Brief Introduction to R package qgg using 1000G data

Palle Duun Rohde, Izel Fourie Sørensen, & Peter Sørensen

2022-10-16

Contents

1 Introduction

The practical is based on the R package `qgg` (Rohde et al. (2021, 2022)). This package provides an infrastructure for efficient processing of large-scale genetic and phenotypic data including core functions for:

- fitting linear mixed models
- constructing genetic relationship matrices
- estimating genetic parameters (heritability and correlation)
- performing genomic prediction and genetic risk profiling
- single or multi-marker association analyses

`qgg` handles large-scale data by taking advantage of:

- multi-core processing using openMP
- multithreaded matrix operations implemented in BLAS libraries (e.g., OpenBLAS, ATLAS or MKL)
- fast and memory-efficient batch processing of genotype data stored in binary files (i.e., PLINK bedfiles)

You can install qgg from CRAN with:

```
install.packages("qgg")
```

The most recent version of `qgg` can be obtained from github:

```r
library(devtools)
devtools::install_github("psoerensen/qgg")
```

Input data/objects commonly used in the qgg package

All functions in `qgg` used for analysis of complex traits relies on a simple data infrastructure that takes the following main input:

`y`: vector, matrix or list of phenotypes
`X`: design matrix for non-genetic factors
`W`: matrix of centered and scaled genotypes (in memory)
`Glist`: list structure providing information on genotypes, sparse LD, and LD scores (on disk)
`stat`: data frame with marker summary statistics
`sets`: list of sets with marker ids
`ids`: vector of ids of individuals
`rsids`: vector marker marker ids

Linking R to multi-threaded math libraries

The multi-core machines of today offer parallel processing power. To take advantage of this, R should be linked to multi-threaded math libraries (e.g. MKL/OpenBLAS/ATLAS). These libraries make it possible for many common R operations, such as matrix multiplication/inversion/decomposition, and some higher-level matrix operations, to compute in parallel and use all of the processing power available to reduce computation times.

This can make a huge difference in computation times: https://mran.microsoft.com/documents/rro/multithread#mt-bench

For Windows/Linux users it is possible to install Microsoft R Open is the enhanced distribution of R from Microsoft Corporation: https://mran.microsoft.com/open

For MAC users the ATLAS (Automatically Tuned Linear Algebra Software) library can be installed from here: https://ports.macports.org/port/atlas/

2 Prepare genotype data

The preparation (including quality control) of genotype data is a key step in quantitative genetic analyses.

```r
library(qgg)
library(data.table)
library(corrplot)
```

In this example we will use the 1000G data downloaded using the following commands:

```r
url <- "https://data.broadinstitute.org/alkesgroup/LDSCORE/1000G_Phase3_plinkfiles.tgz"
dest <- "./1000G_Phase3_plinkfiles.tgz"
download.file(url = url, dest = dest)
cmd <- "tar -xvzf 1000G_Phase3_plinkfiles.tgz"
system(cmd)
```

Summarize genotype information in PLINK files

The function `gprep()` reads genotype information from binary PLINK files, and creates the `Glist` object that contains general information about the genotypes:

```r
bedfiles <- paste("/mydir/1000G.EUR.QC.", 1:22, ".bed", sep = "")
bimfiles <- paste("/mydir/1000G.EUR.QC.", 1:22, ".bim", sep = "")
famfiles <- paste("/mydir/1000G.EUR.QC.", 1:22, ".fam", sep = "")

Glist <- gprep(study = "1000G", bedfiles = bedfiles, bimfiles = bimfiles,
    famfiles = famfiles)
names(Glist)
```

The output from `gprep()` (`Glist`) has a list structure that contains information about the genotypes in the binary file. `Glist` is required for downstream analyses provided in the qgg package. Typically, the `Glist` is prepared once, and saved as an *.RDS-file.

```r
saveRDS(Glist, file = "Glist.RDS", compress = FALSE)
```

Quality control of genotype data

In general it advisable to perform quality control of the genotype data. The quality control include removing markers with low genotyping rate, low minor allele frequency, not in Hardy-Weinberg Equilibrium. The function `gfilter()` can be used for filtering of markers:

```r
rsidsQC <- gfilter(Glist = Glist, excludeMAF = 0.01, excludeMISS = 0.05,
    excludeHWE = 1e-12, excludeCGAT = TRUE, excludeINDEL = TRUE, excludeDUPS = TRUE,
    excludeMHC = FALSE)
```

3 Compute sparse LD matrices

A number of methods used in the genetic analyses of complex traits (e.g. Bayesian linear regression analyses, genomic risk scoring and LD score regression) are based on summary statistics and require the construction of a reference linkage disequilibrium (LD) correlation matrix. The LD matrix corresponds to the correlation between the genotypes of genetic variants across the genome. Here we use a sparse LD matrix approach using a fixed window approach (e.g. number of markers, 1 cM or 1000kb), which sets LD correlation values outside this window to zero.

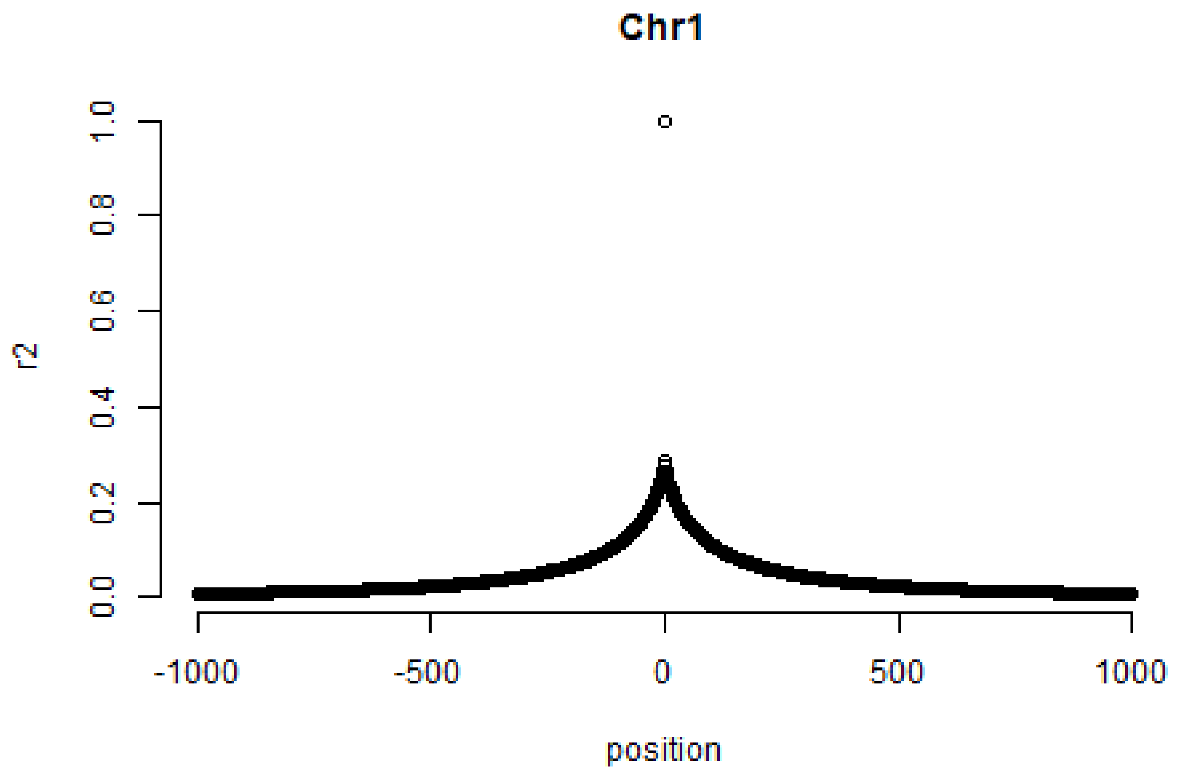
The function `gprep` can be used to compute sparse LD matrices which are stored on disk. The r^2 metric used is the pairwise correlation between markers (allele count alternative allele) in a specified region of the genome. Although this step can be slow unless R is linked to a fast BLAS it is typically only done once (or a few times).

```
# Define filenames for the sparse LD matrices
nchr <- Glist$nchr
ldfiles <- paste0(getwd(), "/sample_chr", 1:nchr, ".ld")

# Compute sparse LD matrices using the filtered rsids only
Glist <- gprep(Glist, task = "sparseld", msize = 1000, rsids = rsidsQC,
    ldfiles = ldfiles, overwrite = TRUE)

# Save the updated Glist object
saveRDS(Glist, file = "Glist.RDS", compress = FALSE)
```

Get the sparse LD matrix for a chromsome

The ´getLD´ function can be used to extract the sparse LD matrix stored on disk. Here we extract the sparse LD for chromosome 1 and plot the mean r2 in a genomic window around the index marker illustrating that marker in close proximity with the index marker have (on average) a higher r2 as compared to distantly located markers:

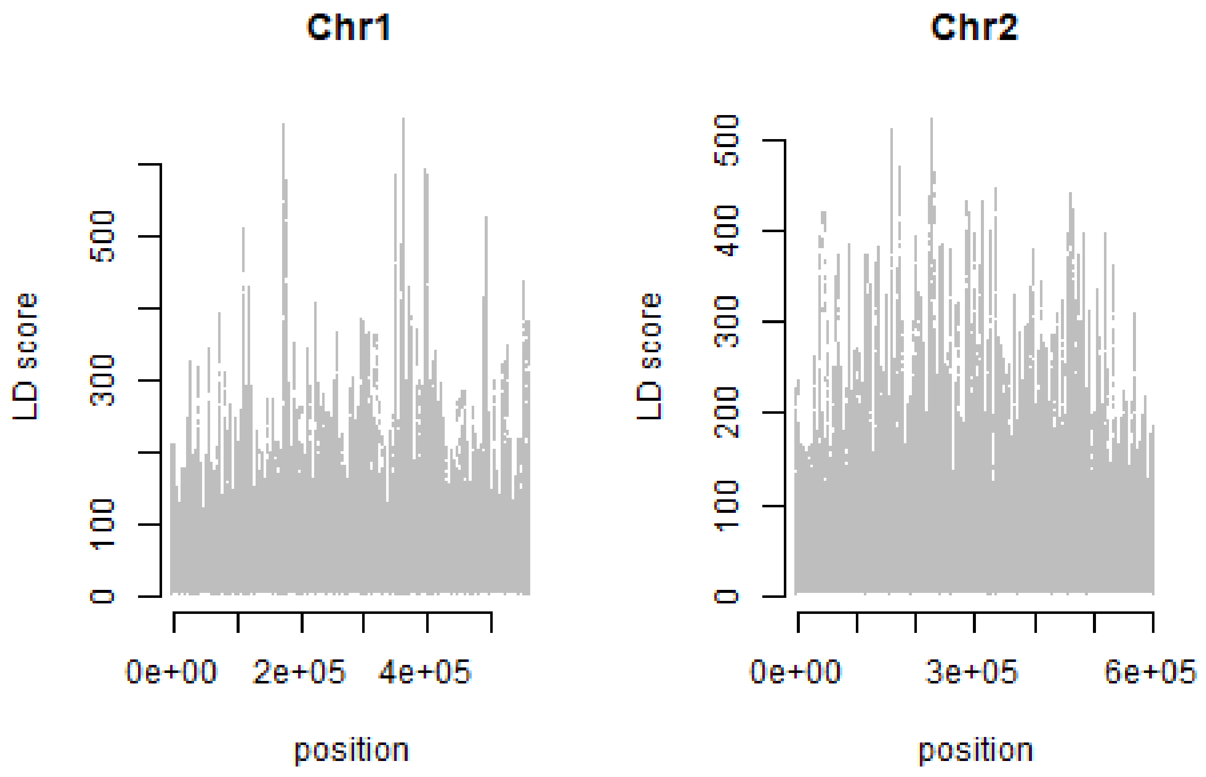
```
ld <- getLD(Glist, chr = 1)
# Plot mean r2 f
plot(y = rowMeans(ld^2), x = rownames(ld), frame.plot = FALSE, ylab = "r2",
    xlab = "position", main = "Chr1")
```



Get the LD scores for a chromosome

The ld scores quantify the degree of linkage disequilibrium in a genomic region and are used LD score regression. They can be extracted from the Glist object in the following way:

```
layout(matrix(1:2, ncol = 2))
plot(Glist$ldscores[[1]], frame.plot = FALSE, ylab = "LD score", xlab = "position",
    main = "Chr1", cex = 0.1, col = "grey")
plot(Glist$ldscores[[2]], frame.plot = FALSE, ylab = "LD score", xlab = "position",
    main = "Chr2", cex = 0.1, col = "grey")
```



Get LD sets for chromosome 1

It can be useful to identify markers linked in a genomic regions. The ´getLDsets´ function can be used to extract linked marker based on a LD threshold such as r2=0.25:

```
sets <- getLDsets(Glist = Glist, chr = 1, r2 = 0.5)
str(sets, list.len = 5)
```

```
## List of 562925
##  $ rs540538026: chr "rs540538026"
##  $ rs62635286 : chr [1:2] "rs62635286" "rs200579949"
##  $ rs200579949: chr [1:2] "rs62635286" "rs200579949"
##  $ rs541940975: chr "rs541940975"
##  $ rs199856693: chr "rs199856693"
##   [list output truncated]
```

4 Quality control of external GWAS summary statistic

Quality control is a critical step for working with summary statistics (in particular for external). Processing and quality control of GWAS summary statistics includes:

- map marker ids (rsids/cpra (chr, pos, ref, alt)) to LD reference panel data in Glist
- check effect allele (ea)
- check effect allele frequency (eaf)
- thresholds for MAF and HWE
- exclude INDELS, CG/AT and MHC region
- remove duplicated marker ids
- check which build version
- check for concordance between marker effect and LD data

The ´qcStat´ function can be used for processing of summary statistics is available in our qgg package.

```
# Prepare T2DM summary statistics used in GSEA
fname_stat <- "C:\\Users\\au223366\\Dropbox\\Projects\\balder\\Mahajan.NatGenet2018b.T2D-noUKBB.European
stat <- fread(fname_stat, data.table = FALSE)
head(stat)
```

```
##           SNP Chr       Pos EA NEA    EAF   Beta     SE Pvalue
## 1 1:100000012   1 100000012  T   G 0.2600 -0.030 0.0095 0.0018
## 2  1:10000006   1  10000006  A   G 0.0047 -0.098 0.0780 0.2100
## 3 1:100000135   1 100000135  A   T 0.9900 -0.089 0.0630 0.1600
## 4 1:100000436   1 100000436  T   C 1.0000  0.240 0.2700 0.3700
## 5 1:100000827   1 100000827  T   C 0.3100 -0.026 0.0090 0.0033
## 6 1:100000843   1 100000843  T   C 0.9400 -0.001 0.0180 0.9600
```

```
dim(stat)
```

```
## [1] 21508698        9
```

```
# check column names of original data
colnames(stat)
```

```
## [1] "SNP"    "Chr"    "Pos"    "EA"     "NEA"    "EAF"    "Beta"   "SE"
## [9] "Pvalue"
```

```
# subset original data by selecting the column needed for downstream
# analysis
stat <- stat[, c("SNP", "Chr", "Pos", "EA", "NEA", "EAF", "Beta", "SE",
    "Pvalue")]

# rename column names of the selected data
colnames(stat) <- c("marker", "chromosome", "position", "effect_allele",
    "non_effect_allele", "effect_allele_freq", "effect", "effect_se", "effect_p")

# QC of summary stat and map to 1000G data
stat <- qcStat(Glist = Glist, stat = stat)
dim(stat)
```

```
## [1] 6546749       9
```

```
head(stat)
```

```
##                 rsids chr    pos ea nea   eaf       b   seb    p
## rs2000096   rs2000096   1 567867  G   A 0.000 -0.5200 0.630 0.41
## rs12238997 rs12238997   1 693731  G   A 0.130 -0.0088 0.017 0.60
## rs72631875 rs72631875   1 705882  A   G 0.063  0.0110 0.037 0.76
## rs55727773 rs55727773   1 706368  A   G 0.500  0.0140 0.015 0.37
## rs12184267 rs12184267   1 715265  T   C 0.041 -0.0610 0.061 0.32
## rs12184277 rs12184277   1 715367  G   A 0.040 -0.0580 0.061 0.34
```

5 LD Score Regression

```
# Effective population size
ncase <- 74124
ncontrol <- 824006
ntotal <- ncase + ncontrol
pcase <- ncase/(ncase + ncontrol)
neff <- ntotal * pcase * (1 - pcase)

# LDSC analysis
h2 <- ldsc(Glist = Glist, stat = stat, n = neff, what = "h2")
h2
```

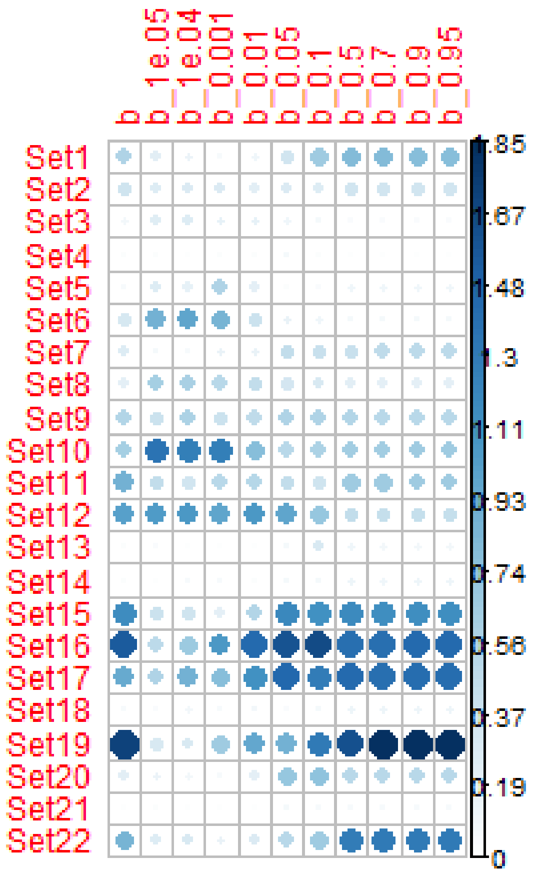
```
##        h2
## 0.2301532
```

6 Gene Set Enrichment Analysis

```
# Adjust summary statistics using clumping and p-value thresholding
statAdj <- adjStat(stat = stat, Glist = Glist, r2 = 0.9, threshold = c(1e-05,
    1e-04, 0.001, 0.01, 0.05, 0.1, 0.5, 0.7, 0.9, 0.95))

# Marker sets defined by chromosomes
sets <- Glist$rsidsLD

# Gene set enrichment analysis
setstat <- gsea(stat = statAdj, sets = sets)
setstat
```

```
## $m
##   Set1   Set2   Set3   Set4   Set5   Set6   Set7   Set8   Set9  Set10  Set11
## 506354 548392 466229 478009 416796 437125 384483 359624 279843 335923 326462
##  Set12  Set13  Set14  Set15  Set16  Set17  Set18  Set19  Set20  Set21  Set22
## 317020 245210 215960 185744 197140 174802 190907 148437 149792  91437  91060
##
## $stat
##               b  b_1e.05  b_1e.04  b_0.001    b_0.01    b_0.05    b_0.1
## Set1  316.00583 0.784484 1.340025 3.862936 16.471837 50.338405 84.13124
## Set2  334.42726 0.939207 1.761653 4.957158 18.850542 52.645754 82.98848
## Set3  275.13384 0.768474 1.500383 4.070760 15.914105 42.574226 64.52643
## Set4  268.08082 0.338026 0.626529 3.077185 14.280380 42.641111 63.23907
## Set5  236.86457 0.678027 1.154620 4.441060 14.163925 35.660004 56.44666
## Set6  265.93780 1.754007 2.851542 5.400608 16.159533 40.356312 61.13183
## Set7  231.58811 0.382453 0.756589 3.268392 12.725194 38.568849 59.66588
## Set8  214.94867 0.894468 1.498439 3.771939 13.613739 34.899919 54.23676
## Set9  176.89183 0.566069 1.161744 2.764007 10.484782 29.296266 45.57101
## Set10 213.77561 3.224738 3.761265 6.015335 16.161269 35.305623 55.62773
## Set11 212.95027 0.638766 1.085478 3.383467 12.357922 33.019849 50.35553
## Set12 211.52727 1.487653 2.161536 4.177904 16.729869 37.702406 54.17904
## Set13 135.22755 0.090233 0.171552 1.203567  6.742237 21.495011 36.92711
## Set14 119.93001 0.043854 0.266198 1.293855  6.016815 18.853437 29.26174
## Set15 129.09928 0.363639 0.632933 1.509686  7.140814 25.651222 37.69067
## Set16 138.04962 0.467657 0.927270 2.851137 12.290879 30.031433 42.03996
## Set17 119.61659 0.456273 0.999763 2.263009 10.740711 25.469463 36.41365
## Set18 108.36866 0.072765 0.308541 1.075226  5.766528 16.004856 25.43795
## Set19 106.62772 0.221054 0.428282 1.822225  7.643014 19.628251 32.15010
## Set20  88.69101 0.098630 0.197171 0.810432  4.876498 16.334542 25.94319
## Set21  45.27943 0.008435 0.020454 0.194950  1.943685  7.545219 11.60817
## Set22  61.79085 0.090456 0.217485 0.609190  3.131874  9.336013 15.24851
##           b_0.5     b_0.7     b_0.9    b_0.95
## Set1  163.30869 173.08101 175.03851 175.08742
## Set2  164.82059 173.95600 176.07790 176.13327
## Set3  125.17712 132.77364 134.41433 134.45334
## Set4  122.58003 129.15429 130.81154 130.85248
## Set5  114.30075 120.57089 122.07644 122.11357
## Set6  118.48477 125.44144 126.93277 126.98418
## Set7  116.37094 123.44452 124.96677 125.01627
## Set8  104.09672 109.46158 110.68776 110.72149
## Set9   86.99724  91.01017  92.14143  92.16677
## Set10 106.36845 112.04363 113.58729 113.62142
## Set11 103.03924 108.44761 109.52055 109.55099
## Set12  97.07657 101.80770 102.87436 102.91506
## Set13  67.12360  70.74389  71.55338  71.57592
## Set14  59.71051  62.92720  63.81031  63.83091
## Set15  69.07708  72.33829  73.12870  73.15065
## Set16  77.06804  81.70264  82.61723  82.64511
## Set17  69.32206  72.82931  73.63462  73.65443
## Set18  52.65910  55.87047  56.66594  56.68229
## Set19  60.50669  65.00110  65.80233  65.83148
## Set20  46.18914  48.76954  49.38502  49.40888
## Set21  23.36973  24.74765  25.09882  25.10661
## Set22  34.38163  36.24058  36.63944  36.65064
##
## $p
##           b b_1e.05 b_1e.04 b_0.001 b_0.01 b_0.05 b_0.1 b_0.5 b_0.7 b_0.9
## Set1  0.275   0.576   0.701   0.845  0.772  0.440 0.220 0.157 0.159 0.160
## Set2  0.410   0.527   0.511   0.510  0.598  0.518 0.546 0.435 0.430 0.431
## Set3  0.779   0.554   0.540   0.651  0.649  0.767 0.817 0.903 0.891 0.890
## Set4  0.944   0.951   0.958   0.956  0.950  0.816 0.955 0.982 0.987 0.987
## Set5  0.864   0.530   0.646   0.275  0.620  0.941 0.900 0.761 0.768 0.766
## Set6  0.471   0.130   0.105   0.139  0.391  0.667 0.759 0.820 0.808 0.810
## Set7  0.548   0.821   0.873   0.662  0.675  0.356 0.378 0.379 0.330 0.327
## Set8  0.618   0.241   0.247   0.311  0.354  0.463 0.499 0.587 0.605 0.610
## Set9  0.281   0.392   0.263   0.393  0.336  0.276 0.267 0.278 0.305 0.306
## Set10 0.241   0.044   0.053   0.053  0.162  0.309 0.255 0.232 0.230 0.224
## Set11 0.133   0.348   0.426   0.306  0.302  0.348 0.416 0.219 0.223 0.224
## Set12 0.099   0.088   0.087   0.107  0.085  0.108 0.196 0.345 0.367 0.371
## Set13 0.944   0.908   0.997   0.958  0.977  0.826 0.490 0.772 0.783 0.783
## Set14 0.923   0.968   0.868   0.860  0.954  0.816 0.834 0.728 0.744 0.738
## Set15 0.068   0.392   0.414   0.614  0.285  0.064 0.075 0.066 0.072 0.073
## Set16 0.028   0.324   0.210   0.086  0.037  0.025 0.022 0.040 0.040 0.036
## Set17 0.114   0.276   0.134   0.162  0.074  0.035 0.052 0.037 0.039 0.038
## Set18 0.858   0.904   0.786   0.866  0.806  0.903 0.839 0.730 0.720 0.717
## Set19 0.019   0.487   0.490   0.216  0.113  0.133 0.049 0.024 0.014 0.014
## Set20 0.588   0.683   0.799   0.868  0.605  0.192 0.176 0.309 0.305 0.307
## Set21 0.995   0.917   0.993   0.998  0.978  0.810 0.857 0.864 0.862 0.862
## Set22 0.140   0.532   0.499   0.662  0.506  0.303 0.220 0.052 0.050 0.050
##       b_0.95
## Set1   0.160
## Set2   0.431
## Set3   0.890
## Set4   0.987
## Set5   0.766
## Set6   0.810
## Set7   0.326
## Set8   0.610
## Set9   0.307
## Set10  0.224
## Set11  0.224
## Set12  0.370
## Set13  0.783
## Set14  0.738
## Set15  0.072
## Set16  0.036
## Set17  0.038
## Set18  0.717
## Set19  0.014
## Set20  0.307
## Set21  0.862
## Set22  0.050
```

```
corrplot(-log10(setstat$p), is.corr = FALSE)
```


b
b_1e.05
b_1e.04
b_0.001
b_0.01
b_0.05
b_0.1
b_0.5
b_0.7
b_0.9
b_0.95
Set1
Set2
Set3
Set4
Set5
Set6
Set7
Set8
Set9
Set10
Set11
Set12
Set13
Set14
Set15
Set16
Set17
Set18
Set19
Set20
Set21
Set22