OneMap Tutorial
Software for constructing genetic maps in outcrossing species

Gabriel Rodrigues Alves Margarido[1]
Anete Pereira de Souza[2]
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

[2]Centro de Biologia Molecular e Engenharia Genética (CBMEG)
Universidade Estadual de Campinas (UNICAMP)
Cidade Universitária "Zeferino Vaz" - CP 6010
CEP: 13083-970 - Campinas São Paulo - Brazil

[*]corresponding author

http://www.ciagri.usp.br/˜aafgarci/OneMap/
June 6, 2008

Overview

OneMap is an environment for constructing linkage maps in outcrossing plant species, using full-sib families derived from two outbred (non-inbreeding) parent plants. 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. (2002) proposed a methodology to construct the 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 a sugarcane data set (Garcia et al., 2006). Actually, the analysis of this sugarcane data set motivated the implementation of OneMap. It is strongly recommended to check Wu et al. paper before using OneMap.

This first version handles the two-point analysis between markers, performs the grouping step and orders the markers in a linkage group using Rapid Chain Delineation - RCD (Doerge, 1996). Subsequently, the three-point test should be used to refine the map distance between adjacent markers.

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 approppriate files directly at the web site above 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, which are listed below:

Function type	Function name	Function description
Input	read.outcross	Read data from an outcross
Data manipulation	extract.group	Set a linkage group ready for mapping
	remove.marker	Remove a marker from a linkage group
	arbitr.rf.2pts	Arbitrarily define linkage phase between markers
Genetic mapping	est.rf.2pts	Estimate recombination fractions (two points)
	modify.rf.2pts	Change criteria for two-point analysis
	group	Assign markers to linkage groups
	map	Order markers on a linkage group using RCD
	make.map	Arbitrarily define a linkage map
	est.rf.3pts	Estimate recombination fractions (three points)

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 enough for any user to understand its functionality.

1. Start R by double-clicking its icon.
2. Load OneMap:

   ```
   > library(onemap)
   ```

3. To save your project anytime, type:

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

   or access the toolbar File → Save Workspace.

4. You can access the command lines of this tutorial by typing:

   ```
   > file.show("C:/.../OneMapTour.R")
   ```

   'OneMapTour.R' is a file that comes when you download OneMap.

Creating the data file

This step can be quite difficult, since the data file is not very simple and many errors can occur while reading it. The input file format is very 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 contains 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. (2002) notation. Its value 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 a lot of detail with respect to the marker type; therefore, we will not discuss this 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. (2002) and the possible values vary according to the marker type. Missing data corresponds to 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, to see how your file should be.

Importing data

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

   ```
   > example <- 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 `setwd()` or the toolbar File → Change dir. If you set your working directory to the one containing the input file, you can just type:

   ```
   > example <- 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 the data 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 and, thus, if you type:

   ```
   > example
   ```

   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 the two-point test:

   ```
   > twopt <- est.rf.2pts(example)
   ```

   This command uses default values for LOD score (3) and maximum recombination fraction (0.35).

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

   ```
   > twopt <- est.rf.2pts(example, LOD = 4, max.rf = 0.4)
   ```

3. Although the two-point test was implemented in C language, which is much faster, this step can take quite some time, depending on the number of markers involved and their segregation type. Besides, the results use a lot of memory and a rather powerful computer is needed. 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:

   ```
   > twopt
   ```

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

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

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

   Note: the column `Posterior` in the output shows the posterior probability of each assignment, according to equation (1) in Wu et al. (2002).

6. You may possibly want to change the criteria used in the two-point analysis. This step is faster than the previous one, since it uses the results generated by `est.rf.2pts` and just changes the thresholds, but can also take a while. For example, the command:

   ```
   > twopt <- modify.rf.2pts(twopt, LOD = 5)
   ```

   updates the object `twopt`, changing the LOD threshold from 4 to 5. The object is still of class `rf.2pts`, thus the `print` command works in the same way.

7. Checking the results for some markers, like `M1` and `M3` above, you should notice that OneMap chooses the most probable linkage phase between two markers, based on the defined criteria. In general, this will be enough to decide the best assignment for every pair of markers. However, if you want to arbitrarily define the linkage phase for a given pair of markers, this can be done with the function `arbitr.rf.2pts`, like follows:

   ```
   > twopt <- arbitr.rf.2pts(twopt, "M1", "M3", "A1")
   ```

   You can choose one of the four assignments: "`A1`" (coupling/coupling), "`A2`" (coupling/repulsion), "`A3`" (repulsion/coupling) and "`A4`" (repulsion/repulsion). Additionally, the option "`ns`" indicates that the two markers are not linked.

8. Again, type:

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

   to see what `arbitr.rf.2pts` did.

9. If you want to set the best assignment back to the original one, the function `arbitr.rf.2pts` can be used again, the same way:

   ```
   > twopt <- arbitr.rf.2pts(twopt, "M1", "M3", "ns")
   ```

10. Moreover, you may want to let OneMap once again decide the most probable linkage phases, using:

    ```
    > twopt <- modify.rf.2pts(twopt, LOD = 3)
    ```

    Note: this will change the best assignments for ALL pairs of markers!

Assigning markers to linkage groups

1. Once the recombination fractions and linkage phases for all pairs of markers have been estimated, they can be assigned to linkage groups. This step is very simple and can be done by using the function `group`:

```
> LGs <- group(twopt)
```

No other options are allowed, since this function uses the criteria previously defined for the object `twopt`. If you want to change the criteria used to group markers, then the function `modify.rf.2pts` must be used first.

2. 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 group.

3. 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 = F)
```

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:

```
> twopt <- modify.rf.2pts(twopt, LOD = 6)
> LGs <- group(twopt)
> LGs
```

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

```
> twopt <- modify.rf.2pts(twopt, LOD = 3, max.rf = 0.4)
> LGs <- group(twopt)
> LGs
```

Genetic mapping

1. When the marker assignment to linkage groups is finished, the mapping step can take place. First of all, 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. This can be done using the following code, starting with the smaller group, which is group 3 in this example:

```
> LG3 <- extract.group(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 `extract.group`, is of class `extracted.group`.

2. Next, to order the markers in linkage group 3 (`LG3`), the function `map` must be used:

```
> map(LG3, twopt)
```

   The first argument is the extracted group `LG3` and the second is the object of class `rf.2pts` that was used to generate the linkage group (in this case, `twopt`). If the objects do not match, an error message will be displayed. The object generated by function `map` is of class `map`.

3. The ordering algorithm used by OneMap is *Rapid Chain Delineation* (Doerge, 1996). In some cases, it makes use of random steps, thus the map may change if the same linkage group is ordered more than once. These random steps may be necessary because recombination fractions between various pairs of markers can be the same, resulting in unsolvable ties. Therefore, it is recommended to use the `map` function a number of times, checking if the order obtained is consistent:

```
> map(LG3, twopt)
> map(LG3, twopt)
> map(LG3, twopt)
> map(LG3, twopt)
```

4. In this case, it is noticed that the position of marker `M7` is not certain. Thus, one may want to remove it from the linkage group. This can be done typing:

```
> LG3 <- remove.marker(LG3, "M7")
```

5. Try to map this linkage group again:

```
> map(LG3, twopt)
> map(LG3, twopt)
> map(LG3, twopt)
> map(LG3, twopt)
```

   The order is now consistent and does not change anymore.

6. Now, look at the recombination fractions between marker `M7` and the others in that linkage group:

```
> print(twopt, "M7", "M18")
> print(twopt, "M7", "M8")
> print(twopt, "M7", "M13")
> print(twopt, "M7", "M22")
```

Notice that marker `M7` is closer to marker `M13` than to `M8` and `M18`. Thus, the order could be `M18 - M8 - M13 - M7 - M22` or `M18 - M8 - M13 - M22 - M7`.

7. To check which one is the best option, you can arbitrarily define an order to the linkage group. Do that with the commands:

```
> make.map(twopt, c("M18", "M8", "M13", "M7", "M22"))
> make.map(twopt, c("M18", "M8", "M13", "M22", "M7"))
```

The first argument is again the object of class `rf.2pts`, which contains the information about recombination fractions. The second argument is a vector of class `character`, defining the order of markers in the linkage group.

8. Recall that marker `M22` is of type D1 and `M7` is of type D2, so the recombinants between them cannot be identified; thus, the distance between them cannot be estimated by the two-point analysis and is shown as infinite (`Inf`).

9. One should also notice that marker `M13` is closer to `M22` than to `M7` (9.89 cM against 19.55 cM, respectively, in Kosambi map distance). This suggests that the best order is `M18 - M8 - M13 - M22 - M7`.

Refining the map with the three-point analysis

Finally, in order to obtain better estimates of the distances between adjacent markers, including the non-informative pairings (D1 x D2), the three-point analysis can be used. Currently, this step is quite manual and laborious, since there is not an implemented function that automatically does these analyses for an entire linkage group. In the future, this will be improved.

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

```
> est.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. This step can take quite some time for calculations, depending on the computer being used.

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 M_i and M_{i+1}, while the second number is the linkage phase between markers M_{i+1} and M_{i+2}.

2. Notice that the distances between markers `M18 - M8` and `M8 - M13` have changed a little bit when compared with 2-point estimates. Also 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:

```
> est.rf.3pts(example, "M18", "M8", "M13", LOD = 10, max.rf = 0.4)
> est.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 the linkage group:

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

Notice that we are now refining the order that we got using RCD, but based on the three-point likelihood, which is a superior criterion. For details, see Wu et al. (2002).

This last command line shows that, based on the likelihoods, the most likely order is now `M13 - M22 - M7`, thus the best order for this linkage group is really `M18 - M8 - M13 - M22 - M7`.

4. Note that if the order is defined incorrectly, OneMap will display a "warning" message. For example, type:

```
> est.rf.3pts(example, "M13", "M7", "M22")
```

5. You can also try carrying out these analyses with several different criteria.

These three-point analyses provide two different estimates of the recombination fraction for the same interval, except for the ones at the two ends of the linkage group. Wu et al. (2002) suggest combining these estimates using a weighted mean, with the weights being the reciprocals of the variances of the two separate estimates. This is currently being implemented and will be made available soon.

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 <- extract.group(LGs, 2)
   ```

2. Use the `map` function various times to map the markers in that linkage group:

   ```
   > map(LG2, twopt)
   > map(LG2, twopt)
   > map(LG2, twopt)
   > map(LG2, twopt)
   ```

   Notice the following consistent partial sub-order: `M16 - M20 - M4 - M19 - M23 - M21`. Markers `M24` and `M29` appear to be linked to each other, but not to the rest of the group. Marker `M9` sometimes appears linked to `M24` and sometimes it does not. Marker `M27` seems to be floating and apparently cannot be mapped.

3. What should be done next? Type

   ```
   > mrktype(twopt, "M21")
   > mrktype(twopt, "M16")
   ```

   to see that `M21` and `M16` (the two markers on the edges of the group) are of segregation type D2 (the first argument is an object of class `rf.2pts` and the second is a marker name). Then, type

   ```
   > mrktype(twopt, "M9")
   > mrktype(twopt, "M29")
   > mrktype(twopt, "M27")
   > mrktype(twopt, "M24")
   ```

   to see that these markers that are difficult to order are of segregation type D1 (with the exception of `M24`). Thus, the non-informative analysis between markers of type D1 and D2 could be leading to this problem in ordering.

4. So, it should be a good idea to remove one of the markers on the edge. Use

   ```
   > LG2 <- remove.marker(LG2, "M21")
   ```

   and again try to map this linkage group:

   ```
   > map(LG2, twopt)
   > map(LG2, twopt)
   > map(LG2, twopt)
   > map(LG2, twopt)
   ```

   Notice now that the consistent sub-order is larger than the previous one: `M16 - M20 - M4 - M19 - M23 - M9 - M24 - M29`. Marker `M27` still could not be mapped.

5. Now, check the results of the two-point analysis between marker `M27` and the others in linkage group 2:

   ```
   > print(twopt, "M27", "M16")
   > print(twopt, "M27", "M20")
   > print(twopt, "M27", "M4")
   > print(twopt, "M27", "M19")
   > print(twopt, "M27", "M23")
   > print(twopt, "M27", "M9")
   > print(twopt, "M27", "M24")
   > print(twopt, "M27", "M29")
   ```

   Marker `M27` is significantly linked only to markers `M20` and `M4`, and therefore probably located somewhere in this region. Similarly to what has occurred in linkage group 3, marker `M27` is of segregation type D1, while `M16` is of type D2; therefore, this pairwise analyses is not informative and this makes it difficult to infer the correct position of marker `M27`.

6. Type:

   ```
   > print(twopt, "M27", "M20")
   > print(twopt, "M27", "M4")
   > print(twopt, "M4", "M20")
   ```

   to notice that `M27` is closer to `M20` than to `M4` (21.18 cM against 34.66 cM, respectively, in Kosambi map distance), while `M4` is closer to `M20` than to `M27` (12.77 cM against 34.66 cM). This suggests that `M20` is located between `M27` and `M4`. Thus, the correct global order may be `M16 - M27 - M20 - M4 - M19 -` . . . or `M27 - M16 - M20 - M4 - M19 -` . . .

7. Next, type:

   ```
   > print(twopt, "M20", "M16")
   > print(twopt, "M20", "M27")
   ```

   and you will see that `M20` is closer to `M16` than to `M27`, so that the correct order is probably: `M27 - M16 - M20 - M4 - M19 - M23 - M9 - M24 - M29`.

8. Marker `M21` still needs to be mapped. Check the results of the two-point analysis between marker `M21` and all the others:

   ```
   > print(twopt, "M21", "M27")
   > print(twopt, "M21", "M16")
   > print(twopt, "M21", "M20")
   > print(twopt, "M21", "M4")
   > print(twopt, "M21", "M19")
   > print(twopt, "M21", "M23")
   > print(twopt, "M21", "M9")
   > print(twopt, "M21", "M24")
   > print(twopt, "M21", "M29")
   ```

   Notice that `M21` is closer to `M19` and `M23`.

9. Use the three-point analysis for this triplet:

   ```
   > est.rf.3pts(example, "M19", "M23", "M21")
   > est.rf.3pts(example, "M19", "M21", "M23")
   > est.rf.3pts(example, "M21", "M19", "M23")
   ```

   It seems that `M21` is located to the left of `M19`. However, if you type:

   ```
   > est.rf.3pts(example, "M4", "M21", "M19")
   ```

   you will see that the above result is inconsistent with the global best order.

   This happens because `M19` and `M23` are linked on repulsion phase and `M23` is of segregation type C.8 (it segregates 3:1). Thus, the recombination fraction between these two markers may be greatly underestimated, which would force them to stay together on the three-point analysis.

10. Finally, it must be noticed that if you add marker `M21` back to the linkage group, it will result in a lot of ordering problems. Therefore, perhaps the most coherent idea would be to keep `M21` as an accessory marker.

11. When you run the three-point analysis for all triplets of markers in the linkage group, a problem appears: one of the sub-orders may be wrong, since it does not meet the criteria under any assignment:

```
> est.rf.3pts(example, "M27", "M16", "M20")
> est.rf.3pts(example, "M16", "M20", "M4")
> est.rf.3pts(example, "M20", "M4", "M19")
> est.rf.3pts(example, "M4", "M19", "M23")
> est.rf.3pts(example, "M19", "M23", "M9")
> est.rf.3pts(example, "M23", "M9", "M24")
> est.rf.3pts(example, "M9", "M24", "M29")
```

All analyses show that the order is correct, except for the first one, between makers `M27`, `M16` and `M20`.

12. Check it again, very carefully:

```
> est.rf.3pts(example, "M27", "M16", "M20")
```

Notice that, under assignment `A11`, `theta12` (recombination fraction between `M27` and `M16`) and `theta13` (recombination fraction between `M27` and `M20`) are almost the same. This seems to be common when markers of segregation types D1 and D2 are included in a three-point analysis. Thus, the user has two options: keep the above order, which is coherent with the two-point analysis, or remove marker `M27` from the linkage group and make it an accessory marker. Either one seems feasible.

13. You can also check the results of the three-point analysis and notice that the recombination fraction between `M19` and `M23` changes greatly when comparing the two triplets where this couple appears. This is the usual case when a marker of type C.8 (3:1) is linked in repulsion phase.

14. The results of this analysis showed that the problem of ordering markers in linkage groups for outcrossing species is a complicated one and more efficient algorithms are required.

Genetic mapping of linkage group 1

1. At last, linkage group 1 (the large one) can be mapped. Extract it and try to map it some times:

```
> LG1 <- extract.group(LGs, 1)
> map(LG1, twopt)
> map(LG1, twopt)
> map(LG1, twopt)
> map(LG1, twopt)
```

Although this is the largest linkage group, and therefore supposed to lead to more ordering problems, there is a very consistent order: `M12 - M30 - M3 - M14 - M2 - M1 - M10 - M17 - M28 - M26 - M6 - M15 - M5 - M25 - M11`. This order does not change running RCD several times, despite some large gaps between markers `M28 - M26` and `M6 - M15`.

2. The function `map` also makes use of thresholds for minimum LOD score and maximum recombination fraction: estimates of recombination fraction for pairs of markers that do not meet these criteria are replaced by the value 0.5 (no linkage). If many cases like this occur in a single linkage group, ordering can be a problem, because of the random steps made by RCD when there are ties.

   By default, the thresholds found in the object of class `rf.2pts` (second argument) are used. To change them, simply type

   ```
   > map(LG1, twopt, LOD = 1.5, max.rf = 0.5)
   ```

   Lower values for `LOD` and higher values for `max.rf` may be desirable, because this should reduce the number of 0.5 values. Besides, after the assignment of markers to linkage groups has been done, all markers in the same group are assumed to be linked.

   On the other hand, higher values for `LOD` and lower values for `max.rf` can lead to many false-negatives. For example, try the command

   ```
   > map(LG1, twopt, LOD = 5, max.rf = 0.3)
   ```

   It is important to note that the `rf.2pts` object is not changed.

3. To finish the analysis, use the three-point function to refine the map:

   ```
   > est.rf.3pts(example, "M12", "M30", "M3")
   > est.rf.3pts(example, "M30", "M3", "M14")
   > est.rf.3pts(example, "M3", "M14", "M2")
   > est.rf.3pts(example, "M14", "M2", "M1")
   > est.rf.3pts(example, "M2", "M1", "M10")
   > est.rf.3pts(example, "M1", "M10", "M17")
   > est.rf.3pts(example, "M10", "M17", "M28")
   > est.rf.3pts(example, "M17", "M28", "M26")
   > est.rf.3pts(example, "M28", "M26", "M6")
   ```

```
> est.rf.3pts(example, "M26", "M6", "M15")
> est.rf.3pts(example, "M6", "M15", "M5")
> est.rf.3pts(example, "M15", "M5", "M25")
> est.rf.3pts(example, "M5", "M25", "M11")
```

The order is apparently correct and no further analyses are required.

4. Finally, sometimes it may be interesting to present the map with cumulative distances. For instance, some softwares used to draw chromosomes require this kind of presentation. This can be done like follows:

```
> final_LG1 <- map(LG1, twopt)
> print(final_LG1, cumulative = TRUE)
```

Final comments

At this point it should be clear that any potential OneMap user must have some knowledge about genetic mapping, the methods presented by Wu et al. (2002) and also the R language, since the analysis is not done with *only one mouse click*. In the future, we expect to make this software a lot easier to use, especially for the three-point analysis. Any suggestions and critics are welcome.

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

References

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.

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.

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, 2002.