG.err.map)
```

   The color scale varies from red (small distances) to dark blue. This scale follows the "rainbow" color palette with `start` argument equal to 0 and `end` argument equal to 0.65. Clicking on the cell corresponding to two markers (off secondary diagonal), you can see some information about them. For example, clicking on the cell corresponding to markers `M4` and `M19` you can see their names, types (`A.4` and `B1.5`), recombination fraction (`rf=0.02281`) and LOD-Scores for each possible linkage phase.

   Looking at the whole matrix, it is possible to see a sub-division in two groups: one with markers from `LG2` (`M27 M16 M20 M4 M19 M21 M23 M9 M24 M29`) and other with markers from `LG3` (`M18 M8 M13 M7 M22`). Following the secondary diagonal of the matrix, there is a gap between markers `M18` and `M29` (`rf=0.4322`). At this position, the group should be divided, that is, a higher LOD-Score should be used. Notice that these two groups were placed together due to a false linkage (false positive) detected between markers `M4` and `M22`, which has LOD-Score 2.9.

   The `rf.graph.table` can also be used to check the order of markers based on the monotonicity of the matrix, i.e. as we get away from the secondary diagonal, the recombination fraction values should increase.

Final comments

At this point it should be clear that any potential *OneMap* user must have some knowledge about genetic mapping and also the R language, since the analysis is not done with *only one mouse click*. In the future, perhaps a graphical interface will be made available to make this software a lot easier to use. Any suggestions and critics are welcome.

Currently, there is not a graphical function included in *OneMap* to draw the map. But once the distances and the linkage phases are estimated, map figures could be easily done. Also, there are several free softwares that can be used, such as MapChart (Voorrips, 2002).

References

Broman, K. W., Wu, H., Churchill, G., Sen, S., Yandell, B. ***qtl: Tools for analyzing QTL experiments*** R package version 1.09-43, 2008.

Doerge, R.W. Constructing genetic maps by rapid chain delineation. ***Journal of Agricultural Genomics*** 2, 1996.

Garcia, A.A.F., Kido, E.A., Meza, A.N., Souza, H.M.B., Pinto, L.R., Pastina, M.M., Leite, C.S., Silva, J.A.G., Ulian, E.C., Figueira, A. and Souza, A.P. Development of an integrated genetic map of a sugarcane (*Saccharum spp.*) commercial cross, based on a maximum-likelihood approach for estimation of linkage and linkage phases. ***Theoretical and Applied Genetics*** 112, 298-314, 2006.

Haldane, J. B. S. The combination of linkage values and the calculation of distance between the loci of linked factors. ***Journal of Genetics*** 8, 299-309, 1919.

Jiang, C. and Zeng, Z.-B. Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. ***Genetica*** 101, 47-58, 1997.

Kosambi, D. D. The estimation of map distance from recombination values. ***Annuaire of Eugenetics*** 12, 172-175, 1944.

Lander, E. S. and Green, P. Construction of multilocus genetic linkage maps in humans. ***Proc. Natl. Acad. Sci. USA*** 84, 2363-2367, 1987.

Lander, E.S., Green, P., Abrahanson, J., Barlow, A., Daly, M.J., Lincoln, S.E. and Newburg, L. MAPMAKER, An interactive computing package for constructing primary genetic linkage maps of experimental and natural populations. ***Genomics*** 1, 174-181, 1987.

Lincoln, S. E., Daly, M. J. and Lander, E. S. Constructing genetic linkage maps with MAPMAKER/EXP Version 3.0: a tutorial and reference manual. ***A Whitehead Institute for Biomedical Research Technical Report*** 1993.

Margarido, G. R. A., Souza, A.P. and Garcia, A. A. F. OneMap: software for genetic mapping in outcrossing species. ***Hereditas*** 144, 78-79, 2007.

Oliveira, K.M., Pinto, L.R., Marconi, T.G., Margarido, G.R.A., Pastina, M.M., Teixeira, L.H.M., Figueira, A.M., Ulian, E.C., Garcia, A.A.F., Souza, A.P. Functional genetic linkage map on EST-markers for a sugarcane (*Saccharum* spp.) commercial cross. ***Molecular Breeding*** 20, 189-208, 2007.

Oliveira, E. J., Vieira, M. L. C., Garcia, A. A. F., Munhoz, C. F.,Margarido, G. R.A., Consoli, L., Matta, F. P., Moraes, M. C., Zucchi, M. I., and Fungaro,M. H. P. An Integrated Molecular Map of Yellow Passion Fruit Based on Simultaneous Maximum-likelihood Estimation of Linkage and Linkage Phases ***J. Amer. Soc. Hort. Sci.*** 133, 35-41, 2008.

Voorrips, R.E. MapChart: software for the graphical presentation of linkage maps and QTLs. ***Journal of Heredity*** 93, 77-78, 2002.

Wu, R., Ma, C.X., Painter, I. and Zeng, Z.B. Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. ***Theoretical Population Biology*** 61, 349-363, 2002a.

Wu, R., Ma, C.-X., Wu, S. S. and Zeng, Z.-B. Linkage mapping of sex-specific differences. ***Genetical Research*** 79, 85-96, 2002b.

Apendix

DEFUNCT - Checking the map with three-point analysis

For historical reasons, three-point analyses are maintained in *OneMap*, but the same (and a lot more) can be done using the multipoint approach.

1. The function `def.rf.3pts` is used as follows:

   ```
   > def.rf.3pts(example, "M18", "M8", "M13")
   ```

   The first argument is the object with the input data, of class `outcross`. Then, three ordered markers are specified.

   In this case, the assignments "`A11`", "`A12`", ..., have similar meanings to those of the two-point analysis: `1` means coupling/coupling, `2` is for coupling/repulsion, `3` is for repulsion/coupling and `4` is for repulsion/repulsion. The first number is the linkage phase between markers $\texttt{M}_\texttt{i}$ and $\texttt{M}_{\texttt{i+1}}$, while the second number is the linkage phase between markers $\texttt{M}_{\texttt{i+1}}$ and $\texttt{M}_{\texttt{i+2}}$.

2. Take a look at the default criteria used by this function: LOD = 5, maximum recombination fraction between adjacent markers = 0.35 and maximum recombination fraction between markers on the two ends = 0.55. Considering, for example, three markers A - B - C, in that order, the last criterion indicates the maximum recombination fraction acceptable between markers A and C. These values are used by the software to decide the most probable assignment and can be changed by the user:

   ```
   > def.rf.3pts(example, "M18", "M8", "M13", LOD = 10, max.rf = 0.4)
   > def.rf.3pts(example, "M18", "M8", "M13", max.rf = 0.4, max.nolink = 0.6)
   ```

   The arguments `max.rf` and `max.nolink` correspond to the maximum recombination fraction between adjacent markers and the maximum recombination fraction between markers on the two ends, respectively.

3. Do this step for all triplets of markers in linkage group 1:

```
> def.rf.3pts(example, "M18", "M8", "M13")
> def.rf.3pts(example, "M8", "M13", "M7")
> def.rf.3pts(example, "M13", "M7", "M22")
```

   This last command line shows that the order `M13 - M7 - M22` is possibly incorrect, and a warning message is displayed. However, the HMM-based analyses use information from every marker in the sequence and, therefore, the order obtained through `compare` is likely to be the best order. Anyway, we had noticed that changing the positions of markers M7 and M22 resulted in an order with LOD-Score -0.02, which is very close to zero. This probably happens because M7 is of type D2 and M22 is of type D1.

These three-point analyses were formerly used to check the final linkage map. In this new version, the best way to do this is using the new function `ripple`.