cgmisc: package tutorial

Marcin Kierczak, Jagoda Jablonska, Simon Forsberg, Matteo Bianchi, Katarina Tengvall, Mats Pettersson, Jennifer Meadows, Patric Jern, Orjan Carlborg, Kerstin Lindblad-Toh

2 Feb 2015

cgmisc is an R package for enhanced genome-wide association studies (GWAS) and visualisation. This document aims at guiding you through the installation process and to demonstrate package capabilities in a series of practical examples based on an example data included in the package.

Package installation

The **cgmisc** package can be installed in the same way as any other R package. One way is to issue the following command in R console:

```
install.packages("cgmisc")
```

Other possibilities include using graphical user interface (GUI) of, e.g. R console or RStudio.

After the package has been installed, to use the package, it is necessary to load it into environment:

```
library("cgmisc")
```

```
## Loading required package: GenABEL
## Loading required package: MASS
## Loading required package: GenABEL.data
##
##  Package cgmisc contains miscellaneous functions, useful for extending
## genome-wide association study (GWAS) analyses.
##
## Package Name: cgmisc
##  Version: 2.9.8
##  Date: 2015-01-28
##  Author: Marcin Kierczak <marcin.kierczak@imbim.uu.se>
##  License GPL (>=2.10)
##
##  Package contains various functions useful in computational
##     genetics, especially in genome-wide association studies.
```

Loading data

Whenever possible, the **cgmisc** package uses data structures used by the GenABEL (Aulchenko et al., 2007) package. In particular, the `gwaa.data-class` and the `gwaa.scan-class` structures are used. The package is shipped with an example dataset called **cgmisc_data** that contains genotyping data (Illumina, canFam2) for N=207 German shepherds originally collected for the project described in (Tengvall et al., 2012). The phenotypes, though, have been simulated in order to be able to illustrate various features of **cgmisc**. To load the example dataset, use the following command:

```
data("data")
```

Example analyses

In order to illustrate how to use particular functions, we will perform a very much simplified GWAS analysis. We begin by initial quality control where we prune the data with per marker of per individual call rates below 95%. Based on 2000 randomly selected markers, we remove one (with lower call rate) from each pair of too similar (more than 95% similarity) individuals. We also set very low (10^{-3}) threshold for pruning on minor allele frequency (in practise only the monomorphic markers will be removed) and turn off checks based on the departure form Hardy-Weinberg equilibrium (p.level=10e-18)

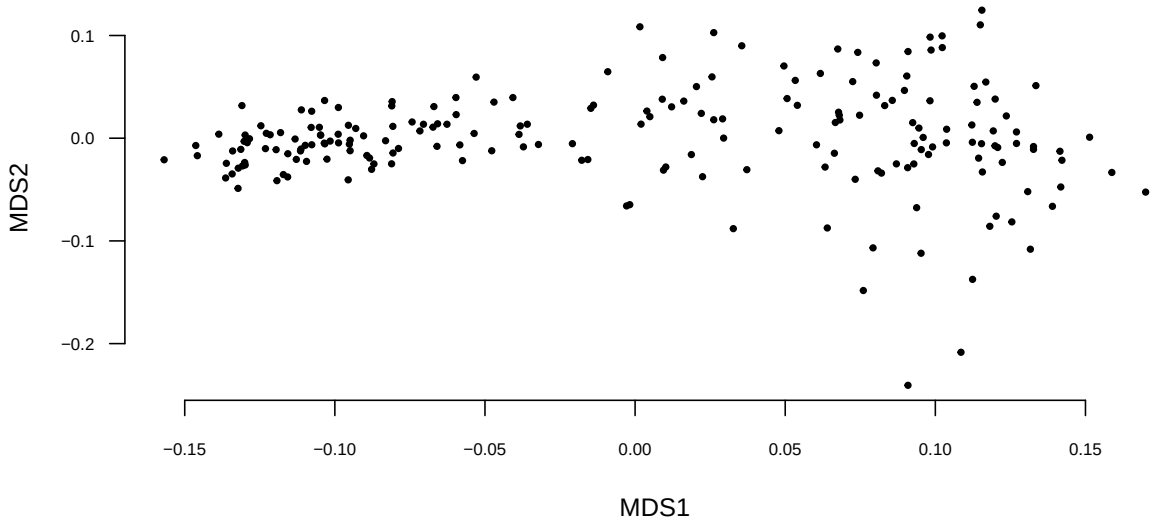
```
qc1 <- check.marker(data, callrate = .95, perid.call = .95, ibs.threshold = .95, ibs.mrk=2000, ibs.exclu
```

```
## Excluding people/markers with extremely low call rate...
## 174375 markers and 207 people in total
## 0 people excluded because of call rate < 0.1
## 1069 markers excluded because of call rate < 0.1
## Passed: 173306 markers and 207 people
##
## RUN 1
## 173306 markers and 207 people in total
## 42743 (24.66331%) markers excluded as having low (<0.1%) minor allele frequency
## 1468 (0.8470567%) markers excluded because of low (<95%) call rate
## 650 (0.3750591%) markers excluded because they are out of HWE (P <1e-17)
## 0 (0%) people excluded because of low (<95%) call rate
## Mean autosomal HET is 0.2658536 (s.e. 0.01917125)
## 0 people excluded because too high autosomal heterozygosity (FDR <1%)
## Mean IBS is 0.7758926 (s.e. 0.01470521), as based on 2000 autosomal markers
## 2 (0.9661836%) people excluded because of too high IBS (>=0.95)
## In total, 128942 (74.40135%) markers passed all criteria
## In total, 205 (99.03382%) people passed all criteria
##
## RUN 2
## 128942 markers and 205 people in total
## 0 (0%) markers excluded as having low (<0.1%) minor allele frequency
## 0 (0%) markers excluded because of low (<95%) call rate
## 0 (0%) markers excluded because they are out of HWE (P <1e-17)
## 0 (0%) people excluded because of low (<95%) call rate
## Mean autosomal HET is 0.2660211 (s.e. 0.01918304)
## 0 people excluded because too high autosomal heterozygosity (FDR <1%)
## Mean IBS is 0.7735147 (s.e. 0.0156286), as based on 2000 autosomal markers
## 0 (0%) people excluded because of too high IBS (>=0.95)
## In total, 128942 (100%) markers passed all criteria
## In total, 205 (100%) people passed all criteria
```

```
data.qc1 <- data[qc1$idok, qc1$snpok]
```

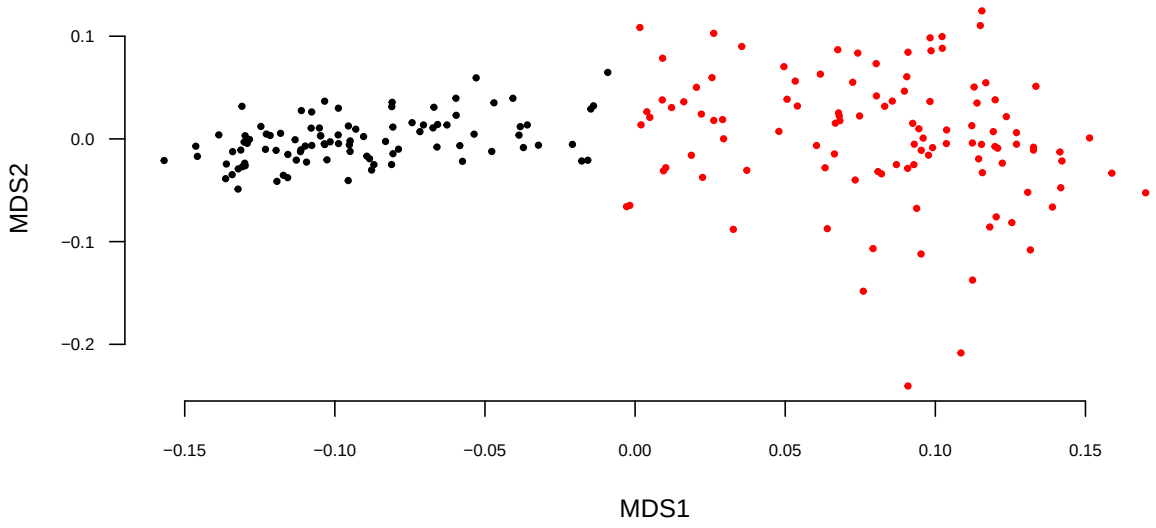
Next, we analyse population structure by means of genomic-kinship:

```
autosomal <- which(data.qc1@gtdata@chromosome != 39)
data.qc1.gkin <- ibs(data.qc1, snpsubset = autosomal, weight = 'freq')
data.qc1.dist <- as.dist(0.5 - data.qc1.gkin)
data.qc1.mds <- cmdscale(data.qc1.dist)
plot(data.qc1.mds, pch=19, cex=.5, las=1, xlab="MDS1", ylab="MDS2", cex.axis=.7, bty='n')
```



We can see that there is possible population structure here. We should investigate this further, but for our purposes, let's just run simple K-means clustering with the number of clusters *a priori* set to $K = 2$

```
kclust <- kmeans(data.qc1.mds, centers = 2)
plot(data.qc1.mds, pch=19, cex=.5, las=1, xlab="MDS1", ylab="MDS2", cex.axis=.7, bty='n', col=kclust$clu
```

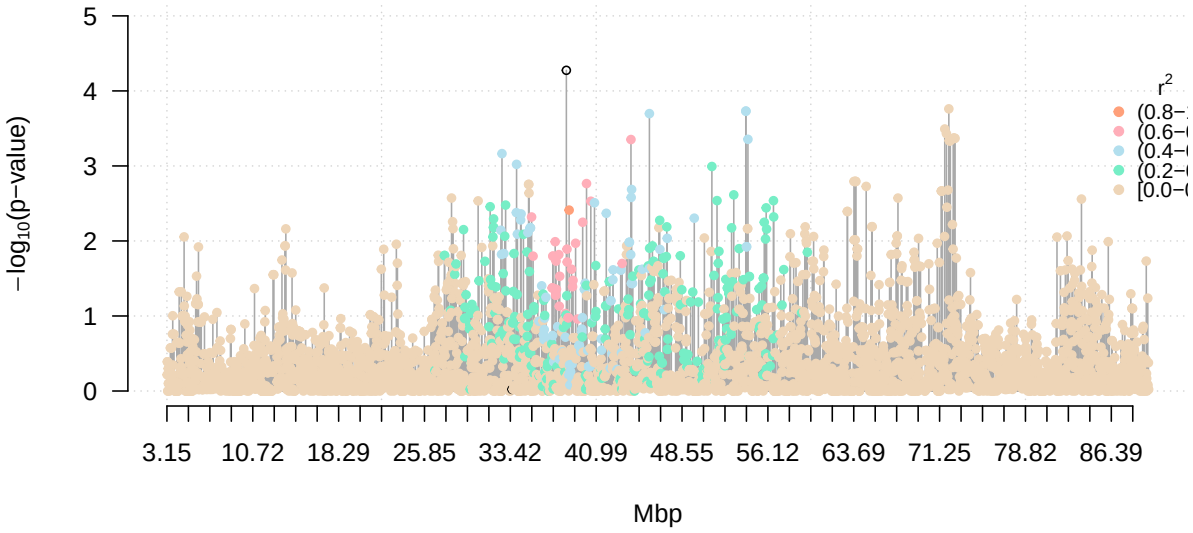



```
pop <- kclust$cluster
```

We can compare subpopulations looking at the differences in the reference allele frequency using the `pop.allele.counts` and the `plot.pac` functions. Here, we just focus on chromosome 2. # Comparing subpopulations

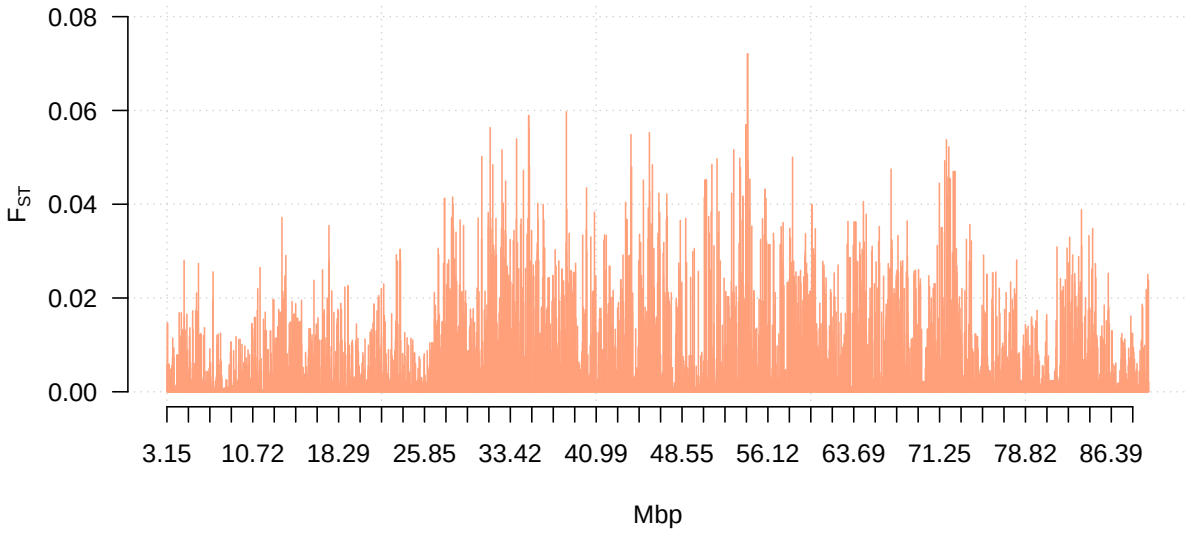
```
pac <- pop.allele.counts(data.qc1[,data.qc1@gtdata@chromosome==2], pop, progress=F)
plot.pac(data.qc1[,data.qc1@gtdata@chromosome==2], allele.cnt = pac, plot.LD = T)
```

```
## Loading required package: wesanderson
```



In a similar way, we can compute and plot fixation index F_{ST}:

```
fst <- compute.fstats(data.qc1[,data.qc1@gtdata@chromosome==2], pop)
plot.fstats(data.qc1[,data.qc1@gtdata@chromosome==2], fst)
```



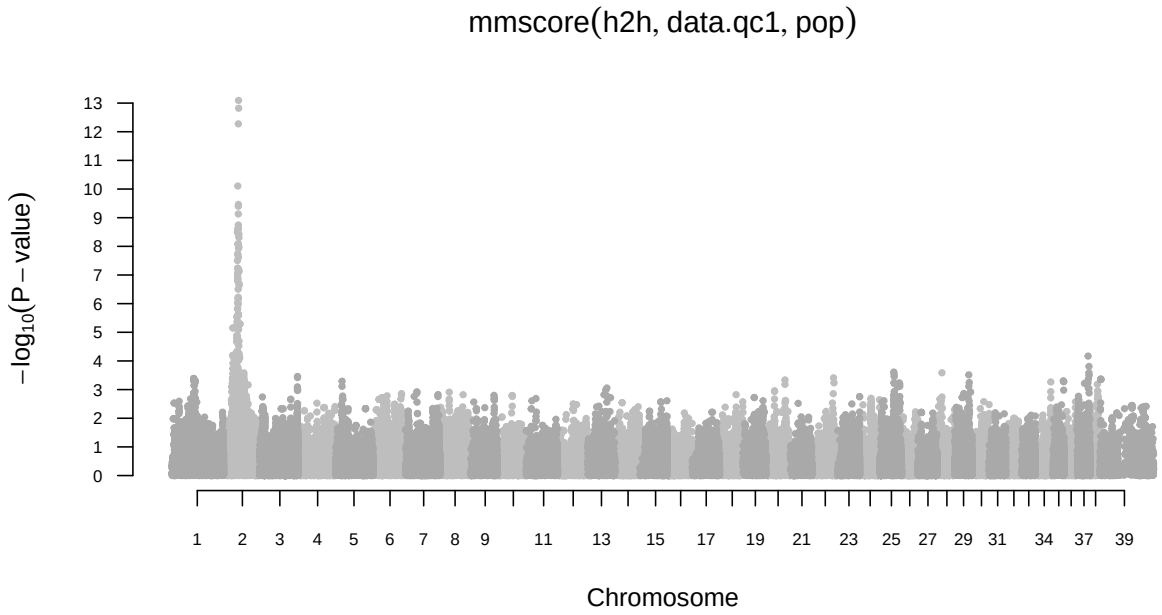
Having defined subpopulations, we can proceed to association analyses using mixed model with genomic kinship as random effect.

```
h2h <- polygenic_hglm(formula = ct ~ sex, data.qc1.gkin, data.qc1)
```

```
## Loading required package: hglm
```

```
## Loading required package: Matrix
## Loading required package: hglm.data
##
## hglm: Hierarchical Generalized Linear Models
## Version 2.0-11 (2014-10-30) installed
## Authors: Xia Shen, Moudud Alam, Lars Ronnegard
## Maintainer: Xia Shen <xia.shen@ki.se>
##
## Use citation("hglm") to know how to cite our work.
##
## Discussion: https://r-forge.r-project.org/forum/?group_id=558
## BugReports: https://r-forge.r-project.org/tracker/?group_id=558
## VideoTutorials: http://www.youtube.com/playlist?list=PLn1OmZECD-n15vnYzvJDy5GxjNpVV5Jr8
```

```
mm <- mmscore(h2h, data.qc1, strata = pop)
par(las=1, cex.axis=.7) # Tweak graphics
plot(mm, cex=.5, pch=19, col=c("darkgrey","grey"))
```



As we can see, there is a very strong association signal on chromosome 2. We can examine it a bit closer using the `plot.manhattan.ld` function.

Visualization and analyses of linkage structure

Say, we would like to zoom in on chromosome 2 and visualise LD to the top-associated marker. First, we need the name and cooridinates of the marker:

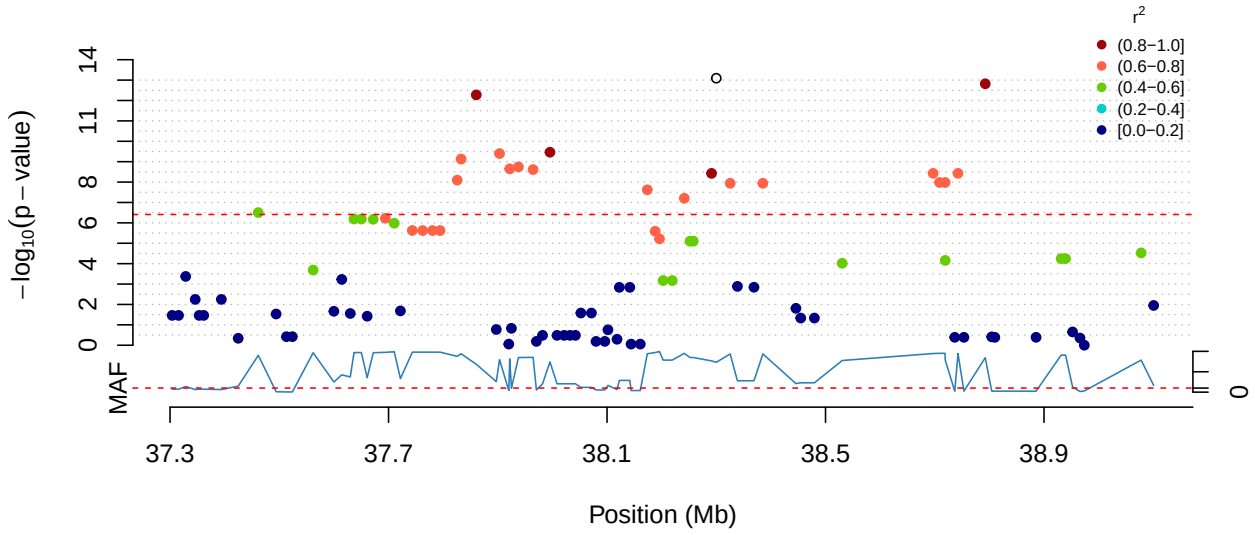
```
summary(mm, top=1)
```

```
## Summary for top 1 results, sorted by P1df
```

```
##                Chromosome Position Strand A1 A2   N     effB   se_effB
## BICF2S2365880           2 38256927      u  T  C 205 1.095257 0.1466398
##                chi2.1df         P1df        Pc1df effAB effBB chi2.2df P2df
## BICF2S2365880 55.78642 8.078765e-14 3.540639e-13    NA    NA        0   NA
```

We see that the top-associated marker is **BICF2S2365880** and its position is **38256927bp**. We will zoom in on a 2Mbp region centered on the marker:

```
plot.manhattan.LD(data = data.qc1, gwas.result = mm, chr = 2, region = c(37256927,39256927), index.snp =
```



top snps

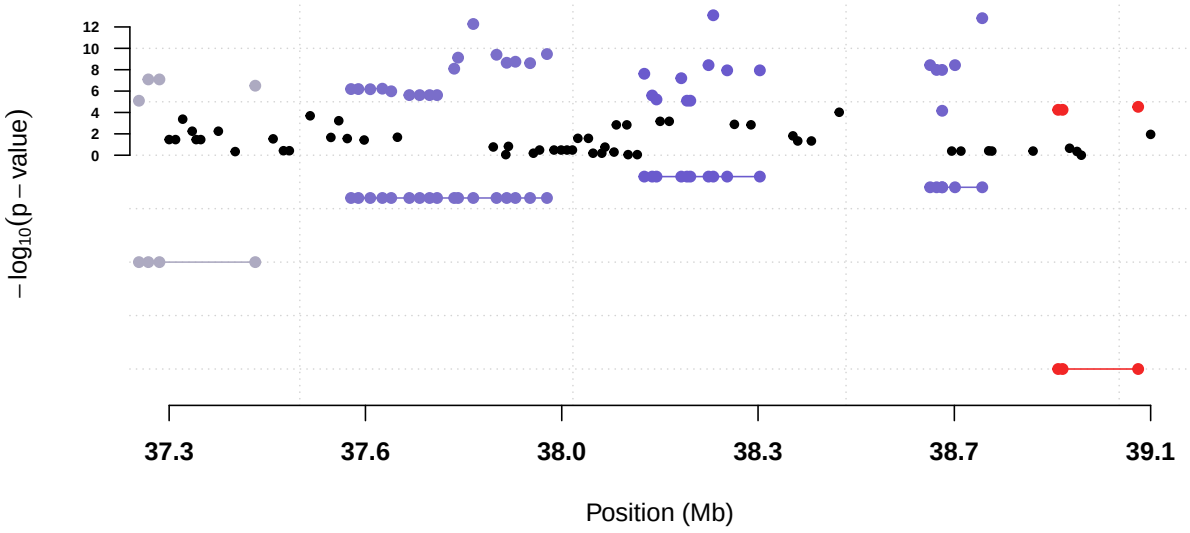
```
top.snps <- choose.top.snps(data = data.qc1, chr = 2, region = c(37256927,39256927), index.snp = "BICF2S

print (top.snps)
```

```
##                          marker    coord
## BICF2S2365880          INDEX SNP 38256927
## BICF2P462003   0.916279689260684 37817567
## BICF2P425207   0.866030555369713 38248275
## BICF2P612394   0.843123203444902 37952464
## BICF2P1340853   0.83585996145732 38749375
## BICF2P558673   0.784841547756073 37860148
## BICF2P201321   0.740325898827365 38342208
## BICF2P1315898  0.740325898827365 38282388
## BICF2S2319660  0.710277601793861 38198355
## BICF2P266917   0.707782841536202 38130649
## BICF2S23339898 0.691211049336204 37782677
```

```
## BICF2P628260   0.678611787509248 37789762
## BICF2P644508   0.662336911797675 37878759
## BICF2S23027232 0.640468741510863 38152969
## BICF2P567151   0.631178965999453 37894702
## BICF2P992165    0.62679552094748 37921591
## BICF2P1191074  0.623164831486209 38675801
## BICF2S23340033 0.623164831486209 38665751
## BICF2S23047517 0.617595381512591 38699448
## BICF2P218913   0.617595381512591 38653901
## BICF2P343829   0.613841113425679 38145114
## TIGRP2P21859   0.608354859526652 37700553
## TIGRP2P21848   0.606012904893604 37651105
## BICF2P1155790  0.604429710732471 37751264
## BICF2P691713   0.604429710732471 37737541
## BICF2P540705   0.604429710732471 37719461
```

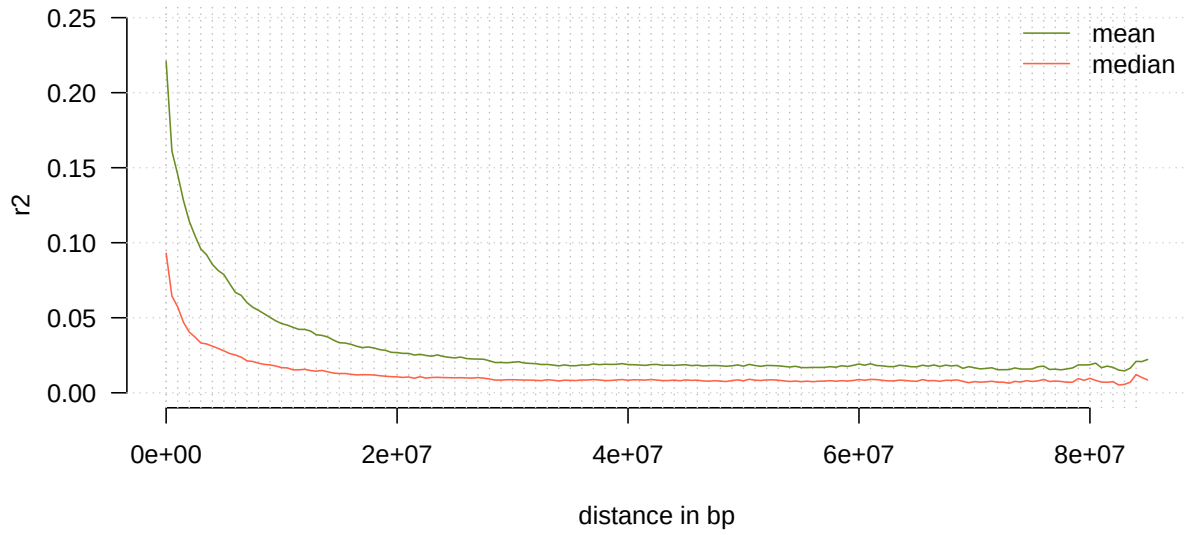
We can also use the clumping procedure as outlined in PLINK documentation [cite] to single out regions of interest.

```
clumps <- clump.markers(data, mm, chr = 2)
plot.clumps(mm, clumps, 2, c(37256927,39256927))
```



To visualise LD decay on chromosome 2, one can call the `plot.ld.decay` function.

```
plot.LD.decay(data.qc1[,data.qc1@gtdata@chromosome==2])
```



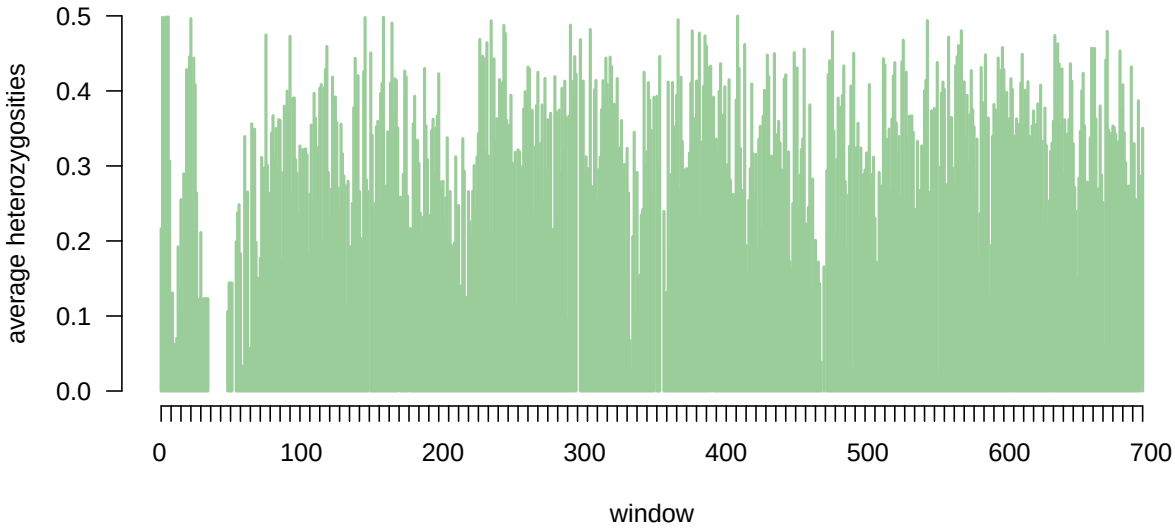
Detecting runs of homozygosity

Computing average heterozygosity using overlapping windows approach.

```
LW <- get.overlap.windows(data.qc1, 2, 125e3, 2500)
het.windows <- het.overlap.wind(data.qc1, LW, F)
```

Now, having calculated average heterozygosity we can visualize them with `plot.overlap`

```
plot.overlap(LW,het.windows)
```



We can use calculated heterozygosity to examine runs of homozygosity across selected chromosome. Let's use `get.roh` and check if we have any stretches of reduced heterozygosity on chromosome 2. Below a given threshold, all windows will be treated as homozygous.

```
get.roh(data = data.qc1, chr=2, LW=LW, hetero.zyg=het.windows, threshold = 0.30, strict = TRUE)
```

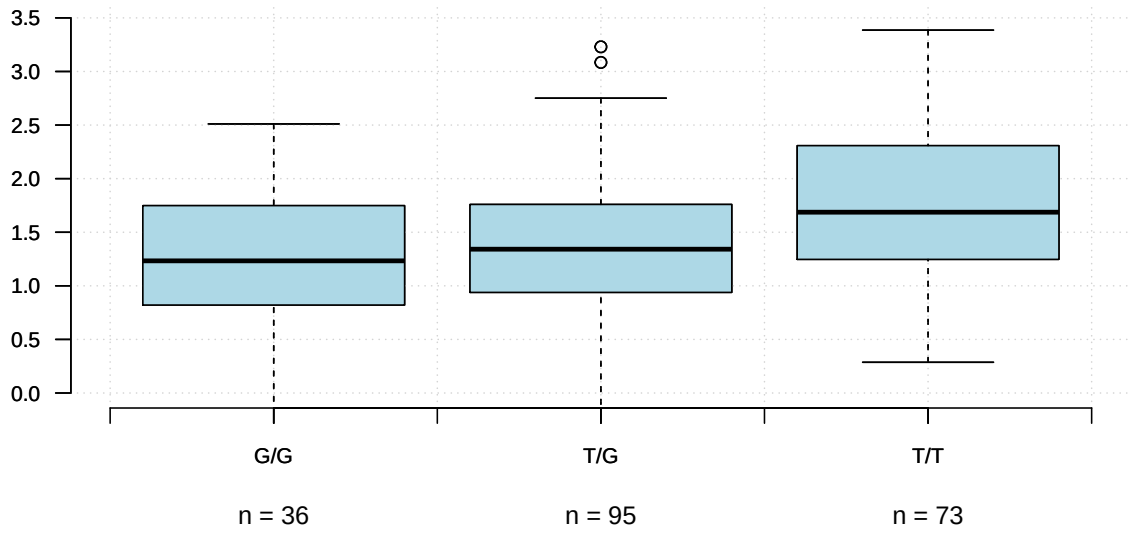
```
##      window    begin      end length
## [1,]      8  4010290  5360290     87
## [2,]    197 27162790 28022790     54
## [3,]    316 41740290 42477790     52
## [4,]    436 56440290 57667790     77
```

As a result we get a matrix runs coordinates, length (in windows) and first window that starts a stretch.

For quantitative traits, the `boxplot.snp` function can be used to visually examine allele or genotype effect by plotting phenotype boxplots for the individuals in every genotypic class. The function works for both outbreed (three boxes) and inbreed (two boxes) data.

```
## Loading required package: genetics
## Loading required package: combinat
##
## Attaching package: 'combinat'
##
## The following object is masked from 'package:utils':
##
##     combn
##
## Loading required package: gdata
## gdata: Unable to locate valid perl interpreter
## gdata:
## gdata: read.xls() will be unable to read Excel XLS and XLSX files
```

```
## gdata: unless the 'perl=' argument is used to specify the location
## gdata: of a valid perl intrpreter.
## gdata:
## gdata: (To avoid display of this message in the future, please
## gdata: ensure perl is installed and available on the executable
## gdata: search path.)
## gdata: Unable to load perl libaries needed by read.xls()
## gdata: to support 'XLX' (Excel 97-2004) files.
##
## gdata: Unable to load perl libaries needed by read.xls()
## gdata: to support 'XLSX' (Excel 2007+) files.
##
## gdata: Run the function 'installXLSXsupport()'
## gdata: to automatically download and install the perl
## gdata: libaries needed to support Excel XLS and XLSX formats.
##
## Attaching package: 'gdata'
##
## The following object is masked from 'package:stats':
##
##     nobs
##
## The following object is masked from 'package:utils':
##
##     object.size
##
## Loading required package: gtools
## Loading required package: mvtnorm
##
##
## NOTE: THIS PACKAGE IS NOW OBSOLETE.
##
##
##
##   The R-Genetics project has developed an set of enhanced genetics
##
##   packages to replace 'genetics'. Please visit the project homepage
##
##   at http://rgenetics.org for informtion.
##
##
##
##
## Attaching package: 'genetics'
##
## The following object is masked from 'package:GenABEL':
##
##     as.genotype
##
## The following objects are masked from 'package:base':
##
##     %in%, as.factor, order
```



Endogenous retroviral sequences (ERV)

The get.erv returns information about endogenous retroviral sequences (ERV) in an analysed region. At first, we need to obtaint the list of ERV sequences from defined region of genome using `get.erv`. In package, we provide collection of canine ERVs identified in canFam3.1 assembly but you can also supply any other database.

Let's search for ERVs on chromosome 2:

```
ervs <- get.erv(chr = "chr2", coords=c(10e6, 40e6))
```

```
## Loading required package: RSQLite
## Loading required package: DBI
```

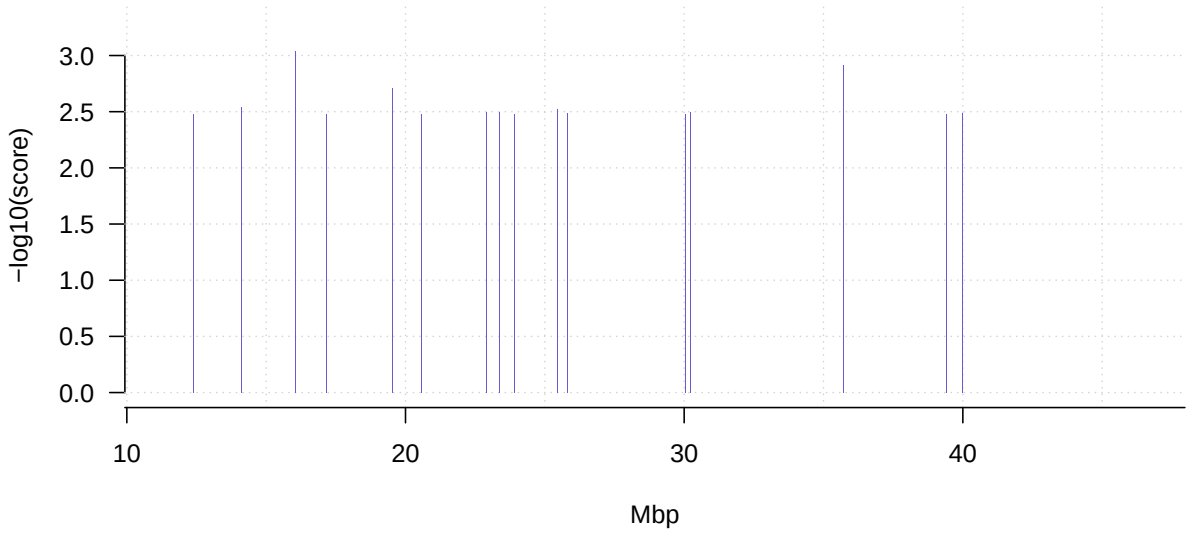
```
print(ervs)
```

```
##     id chromosome strand    start      end length score
## 1 2651       chr2      S 10015031 10006437   8594   309
## 2 2653       chr2      S 10128297 10123555   4742   847
## 3 2659       chr2      S 11537301 11525069  12232   319
## 4 2667       chr2      P 13840823 13856123  15300   324
## 5 2701       chr2      P 24590243 24597742   7499   319
## 6 2738       chr2      S 35706396 35691633  14763   318
## 7 2745       chr2      S 36488903 36477513  11390   309
## 8 2761       chr2      S 39385910 39371155  14755   303
##                                 subgenes
## 1 5LTR PBS MA CA NC Prot IN TM PPT 3LTR
## 2           5LTR PBS MA NC Prot RT 3LTR
## 3       5LTR PBS CA Prot IN TM PPT 3LTR
```

```
## 4           5LTR CA Prot RT IN PPT 3LTR
## 5      5LTR PBS CA DU RT IN TM PPT 3LTR
## 6            5LTR MA CA Prot IN TM 3LTR
## 7          5LTR MA NC RT IN TM PPT 3LTR
## 8       5LTR PBS CA Prot IN TM PPT 3LTR
```

Having returned a list of ERV sequences, we are able to plot them with `plot.erv` function using the same region:

```
plot.erv(chr = "chr10", coords = c(10e6, 40e6))
```



Converting functions