1 Introduction

This package was initiated to integrate some C/Fortran/SAS programs I have written or used over the years. As such, it would rather be a long-term project, but an immediate benefit would be something complementary to other packages currently available from CRAN, e.g. genetics, hwde,
etc. I hope eventually this will be part of a bigger effort to fulfill most of the requirements foreseen by many, e.g. Guo and Lange (2000), within the portable environment of R for data management, analysis, graphics and object-oriented programming. My view has been outlined more formally in Zhao and Tan (2006a) and Zhao and Tan (2006b) in relation to other package systems. Also reported are Zhao (2005) and Zhao (2006) on package kinship.

The number of functions are quite limited and experimental, but I already feel the enormous advantage by shifting to R and would like sooner rather than later to share my work with others. I will not claim this work as exclusively done by me, but would like to invite others to join me and enlarge the collections and improve them.

With my recent work on genomewide association studies (GWAS) especially protein GWAS, I have added many functions such as METAL_forestplot which handles data from software METAL and sentinels which extracts sentinels from GWAS summary statistics in a way that is very appealing to us compared to some other established software. At the meantime, the size of the package surpasses the limit as imposed by CRAN, thus the old good feature of S as with R that value both code and data alike has to suffer slightly in that gap.datasets and gap.examples are spun off as two separate packages; they do deserve a glimpse however for some general ideas.

2 Implementation

The following list shows the data and functions currently available.
* ANALYSIS *

AE3  AE model using nuclear family trios
bt   Bradley-Terry model for contingency table
ccs  Power and sample size for case-cohort design
cs   Credibel set
fbs  Sample size for family-based linkage and association design
gc.em Gene counting for haplotype analysis
gcontrol genomic control
gcontrol2 genomic control based on p values
gcp  Permutation tests using GENECOUNTING
gc.lambda Estionmation of the genomic control inflation statistic (lambda)
genecounting Gene counting for haplotype analysis
gif  Kinship coefficient and genetic index of familiality
grnMCMC Mixed modeling with genetic relationship matrices
gsmr Mendelian randomization analysis
hap  Haplotype reconstruction
hap.em Gene counting for haplotype analysis
hap.score Score statistics for association of traits with haplotypes
htr  Haplotype trend regression
hwe Hardy-Weinberg equilibrium test for a multiallelic marker
hwe.cc A likelihood ratio test of population Hardy-Weinberg equilibrium
hwe.hardy Hardy-Weinberg equilibrium test using MCMC
invnormal Inverse normal transformation
kin.morgan kinship matrix for simple pedigree
LD22  LD statistics for two diallelic markers
LDkl  LD statistics for two multiallelic markers
lambda1000 A standardized estimate of the genomic inflation scaling to a study of 1,000 cases and 1,000 controls
log10p log10(p) for a standard normal deviate
logp  log(p) for a normal deviate
masize Sample size calculation for mediation analysis
mia   multiple imputation analysis for hap
mtdt Transmission/disequilibrium test of a multiallelic marker
mtdt2 Transmission/disequilibrium test of a multiallelic marker by Bradley-Terry model
mvmeta Multivariate meta-analysis based on generalized least squares
pbsize Power for population-based association design
pbsize2 Power for case-control association design
pfc Probability of familial clustering of disease
pfc.sim Probability of familial clustering of disease
pgc  Preparing weight for GENECOUNTING
print.hap.score Print a hap.score object
s2k  Statistics for 2 by K table
sentinels Sentinel identification from GWAS summary statistics
tscc  Power calculation for two-stage case-control design
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>asplot</td>
<td>Regional association plot</td>
</tr>
<tr>
<td>ESplot</td>
<td>Effect-size plot</td>
</tr>
<tr>
<td>circos.cnvplot</td>
<td>Circos plot of CNVs</td>
</tr>
<tr>
<td>circos.cis.vs.trans.plot</td>
<td>Circos plot of cis/trans classification</td>
</tr>
<tr>
<td>circos.mhtplot</td>
<td>Circos Manhattan plot with gene annotation</td>
</tr>
<tr>
<td>cnvplot</td>
<td>Genomewide plot of CNVs</td>
</tr>
<tr>
<td>METAL_forestplot</td>
<td>Forest plot as R/meta’s forest for METAL outputs</td>
</tr>
<tr>
<td>makeRLEplot</td>
<td>Make relative log expression plot</td>
</tr>
<tr>
<td>mhtplot</td>
<td>Manhattan plot</td>
</tr>
<tr>
<td>mhtplot2</td>
<td>Manhattan plot with annotations</td>
</tr>
<tr>
<td>mhtplot2d</td>
<td>2D Manhattan plot</td>
</tr>
<tr>
<td>mhtplot3d</td>
<td>3D Manhattan plot</td>
</tr>
<tr>
<td>mhtplot.trunc</td>
<td>Truncated Manhattan plot</td>
</tr>
<tr>
<td>miamiplot</td>
<td>Miami plot</td>
</tr>
<tr>
<td>pedtodot</td>
<td>Converting pedigree(s) to dot file(s)</td>
</tr>
<tr>
<td>plot.hap.score</td>
<td>Plot haplotype frequencies versus haplotype score statistics</td>
</tr>
<tr>
<td>qqfun</td>
<td>Quantile-comparison plots</td>
</tr>
<tr>
<td>qqunif</td>
<td>Q-Q plot for uniformly distributed random variable</td>
</tr>
</tbody>
</table>


** UTILITIES **

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNP</td>
<td>Functions for single nucleotide polymorphisms (SNPs)</td>
</tr>
<tr>
<td>BFDP</td>
<td>Bayesian false-discovery probability</td>
</tr>
<tr>
<td>FPRP</td>
<td>False-positive report probability</td>
</tr>
<tr>
<td>ab</td>
<td>Test/Power calculation for mediating effect</td>
</tr>
<tr>
<td>b2r</td>
<td>Obtain correlation coefficients and their variance-covariances</td>
</tr>
<tr>
<td>chow.test</td>
<td>Chow’s test for heterogeneity in two regressions</td>
</tr>
<tr>
<td>chr_pos_a1_a2</td>
<td>Form SNPID from chromosome, position and alleles</td>
</tr>
<tr>
<td>cis.vs.trans.classification</td>
<td>a cis/trans classifier</td>
</tr>
<tr>
<td>comp.score</td>
<td>score statistics for testing genetic linkage of quantitative trait</td>
</tr>
<tr>
<td>GRM functions</td>
<td>ReadGRM, ReadGRMBin, ReadGRMPLINK, ReadGRMPCA, WriteGRM, WriteGRMBin, WriteGRMSAS</td>
</tr>
<tr>
<td>heritability functions</td>
<td>h2G, VR, h2GC, h2l give point estimates as with their variances</td>
</tr>
<tr>
<td>h2</td>
<td>Heritability estimation according to twin correlations</td>
</tr>
<tr>
<td>klem</td>
<td>Haplotype frequency estimation based on a genotype table of two multiallelic markers</td>
</tr>
<tr>
<td>makeped</td>
<td>A function to prepare pedigrees in post-MAKEPED format</td>
</tr>
<tr>
<td>metap</td>
<td>Meta-analysis of p values</td>
</tr>
<tr>
<td>metareg</td>
<td>Fixed and random effects model for meta-analysis</td>
</tr>
<tr>
<td>muvar</td>
<td>Means and variances under 1- and 2- locus (diallelic) QTL model</td>
</tr>
<tr>
<td>read.ms.output</td>
<td>A utility function to read ms output</td>
</tr>
<tr>
<td>snptest_sample</td>
<td>A utility to generate SNPTEST sample file</td>
</tr>
<tr>
<td>twinan90</td>
<td>Classic twin models</td>
</tr>
<tr>
<td>whscore</td>
<td>Whittemore-Halpern scores for allele-sharing</td>
</tr>
<tr>
<td>weighted.median</td>
<td>Weighted median with interpolation</td>
</tr>
</tbody>
</table>

Assuming proper installation, you will be able to obtain the list by typing `library(help=gap)` or view the list within a web browser via `help.start()`. See Appendix on how to obtain a full list of functions.

A PDF version of this file can be viewed with command `vignette("gap",package="gap")`.

You can cut and paste examples at end of each function’s documentation.

Both `genecounting` and `hap` are able to handle SNPs and multiallelic markers, with the former being flexible enough to include features such as X-linked data and the latter being able to handle large number of SNPs. But they are unable to recode allele labels automatically, so functions `gc.em` and `hap.em`
are in *haplo.em* format and used by a modified function *hap.score* in association testing.

It is notable that multilocus data are handled differently from that in *hwde* and elegant definitions of basic genetic data can be found in the *genetics* package.

Incidentally, I found my C mixed-radixed sorting routine as in Zhao and Sham (2003) is much faster than R’s internal function.

With exceptions such as function *pfc* which is very computer-intensive, most functions in the package can easily be adapted for analysis of large datasets involving either SNPs or multiallelic markers. Some are utility functions, e.g. *muvar* and *whscore*, which will be part of the other analysis routines in the future.

The benefit with R compared to standalone programs is that for users, all functions have unified format. For developers, it is able to incorporate their C/C++ programs more easily and avoid repetitive work such as preparing own routines for matrix algebra and linear models. Further advantage can be taken from packages in *Bioconductor*, which are designed and written to deal with large number of genes.

### 3 Independent programs

To facilitate comparisons and individual preferences, The source codes for 2LD, EHPPLUS, GENECOUNTING, HAP, now hosted at GitHub, have enjoyed great popularity ahead of the genomewide association studies (GWAS) therefore are likely to be more familiar than their R counterparts in *gap*. However, you need to follow their instructions to compile for a particular computer system.

I have included ms code (which is required by *read.ms.output*) and .xls files to accompany *read.ms.output* and *FPRP* and *BFDP* functions as with a classic twin example for ACE model in *OpenMx*. The package is now available from CRAN.

For these models it is actually simpler to use facilities as in package *mets*, which I now suggest.

A final category is *twinan90*, which is now dropped from the package function list due to difficulty to keep up with the requirements by the R environment but nevertheless you will still be able to compile and use otherwise.
4 Demos

You can also try several simple examples via demo:

```r
library(gap)
demo(gap)
```

5 Examples

I would like to highlight `pedtodot pbsize`, `fbsize` and `ccsize` functions used for pedigree drawing and power/sample calculations in a genome-wide association study as reported in Zhao (2007).

5.1 Pedigree drawing

I have included the original file for the *R News* as well as put examples in separate vignettes. They can be accessed via `vignette("rnews",package="gap.examples")` and `vignette("pedtodot", package="gap.examples")`, respectively.

5.2 Kinship calculation

Next, I will provide an example for kinship calculation using `kin.morgan`. It is recommended that individuals in a pedigree are ordered so that parents always precede their children. In this regard, package `pedigree` can be used, and package `kinship2` can be used to produce pedigree diagram as with kinship matrix.

**Pedigree diagram**

```r
> library(gap)
> # pedigree diagram
> data(lukas)
> library(kinship2)
> ped <- with(lukas,pedigree(id,father,mother,sex))
> pdf("figures/lukas.pdf",height=14,width=15)
> plot(ped)
> dev.off()

null device
 1
```

The pedigree diagram is as follows,
Kinship calculation

We then turn to the kinship calculation.

```r
> # unordered individuals
> library(gap)
> gk1 <- kin.morgan(lukas)
> write.table(gk1$kin.matrix, "results/gap_1.txt", quote=FALSE)
> library(kinship2)
> kk1 <- kinship(lukas[,1], lukas[,2], lukas[,3])
> write.table(kk1, "results/kinship_1.txt", quote=FALSE)
> d <- gk1$kin.matrix-kk1
> sum(abs(d))
```

```
[1] 2.443634
```
# order individuals so that parents precede their children
> library(pedigree)
> op <- orderPed(lukas)
> olukas <- lukas[order(op),]
> gk2 <- kin.morgan(olukas)
> write.table(olukas,"olukas.csv",quote=FALSE)
> write.table(gk2$kin.matrix,"results/gap_2.txt",quote=FALSE)
> kk2 <- kinship(olukas[,1],olukas[,2],olukas[,3])
> write.table(kk2,"results/kinship_2.txt",quote=FALSE)
> z <- gk2$kin.matrix-kk2
> sum(abs(z))

[1] 0

We see that in the second case, the result agrees with \texttt{kinship2}.

5.3 Study design

Family-based design

The example involving family-based design is as follows,

> library(gap)
> models <- matrix(c(  
+ 4.0, 0.01,  
+ 4.0, 0.10,  
+ 4.0, 0.50,  
+ 4.0, 0.80,  
+ 2.0, 0.01,  
+ 2.0, 0.10,  
+ 2.0, 0.50,  
+ 2.0, 0.80,  
+ 1.5, 0.01,  
+ 1.5, 0.10,  
+ 1.5, 0.50,  
+ 1.5, 0.80), ncol=2, byrow=TRUE)
> outfile <- "fbsize.txt"
> cat("gamma","p","Y","N_{asp","P_{A","H1","N_{tdt","H2","N_{asp/tdt","L_{o","L_{s}\n",  
+ file=outfile,sep="\t")
> for(i in 1:12) {
+ g <- models[i,1]
+ p <- models[i,2]
+ z <- fbsize(g,p)
+ cat(z$gamma,z$p,z$Y,z$n1,z$pA,z$h1,z$n2,z$h2,z$n3,z$lambdao,z$lambdaas,  
+ file=outfile,append=TRUE,sep="\t")


Alzheimer's:

```r
fbsize(g,p)
```

```r
$gamma
[1] 4.5

$p
[1] 0.15

$y
[1] 0.6256916

$n1
[1] 162.6246

$pA
[1] 0.8181818

$h1
[1] 0.4598361

$n2
[1] 108.994

$h2
[1] 0.6207625

$n3
[1] 39.97688
```
Population-based design

The example involving population-based design is as follows,

```r
library(gap)
kp <- c(0.01, 0.05, 0.10, 0.2)
models <- matrix(c(
  4.0, 0.01,
  4.0, 0.10,
  4.0, 0.50,
  4.0, 0.80,
  2.0, 0.01,
  2.0, 0.10,
  2.0, 0.50,
  2.0, 0.80,
  1.5, 0.01,
  1.5, 0.10,
  1.5, 0.50,
  1.5, 0.80), ncol=2, byrow=TRUE)
 outfile <- "pbsize.txt"
 cat("gamma","p","Y","N_asp","P_A","H1","N_tdt","H2","N_asp.tdt","L_o","L_s",sep="\t",file=outfile)
 for(i in 1:dim(models)[1])
 11
```
for(k in kp) n <- c(n,ceiling(pbsize(k,g,p)))
+  cat(models[i,1:2],n,sep="\t",file=outfile,append=TRUE)
+  cat("\n",file=outfile,append=TRUE)
+ }
> table5 <- read.table(outfile,header=TRUE,sep="\t")
> table5

Case-cohort design

For case-cohort design, we obtain results for ARIC and EPIC studies.

> library(gap)
> # ARIC study
> outfile <- "aric.txt"
> n <- 15792
> pD <- 0.03
> p1 <- 0.25
> alpha <- 0.05
> theta <- c(1.35,1.40,1.45)
> beta1 <- 0.8
> s_nb <- c(1463,722,468)
> cat("n","pD","p1","hr","q","power","ssize\n",file=outfile,sep="\t")
> for(i in 1:3)
+  { q <- s_nb[i]/n
+    power <- ccsie(n,q,pD,p1,alpha,log(theta[i]))
+    ssize <- ccsie(n,q,pD,p1,alpha,log(theta[i]),beta1)
```r
+ cat(n, "\t", pD, "\t", p1, "\t", theta[i], "\t", q, "\t",
+ signif(power, 3), "\t", ssize, "\n",
+ file=outfile, append=TRUE)
+ }
>
+ read.table(outfile, header=TRUE, sep="\t")

    n  pD  p1  hr  q  power  ssize
1 15792 0.03 0.25 1.35 0.09264184 0.8 1463
2 15792 0.03 0.25 1.40 0.04571935 0.8 722
3 15792 0.03 0.25 1.45 0.02963526 0.8 468

> unlink(outfile)

> # EPIC study
> outfile <- "epic.txt"
> n <- 25000
> alpha <- 0.00000005
> power <- 0.8
> s_pD <- c(0.3, 0.2, 0.1, 0.05)
> s_p1 <- seq(0.1, 0.5, by=0.1)
> s_hr <- seq(1.1, 1.4, by=0.1)
> cat("n","pD","p1","hr","alpha","ssize\n",file=outfile,sep="\t")
>
> # direct calculation
> for(pD in s_pD)
+ {
+   for(p1 in s_p1)
+     {
+       for(hr in s_hr)
+         {
+           ssize <- ccsize(n, q, pD, p1, alpha, log(hr), power)
+           if (ssize>0) cat(n, "\t", pD, "\t", p1, "\t", hr, "\t", alpha, "\t",
+           ssize, "\n",
+           file=outfile, append=TRUE)
+         }
+     }
+   }
>
+ read.table(outfile, header=TRUE, sep="\t")

    n  pD  p1  hr  alpha  ssize
1 25000 0.3 0.1 1.3 5e-08 14391
2 25000 0.3 0.1 1.4 5e-08 5732
3 25000 0.3 0.2 1.2 5e-08 21529
4 25000 0.3 0.2 1.3 5e-08 5099
5 25000 0.3 0.2 1.4 5e-08 2613
6 25000 0.3 0.3 1.2 5e-08 11095
```
14

5.4 Graphics examples

I now include some figures from the documentation that may be of interest.

Genome-wide association

The following code is used to obtain a Q-Q plot via qqunif function,

```r
> library(gap)
> pdf("figures/qqunif.pdf", height=10, width=10)
> u_obs <- runif(1000)
> r <- qqunif(u_obs, pch=21, bg="blue", bty="n")
> u_exp <- r$y
> hits <- u_exp >= 2.30103
> points(r$x[hits], u_exp[hits], pch=21, bg="green")
> legend("topleft", sprintf("GC.lambda=%.4f", gc.lambda(u_obs)))
> dev.off()
```
Based on a chicken genome scan data, the code below generates a Manhattan plot, demonstrating the use of gaps to separate chromosomes.

```r
> library(gap)
> ord <- with(w4, order(chr, pos))
> w4 <- w4[ord,]
> pdf("figures/w4.pdf", height=9, width=10)
> oldpar <- par()
> par(cex=0.6)
> colors <- c(rep(c("blue", "red"), 15), "red")
> mhtplot(w4, control=mht.control(colors=colors, gap=1000), pch=19, srt=0)
> axis(2, cex.axis=2)
> suggestiveline <- -log10(3.60036E-05)
> genomewideline <- -log10(1.8E-06)
> abline(h=suggestiveline, col="blue")
> abline(h=genomewideline, col="green")
> abline(h=0)
> dev.off()
```
The code below obtains a Manhattan plot with gene annotation,

```r
> library(gap)
> png("figures/mhtplot.png",height=10,width=16,units="cm",res=300)
> data <- with(mhtdata,cbind(chr,pos,p))
> glist <- c("IRS1","SPRY2","FTO","GRIK3","SNED1","HTR1A","MARCH3","WISP3","PPP1R3B"+
   "RP1L1","FDFT1","SLC39A14","GFRA1","MC4R")
> hdata <- subset(mhtdata,gene%in%glist)[c("chr","pos","p","gene")]
> color <- rep(c("lightgray","gray"),11)
> glen <- length(glist)
> hcolor <- rep("red",glen)
> par(las=2,xpd=TRUE,cex.axis=1.8,cex=0.4)
> ops <- mht.control(colors=color,yline=1.5,xline=3)
> hops <- hmht.control(data=hdata,colors=hcolor)
> mhtplot(data,ops,hops,pch=19)
> axis(2,pos=2,at=1:16,cex.axis=0.5)
> title("Manhattan plot with genes highlighted",cex.main=1)
> dev.off()
```
All these look familiar, so revised form of the function called `mhtplot2` was created for additional features such as centering the chromosome ticks, allowing for more sophisticated coloring schemes, using prespecified fonts, etc. Please refer to the function’s documentation example.

We could also go further with a circos Manhattan plot,

```r
> library(gap)
> library(gap.datasets)
> pdf("figures/circos-mhtplot.pdf")
> circos.mhtplot()
> dev.off()
```
We now experiment with Miami plot,

```r
> library(gap.datasets)
> mhtdata <- within(mhtdata, {pr=p})
> png("figures/mamiplot.png")
> mamiplot(mhtdata, chr="chr", bp="pos", p="p", pr="pr", snp="rsn")
> dev.off()
```
The code below obtains a regional association plot with the `asplot` function,

```r
> library(gap)
> library(gap.datasets)
> pdf("figures/asplot.pdf",height=14,width=14)
> asplot(CDKNlocus, CDKNmap, CDKNgenes, best.pval=5.4e-8, sf=c(3,6))
> title("CDKN2A/CDKN2B Region")
> dev.off()
```
The function predates the currently popular **locuszoom** software but leaves the option open for generating such plots on the fly and locally.

**Effect size plot**

The code below obtains an effect size plot via the `ESplot` function.

```r
> library(gap)
> options(stringsAsFactors=FALSE)
> testdata <- data.frame(models=c("Basic model","Adjusted",
+ "Moderately adjusted",
+ "Heavily adjusted","Other"),
+ OR = c(4.5,3.5,2.5,1.5,1),
+ SElogOR = c(0.2,0.1,0.5,0.5,0.2))
> ESplot(testdata,v=1)
```
Note that all these can serve as templates to customize features of your own.

6 Polygenic modeling

In line with the recent surge of interest in the polygenic models, a separate vignette is available through `vignette("h2", package="gap.examples")` demonstrating aspect of the models on heritability.

7 Mendelian randomization

The function `gsmr()` implements several methods and gives effect size plots.

```r
> r <- gsmr(mr, "LIF.R", c("CAD", "FEV1"))
```
8 Known bugs

Unaware of any bug. However, better memory management is expected.

9 Summary

I believe by now the package should have given you a flavour of initiatives I have made so far in relation to how the project was envisaged. More importantly, it is clear that availability of the package will serve as a platform on which future work can be accumulated and collaboration can be built.

Appendix

Assuming that you have already loaded the package via library(gap), you can use lsf.str("package:gap") and data(package="gap") to generate a list of functions and a list of datasets, respectively. If this looks odd to you, you might try search() within R to examine what is available in your environment before issuing the lsf.str command.
AE3 : function (model, random, data, seed = 1234, n.sim = 50000, verbose = TRUE)
BFDP : function (a, b, pi1, W, logscale = FALSE)
Cox.T : function (parms, case, control, k)
Cox.est : function (case, ctl, k0, initial)
DevH0dominant : function (parms, case, control, k)
DevH0dominant.est : function (case, ctl, k0, initial)
DevH0recessive : function (parms, case, control, k)
DevH0recessive.est : function (case, ctl, k0, initial)
DevHaGdominant : function (parms, case, control, k)
DevHaGdominant.est : function (case, ctl, k0, initial)
DevHaGrecessive : function (parms, case, control, k)
DevHaGrecessive.est : function (case, ctl, k0, initial)
ESplot : function (ESdat, SE = TRUE, logscale = TRUE, alpha = 0.05, xlim = c(-2, 8), v = 1, ...)
FPRP : function (a, b, pi0, ORlist, logscale = FALSE)
HapDesign : function (HaploEM)
HapFreqSE : function (HaploEM)
KCC : function (model, GRR, p1, K)
LD22 : function (h, n)
LDkl : function (n1 = 2, n2 = 2, h, n, optrho = 2, verbose = FALSE)
MCMCgrm : function (model, prior, data, GRM, eps = 0, n.thin = 10, n.burnin = 3000, n.iter = 13000, ...)
METAL_forestplot : function (tbl, all, rsid, package = "meta", ...)
PARn : function (p, RRlist)
ReadGRM : function (prefix = 51)
ReadGRMBin : function (prefix, AllN = FALSE, size = 4)
ReadGRMPCA : function (prefix)
ReadGRMPLINK : function (prefix, diag = 1)
VR : function (v1, vv1, v2, vv2, c12)
WriteGRM : function (prefix = 51, id, N, GRM)
WriteGRMBin : function (prefix, grm, N, id, size = 4)
WriteGRMSAS : function (grmlist, outfile = "gwas")
a2g : function (a1, a2)
ab : function (type = "power", n = 25000, a = 0.15, sa = 0.01, b = log(1.19), sb = 0.01, alpha = 0.05, fold = 1)
allele.recode : function (a1, a2, miss.val = NA)
asplot : function (locus, map, genes, flanking = 1000, best.pval = NULL, sf = c(4,
hap : function (id, data, nloci, loci = rep(2, nloci), names = paste("loci",
1:nloci, sep = ""), control = hap.control())

hap.control : function (mb = 0, pr = 0, po = 0.001, to = 0.001, th = 1, maxit = 100,
n = 0, ss = 0, rs = 0, rp = 0, ro = 0, rv = 0, sd = 0, mm = 0, mi = 0,
mc = 50, ds = 0.1, de = 0, q = 0, hapfile = "hap.out", assignfile = "assign.out")

hap.em : function (id, data, locus.label = NA, converge.eps = 1e-06, maxiter = 500, miss.val = 0)

hap.score : function (id, data, loci.label = NA, converge.eps = 1e-06, maxiter = 500,
miss.val = 0)

hmht.control : function (data = NULL, colors = NULL, yoffset = 0.25, cex = 1.5, boxed = FALSE)

k : function (r, N, adjust = TRUE)

lambda1000 : function (lambda, ncases, ncontrols)

log10p : function (z)

log10pvalue : function (p = NULL, base = NULL, exponent = NULL)

logit : function (p = 0.5)

m2plem : function (a1, a2)

mht.control : function (type = "p", usepos = FALSE, logscale = TRUE, base = 10, cutoffs = NULL,
colors = NULL, labels = NULL, gap = NULL, cex = 0.4, yline = 3,
xline = 3)

mhtplot : function (data, control = mht.control(), hcontrol = hmht.control(), ...)

mhtplot.trunc : function (x, chr = "CHR", bp = "BP", p = NULL, log10p = NULL, z = NULL,
snp = "SNP", col = c("gray10", "gray60"), chrlabs = NULL, suggestiveline = -log10,
genomewideline = -log10(5e-08), highlight = NULL, annotatelogoP = NULL, annotateTop = FALSE, cex.mtext = 1.5, cex.text = 0.7, mtext.line = 2,
cex.y = 1, y.ax.space = 5, y.brk1, y.brk2, delta = 0.05, ...

mhtplot2 : function (data, control = mht.control(), hcontrol = hmht.control(), ...)
mhtplot2d : function (data, plot = TRUE, cex = 0.6)
mhtplot3d : function (xyz = "INF1.merge.cis.vs.trans", cols = c("id", "chr1", "pos1", "chr2", "pos2", "gene", "target", "log10p", "x", "y", "col"), xy.scale = c(1.3e+1, 1.3e+08), marker.size = 4, log10p.max = 400, prefix = c("Sentinel", "CHR", "POS", "CHR", "POS", "Gene", "Target", "-log10(p)"), postfix = "</br>", json.file = "d3.json", pretty = TRUE)

mia : function (hapfile = "hap.out", assfile = "assign.out", miafile = "mia.out", so = 0, ns = 0, mi = 0, allsnps = 0, ssas = 0)
miamiplot : function (x, chr = "CHR", bp = "BP", p = "P", pr = "PR", snp = "SNP", col = "chartreuse4"), col2 = c("royalblue1", "seagreen1"), ymax = NULL, highlight = NULL, highlight.add = NULL, pch = 19, cex = 0.75, col = c("red", "black"), lwd = c(5, 2), lty = c(1, 2), main = "", ...)
micombine : function (est, std.err, confidence = 0.95)
mr.boot : function (bXG, sebXG, bYG, sebYG, w, n.boot = 1000, method = "median")
mtdt : function (x, n.sim = 0)
mtdt2 : function (x, verbose = TRUE, n.sim = NULL, ...)
muvar : function (n.loci = 1, y1 = c(0, 1, 1), y12 = c(1, 1, 1, 1, 1, 0, 0, 0, 0), p1 = 0.99, p2 = 0.9)
mvmeta : function (b, V)
pbsize : function (kp, gamma = 4.5, p = 0.15, alpha = 5e-08, beta = 0.2)
pbsize2 : function (N, p = 0.5, gamma = 4.5, p = 0.15, kp = 0.1, model = "additive")
pedtodot : function (pedfile, makeped = FALSE, sink = TRUE, page = "B5", url = "http://www.mrc-epid.cam.ac.uk", height = 0.5, width = 0.75, rotate = 0, dir = "none")
pfc : function (famdata, enum = 0)
pfc.sim : function (famdata, n.sim = 1e+06, n.loop = 1)
pgc : function (data, handle.miss = 1, is.genotype = 0, with.id = 0)
plem2m : function (a)
plot.hap.score : function (x, ...) print.hap.score : function (x, ...) pvalue : function (z, decimals = 2)
qqfun : function (x, distribution = "norm", ylab = deparse(substitute(x)), xlab = paste(distribution, "quantiles"), main = "NULL", las = par("las"), envelope = 0.95, labels = FALSE, col = palette()[4], lcol = palette()[2], xlim = NULL, ylim = NULL, lwd = 1, lch = 1, bg = palette()[4], cex = 0.4, line = c("quartiles", "robust", "none"), ...)
qqunif : function (u, type = "unif", logscale = TRUE, base = 10, col = palette()[4], lcol = palette()[2], ci = FALSE, alpha = 0.05, ...) read.ms.output : function (msout, is.file = TRUE, xpose = TRUE, verbose = TRUE, outfileonly = FALSE)
revhlap : function (loci, hapid)
revhlap.i : function (loci, hapid)
s2k : function (y1, y2)
se.exp : function (p, se.p)
se.invlogit : function (logit.p, se.logit)
sentinels : function (p, pid, st, debug = FALSE, flanking = 1e+06, chr = "Chrom", pos = "End", b = "Effect", se = "StdErr", log_p = NULL, snp = "MarkerName", sep = ",")
snp.ES : function (beta, SE, N)
snp.HWE : function (g)
snp.PAR : function (RR, MAF, unit = 2)
snptest_sample : function (data, sample_file = "snptest.sample", ID_1 = "ID_1", ID_2 = "ID_2", missing = "missing", C = NULL, D = NULL, P = NULL)
solve_skol : function (rootfun, target, lo, hi, e)
toETDT : function (a)
tscc : function (model, GRR, p1, n1, n2, M, alpha.genome, pi.samples, pi.markers, K)
ungcode : function (g)
weighted.median : function (x, w)
whscore : function (allele, type)
x2 : function (p1, p2, n1, n2)
z : function (p1, p2, n1, n2, r)
10 Bibliographic note

The main references are Chow (1960); Guo and Thompson (1992); Williams et al. (1992); Gholamie and Thomas (1994); Hartung et al. (2008); Risch and Merikangas (1996); Spielman and Ewens (1996); Risch and Merikangas (1997); Miller (1997); Sham (1997); Elston (1975); Sham (1998); Devlin and Roeder (1999); Zhao et al. (1999); Guo and Lange (2000); Hirotu et al. (2001); Zhao et al. (2002); Zaykin et al. (2002); Zhao (2004); Wacholder et al. (2004); Wang (2005); Skol et al. (2006); Wakefield (2007).

References


K. Gholamie and A. Thomas. A linear time algorithm for calculation of mul-


