A Genetic Analysis Package with R

Jing Hua Zhao

Department of Public Health and Primary Care, University of Cambridge, Cambridge, UK
https://jinghuazhao.github.io/

Contents

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
ccsize	Power and sample size for case-cohort design
cs	Credibel set
fbsize	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
grmMCMC	Mixed modeling with genetic relationship matrices
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

* GRAPHICS *

asplot	Regional association plot
ESplot	Effect-size plot
circos.cnvplot	circos plot of CNVs
circos.cis.vs.trans.plot	circos plot of cis/trans classification
circos.mhtplot	circos Manhattan plot with gene annotation
cnvplot	genomewide plot of CNVs
METAL_forestplot	forest plot as R/meta's forest for METAL outputs
makeRLEplot	make relative log expression plot
mhtplot	Manhattan plot
mhtplot2	Manhattan plot with annotations
mhtplot2d	2D Manhattan plot
mhtplot3d	3D Manhattan plot
mhtplot.trunc	truncated Manhattan plot
miamiplot	Miami plot
pedtodot	Converting pedigree(s) to dot file(s)
plot.hap.score	Plot haplotype frequencies versus haplotype score statistics
qqfun	Quantile-comparison plots
qqunif	Q-Q plot for uniformly distributed random variable

* UTILITIES *

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

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 be flexible enough to include features such as X-linked data and the later 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 asso-

ciation 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, EHPLUS, 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 couunterparts 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*:

```
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 asssociatoin 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

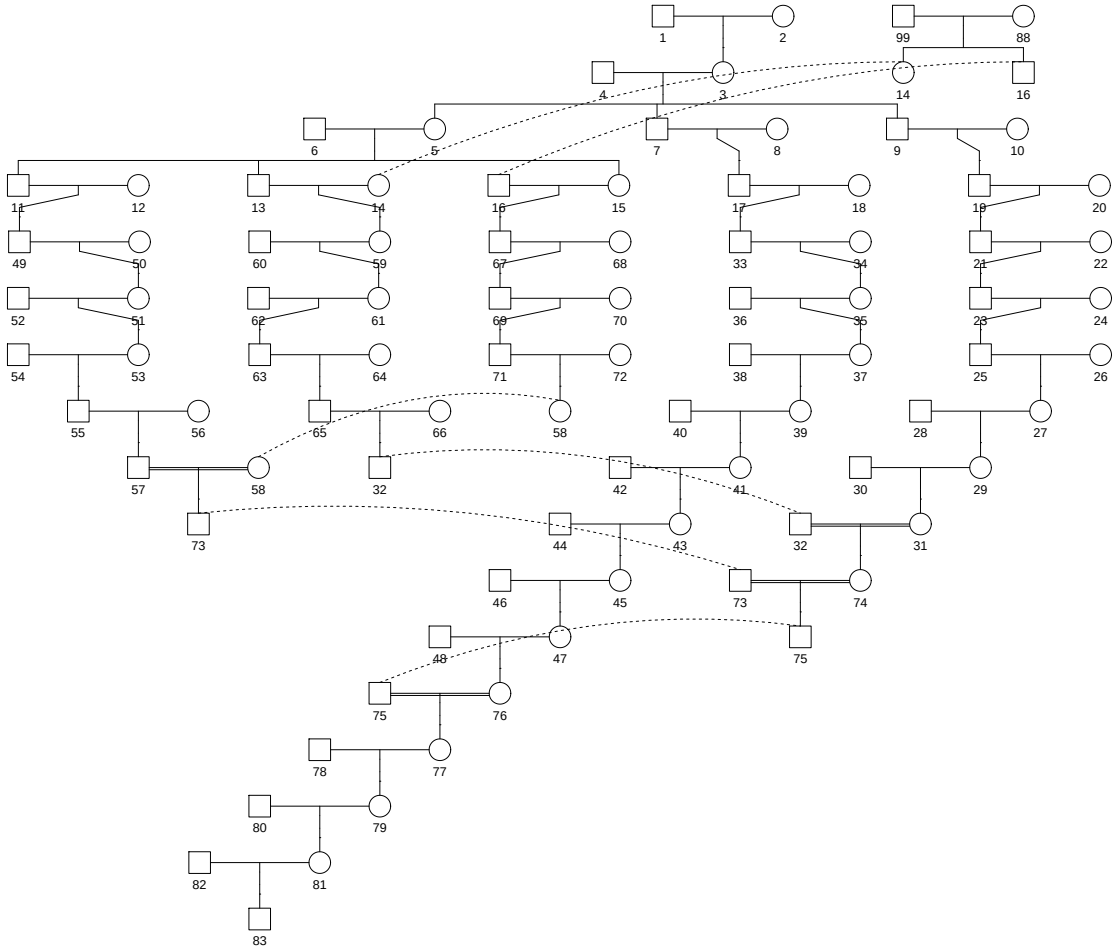
Next, I will provide an example for kinship calculation using *kin.morgan*. It is recommended that individuals in a pedigree are ordered so that parents always precede their children. In this regard, package **pedigree** can be used, and package **kinship2** can be used to produce pedigree diagram as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

Pedigree diagram

```
> 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.

```
> # 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 **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$lambdas,
+         file=outfile,append=TRUE,sep="\t")
```

```
+     cat("\n",file=outfile,append=TRUE)
+ }
> table1 <- read.table(outfile,header=TRUE,sep="\t")
> nc <- c(4,7,9)
> table1[,nc] <- ceiling(table1[,nc])
> dc <- c(3,5,6,8,10,11)
> table1[,dc] <- round(table1[,dc],2)
> unlink(outfile)
> # APOE-4, Scott WK, Pericak-Vance, MA & Haines JL
> # Genetic analysis of complex diseases 1327
> g <- 4.5
> p <- 0.15
> cat("\nAlzheimer's:\n\n")

Alzheimer's:

> fbsize(g,p)

$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
```

```
$lambdao
[1] 1.671594

$lambdas
[1] 1.784353
```

```
> table1
```

	gamma	p	Y	N_asp	P_A	H1	N_tdt	H2	N_asp.tdt	L_o	L_s
1	4.0	0.01	0.52	6402	0.80	0.05	1201	0.11	257	1.08	1.09
2	4.0	0.10	0.60	277	0.80	0.35	165	0.54	53	1.48	1.54
3	4.0	0.50	0.58	446	0.80	0.50	113	0.42	67	1.36	1.39
4	4.0	0.80	0.53	3024	0.80	0.24	244	0.16	177	1.12	1.13
5	2.0	0.01	0.50	445964	0.67	0.03	6371	0.04	2155	1.01	1.01
6	2.0	0.10	0.52	8087	0.67	0.25	761	0.32	290	1.07	1.08
7	2.0	0.50	0.53	3753	0.67	0.50	373	0.47	197	1.11	1.11
8	2.0	0.80	0.51	17909	0.67	0.27	701	0.22	431	1.05	1.05
9	1.5	0.01	0.50	6944779	0.60	0.02	21138	0.03	8508	1.00	1.00
10	1.5	0.10	0.51	101926	0.60	0.21	2427	0.25	1030	1.02	1.02
11	1.5	0.50	0.51	27048	0.60	0.50	1039	0.49	530	1.04	1.04
12	1.5	0.80	0.51	101926	0.60	0.29	1820	0.25	1030	1.02	1.02

Population-based design

The example involving population-based design is as follows,

```
> 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","p1","p5","p10","p20\n",sep="\t",file=outfile)
> for(i in 1:dim(models)[1])
```

```
+ {
+    g <- models[i,1]
+    p <- models[i,2]
+    n <- vector()
+    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

   gamma    p      p1     p5    p10   p20
1    4.0 0.01   46681   8959   4244  1887
2    4.0 0.10    8180   1570    744   331
3    4.0 0.50   10891   2091    991   441
4    4.0 0.80   31473   6041   2862  1272
5    2.0 0.01  403970  77530  36725 16323
6    2.0 0.10   52709  10116   4792  2130
7    2.0 0.50   35285   6772   3208  1426
8    2.0 0.80   79391  15237   7218  3208
9    1.5 0.01 1599920 307056 145448 64644
10   1.5 0.10  192105  36869  17465  7762
11   1.5 0.50   98013  18811   8911  3961
12   1.5 0.80  192105  36869  17465  7762
```

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 <- ccsize(n,q,pD,p1,alpha,log(theta[i]))
+   ssize <- ccsize(n,q,pD,p1,alpha,log(theta[i]),beta1)
```

```
+   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
```

```
7  25000 0.3 0.3 1.3 5e-08  3490
8  25000 0.3 0.3 1.4 5e-08  1882
9  25000 0.3 0.4 1.2 5e-08  8596
10 25000 0.3 0.4 1.3 5e-08  2934
11 25000 0.3 0.4 1.4 5e-08  1611
12 25000 0.3 0.5 1.2 5e-08  7995
13 25000 0.3 0.5 1.3 5e-08  2786
14 25000 0.3 0.5 1.4 5e-08  1538
15 25000 0.2 0.1 1.4 5e-08  9277
16 25000 0.2 0.2 1.3 5e-08  7725
17 25000 0.2 0.2 1.4 5e-08  3164
18 25000 0.2 0.3 1.3 5e-08  4548
19 25000 0.2 0.3 1.4 5e-08  2152
20 25000 0.2 0.4 1.2 5e-08 20131
21 25000 0.2 0.4 1.3 5e-08  3648
22 25000 0.2 0.4 1.4 5e-08  1805
23 25000 0.2 0.5 1.2 5e-08 17120
24 25000 0.2 0.5 1.3 5e-08  3422
25 25000 0.2 0.5 1.4 5e-08  1713
26 25000 0.1 0.2 1.4 5e-08  8615
27 25000 0.1 0.3 1.4 5e-08  3776
28 25000 0.1 0.4 1.3 5e-08 13479
29 25000 0.1 0.4 1.4 5e-08  2824
30 25000 0.1 0.5 1.3 5e-08 10837
31 25000 0.1 0.5 1.4 5e-08  2606
```

```
> unlink(outfile)
```

5.4 Graphics examples

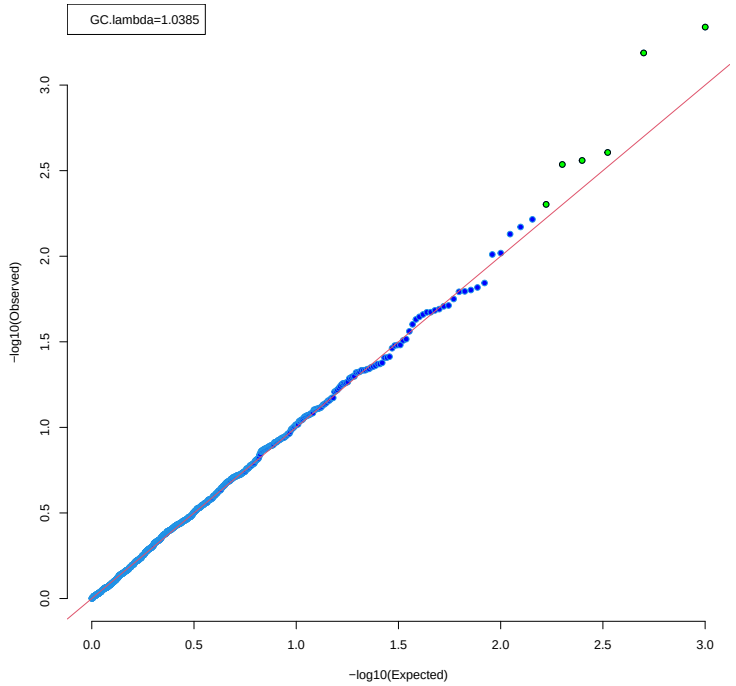
I now include some figures from the documentation that may be of interest.

Genome-wide association

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

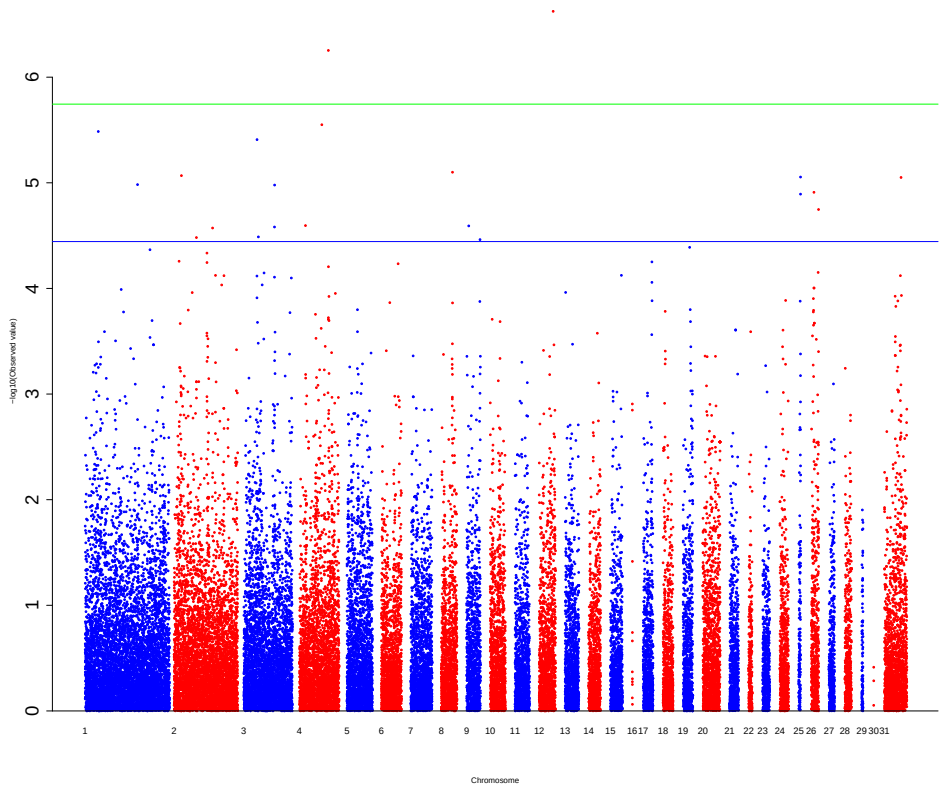
```
> library(gap)
> pdf("figures/qqunif.pdf",height=10,width=10)
> u_obs <- runif(1000)
> r <- qqunif(u_obs,pch=21,bg="blue",bty="n")
> u_exp <- r$y
> hits <- u_exp >= 2.30103
> points(r$x[hits],u_exp[hits],pch=21,bg="green")
> legend("topleft",sprintf("GC.lambda=%.4f",gc.lambda(u_obs)))
> dev.off()
```

```
null device
          1
```



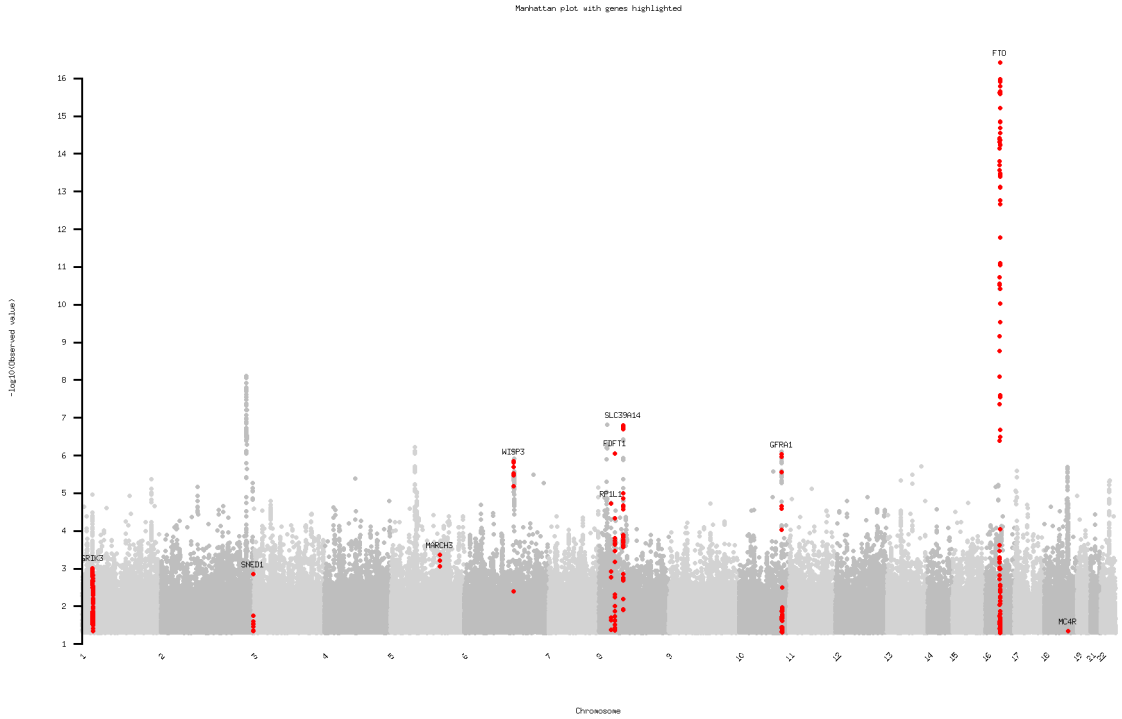
Based on a chicken genome scan data, the code below generates a Manhattan plot, demonstrating the use of gaps to separate chromosomes.

```
> 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()
```



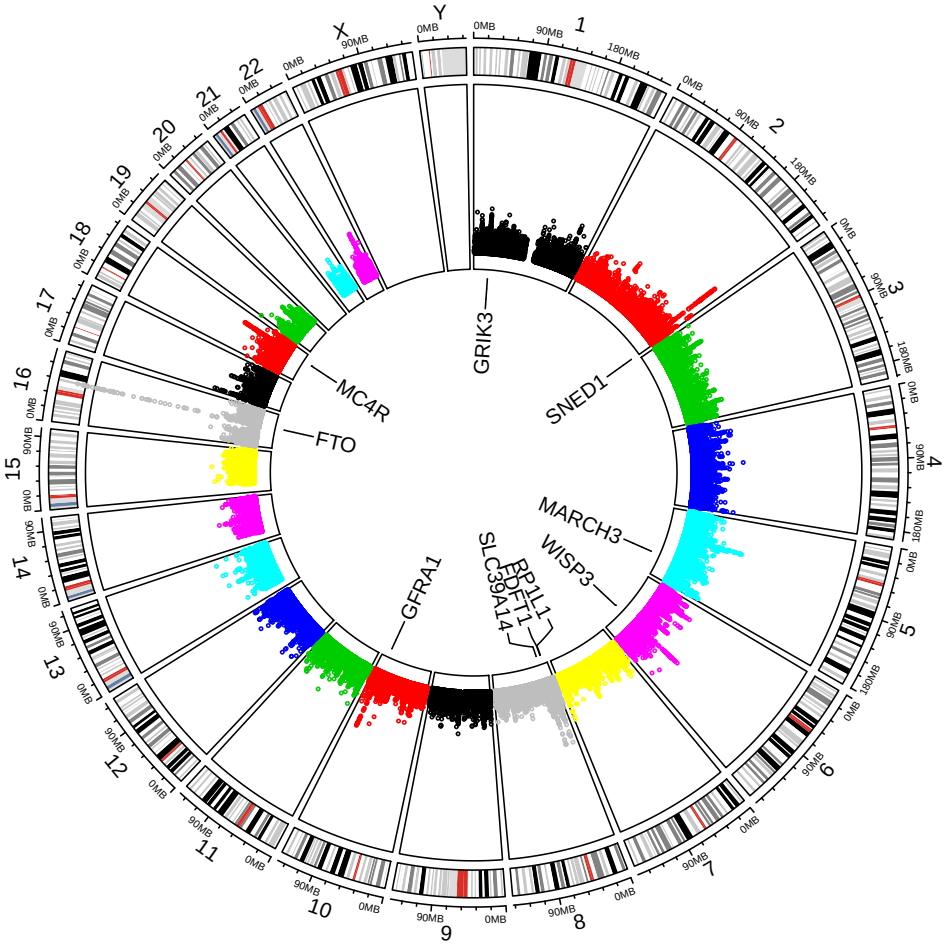
The code below obtains a Manhattan plot with gene annotation,

```
> 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()
```



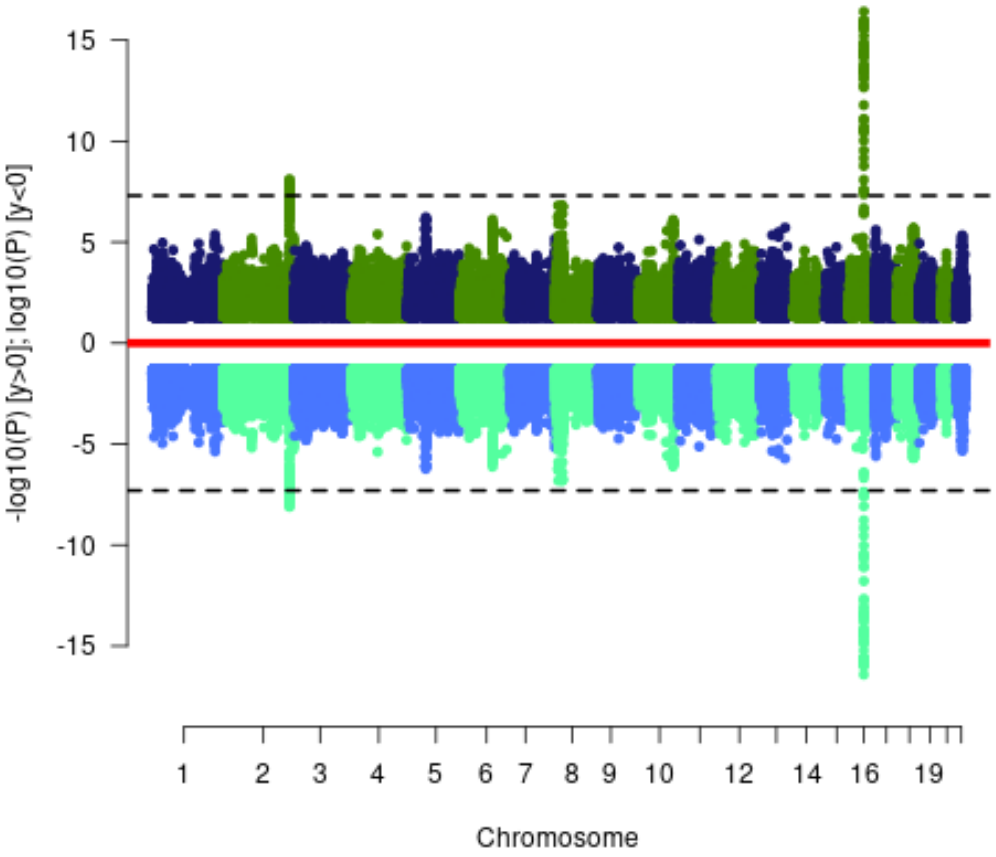
All these look familiar, so revised form of the function called **mhtplot2** was created for additional features such as centering the chromosome ticks, allowing for more sophisticated coloring schemes, using prespecified fonts, etc. Please refer to the function's documentation example.
We could also go further with a circos Manhattan plot,

```
> library(gap)
> library(gap.datasets)
> pdf("figures/circos-mhtplot.pdf")
> circos.mhtplot()
> dev.off()
```



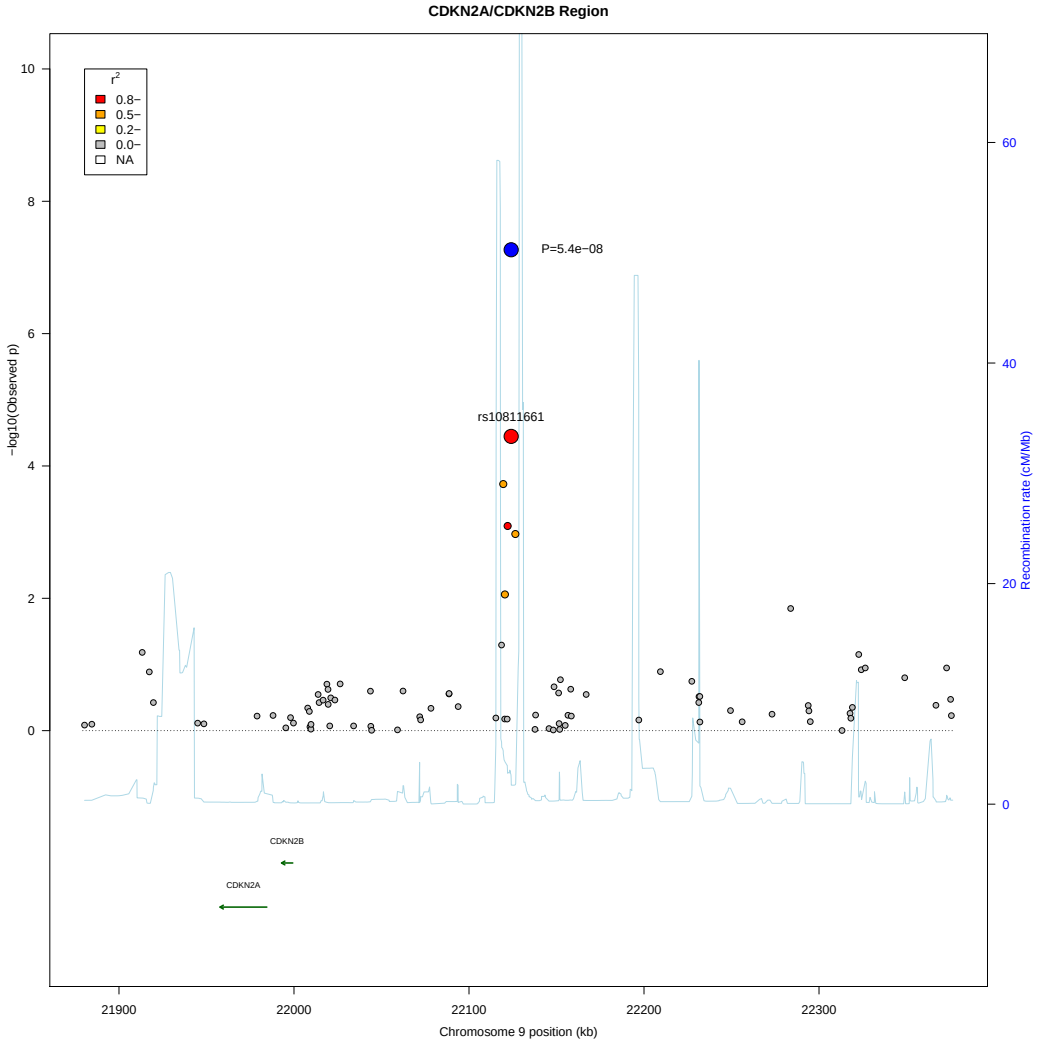
We now experiment with Miami plot,

```
> library(gap.datasets)
> mhtdata <- within(mhtdata,{pr=p})
> png("figures/miamiplot.png")
> miamiplot(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,

```
> 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.

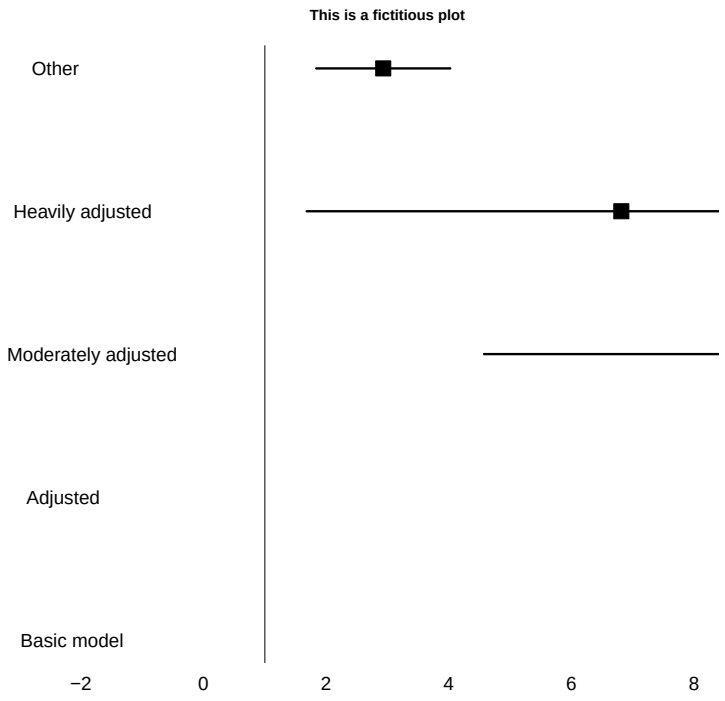
Effect size plot

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

```
> library(gap)
> pdf("figures/ESplot.pdf",height=10,width=10)
> 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)
> title("This is a fictitious plot")
> dev.off()

null device
          1
```



Note that all these can serve as templates to customize features of your own.

6 Polygenic modeling

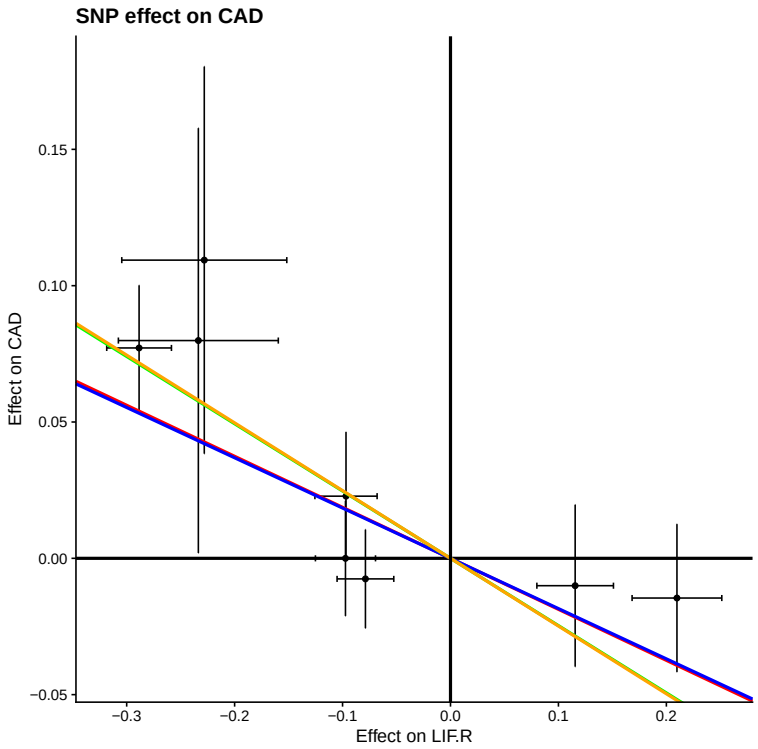
In line with the recent surge of interest in the polygenic models, a separate vignette is available through `vignette("h2",package="gap.examples")` demonstrating aspect of the models on heritability.

7 Mendelian randomization

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

```
> library(cowplot)
> library(ggplot2)
```

```
> library(gap)
> 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, respectvely. 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.

```
 [1] ".GlobalEnv"        "package:ggplot2"   "package:cowplot"
 [4] "package:pedigree"  "package:reshape"   "package:HaploSim"
 [7] "package:kinship2"  "package:quadprog"  "package:Matrix"
[10] "package:gap"       "package:stats"     "package:graphics"
[13] "package:grDevices" "package:utils"     "package:datasets"
[16] "package:methods"   "Autoloads"         "package:base"
```

```
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,
    4), logpmax = 10, pch = 21)
b2r : function (b, s, rho, n)
bt : function (x)
ccsize : function (n, q, pD, p1, alpha, theta, power = NULL, verbose = FALSE)
chow.test : function (y1, x1, y2, x2, x = NULL)
chr_pos_a1_a2 : function (chr, pos, a1, a2, prefix = "chr", seps = c(":", "_", "_"),
circos.cis.vs.trans.plot : function (hits, panel, id, radius = 1e+06)
circos.cnvplot : function (data)
circos.mhtplot : function (data, glist)
cis.vs.trans.classification : function (hits, panel, id, radius = 1e+06)
cnvplot : function (data)
comp.score : function (ibddata = "ibd_dist.out", phenotype = "pheno.dat", mean = 0,
    var = 1, h2 = 0.3)
cov.invlogit : function (logit.p1, logit.p2, cov.logit)
cs : function (tbl, b = "Effect", se = "StdErr", log_p = NULL, cutoff = 0.95)
fbsize : function (gamma, p, alpha = c(1e-04, 1e-08, 1e-08), beta = 0.2, debug = 0,
    error = 0)
g2a : function (g)
g2a.c : function (g)
gc.control : function (xdata = FALSE, convll = 1, handle.miss = 0, eps = 1e-06, tol
    maxit = 50, pl = 0.001, assignment = "assign.dat", verbose = T)
gc.em : function (data, locus.label = NA, converge.eps = 1e-06, maxiter = 500,
    handle.miss = 0, miss.val = 0, control = gc.control())
gc.lambda : function (p)
gcode : function (a1, a2)
gcontrol : function (data, zeta = 1000, kappa = 4, tau2 = 1, epsilon = 0.01, ngib =
    burn = 50, idum = 2348)
gcontrol2 : function (p, col = palette()[4], lcol = palette()[2], ...)
gcp : function (y, cc, g, handle.miss = 1, miss.val = 0, n.sim = 0, locus.label = NU
    quietly = FALSE)
genecounting : function (data, weight = NULL, loci = NULL, control = gc.control())
geno.recode : function (geno, miss.val = 0)
getPTE : function (b1, b2, rho, sdx1 = 1, sdx2 = 1)
getb1star : function (b1, b2, rho, sdx1 = 1, sdx2 = 1)
gif : function (data, gifset)
grec2g : function (h, n, t)
gsmr : function (data, X, Y, alpha = 0.05, other_plots = FALSE)
h2 : function (mzDat = NULL, dzDat = NULL, rmz = NULL, rdz = NULL, nmz = NULL,
    ndz = NULL, selV = NULL)
h2.jags : function (y, x, G, eps = 1e-04, sigma.p = 0, sigma.r = 1, parms = c("b",
    "p", "r", "h2"), ...)
h2G : function (V, VCOV, verbose = TRUE)
```

```
h2GE : function (V, VCOV, verbose = TRUE)
h2l : function (K = 0.05, P = 0.5, h2, se, verbose = TRUE)
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 (y, geno, trait.type = "gaussian", offset = NA, x.adj = NA, ski
    locus.label = NA, miss.val = 0, n.sim = 0, method = "gc", id = NA,
    handle.miss = 0, mloci = NA, sexid = NA)
hmht.control : function (data = NULL, colors = NULL, yoffset = 0.25, cex = 1.5, boxe
htr : function (y, x, n.sim = 0)
hwe : function (data, data.type = "allele", yates.correct = FALSE, miss.val = 0)
hwe.cc : function (model, case, ctrl, k0, initial1, initial2)
hwe.hardy : function (a, alleles = 3, seed = 3000, sample = c(1000, 1000, 5000))
hwe.jags : function (k, n, delta = rep(1/k, k), lambda = 0, lambdamu = -1, lambdasd
    parms = c("p", "f", "q", "theta", "lambda"), ...)
inv_chr_pos_a1_a2 : function (chr_pos_a1_a2, prefix = "chr", seps = c(":", "_", "_")
invlogit : function (x = 0)
invnormal : function (x)
k : function (r, N, adjust = TRUE)
kin.morgan : function (ped, verbose = FALSE)
klem : function (obs, k = 2, l = 2)
lambda1000 : function (lambda, ncases, ncontrols)
log10p : function (z)
logit : function (p = 0.5)
logp : function (z)
m2plem : function (a1, a2)
makeRLEplot : function (E, log2.data = TRUE, groups = NULL, col.group = NULL, showTi
    title = "Relative log expression (RLE) plot", ...)
makeped : function (pifile = "pedfile.pre", pofile = "pedfile.ped", auto.select = 1,
    with.loop = 0, loop.file = NA, auto.proband = 1, proband.file = NA)
masize : function (model, opts, alpha = 0.025, gamma = 0.2)
metap : function (data, N, verbose = "Y", prefixp = "p", prefixn = "n")
metareg : function (data, N, verbose = "Y", prefixb = "b", prefixse = "se")
mht.control : function (type = "p", usepos = FALSE, logscale = TRUE, base = 10, cuto
    colors = NULL, labels = NULL, srt = 45, 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 = "P", snp = "SNP", z = NULL,
    "gray60"), chrlabs = NULL, suggestiveline = -log10(1e-05), genomewideline = -log
    highlight = NULL, logp = TRUE, annotatePval = NULL, annotateTop = TRUE,
```

```
    cex.mtext = 0.6, cex.text = 0.8, mtext.line = 2, cex.y = 1, y.ax.space = 5,
    y.brk1, y.brk2, ...)
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+
    1.3e+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, sas = 0)
miamiplot : function (x, chr = "CHR", bp = "BP", p = "P", pr = "PR", snp = "SNP", co
    "chartreuse4"), col2 = c("royalblue1", "seagreen1"), ymax = NULL, highlight = NU
    highlight.add = NULL, pch = 19, cex = 0.75, cex.lab = 1, xlab = "Chromosome",
    ylab = "-log10(P) [y>0]; log10(P) [y<0]", lcols = c("red", "black"),
    lwds = c(5, 2), ltys = 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, fc = 0.5, alpha = 0.05, gamma = 4.5, p = 0.15, kp = 0.1, mode
pedtodot : function (pedfile, makeped = FALSE, sink = TRUE, page = "B5", url = "http
    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 = pa
    "quantiles"), main = NULL, las = par("las"), envelope = 0.95, labels = FALSE,
    col = palette()[4], lcol = palette()[2], xlim = NULL, ylim = NULL,
    lwd = 1, pch = 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, outf
    outfileonly = FALSE)
revhap : function (loci, hapid)
```

```
revhap.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", po
    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
    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)

      Package LibPath
 [1,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
 [2,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
 [3,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
 [4,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
 [5,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
 [6,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
 [7,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
 [8,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
 [9,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[10,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[11,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[12,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[13,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[14,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[15,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[16,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[17,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[18,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[19,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[20,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
[21,] "gap"   "/rds/user/jhz22/hpc-work/work/Rtmp6r0rdt/Rinst3c2db13bba089"
      Item            Title
```

```
 [1,] "OPGall (OPG)"  "Internal functions for gap"
 [2,] "OPGrsid (OPG)" "Internal functions for gap"
 [3,] "OPGtbl (OPG)"  "Internal functions for gap"
 [4,] "PD"            "Internal functions for gap"
 [5,] "aldh2"         "Internal functions for gap"
 [6,] "apoeapoc"      "Internal functions for gap"
 [7,] "cf"            "Internal functions for gap"
 [8,] "cnv"           "Internal functions for gap"
 [9,] "crohn"         "Internal functions for gap"
[10,] "fa"            "Internal functions for gap"
[11,] "fsnps"         "Internal functions for gap"
[12,] "hla"           "Internal functions for gap"
[13,] "inf1"          "Internal functions for gap"
[14,] "jma.cojo"      "Internal functions for gap"
[15,] "l51"           "Internal functions for gap"
[16,] "lukas"         "Internal functions for gap"
[17,] "mao"           "Internal functions for gap"
[18,] "meyer"         "Internal functions for gap"
[19,] "mfblong"       "Internal functions for gap"
[20,] "mr"            "Internal functions for gap"
[21,] "nep499"        "Internal functions for gap"
```

10 Bibliographic note

The main references are Chow (1960); Guo and Thompson (1992); Williams et al. (1992); Gholamic 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); Hirotsu 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

G. C. Chow. Tests of equality between sets of coefficients in two linear regression. *Econometrica*, 28:591–605, 1960.

B. Devlin and K. Roeder. Genomic control for association studies. *Biometrics*, 55(4):997–1004, 1999.

R. C. Elston. On the correlation between correlations. *Biometrika*, 62: 133–140, 1975.

K. Gholamic and A. Thomas. A linear time algorithm for calculation of multiple pairwise kinship coefficients and genetic index of familiality. *Comp Biomed Res*, 27:342–350, 1994.

S. W. Guo and K. Lange. Genetic mapping of complex traits: promises, problems, and prospects. *Theor Popul Biol*, 57:1–11, 2000.

S. W. Guo and E. A. Thompson. Performing the exact test of hardy-weinberg proportion for multiple alleles. *Biometrics*, 48:361–372, 1992.

J. Hartung, G. Knapp, and B. K. Sinha. *Statistical Meta-analysis with Applications*. Wiley, 2008.

C. Hirotsu, S. Aoki, T. Inada, and Y. Kitao. An exact test for the association between the disease and alleles at highly polymorphic loci with particular interest in the haplotype analysis. *Biometrics*, 57:769–778, 2001.

M. B. Miller. Genomic scanning and the transmission/disequilibrium test: analysis of error rates. *Genet Epidemiol*, 14:851–856, 1997.

N. Risch and K. Merikangas. The future of genetic studies of complex human diseases. *Science*, 273(September):1516–1517, 1996.

N. Risch and K. Merikangas. Reply to Scott el al. *Science*, 275:1329–1330., 1997.

P. C. Sham. Transmission/disequilibrium tests for multiallelic loci. *Am J Hum Genet*, 61:774–778, 1997.

P. C. Sham. *Statistics in Human Genetics*. Arnold Applications of Statistics Series. Edward Arnold, London, 1998. 11-1-1999.

A. D. Skol, L. J. Scott, G. R. Abecasis, and M. Boehnke. Joint analysis is more efficient than replication-based analysis for two-stage genome-wide association studies. *Nat Genet*, 38(2):209–13, 2006.

R. S. Spielman and W. J. Ewens. The TDT and other family-based tests for linkage disequilibrium and association. *Am J Hum Genet*, 59(5):983–9, 1996.

S. Wacholder, S. Chanock, M. Garcia-Closas, L. El Ghormli, and N. Rothman. Assessing the probability that a positive report is false: an approach for molecular epidemiology studies. *J Natl Cancer Inst*, 96(6):434–42, 2004.

J. Wakefield. A bayesian measure of the probability of false discovery in genetic epidemiology studies. *Am J Hum Genet*, 81:208–226, 2007.

K. Wang. A likelihood approach for quantitative-trait-locus mapping with selected pedigrees. *Biometrics*, 61:465–473, 2005.

C. J. Williams, J. C. Christian, and J.A. Jr. Norton. Twinan90: A fortran programfor conducting anova-based and likelihood-based analyses of twin data. *Comp Meth Prog Biomed*, 38(2-3):167–76, 1992.

D. V. Zaykin, P. H. Westfall, S. S. Young, M. A. Karnoub, M. J. Wagner, and M. G. Ehm. Testing association of statistically inferred haplotypes with discrete and continuous traits in samples of unrelated individuals. *Hum Hered*, 53(2):79–91, 2002.

J. H. Zhao. 2LD. GENECOUNTING and HAP: computer programs for linkage disequilibrium analysis. *Bioinformatics*, 20:1325–6, 2004.

J. H. Zhao. Mixed-effects Cox models of alcohol dependence in extended families. *BMC Genet*, 6(Suppl):S127, 2005.

J. H. Zhao. Pedigree-drawing with R and graphviz. *Bioinformatics*, 22(8): 1013–4, 2006.

J. H. Zhao. gap: genetic analysis package. *Journal of Statistical Software*, 23(8):1–18, 2007.

J. H. Zhao and P. C. Sham. Generic number systems and haplotype analysis. *Comp Meth Prog Biomed*, 70:1–9, 2003.

J. H. Zhao and Q. Tan. Integrated analysis of genetic data with R. *Hum Genomics*, 2(4):258–65, 2006a.

J. H. Zhao and Q. Tan. Genetic dissection of complex traits in silico: approaches, problems and soluations. *Current Bioinformatics*, 1(3):359–369, 2006b.

J. H. Zhao, P. C. Sham, and D. Curtis. A program for the Monte Carlo evaluation of significance of the extended transmission/disequilibrium test. *Am J Hum Genet*, 64:1484–1485, 1999.

J. H. Zhao, S. Lissarrague, L. Essioux, and P. C. Sham. GENECOUNTING: haplotype analysis with missing genotypes. *Bioinformatics*, 18(12):1694–5, 2002.