MAPfastR Package Tutorial

Marcin Kierczak, Ronald M. Nelson, Xia Shen,
Mats Pettersson, Zheya Sheng, Örjan Carlborg

August 16, 2012

Abstract

MAPfastR is an R package for mapping quantitative trait loci (QTL) in outbred crosses. It consists of several functions that can be applied in a sequence forming a pipeline for QTL analysis in outbred lines. Some of the functions implement different algorithms that can be used interchangeably and it is up to the user to decide which will be applied. This gives the user a certain degree of flexibility when constructing the analyses pipeline and helps creating custom-tailored solutions depending on the type of analysed data and the aims of the research.

This document is a tutorial that aims at presenting several aspects of working with the package by presenting a complete analysis workflow, beginning with installing MAPfastR.

Introduction

MAPfastR is a software package developed to analyse QTL data from outbred line-crosses. It has been developed for fast and comprehensive analyses of large datasets implemented in the R language. The package has been developed for flexible analyses of large datasets. It includes an online developer and community-based support system. MAPfastR is based on a computationally efficient algorithm that uses all available data from dense SNP-chips [4]. MAPfastR provides functionality for F2 crosses and backcrosses under the assumption that different QTL alleles are fixed in the founder lines [4], line-cross analyses allowing for withinline segregation [7] and tests for epistatic interactions [3]. In addition to the standard func-tionality, the software comes with add-on packages that allow more experienced users to take advantage of modules for analyses of deep (Advanced Intercross Line) pedigrees. MAPfastR is implemented in the R language (with optimization of the more computationally intensive algorithms in C++), accepts several standard input formats and is available for Windows, Unix and MacOS.

Support

Users' forum is available at https://groups.google.com/d/forum/mapfastr. There you can post your questions and opinions and get the newest information directly from package developers.

Some useful information you can access via Computational Genetics Group website at http://www.computationalgenetics.se.

Construction of this tutorial

This tutorial brings you along an example workflow. Currently, not all the modules are documented here. We skipped the discussion of some still-in-development modules for later versions of this document. This tutorial will be continuously updated in response to the feedback from the users community and as the new functionalities are added. Please keep track to make sure you have the latest version. An overview of the package structure is presented in Figure 1.

Installation

Before using MAPfastR, the package has to bee installed in the R environment. This can be accomplished by using a standard GUI functionality like "Packages & Data" in native R GUI or in the "Packages" window in RStudio GUI. Alternatively, command line `install.packages("MAPfastR", source="")` can be used. Once the package is installed, it has to be loaded into the R environment:

```
> library("MAPfastR")
```

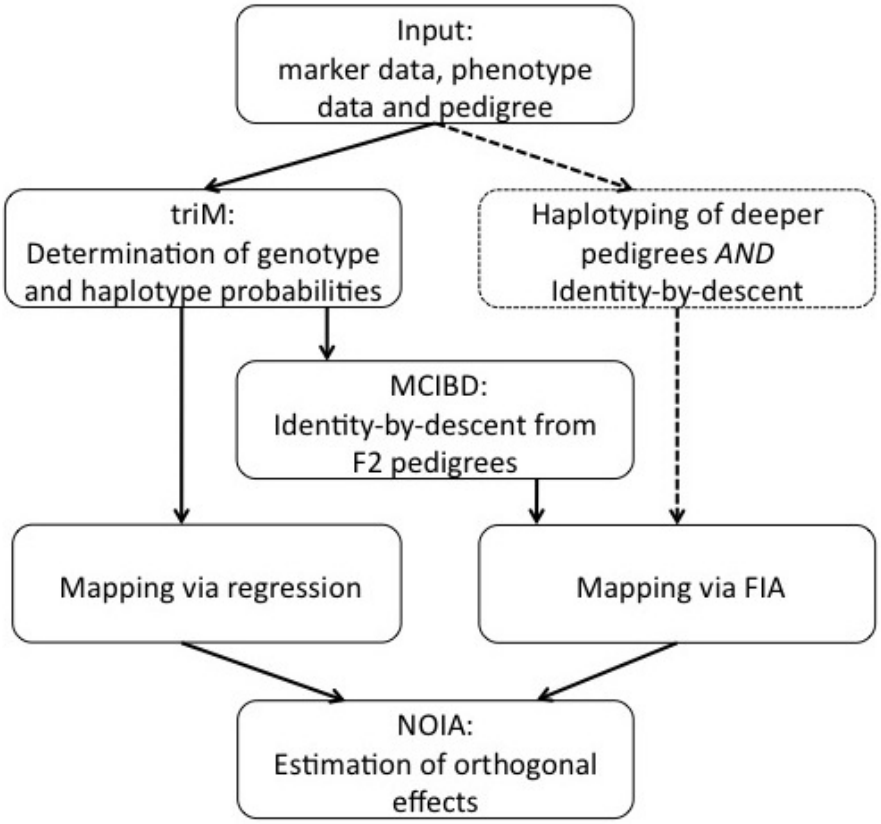

Figure 1: A diagrammatic representation of the package. Dashed arrows represent modules under development that are not included in this tutorial.

One can list the vignettes in the package and open this document by

```
> vignette(package = "MAPfastR")
> vignette("MAPfastR")
```

The package needs to be loaded every time, a new R session is started!

Known issue! Some of the MAPfastR package functions depend on other packages, one of them being "Rcpp". At the time of writing this tutorial, the newest version of the "Rcpp" package (ver. 0.9.12) contained a bug that was causing an error when calling `calc.probs` function. To remedy this, a downgrade of "Rcpp" to version 0.9.10 is required.

Importing Data

Once the package is loaded and ready to use, we have to load the data. We have prepared an example dataset that will be used throughout this tutorial.

In the following section, we will describe data format in more detailed way. Please note that MAPfastR package accepts data in two already existing and popular formats: cnf2freq/triM format and CriMap format. We encourage you to consult the documentation of these software when you need more details. The data in either of the supported formats can be directly imported to MAPfastR package using `import_data function` as described in the coming sections.

Example data

Throughout this tutorial, we are using a simple dataset that consists of two chromosomes, neither of which is a sex chromosome. There is data for NN individuals and MM phenotypes. No backcross has been used. The data is stored in four files in cnf2freq format, which can be found in the "inst" folder in the package installation directory.

- `marker_test.txt` – file containing genotyping data
- `mrkinfo_test.txt` – file containing information about markers
- `ped_test.txt` – file containing pedigree information
- `pheno_test.txt` – file containing phenotypes and co-variates, e.g. sex

Importing the example data

```
> mydata <- import_data(pedigree_file = "ped_test.txt",
                        phenotype_file = "pheno_test.txt",
                        genotype_file = "marker_test.txt",
                        marker_file = "mrkinfo_test.txt",
                        n_gen_map = 1,
                        sex.chrom=NA)
```

Now the `mydata` variable contains a data object in an internal MAPfastR format. You can see how does it look like by displaying variable names within the object:

```
> # Show variable names inside my data object
> names(mydata)
[1] "pheno"            "geno"             "backcross"
[4] "backcross.line"   "backcross.parent" "sex.chrom"
[7] "heterogam"        "sex.restrict"
> # geno itself is a data structure
> # here we show only 5 last names
> tail(names(mydata$geno), 5)
[1] "92140"    "chr"      "sex_1_cM" "sex_2_cM" "ref_cM"
> # if we want to display unique chromosomes
> unique(mydata$geno$chr)
[1] 1 2
```

Calculating line origin probabilities

Calculation of genotype and phenotype probabilities is done using triM algorithm implemented in the cnf2freq package. For details see [5] and [6]. To calculate line origin probabilities, we will be using `calc.probs` function. It will add the calculated probabilities as extra columns to the existing data structure.

```
> mydata <- calc.probs(data = mydata, interval = 1.0)
```

The package asks you which type of analysis you will perform using the calculated probabilities, and here we start by regression models, so enter "1".

```
Are you calculating probabilities for:
(1) Regression models or (2) Variance component models?
1:1
```

Now, we are ready to perform the first genome scan looking for QTL associated with "SC" phenotype and using "sex" as covariate.

```
> testscan <- autosome.genome.scan("SC", data = mydata, covariates = "sex")
The results for the starting model fitting sex to SC in mydata dataset are:

          Estimate Std. Error t value  Pr(>|t|)
Intercept   4.6640     0.1878  24.840 8.426e-89
sex        -0.4319     0.1855  -2.328 2.033e-02

F value 5.419 on 1 and 491 degrees of freedom
```

Once the scan is completed, we will have its results stored in the "testscan" object:

```
> # See what is the structure of the
> # object returned by scan
> names(testscan)
[1] "max.F.value"
[2] "position.max.F"
[3] "estimated.effects.max.F"
[4] "variables.in.starting.model"
[5] "all.F.values"
[6] "chrom.boundaries"
[7] "results.by.chromosome"
```

In addition, a nice plot summarising results of the scan will be displayed as shown in Figure 2

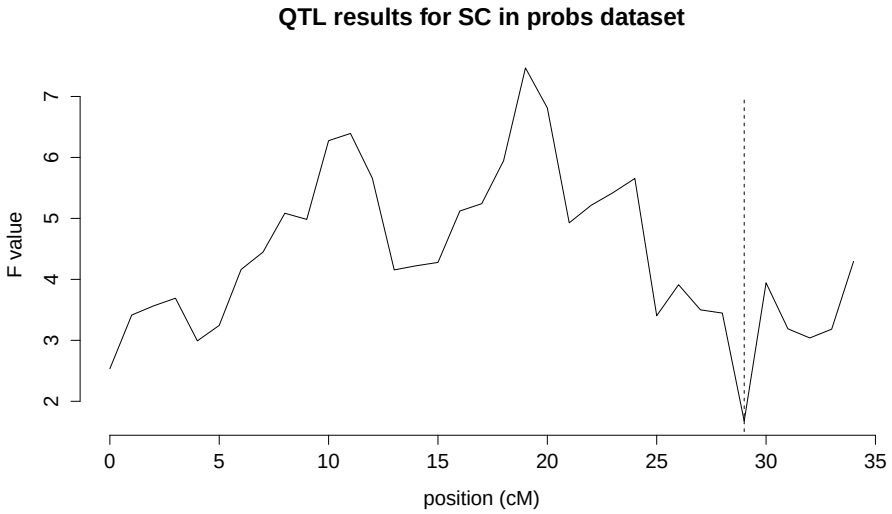

Figure 2: An example scan result. Dashed line denotes chromosome border.

NOIA

Finally, the Normal and Orthogonal Interactions model [1] implemented in the "noia" package [2] can be used to check for interactions between loci. At first, we will pick up some promising loci looking at the scan results. We can see that there are two peaks on chromosome 1: one around 10-11cM, another around 19-20cM. We can retrieve their positions in the vector of markers"

```
> # Select chromosome 1
> ref_cM.chr1 <- mydata$geno$ref_cM[which(mydata$geno$chr == 1)]
> #
> # Which F-values were above 6?
> myFvals <- testscan$results.by.chromosome$chrom.1$F.values
> names(myFvals[myFvals >= 6])
[1] "10 cM" "11 cM" "19 cM" "20 cM"
> #
> # Define 2 peaks
> peak1 <- which(ref_cM.chr1 >= 10 & ref_cM.chr1 <= 11)
> peak2 <- which(ref_cM.chr1 >= 19 & ref_cM.chr1 <= 20)
```

Now, we will select only 2 loci (for demonstration purposes), one from each peak and try to fit linear NOIA.

```
> # Select one (the first) locus from each peak
> mymarkers <- c(peak1[1], peak2[1])
> #
> # Run NOIA
```

```
> noia_result <- run_NOIA(data = mydata, phenotype = "SC",
                markers = mymarkers, noia_func="linear")
> noia_result
Phenotype:
        n= 493  min:  3.2  max:  5.1  mean:  4.2284
Genotype:
        n= 493 , 2 loci
                Locus 1                1: 0.819        2: 0.168        3: 0.0122
                Locus 2                1: 0.415        2: 0.441        3: 0.144

       Effects   Variances Std.err Pr(>|t|)
..  4.2318e+00  0.0000e+00  0.0176  < 2e-16 ***
a. -3.0998e-03  1.7287e-06  0.0555  0.95551
d.  3.6912e-01  5.0866e-03  0.1609  0.02217 *
.a -2.4486e-02  2.9131e-04  0.0267  0.35970
aa -4.1243e-02  2.0884e-04  0.0778  0.59623
da -2.7147e-01  2.2152e-03  0.1800  0.13211
.d -1.9931e-02  8.6310e-05  0.0379  0.59915
ad -4.0508e-02  9.2991e-05  0.0934  0.66483
dd          NA          NA      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Variances
        Total (phen)        0.13781
        Residual         0.13471
        Explained         0.0030975        (2.25%)
        Genetic          0.007983
```

Variance component QTL analysis & FIA

The MAPfastR package provides functions for IBD (identity-by-descent) matrix estimation and variance component QTL analysis, where the latter includes the flexible intercross analysis (FIA) [7]. Alternatively to the `import_data` function, one can load the example dataset directly via:

```
> data(mydata)
```

Now we re-run the `calc.prob` function to get probabilities for IBD estimation. Here, there is no need to assign the function call to a data object. An object named `output` will be created automatically.

```
> calc.probs(data = mydata, interval = 100)
```

It takes long to run the calculation for every centi-Morgan, so we set the interval argument to be 100 as an example. The package asks you which type of analysis you will perform using the calculated probabilities, enter “2”.

```
Are you calculating probabilities for:
(1) Regression models or (2) Variance component models?
1:2
```

It takes longer time to calculate probabilities for variance component analysis than those for regression models. An object `output` is created in the working memory space. Now we use the MCIBD routine (see package vignette "MCIBD" for details) to calculate IBD matrices across the genome.

```
> MCIBD.genome(data = mydata, trim.output = output, mc.size = 9)
```

99 Monte Carlo imputes for IBD estimation is the default setting. One can increase this number to gain precision, and here we decrease it to gain efficiency as an example. After this function call of MCIBD, two folders "chr1" and "chr2" will be created in the working directory since there are two chromosomes in this example dataset. In each folder, the IBD matrix at position index i is stored as "i.ibd.RData". Now we can run an ordinary variance component QTL scan for trait "SC" by:

```
> VC.result <- FIA.scan(mydata, "SC", fixed.effects = "sex",
+ chr = 1:2, figure.file = "VC_scan.pdf")
```

The function call returns the FIA scores of each tested locus into `VC.result` and produces a figure for each chromosome in `VC_scan.pdf` (Figure 3). Furthermore, we can do a FIA scan considering within-line correlation instead of assuming no correlation within each parental line. In order to do this, we need to calculate another set of IBD matrices assuming within-line fixation. There is detailed description of how this should be defined in `MCIBD` documentation. Basically, we need to know how many males and females there are in the founders. We run:

```
> sum(mydata$pheno$generation == 1 & mydata$pheno$sex == 1)
[1] 14
> sum(mydata$pheno$generation == 1 & mydata$pheno$sex == 2)
[1] 19
```

So the new `MCIBD.genome` function call is written as:

```
> MCIBD.genome(data = mydata, trim.output = output, mc.size = 99,
+ segregation = rep(1:2, c(14*2, 19*2)))
```

After this function call of `MCIBD`, in the two folders "chr1" and "chr2", a new set of IBD matrices are stored, such as "i.ibd1.RData" for position with index i. Now the FIA scan can be executed by specifying the argument `estimate.ro = TRUE`.

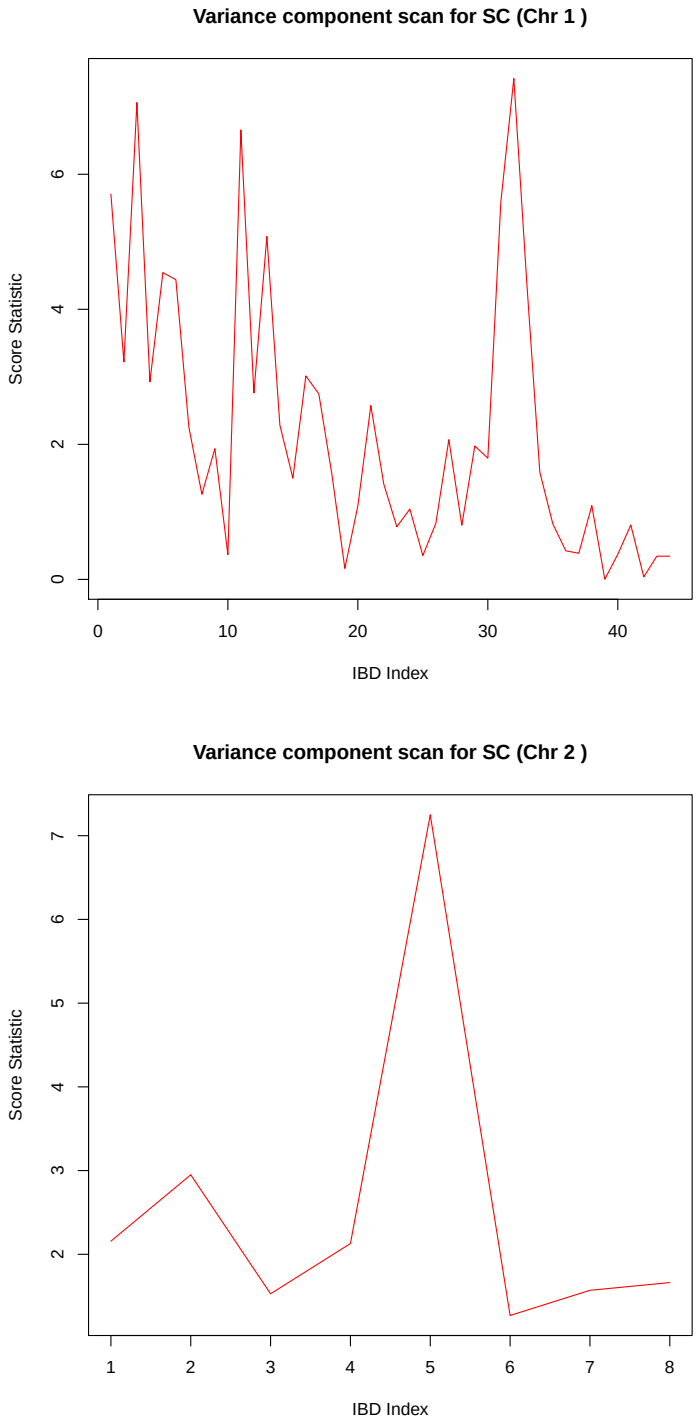

Figure 3: An example variance component QTL scan result.

```
> FIA.result <- FIA.scan(mydata, "SC", fixed.effects = "sex",
+ chr = 1:2, estimate.ro = TRUE, figure.file = "FIA_scan.pdf")
```

The function call returns the FIA scores of each tested locus into `FIA.result` and produces a figure for each chromosome in `FIA_scan.pdf` (Figure 4). We see that position 32 has a high score in the scan, therefore we can try to fit a FIA model for this locus to estimate the within-line correlation.

```
> model <- FIA.model(mydata, "SC", fixed.effects = "sex",
+ chr = 1, IBD.index = 32)
Estimated within-line correlation: rho = 0.9125732
```

We get an within-line correlation estimate of 0.91 which is close to fixation in this particular example locus. The fitted model as a `hglm` object together with the within-line correlation estimate ρ (`rho`) are returned into `model`, so that the generic function such as `summary` can be applied to check the fitted random effects model.

```
> summary(model$model)
Call:
hglm.default(X = X, y = y, Z = cbind(L1, L2), conv = 1e-06,
    RandC = rep(length(y), 2))


----------
MEAN MODEL
----------

Summary of the fixed effects estimates:

                                                        Estimate Std. Error
(Intercept)                                               4.7853     0.2157
data$pheno[data$pheno$generation == 3, fixed.effects]  -0.5193     0.1953
                                                        t-value Pr(>|t|)
(Intercept)                                              22.183  < 2e-16 ***
data$pheno[data$pheno$generation == 3, fixed.effects]  -2.659  0.00811 **
---
Signif. codes:  0 Ô***Õ 0.001 Ô**Õ 0.01 Ô*Õ 0.05 Ô.Õ 0.1 Ô Õ 1
Note: P-values are based on 461 degrees of freedom

Summary of the random effects estimates:

      Estimate Std. Error
Z.1     0.0276     0.0761
Z.2    -0.0348     0.0649
Z.3    -0.0536     0.0657
Z.4    -0.0009     0.0670
```

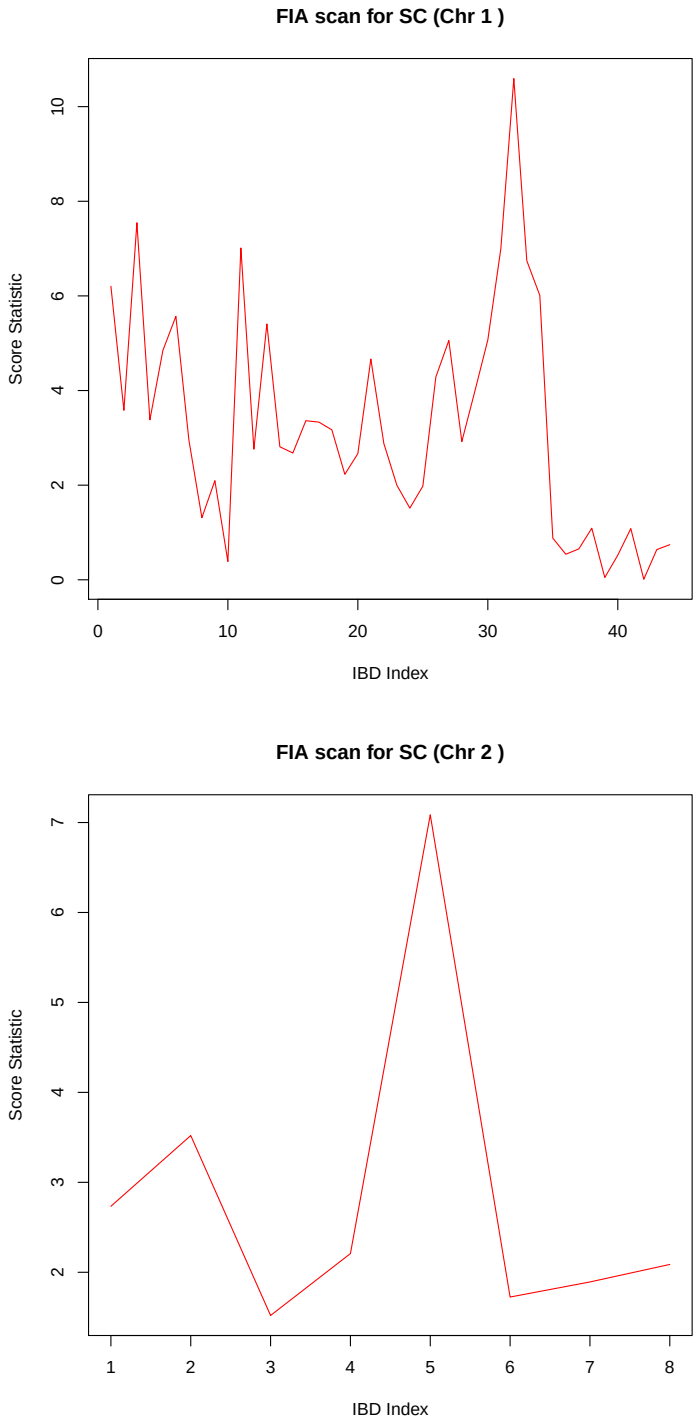

Figure 4: An example FIA QTL scan result.

```
Z.5    -0.0333     0.0671
...
Z.489   0.0000     0.0830
Z.490   0.0000     0.0830
Z.491   0.0000     0.0830
Z.492   0.0000     0.0830
Z.493   0.0000     0.0830

Summary of the random effects estimates:

      Estimate Std. Error
Z.494   0.0314     0.0682
Z.495  -0.0496     0.0529
Z.496  -0.0306     0.0640
Z.497  -0.0496     0.0646
Z.498  -0.0012     0.0649
...
Z.982   0.0000     0.0800
Z.983   0.0000     0.0800
Z.984   0.0000     0.0800
Z.985   0.0000     0.0800
Z.986   0.0000     0.0800

----------------
DISPERSION MODEL
----------------

WARNING: h-likelihood estimates through EQL can be biased.

Dispersion parameter for the mean model:
[1] 0.1255077

Model estimates for the dispersion term:

Link = log

Effects:
  Estimate Std. Error
   -2.0754     0.0659

Dispersion = 1 is used in Gamma model on deviances to
calculate the standard error(s).

Dispersion parameter for the random effects:
[1] 0.006356 0.005800

Dispersion model for the random effects:

Link = log
```

```
Effects:
.|Random1
  Estimate Std. Error
   -5.0584     0.3645

.|Random2
  Estimate Std. Error
   -5.1499     0.3674

Dispersion = 1 is used in Gamma model on deviances to
calculate the standard error(s).

EQL estimation converged in 7 iterations.
```

We hope that this tutorial has been useful for you and that our package facilitates your research. We are happy to receive any suggestions and we will do our best to improve MAPfastR in response to user feedback. Thank you!

THE END!

Bibliography

[1] J Alvarez-Castro and Ö Carlborg. A general model for functional and statistical epistasis and its application in qtl analysis. *Genetics*, (176), 2007.

[2] J Alvarez-Castro, A Le Rouzic, and Ö Carlborg. How to perform meaningful estimates of genetic effects. *PLOS Genetics*, (4), 2008.

[3] Ö Carlborg and L Andersson. The use of randomization testing for detection of multiple epistatic qtl. *Genetical Research*, (79), 2002.

[4] L Crooks, C Nettelblad, and Ö Carlborg. An improved method for estimating chromosomal line origin in qtl analysis of crosses between outbred lines. *G3: Genes Genomes Genetics*, (1: 57-64), 2011.

[5] RM Nelson, C Nettelblad, L Crooks, ME Pettersson, F Besnier, X Shen, J Álvarez Castro, L Rönnegård, W Ek, Z Sheng, M Kierczak, S Holmgren, and Ö Carlborg. Mapfastr: Qtl mapping in outbred line crosses. *Bioinformatics*, 2012.

[6] C Nettelblad, S Holmgren, L Crooks, and Ö Carlborg. cnf2freq: Efficient determination of genotype and haplotype probabilities in outbred populations using markov models. In *Bioinformatics and Computational Biology*, number 5462 in Lecture notes in Bioinformatics. Springer, 2009.

[7] L Rönnegård, F Besnier, and Ö Carlborg. An improved method for qtl detection and identification of within-line segregation in f2 intercross designs. *Genetics*, (178), 2008.