An Introduction to the use of the R-package GeneticTools

Daniel Fischer

March 16, 2013

1 Installation

There are no special requirements for the installation required. However, in case that the Bioconductor packages are not part of the local R repository the package `snpStats` has be to installed beforehand by typing

```
R>  source("http://bioconductor.org/biocLite.R")
R>  biocLite("snpStats")
```

After that the package can be installed as usually

```
R>  install.packages("GeneticTools")
```

After installation the package can be loaded into R's workspace

```
R>  library(GeneticTools)
```

2 Performing an eQTL analysis

The package *GeneticTools* can be used to perform an eQTL analysis and the function name for that is `eQTL()`. Currently we use a simulated dataset to explain the use of the package, but two manuscripts that use real data are currently submitted. As soon as their data is publical available it will be added into this vignette. Hence, at this stage it is not possible to provide here real data, but upon request the author is happy to provide further insights into that analysis.

2.1 Required data formats

We start by making the provided example data available. This data comes with the package. In future there will be also real data available.

```
R> data(Xgene)
R> data(genotData)
R> data(annotTrack)
```

The data format of the expression data `Xgene` looks like this:

```
R>    Xgene[1:4,1:5]

          CHPT1       DRAM1     GNPTAB     MYBPC1  RNA5SP368
Ind1 -1.6564078 -1.19148701 -0.4498994  1.2559992 -0.3344102
Ind2  0.8467278  0.07740516  0.5691444  0.8438009 -2.3245996
Ind3 -1.5724256  0.10391143  0.5989183  0.3593218 -1.0434410
Ind4  0.9476815 -1.89861866 -0.5046581 -0.4820447 -1.3604475
```

The object `Xgene` is a matrix, containing a gene in each column and the Individuals are in the rows. The `genotData` is an object as imported from the `snpStats` package and contains the following attributes

```
R> attributes(genotData)

$names
[1] "genotypes" "fam"       "map"
```

If we would have it as ped/map file pair we could import it via

```
>R  genotData <- read.pedfile(file="genotData.ped",snps="genotData.map")
```

or alternatively give it to the `eQTL` function itself as string object and the function will import the data.

The objects within this list look like this

```
R> genotData$genotypes[1:10,1:10]

A SnpMatrix with  10 rows and  10 columns
Row names:  Ind1 ... Ind10
Col names:  SNP1 ... SNP10
```

```
R> head(genotData$fam)

     pedigree member father mother sex affected
Ind1     Ind1   Ind1   <NA>   <NA>  NA        2
Ind2     Ind2   Ind2   <NA>   <NA>  NA        2
Ind3     Ind3   Ind3   <NA>   <NA>  NA        2
Ind4     Ind4   Ind4   <NA>   <NA>  NA        2
Ind5     Ind5   Ind5   <NA>   <NA>  NA        2
Ind6     Ind6   Ind6   <NA>   <NA>  NA        2
```

```
R> head(genotData$map)

  V1 snp.names V3        V4 allele.1 allele.2
1  2      SNP1  0 100134631        G        A
2  2      SNP2  0 101454156        G        C
3  2      SNP3  0 100289244        G        C
4  2      SNP4  0 102006689        C        G
5  2      SNP5  0 102805453        G        T
6  2      SNP6  0 100657089        C        G
```

Please notice that the names of the individuals in the expression data match with those in the Genotype object `genotData`. If this wouldn't be the case we could specify the names in a separate vector. Lets assume that the order within the expression matrix is the same as in the genotype object, then we could use the rownames as such an object. This could be done via

```
R> genoNames <- rownames(Xgene)
```

and the vector `genoNames` can then be used later in the `eQTL` call.

We need to have the gene annotation in bed format (Please notice the change to the official convention, this is on high priority of the ToDo list of the package to change this to common notation.)

When we download the annotation track from the ensemble webpage in gtf format (e.g. here: `http://www.ensembl.org/info/data/ftp/index.html`) this track looks like this:

```
R> annotTrack[1:6,1:8]
```

```
        V1             V2         V3        V4        V5 V6 V7 V8
1500000 12 protein_coding        CDS 102071880 102072016  .  +  2
1500001 12 protein_coding       exon 102074121 102074307  .  +  .
1500002 12 protein_coding        CDS 102074121 102074307  .  +  0
1500003 12 protein_coding       exon 102079360 102079584  .  +  .
1500004 12 protein_coding        CDS 102079360 102079457  .  +  2
1500005 12 protein_coding stop_codon 102079458 102079460  .  +  0
```

(we omitted to print here column `V9`) due to it's length. A typical entry of column `V9` looks like

```
R> annotTrack[1,9]
```

```
[1] " gene_id ENSG00000196091; transcript_id ENST00000392934; exon_number 25; gene_name MY
```

The ■`annotTrack` was imported by using the command

```
R>  annotTrack <- read.table(file="ensembleAnnot.gtf",sep="\t")
```

Our method requires the gene annotation in bed format and there is a function included that transforms the standard ensemble gtf file into the format required by the `eQTL` function. Please notice, that this function currently works only for human genomes as the chromosomal information is required. This function is called `gtfToBed` and takes as an input the gtf object.

```
R> annotBed <- gtfToBed(annotTrack)
R> annotBed
```

```
            Name Chr     Start      Stop
1         MYBPC1   2 101988749 102079796
2  RP11-755O11.2   2 102040498 102053748
3          CHPT1   2 102090725 102137918
4          Y_RNA   2 102113586 102113696
5          SYCP3   2 102122426 102133250
6         GNPTAB   2 102139275 102224716
```

```
7             U6   2 102159184 102265579
8   RP11-511H9.3   2 102166892 102167533
9      RNA5SP368   2 102173592 102173707
10        snoU13   2 102183593 102183693
11     RNA5SP369   2 102230598 102230705
12  RP11-511H9.4   2 102267094 102267401
13         DRAM1   2 102271129 102317401
```

2.2 Performing an *cis*-eQTL

To perform a basic *cis*-eQTL with the minimum required input, using a linear model we type:

```
R> lm.myEQTL <- eQTL(gex=Xgene,geno=genotData, xAnnot=annotBed)

We will transform the gene annotations into a list at Sat Mar 16 20:39:06 2013 !
We finished to transform the gene annotations at Sat Mar 16 20:39:06 2013
We have for 100 % of the samples in the expression data the genotype information.
We have for 100 % of the expression data the annotations.
We will investigate for 13 possible eQTLs!
We managed to calculate 1 eQTLs at Sat Mar 16 20:39:07 2013
We managed to calculate 2 eQTLs at Sat Mar 16 20:39:08 2013
We managed to calculate 3 eQTLs at Sat Mar 16 20:39:09 2013
We managed to calculate 4 eQTLs at Sat Mar 16 20:39:10 2013
We managed to calculate 5 eQTLs at Sat Mar 16 20:39:11 2013
We managed to calculate 6 eQTLs at Sat Mar 16 20:39:12 2013
We managed to calculate 7 eQTLs at Sat Mar 16 20:39:13 2013
We managed to calculate 8 eQTLs at Sat Mar 16 20:39:14 2013
We managed to calculate 9 eQTLs at Sat Mar 16 20:39:15 2013
We managed to calculate 10 eQTLs at Sat Mar 16 20:39:16 2013
We managed to calculate 11 eQTLs at Sat Mar 16 20:39:17 2013
We managed to calculate 12 eQTLs at Sat Mar 16 20:39:18 2013
We managed to calculate 13 eQTLs at Sat Mar 16 20:39:20 2013
```

The eQTLs are calculated in that case for a window of 1MB up and downstream of the center gene. In order to switch to the directional test we have only to add the parameter `method="directional"`. By default the method gives detailed information about the ongoing calculation steps. If this behavior is not desired, it can be switched of by adding the option `versatile=FALSE`.

The standard output just shows the eQTL with p-values smaller than 0.01 in bed format:

```
R> lm.myEQTL

   Chr     Start       End    Name
2    2 101768986 101768986  SNP891
3    2 102541483 102541483  SNP993
4    2 102454065 102454065   SNP41
5    2 102479126 102479126  SNP240
6    2 102228497 102228497  SNP991
7    2 102341206 102341206 SNP1072
```

```
8    2 102661696 102661696  SNP548
9    2 101981603 101981603 SNP1027
10   2 102561538 102561538 SNP1077
11   2 101614125 101614125  SNP356
12   2 101900654 101900654 SNP1062
13   2 102339524 102339524  SNP760
14   2 101842406 101842406 SNP1013
15   2 102432170 102432170  SNP597
16   2 102028702 102028702  SNP731
17   2 102664917 102664917  SNP589
18   2 102107602 102107602  SNP742
19   2 102440639 102440639  SNP334
20   2 102333854 102333854  SNP644
21   2 101939276 101939276  SNP626
22   2 102085552 102085552   SNP28
23   2 102447857 102447857   SNP40
24   2 102064643 102064643   SNP64
25   2 102028702 102028702  SNP731
26   2 101722161 101722161   SNP29
27   2 102572349 102572349   SNP45
28   2 102175935 102175935  SNP106
29   2 101996387 101996387  SNP994
```

If we are interested in other significance levels we can change that by typing e.g.

```
R> print(lm.myEQTL,sig=0.001)

  Chr     Start       End    Name
2   2 102228497 102228497  SNP991
3   2 102561538 102561538 SNP1077
4   2 102028702 102028702  SNP731
5   2 102440639 102440639  SNP334
```

We can also specify which SNP-gene associations should be displayed. If we would like to see only those SNPs that are associated to genes MYBPC1 and U6 we type

```
R> print(lm.myEQTL, which=c("MYBPC1","U6"))

  Chr     Start       End    Name
2   2 101614125 101614125  SNP356
3   2 101900654 101900654 SNP1062
4   2 102064643 102064643   SNP64
5   2 102028702 102028702  SNP731
```

The standard output omits some information, e.g. the information what SNP is tested to be associated to what gene. The output of the `print` function can be extended with the option `output="full"`, e.g.:

```
R> print(lm.myEQTL, which=c("MYBPC1","U6"), output="full")
```

```
$MYBPC1
     Chr     SNP  Position Allele1 Allele2     p.value
356    2  SNP356 101614125       G       A 0.005240337
1062   2 SNP1062 101900654       T       C 0.003763006

$U6
    Chr    SNP  Position Allele1 Allele2     p.value
64    2  SNP64 102064643       C       T 0.007003025
731   2 SNP731 102028702       G       C 0.005917425
```

This output shows then in detail the significant associations listed for each gene separately, showing the p-value as well as the alleles.

Of course it is also possible to perform the eQTL for a single gene. This can be done by giving the annotations just this particular gene:

```
R> lm.myEQTL <- eQTL(gex=Xgene,geno=genotData, xAnnot=annotBed[1,])

We will transform the gene annotations into a list at Sat Mar 16 20:39:20 2013 !
We finished to transform the gene annotations at Sat Mar 16 20:39:20 2013
We have for 100 % of the samples in the expression data the genotype information.
We have for 7.7 % of the expression data the annotations.
We will investigate for 1 possible eQTLs!
We managed to calculate 1 eQTLs at Sat Mar 16 20:39:22 2013
```

or by specifying it's name:

```
lm.myEQTL <- eQTL(gex=Xgene,geno=genotData, xAnnot=annotBed, which="MYBPC1")
```

A similar situation happens, when there is also the expression values only for this gene available. A typical situation for that case is

```
R> gexVector <- Xgene[,1]
R> gexVector[1:5]

      Ind1       Ind2       Ind3       Ind4       Ind5
-1.6564078  0.8467278 -1.5724256  0.9476815 -1.0997639

R> anOne<- annotBed[1,]
R> anOne

    Name Chr     Start      Stop
1 MYBPC1   2 101988749 102079796
```

and the eQTL is then performed by

```
R> lm.myEQTL <- eQTL(gex=gexVector,geno=genotData, xAnnot=anOne)

We will transform the gene annotations into a list at Sat Mar 16 20:39:22 2013 !
We finished to transform the gene annotations at Sat Mar 16 20:39:22 2013
We have for 100 % of the samples in the expression data the genotype information.
We have for 100 % of the expression data the annotations.
We will investigate for 1 possible eQTLs!
We managed to calculate 1 eQTLs at Sat Mar 16 20:39:23 2013
```

```
R> lm.myEQTL

  Chr     Start       End   Name
2   2 101768986 101768986 SNP891
3   2 101522440 101522440 SNP964
4   2 102541483 102541483 SNP993
```

Please notice here, that `anOne` is still a `data.frame`, as it contain different input types.

If you want to type the location by hand you could use the command

```
R> anManual <- data.frame(Name="MYSP", Chr="2", Start=12344, Stop=31234)
R> anManual

  Name Chr Start  Stop
1 MYSP   2 12344 31234
```

When the eQTL is performed with standard setting, it is performed in so-called *full-mode*. This means, that it stores besides the significant test results plenty of other information, e.g the test results for the non-significant tests. This is on the one hand a desired behavior, e.g. when the test results for *cis*-eQTL tests should be visualized, but e.g. in case of a *trans*-eQTL this might cause memory problems. Hence, especially for longer gene lists or for *trans*-eQTL experiments, it is adviceable to store only the significant test results in order to reduce the required memory.

All that has to be done for that is to specify the significance level in the `eQTL`-call:

```
R> lm.myEQTL <- eQTL(gex=gexVector,geno=genotData, xAnnot=anOne, sig=0.01)

We will transform the gene annotations into a list at Sat Mar 16 20:39:23 2013 !
We finished to transform the gene annotations at Sat Mar 16 20:39:23 2013
We have for 100 % of the samples in the expression data the genotype information.
We have for 100 % of the expression data the annotations.
We will investigate for 1 possible eQTLs!
We managed to calculate 1 eQTLs at Sat Mar 16 20:39:24 2013

R> lm.myEQTL

    chr    SNP  Location     p.value Assoc.Gene
891   2 SNP891 101768986 0.002993818     MYBPC1
964   2 SNP964 101522440 0.000969900     MYBPC1
993   2 SNP993 102541483 0.007711215     MYBPC1
```

In case that the `sig` option is set, the standard output of the function changes and the test results and the gene names are given as well.

As mentioned earlier, the search window around the center gene is 1MB up- and downstream. This can be changed with the option `windowSize=2`. In that case we would perform tests for all SNPs in a window of size 2MB around the center genes (i.e. in total 4MB).

2.3 Performing an *trans*-eQTL

To perform a *trans*-eQTL the same options as above are possible, and all that we have to do is to set the window size to `NULL`, by setting the option `windowSize=NULL`. The eQTL is then performed by typing.

```
R> lm.myTransEQTL <- eQTL(gex=Xgene,geno=genotData, xAnnot=annotBed, windowSize=NULL, sig=
```

```
We will transform the gene annotations into a list at Sat Mar 16 20:39:24 2013 !
We finished to transform the gene annotations at Sat Mar 16 20:39:24 2013
We have for 100 % of the samples in the expression data the genotype information.
We have for 100 % of the expression data the annotations.
We will investigate for 13 possible eQTLs!
We managed to calculate 1 eQTLs at Sat Mar 16 20:39:29 2013
We managed to calculate 2 eQTLs at Sat Mar 16 20:39:34 2013
We managed to calculate 3 eQTLs at Sat Mar 16 20:39:38 2013
We managed to calculate 4 eQTLs at Sat Mar 16 20:39:42 2013
We managed to calculate 5 eQTLs at Sat Mar 16 20:39:48 2013
We managed to calculate 6 eQTLs at Sat Mar 16 20:39:55 2013
We managed to calculate 7 eQTLs at Sat Mar 16 20:40:01 2013
We managed to calculate 8 eQTLs at Sat Mar 16 20:40:06 2013
We managed to calculate 9 eQTLs at Sat Mar 16 20:40:12 2013
We managed to calculate 10 eQTLs at Sat Mar 16 20:40:20 2013
We managed to calculate 11 eQTLs at Sat Mar 16 20:40:27 2013
We managed to calculate 12 eQTLs at Sat Mar 16 20:40:32 2013
We managed to calculate 13 eQTLs at Sat Mar 16 20:40:38 2013
```

Please notice here the longer calculation times, as each gene is tested against each given SNP.

2.4 Visualization of the eQTL results

When the eQTL is performed in *full-mode* (default, no sig option is mentioned) then we can also visualize the result in two possible ways Assume again, we performed this *cis*-eQTL

```
R> lm.myCisEQTL <- eQTL(gex=Xgene, geno=genotData, xAnnot=annotBed, which="U6", verbose=FA
```

We can visualize then the test results around this center gene by typing:

```
R> plot(lm.myCisEQTL)
```

A significance level could be specified here also with the option `sig` and it determines which test results will be highlighted in red.

Alternative the y-axis can be on log scale, by setting `log=TRUE`. In that case are bars instead of dots used. In that case has

```
R> plot(lm.myCisEQTL,log=TRUE, sig=2)
```

In case we performed the *cis*-eQTL not only for a single gene but for a whole set, we can either specify for what gene the plot should be created

```
plot(lm.myEQTL2, which="U6")
```

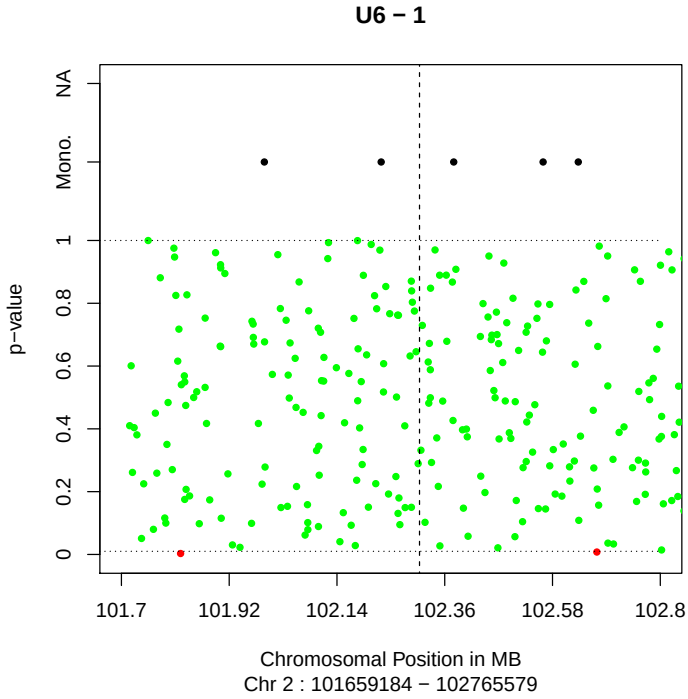

Figure 1: Standard plot of a *cis*-eQTL

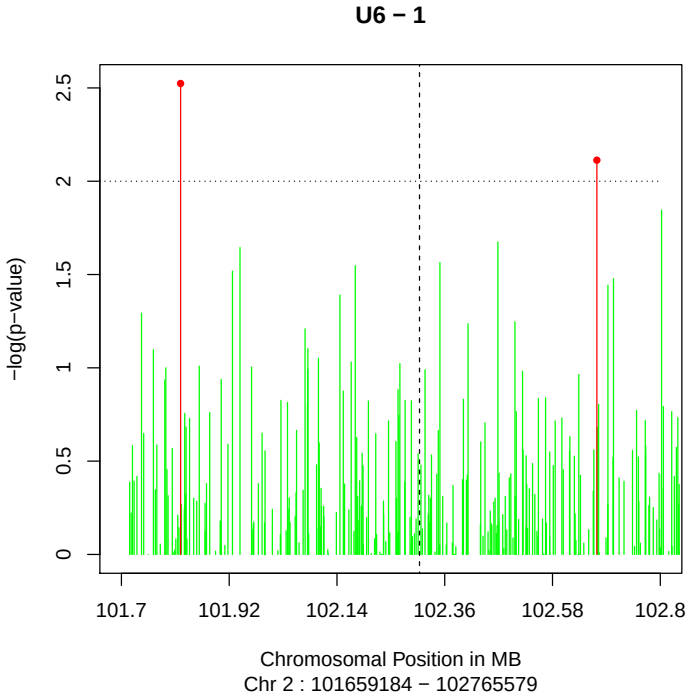

Figure 2: Log-scale plot of a *cis*-eQTL

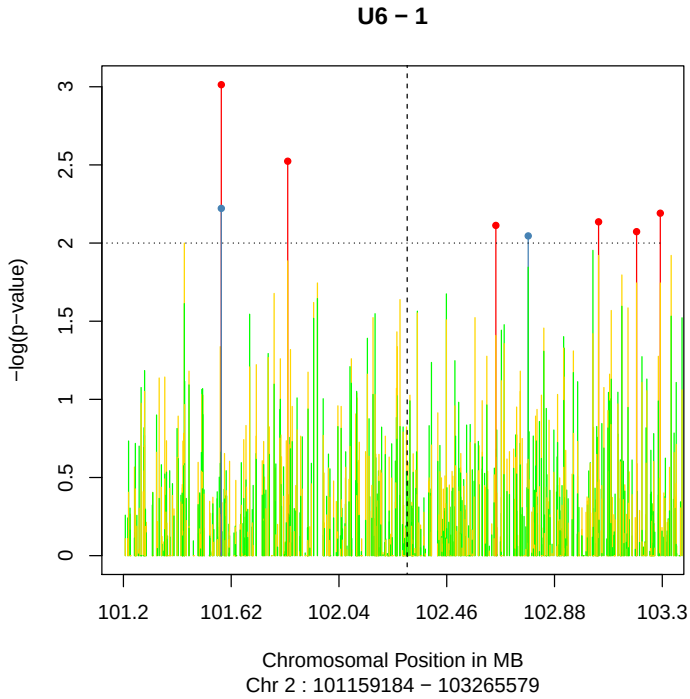

Figure 3: Comparions-plot of a *cis*-eQTL

or we could specify a file name and then all plots will be plotted into one pdf file where each page contains then one figure.

```
plot(lm.myEQTL2, file="/home/ejo138/eqtlExample.pdf")
```

If we run another eQTL testing method, e.g the linear test we can compare also the results, using the plotting function:

```
R> lm.myEQTLcom <- eQTL(gex=Xgene,geno=genotData, xAnnot=annotBed, which="U6", method="LM"
R> dir.myEQTLcom <- eQTL(gex=Xgene,geno=genotData, xAnnot=annotBed, which="U6", method="di
\
```

The comparison can only be done on the log scale:

```
R> plot(lm.myEQTLcom, x2=dir.myEQTLcom, log=TRUE, sig=2)
```

When we test for *trans*-eQTL it is often desired to visualize the results also in an informative way. Here is the function `visTrans` of help

```
visTrans(snpGene=lm.myTransEQTL$bed,annotBed)
```

Since we used here simplified, simulated data to show the use of the methods, we wouldn't see anything special inside this data set. A typical figure from a real data set would like e.g. like the one shown in Figure 4.

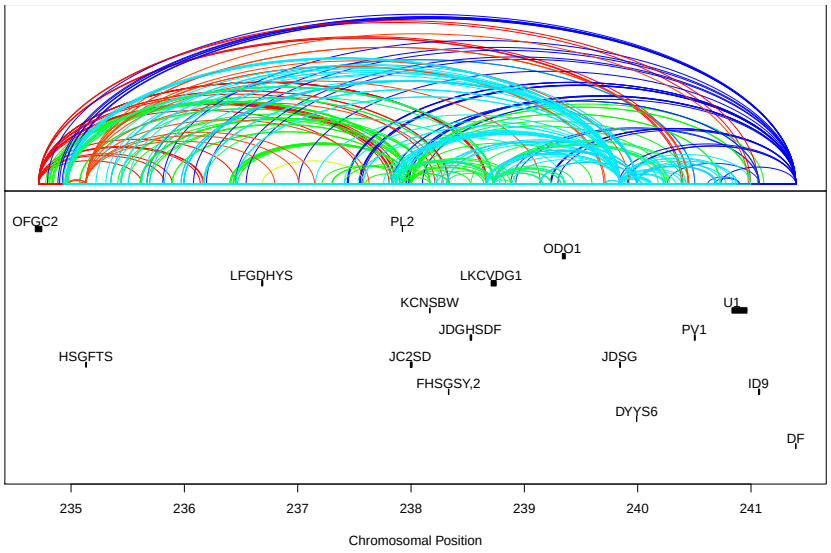

Figure 4: Visualization of a *trans*-eQTL object on real data.

We can see here clearly some regions that are associated with certain genes and these regions deserve some closer investigations.

After having an eQTL performed we often want to visualize also how the expression values are assigned to the different genotype groups. This can be done with the command `genotypePlot`.

```
R> genotypePlot(snp="SNP10", gene="MYBPC1", eqtl=lm.myEQTL, ylab="Expression values",xlab=
```

3 Performing an MDR

To be added.

4 Performing a time series clustering for gene expression profiles

This part of the apckage is currently under active development and shouldn't be used for productive code. Hence, no further advices are given on that topic.

5 Further information

Currently there is a project page under development that is designed for direct feedback and help. It can be found under

`http://genetictools.danielfischer.name`.

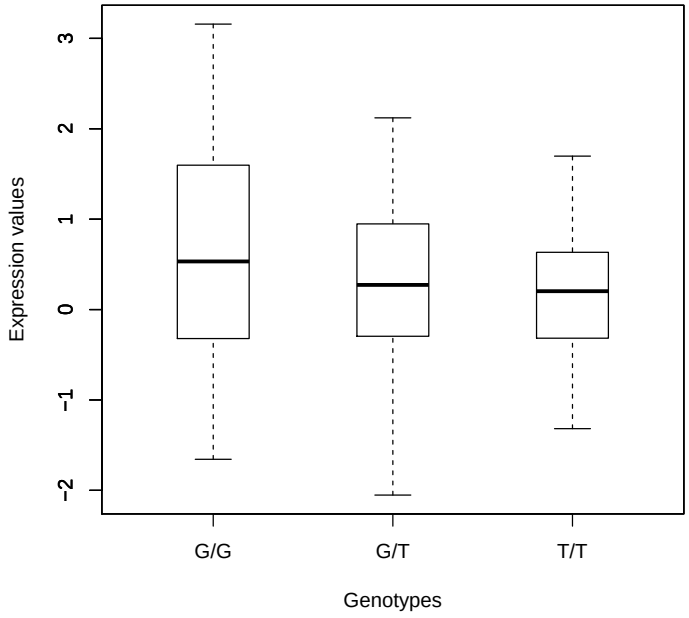

Figure 5: The `genotypePlot` - plot