pheno2geno

Konrad Zych [1,2], Danny Arends [2,], Ritsert C. Jansen [2]

July 17, 2011

1. Faculty of Biochemistry, Biophysics and Biotechnology, Jagiellonian University in Krakow, Poland
2. Groningen Bioinformatics Center, University of Groningen, The Netherlands

1 General introduction

Genetical Genomics (Jansen and Nap, 2001) is powerful method, providing world of life sciences with tool to look deep inside the complex relation between genetic information stored by DNA and final outcome of its processing - phenotype. And because this is the core dogma of modern biology, every method helping us to understand it better is of high importance to scientific world.
Great power often comes for high price, though. To perform GG studies one has to obtain both phenotypic and genotypic data, that are subsequently being matched. This not only elevates costs of experiment, but also introduces number of human errors, e.g. mismatching/mislabeling of arrays.
This brought us back to DNA to phenotype dogma. And we came up with idea of creating genetic map out of gene expression data. This means the same power for less then half of the cost and effort. Procedure is easy enough to be conducted by inexperienced R user and for advanced users we offer variety of extra functionalities to make their analysis fit their needs.

1.1 R programming language

R programming language is powerful, yet easy-to-use. There is graphical interface available for Windows, Mac OS and Linux. Every package/function comes with easily accessible help file and, most importantly, R provides user with handfuls of statistical functionalities. To start your adventure with R just go to: http://cran.r-project.org/, select your operating system, install it, and you're ready to enter the world of R.

1.2 Downloading and installing package

R packages contribute to power of this language more than anything. Using then, you can extend basic R to powerful tool perfectly suited to your personal needs. Installing package is really easy, just open R gui and type: > install me > or what? or use Packages menu (just click on Install Packages, select a mirror and than select package pheno2geno).

2 Data

2.1 Data files' structure

In order to ensure smooth work of our package, we specified strict rules about data files provided. If below mentioned requirements are met and filenames are default, user don't have to provide reading function with any parameters. If your more advanced R user, your data files are of different format or you have data already inside R environment, please see section 4.1.

2.1.1 Phenotype data for offspring

This data is crucial and analysis cannot be run without it. Default filename is offspring_phenotypes.txt. Rows correspond to markers and columns to individuals. File should contain only numeric values apart from first row, containing unique column names and first column containing unique row names. Rows and columns with non-numeric values are removed. All values should be separated by tabs. In normal practice first row has one element less then others (no column name for column containing rownames).

2.1.2 Phenotype data for founders

This data is optional but really important for the analysis, so we recommend to provide it. Default filename is founders_phenotypes.txt. Data structure inside the file should be the same as in offspring phenotype file. Also row names should match, because non-matching rows are removed.

2.1.3 Genotype data for offspring

This data is optional and doesn't have any impact on analysis, could be only used for creating cross object out of existing data using our software. Default filename is offspring_genotypes.txt. Genotypes should be coded as 0,1 and NA. Data structure inside the file should be the same as in offspring and founders phenotype files. Row names should match, because non-matching rows are removed.

2.1.4 Genetic map

This data is optional and could be used for comparison with map created by our software or for ordering markers in output cross. Default filename is maps_genetic.txt. File should contain three columns - marker name, chromosome number and position on chromosome in cM. Second and third column should contain numeric values without NAs, NaNs, etc. First column should contain unique marker names, matching ones in phenotype files.

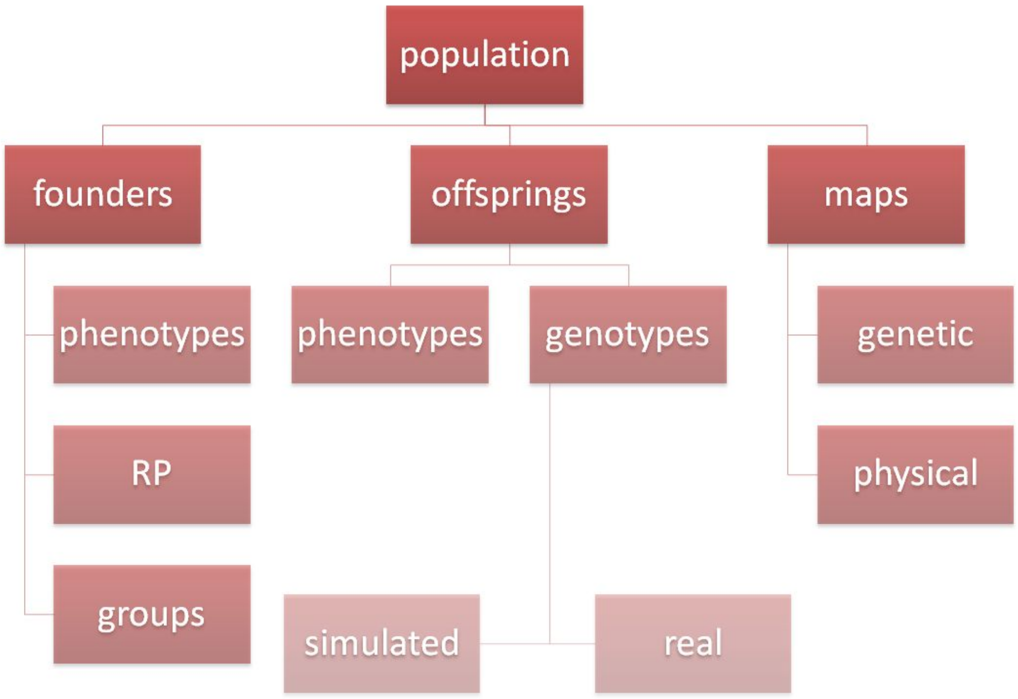
2.1.5 Physical map

Exactly the same as in genetic map, but position on chromosome should be in Mbp.

2.2 Population object

To store all the necessary information and mainupulate it easily we developed object of class population (schematically shown on figure 1). It's divided in three main parts:

- founders part, storing founders phenotype data, information about founders groups used by RankProduct analysis (see 3.2) and results of this analysis (RP object, futher documentation about it is available in help file of RP function),
- offspring part, storing founders phenotype and genotype data (read from file and/or simulated by our software),
- maps part, storing genetic and/or physical map


population
founders
offsprings
maps
phenotypes
RP
groups
phenotypes
genotypes
simulated
real
genetic
physical

3 Basic workflow

3.1 Loading data into workflow

3.1.1 readFiles function

If data is formatted as specified in chapter 2 and files are having default names, you just have to type:

```
> population <- readFiles(founders_groups = c(0, 0, 1, 1))
```

into Rgui (make sure your working directory is the one containing data files) and it's all done. Parameter founders_groups specifies in which column of parental file which parent is. Eg. you measured gene expression of both parents twice. Results for one of those are in first two columns, for the other - in following two. Then your function call will look exactly like one given above. If you don't like words founders, offspring and map and would like to use more friendly parents, children and treasure_map instead, just type:

```
> population <- readFiles(offspring = "children", founders = "parents",
+     map = "treasure_map", founders_groups = c(0, 0, 1, 1))
```

Function will then look for eg genetic map in file treasure_map_genetic.txt.

3.1.2 createPopulation function

Loading data from files is not always the best idea. Imagine you loaded your data into R already, for example to normalize it. Saving it back to disk and reading again into R means wasting your time. In this case you can use createPopulation function:

```
> population <- createPopulation(offspring_phenotypes, founders,
+     founders_groups, offspring_genotypes, maps_genetic, maps_physical)
```

just listing objects that contain phenotype, genotype data and maps. For example if you have phenotype matrix called phenotypes for both founders(columns 1-4) and offspring(5-100) you can load that into population object using:

```
> population <- createPopulation(phenotypes[, 1:4], phenotypes[,
+     5:100], c(0, 0, 1, 1))
```

3.1.3 faking population object

Sometimes you nede pretty simple and small dataset to do samoe testing on. Or to be able to play with, to learn how to use the package. Like in this manual. In this step fakePopulation function comes handy:

```
> population <- fakePopulation()
```

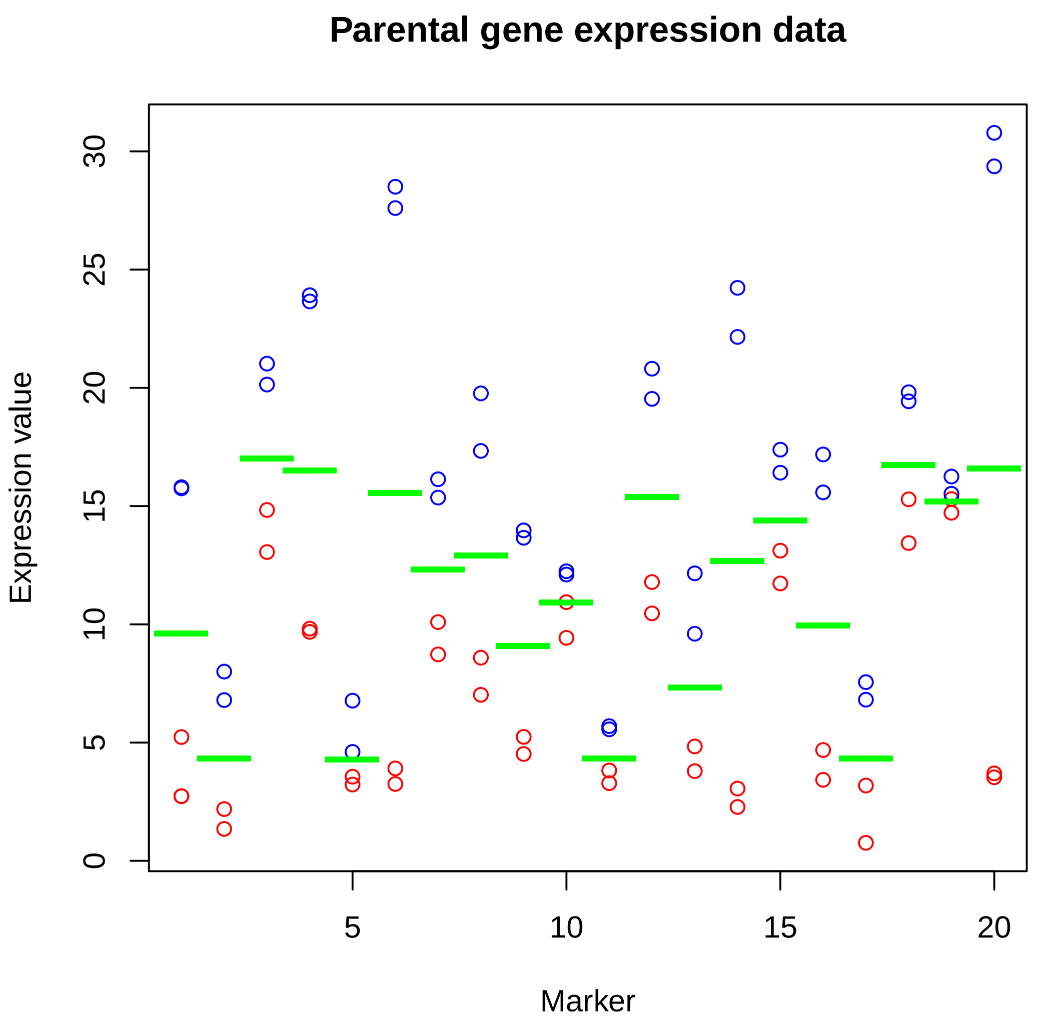
There is a lot of parameters you can provide to this function to get more sophisticated datasets, but for our use default one (founders measured twice, 250 offspring individuals, 1000 markers spread evenly over 10 chromosomes, RIL population obtained using selfing) will be perfect.

3.2 Inspecting data

3.2.1 Founders data

We provide you we function to plot founders data for brief inspection. Let's inspect first 20 markers of just faked population object:

```
> plotParentalExpression(population, 1:20)
```

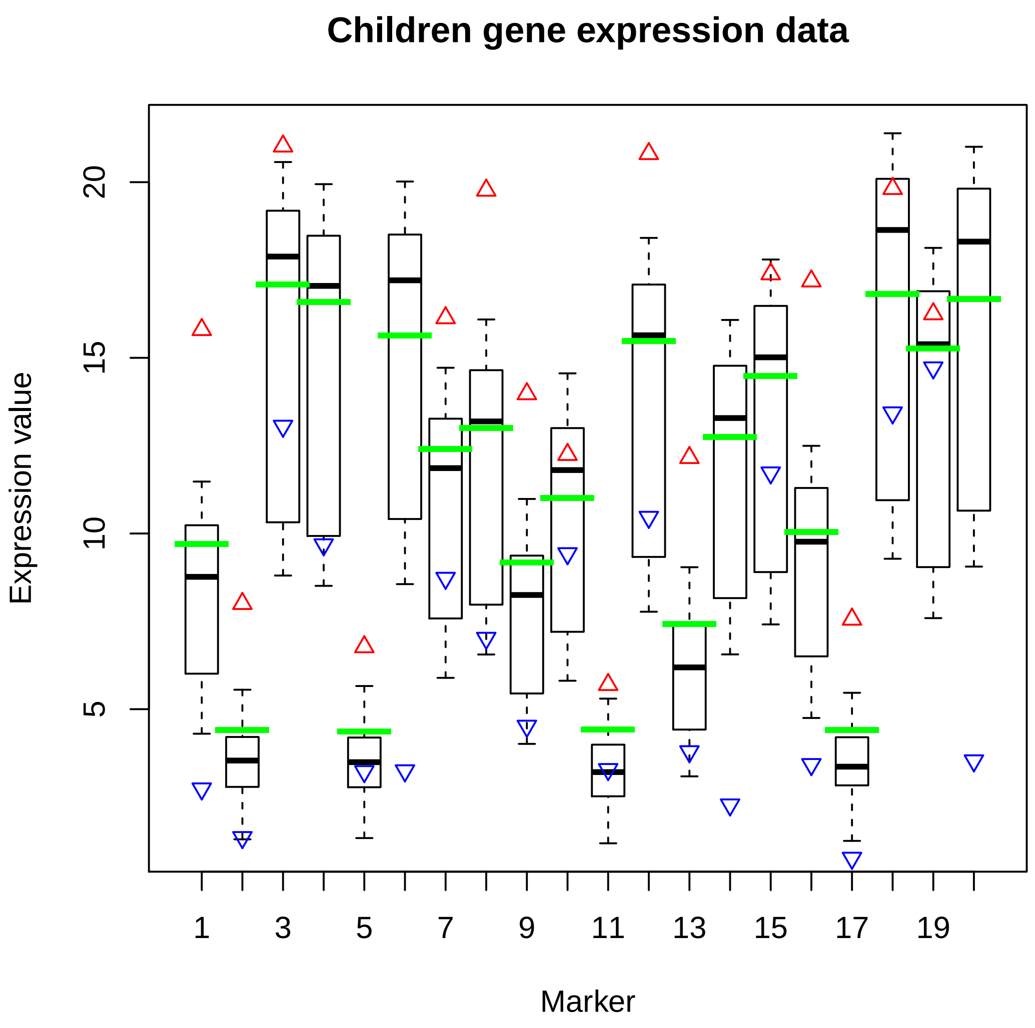


Empty circles (blue for parent "0" and red for parent "1") on the plot show gene expression values, green lines - mean values for each marker.

3.2.2 Offspring data

We can also inspect gene expression values for offspring. To show plot for first 20 markers:

```
> plotChildrenExpression(population, 1:20)
```

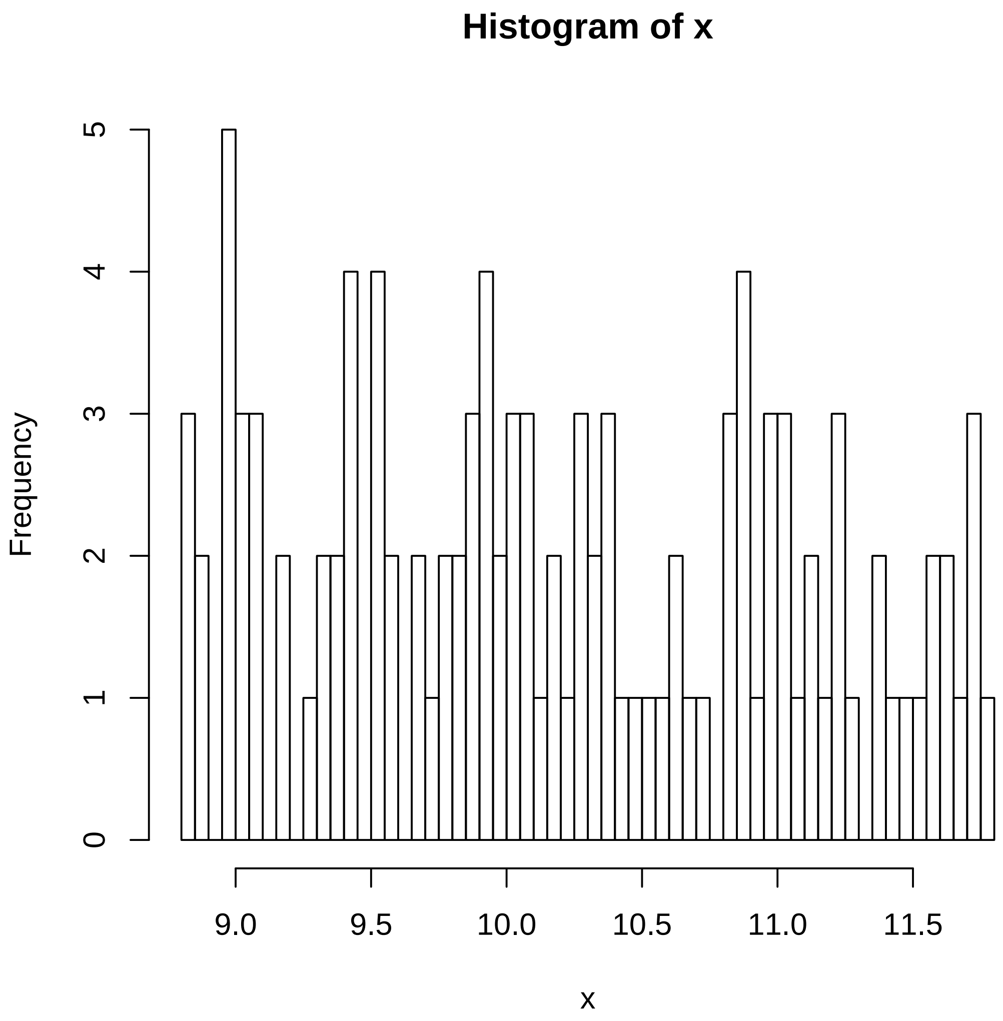


Boxplot (in black) shows gene expression values for offspring. On those, founders data is overlayed, depicted as in plotParentalExpression.

3.2.3 Single markers distribution

This kind of plot gives possibility to make sure that our data is distributed as we supposed. In our case, RIL with selfing, we expect in all markers mixture of two normal distributions. In our example we will look at values for third marker:

```
> plotMarkerDistribution(population, 3, 2)
```



First parameter is object of class population, then number or name of marker to be shown and number of normal distributions to be fitted (more than 1).

3.3 Rank product analysis

RankProd (Hong et al., 2006) is a R package using Rank Product (Breitling et al., 2004) method to detect genes showing differential expression, especially in microarray experiments. As so, it fits our needs perfectly and that's why it's incorporated into our workflow. In majority of cases object of class population is the only parameter while calling findDiffExpressed function. RP, main function of RankProd, is always verbose, so exptect something like:

```
> population <- findDiffExpressed(population)
```

```
Rank Product analysis for two-class case

Starting 100 permutations...
Computing pfp ..
Outputing the results ..
```

If you're more experienced user and would like to provide RP function with additional parameters, it's possible. Just provide preprocessData funtion with them and they will be send directly to RP function. For further information see help file of RP.

3.4 toGenotypes function - introduction

After RankProduct analysis is done, it's time for magic to happen, very essence of this package - creating genetic map out of gene expression data. Because this function call is much more complicated than any other in the package, we summarized parameters in table below:

```
> cross <- toGenotypes(population, genotype = c("simulated", "real"),
+     orderUsing = c("none", "map_genetic", "map_physical"), treshold = 0.01,
+     overlapInd = 0, proportion = c(50, 50), margin = 15, verbose = FALSE,
+     debugMode = 0)
```

parameter name	description	possible values	description
population	object of class population		
genotype	which genotype data should be saved in cross object	simulated	simulated by toGenotypes
		real	provided by user
orderUsing	which map should be used to order markers in output cross object	none	markers are placed on one chromosome with 1 cM distance in between
		map_genetic	genetic map
		map_physical	physical map
treshold	treshold p-value to classify gene as differentially expressed	0-1	
overlapInd	how many individuals may overlap between distributions	0-nr of individuals	
proportion	specifying how many normal distributions are expected and percentage of individuals in every group	vector of 0-100 summing up to 100	
margin	proportion is allowed to varry between this margin (2 sided)	0-min(proportion)	

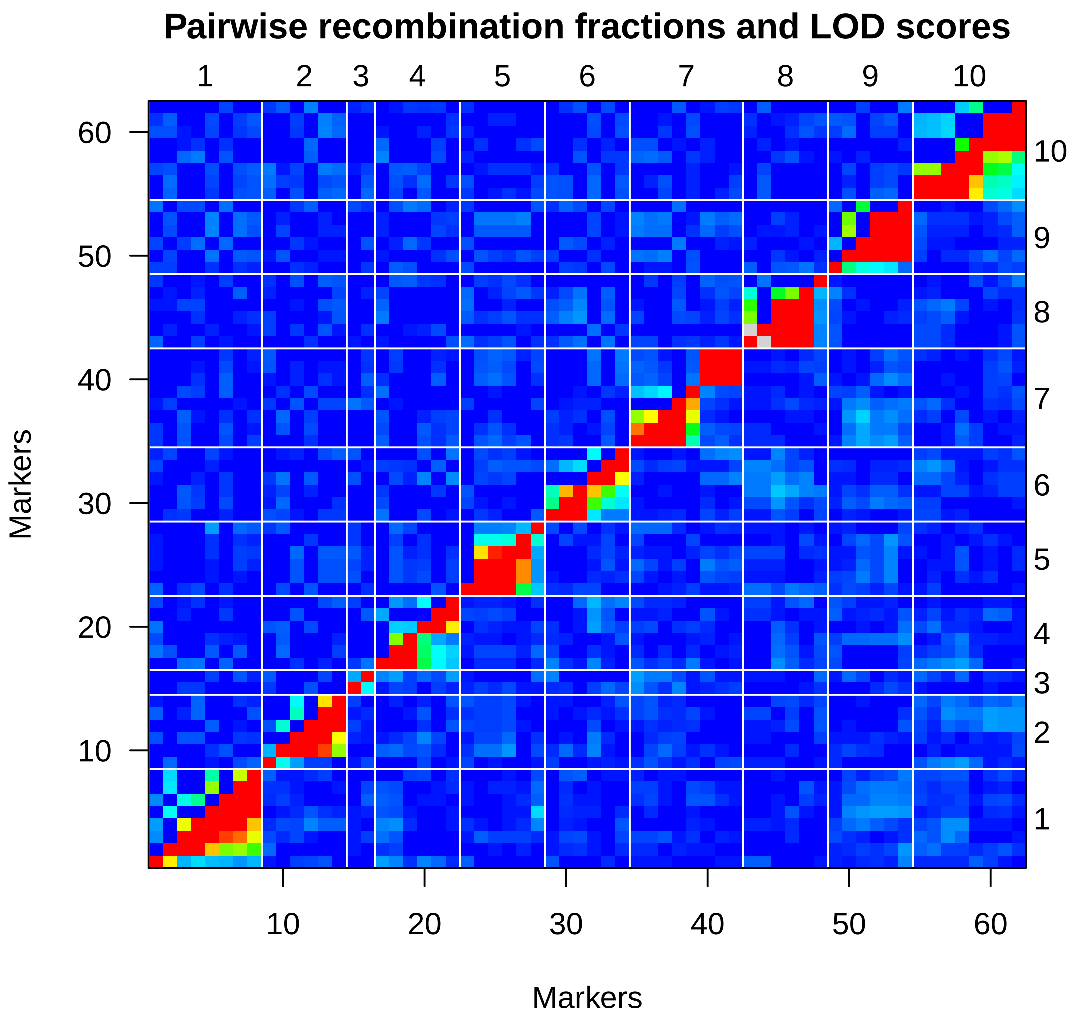
3.5 Example function call for RILs

In data from RIL population (like one we faked in section 3.1.3), we expect two normal distributions in gene expression data. Each group should contain 50 percent of individuals. Proportion parameter should then have value c(50,50). This is default one. We would like to save genotype data created by toGenotypes and order it using genetic map. For all the other parameters we use default values. Function call then should look like:

```
> cross <- toGenotypes(population, genotype = "simulated", orderUsing = "map_genetic",
+     treshold = 0.01, overlapInd = 0, proportion = c(50, 50),
+     margin = 15, verbose = FALSE, debugMode = 0)
```

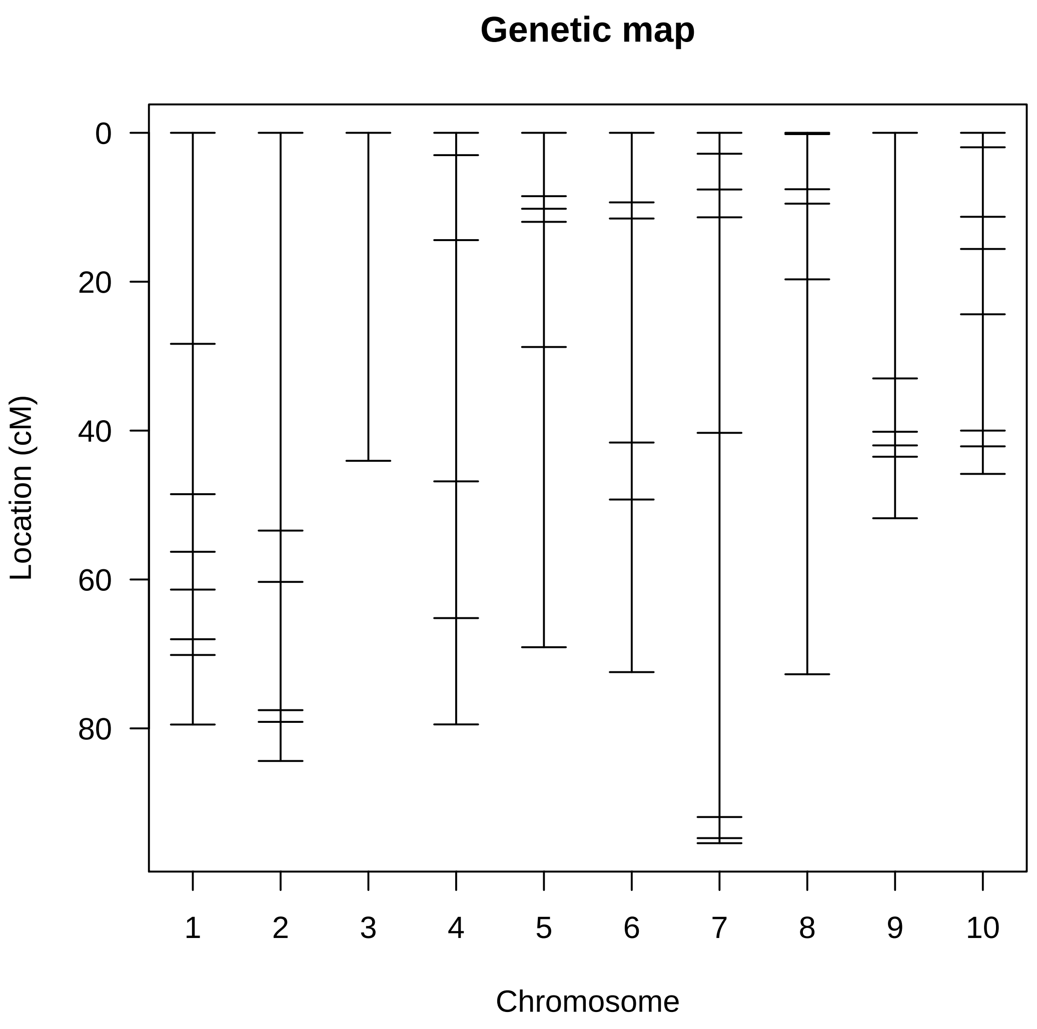
After function is executed, our data is stored inside (a bit enhanced) cross object. We can use R/qtl functions to plot it. Most informative plots will be:
- plot.rf - showing recombination factors between markers

```
> plot.rf(cross)
```



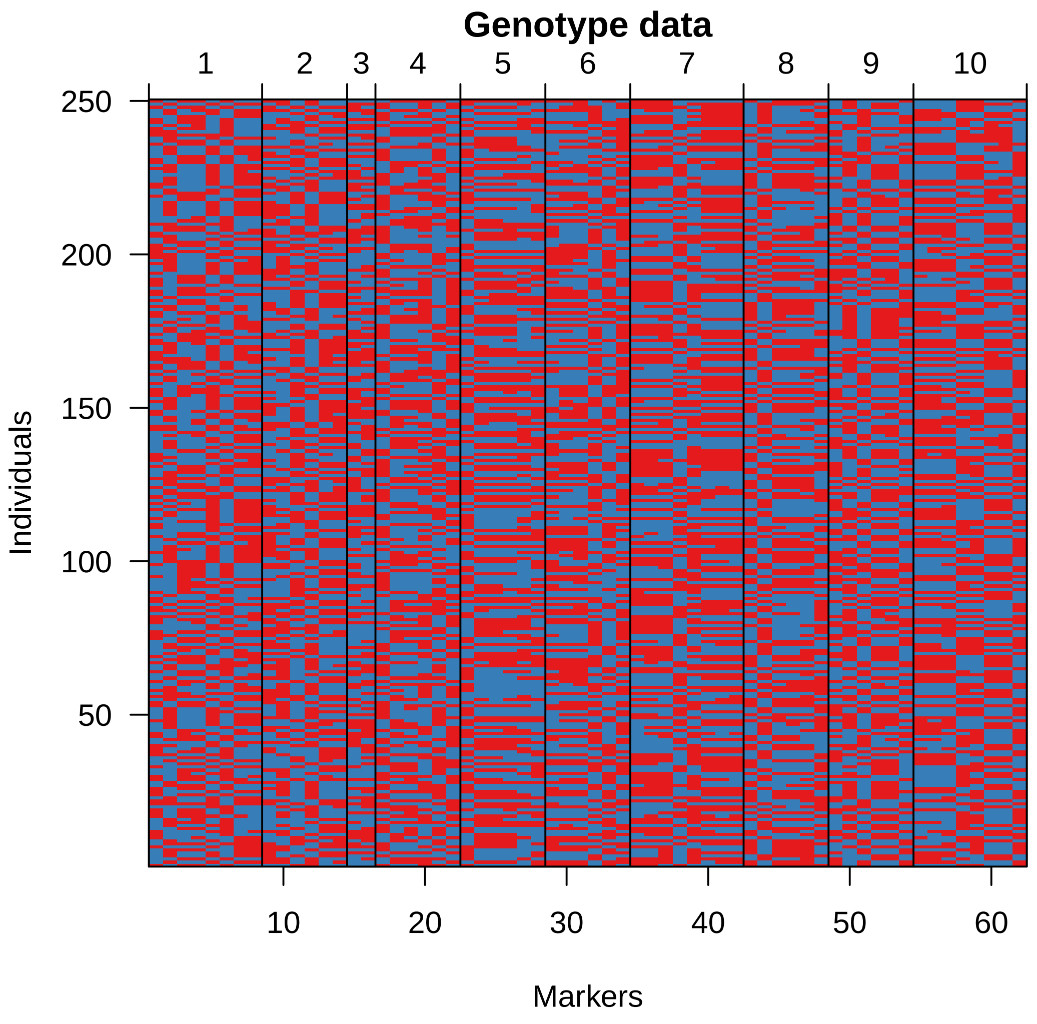
- plot.map - showing how markers are spread on the chromosomes

```
> plot.map(cross)
```



- geno.image showing our newly created genotypes

```
> geno.image(cross)
```



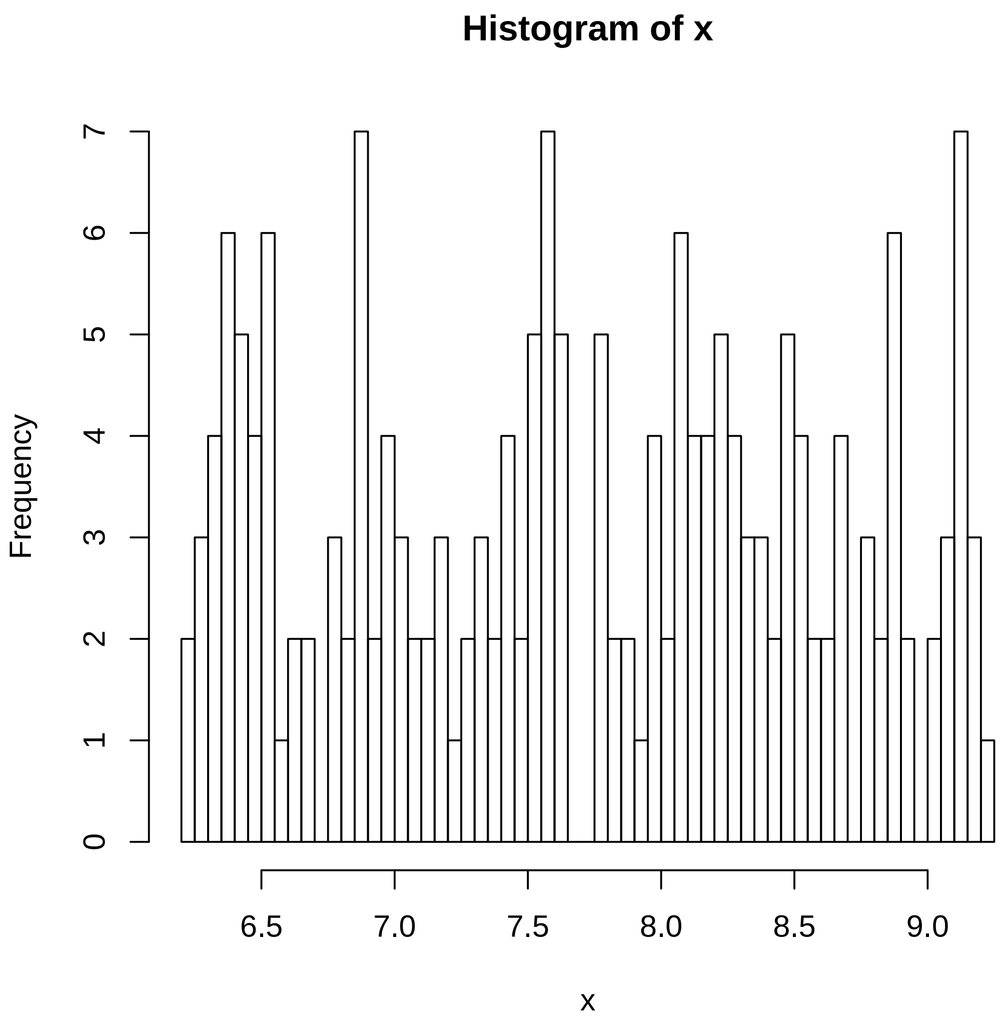
3.6 Example workflow for BC

plot.rf(cross, main="bc toGenotypes example")
We start with faking population object with data for Back Cross:

```
> population <- fakePopulation(type = "bc")
```

The only, but very important difference lays in distribution of the data. We can inspect it using plotMarkerDistribution:

```
> plotMarkerDistribution(population, 3, 2)
```



Next step, Rank Product analysis, is exactly the same as for RILs:

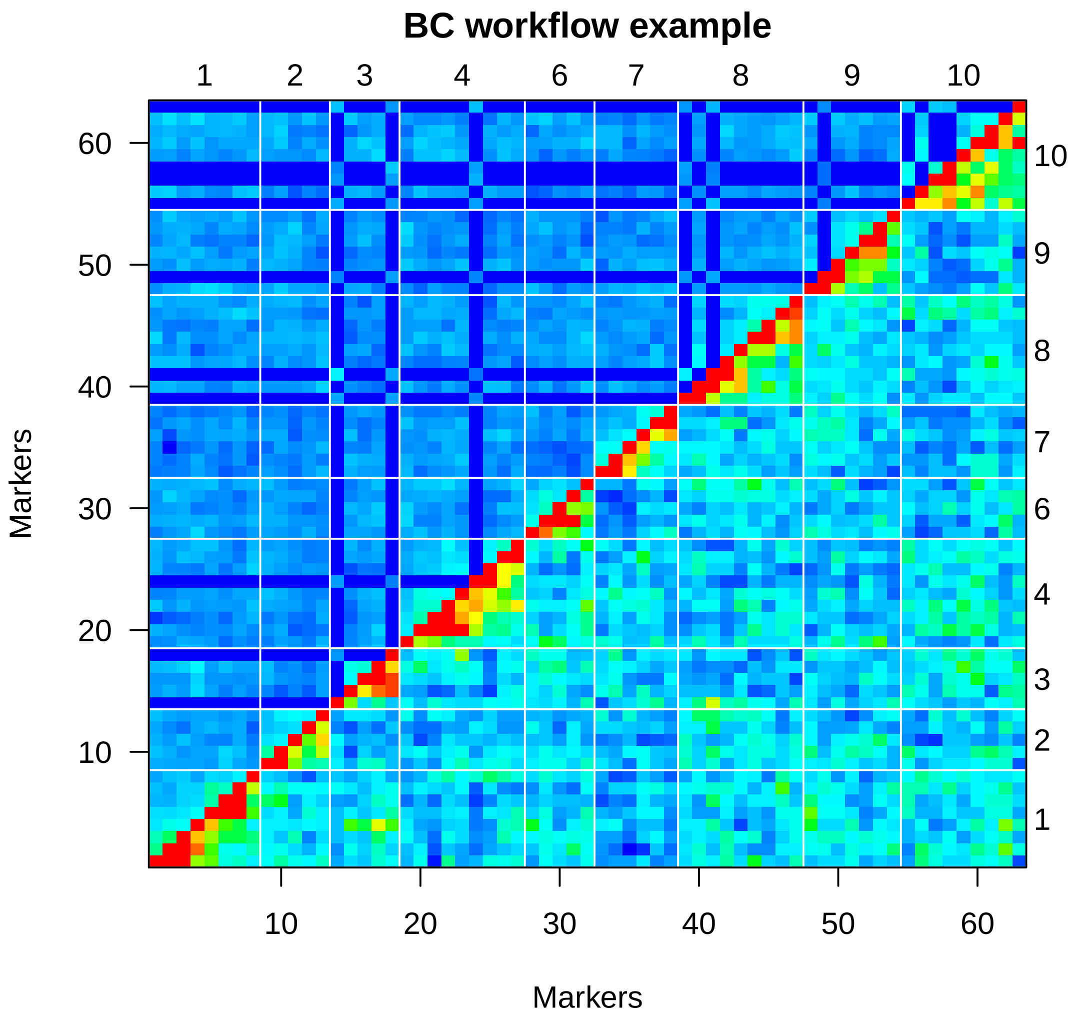
```
> population <- findDiffExpressed(population)
```

After checking distribution of the data (obviously checking distribution of one marker is not enough, but that step is just for visual inspection, toGenotypes is checking distribution of markers automatically) and finding differentially expressed genes, we can use toGenotypes function:

```
> cross <- toGenotypes(population, genotype = "simulated", proportion = c(25,
+     75), orderUsing = "map_genetic", treshold = 0.1)
```

And plot recombination factors between individuals:

```
> plot.rf(cross, main = "BC workflow example")
```



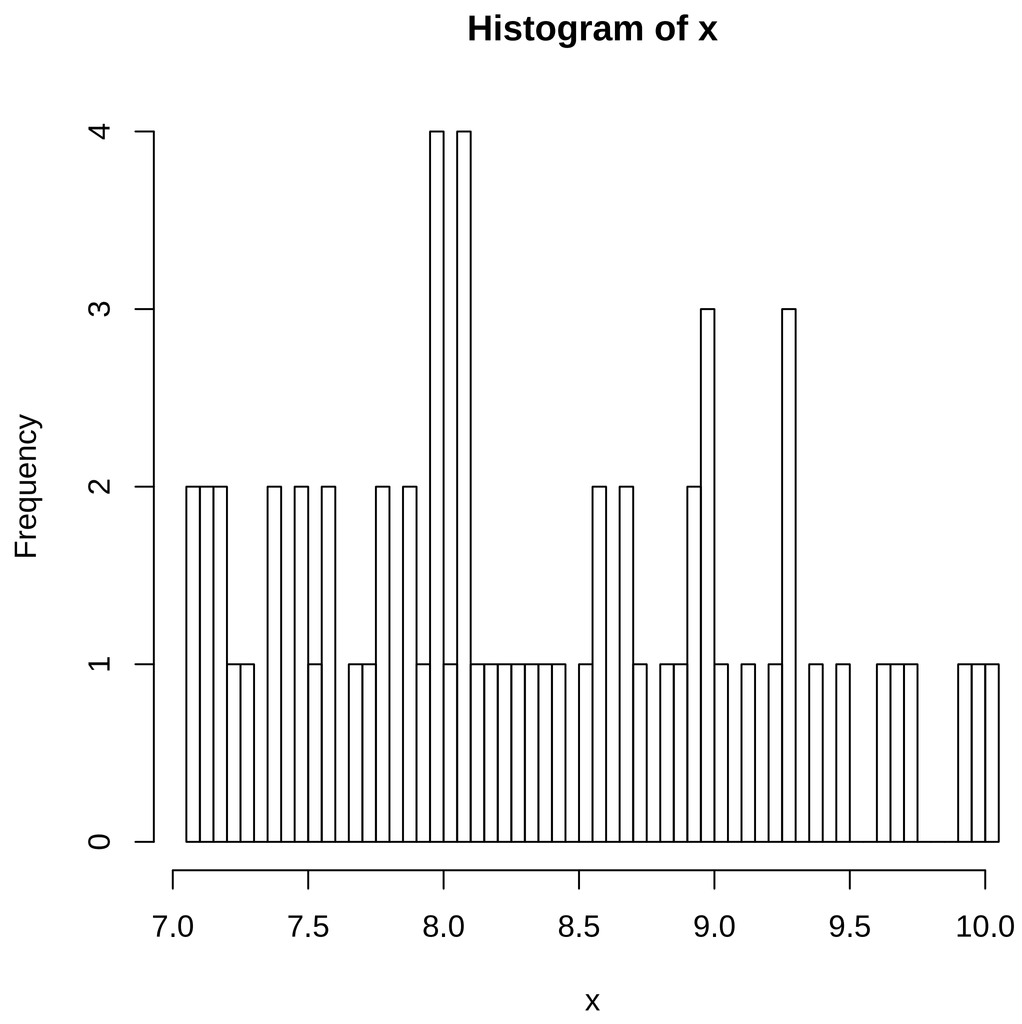
3.7 Example workflow for F2 cross

Again we start with faking population object. This time with data for F2 cross:

```
> population <- fakePopulation(type = "f2")
```

This time we expect totally different distribution of data. Mixture of three normal distributions with proportions 25/50/25:

```
> plotMarkerDistribution(population, 3, 3)
```



Finding differentially expressed genes we still do the same:

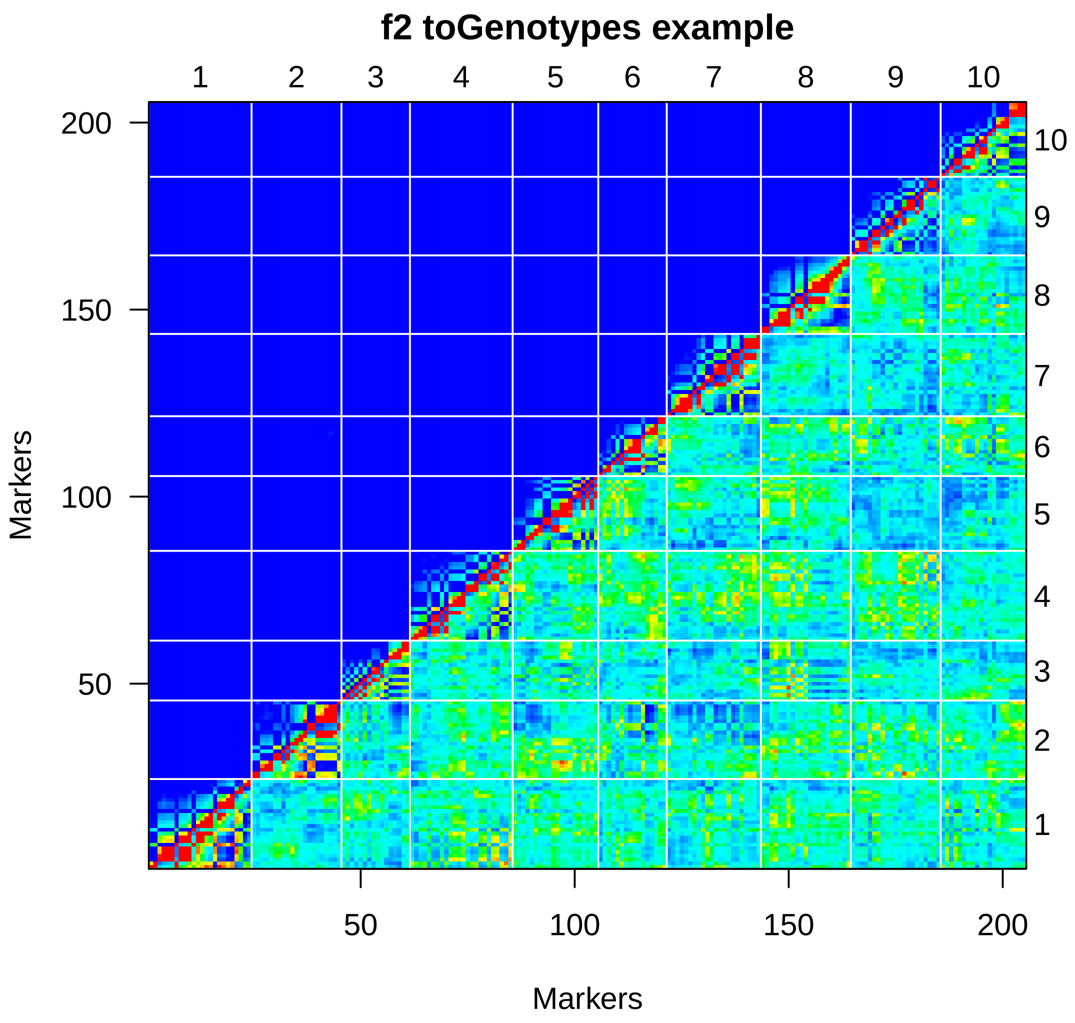
```
> population <- findDiffExpressed(population)
```

Finally, we can use toGenotypes function:

```
> cross <- toGenotypes(population, genotype = "simulated", proportion = c(25,
+     50, 25), orderUsing = "map_genetic", treshold = 0.1)
```

And plot recombination factors between individuals:

```
> plot.rf(cross, main = "f2 toGenotypes example")
```



4 Advanced options/modifications

4.1 Using data files with different structure

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

4.2 Rank product analysis

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

4.3 Uncommon types of crosses

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

4.4 Modifying splitting options

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

4.5 Filtering markers

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

4.6 Cross object

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

4.6.1 Forming linkage groups and ordering markers

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

4.6.2 Post-processing of cross object

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

5 Built-in plotting routines

5.1 plotChildrenExpression

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language. Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

5.2 plotParentalExpression

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language. Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

5.3 plotMapComparison

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language. Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

5.4 plotMarkerDistribution

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language. Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

6 Big datasets

6.1 Problematic handling of big data by R

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

6.2 C preprocessing

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

6.3 Other solutions

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like ■Huardest gefburn■. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

7 Package development & collaboration

8 References

References

Rainer Breitling, Patrick Armengaud, Anna Amtmann, and Pawel Herzyk. Rank products: a simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments. *FEBS Letters*, 573(1-3):83 – 92, 2004. ISSN 0014-5793. doi: DOI:10.1016/j.febslet.2004.07.055. URL `http://www.sciencedirect.com/science/article/pii/S0014579304009354`.

F. Hong, R. Breitling, C. W. McEntee, B. S. Wittner, J. L. Nemhauser, and J. Chory. Rankprod: A bioconductor package for detecting differentially expressed genes in meta-analysis. *Bioinformatics*, 22(22):2825–2827, 2006. doi: DOI:10.1093/bioinformatics/btl476. URL `http://bioinformatics.oxfordjournals.org/content/22/22/2825.full`.

Ritsert C. Jansen and Jan-Peter Nap. Genetical genomics: the added value from segregation. *Trends in Genetics*, 17(7):388 – 391, 2001. doi: DOI:10.1016/S0168-9525(01)02310-1. URL `http://www.sciencedirect.com/science/article/pii/S0168952501023101`.