bigsnpr-example

Elisabeth Rosenthal (erosen [at] uw.edu)

11/4/2021

Vignette

The bigSNPr vignette can be found at
https://privefl.github.io/bigsnpr/articles/demo.html

Setup

```
library(bigsnpr)
```

```
## Warning: package 'bigsnpr' was built under R version 4.1
```

```
## Loading required package: bigstatsr
```

```
## Warning: package 'bigstatsr' was built under R version 4
```

Note that bigstatsr was also loaded. This package provides functions for fast statistical analysis of large-scale data encoded as matrices. We'll discuss this later

Read in the data

```
bedfile <- system.file("extdata", "example.bed",
                       package = "bigsnpr")
file.size(bedfile)
```

```
## [1] 590463
```

Get a temporary directory for the analysis

```
tmpfile <- tempfile()
snp_readBed(bedfile, backingfile = tmpfile)
obj.bigSNP <- snp_attach(paste0(tmpfile, ".rds"))
```

Look at the structure of the object

```
str(obj.bigSNP, max.level = 2, strict.width = "cut")
```

```
## List of 3
##  $ genotypes:Reference class 'FBM.code256' [package "bigstatsr"] with 16 fields
##   ..and 26 methods, of which 12 are  possibly relevant:
##   ..  add_columns, as.FBM, bm, bm.desc, check_dimensions,
##   ..  check_write_permissions, copy#envRefClass, initialize, initialize#FBM,
##   ..  save, show#envRefClass, show#FBM
##  $ fam      :'data.frame':   517 obs. of  6 variables:
##   ..$ family.ID  : chr [1:517] "POP1" "POP1" "POP1" "POP1" ...
##   ..$ sample.ID  : chr [1:517] "IND0" "IND1" "IND2" "IND3" ...
##   ..$ paternal.ID: int [1:517] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ maternal.ID: int [1:517] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ sex        : int [1:517] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ affection  : int [1:517] 1 1 2 1 1 1 1 1 1 1 ...
##  $ map      :'data.frame':   4542 obs. of  6 variables:
##   ..$ chromosome  : int [1:4542] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ marker.ID   : chr [1:4542] "SNP0" "SNP1" "SNP2" "SNP3" ...
##   ..$ genetic.dist: int [1:4542] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ physical.pos: int [1:4542] 112 1098 2089 3107 4091 5091 6107 7103 8090 9..
##   ..$ allele1     : chr [1:4542] "A" "T" "T" "T" ...
##   ..$ allele2     : chr [1:4542] "T" "A" "A" "A" ...
##  - attr(*, "class")= chr "bigSNP"
```

Look at the structure of the object

obj.bigSNP is a list, where the genotypes are stored as a File Backed Matrix (FBM), a reference class for storing and accessing matrix-like data stored in files on disk.

The genotypes are stored as a special FBM, called FBM.code256 which adds a slot code used as a lookup table of size 256.

See https://academic.oup.com/bioinformatics/article/34/16/2781/4956666 for more

- Can apply algorithms on data larger than your RAM
- Can easily parallelize your algorithms because the data on disk is shared

Get aliases and phenotype data

```
G   <- obj.bigSNP$genotypes
CHR <- obj.bigSNP$map$chromosome
POS <- obj.bigSNP$map$physical.pos
y01 <- obj.bigSNP$fam$affection - 1
```

Run simple GWAS

```
(NCORES <- nb_cores())
```

```
## [1] 2
```

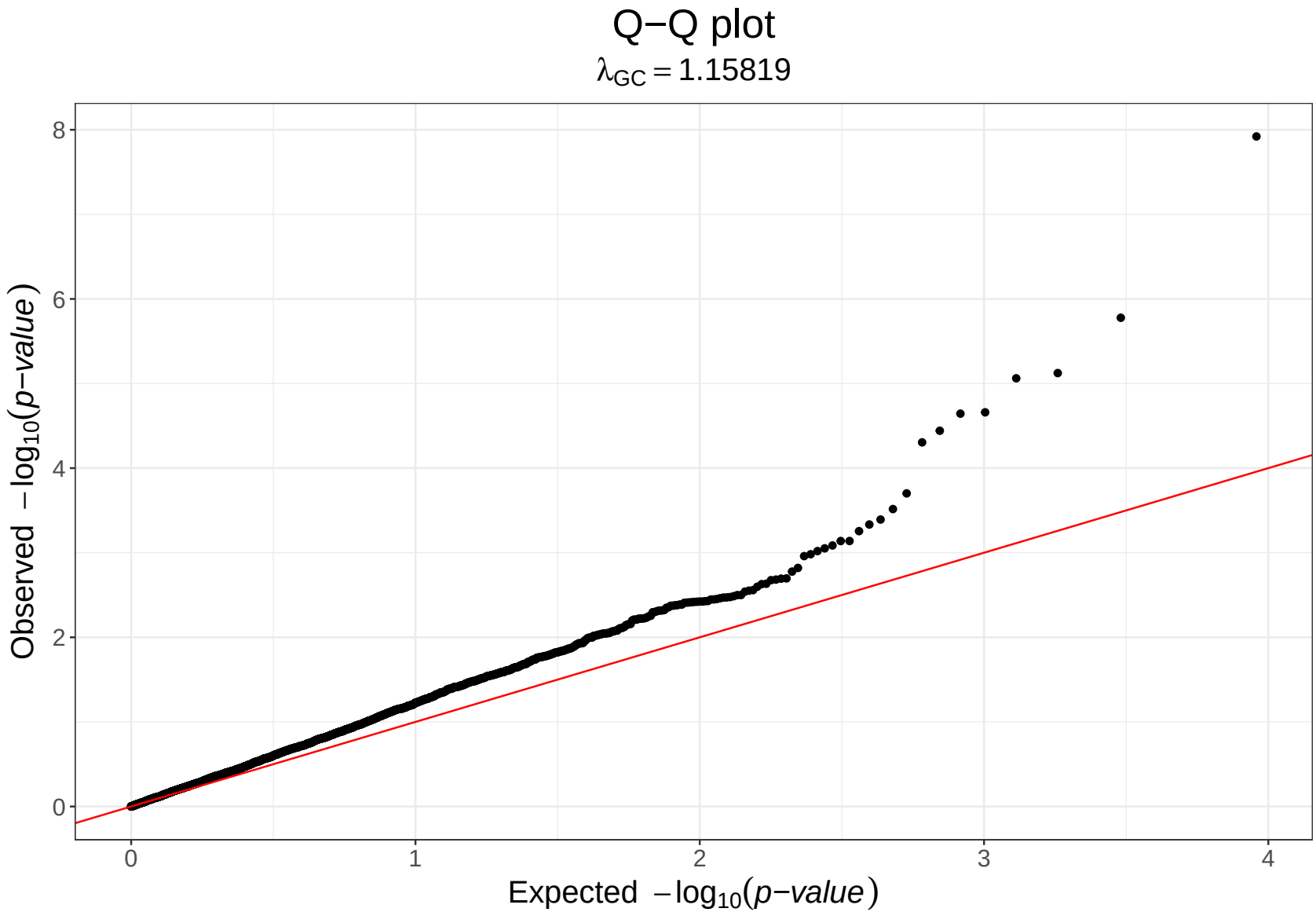
```
obj.gwas <- big_univLogReg(G, y01.train = y01,
                           ncores = NCORES)
```

big_univLogReg() is a function from bigstatsr that runs Logistic Regression

Other similiar functions include big_univLinReg(), big_spLinReg(), and big_spLogReg().

Unadjusted QQ Plot

```
snp_qq(obj.gwas)
```



Genomic Inflation (Genomic Control)

The p-values can be inflated (or deflated) due to bias. Possible sources of bias include

- Relatedness in the data
- Multiple ancestries in the data
- Ancestral admixture
- Covariates left out of the model

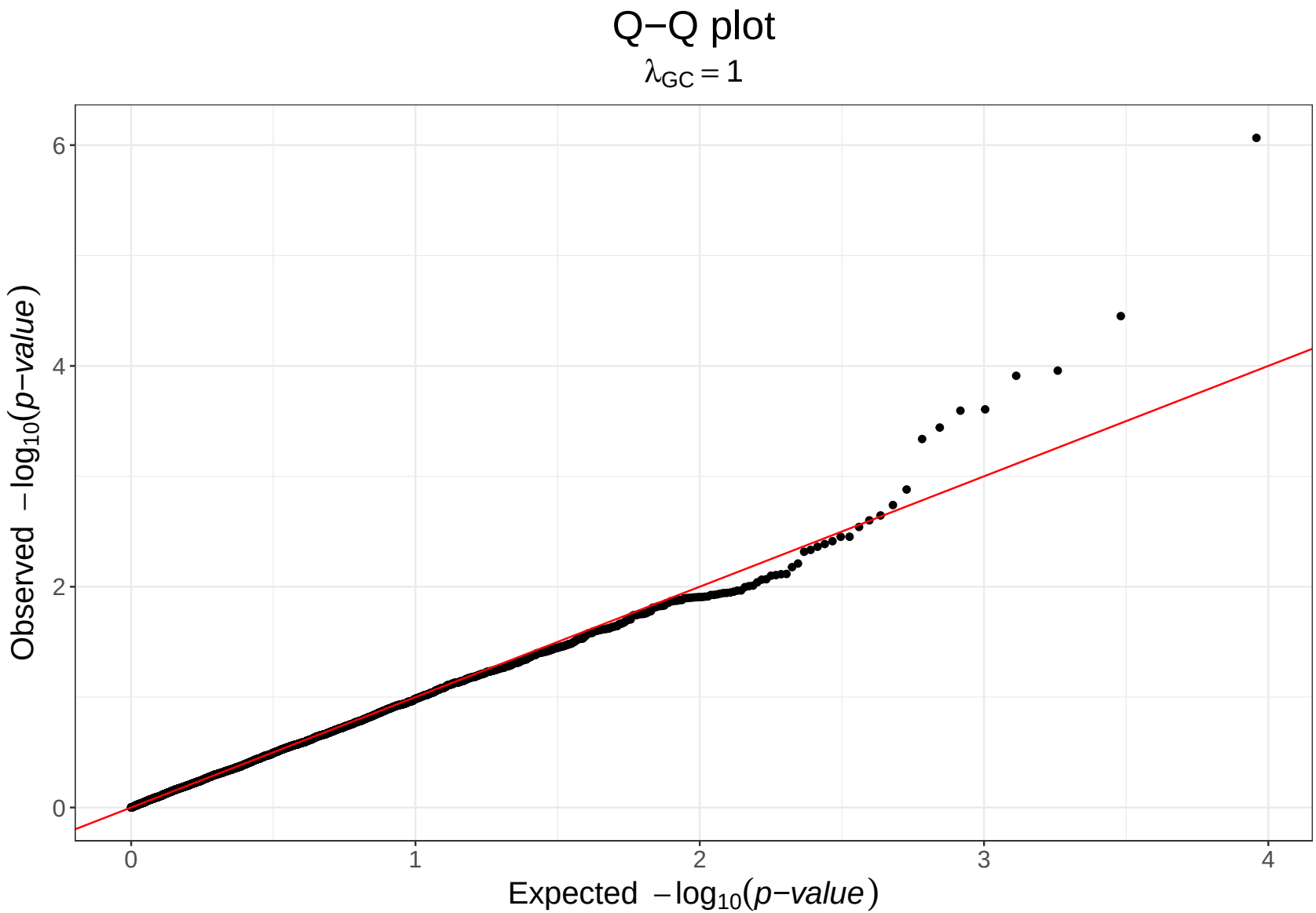
Accounting for Genomic Inflation

Under the null hypothesis, the test statitics, q, follows a χ^2_1 distribution

We adjust for the inflation $\lambda_{GC} = \frac{\text{median(q)}}{0.4549364}$ where 0.4549364 is the median of the χ^2_1 distribution

Adjusting for λ_{GC}

```
obj.gwas.gc <- snp_gc(obj.gwas)
snp_qq(obj.gwas.gc)
```



A note on λ_{GC} in bigSNPr

The package reports the inflation factor for the Z-statistic

$$\lambda_{GC}(\chi^2) = (\lambda_{GC}(Z))^2$$

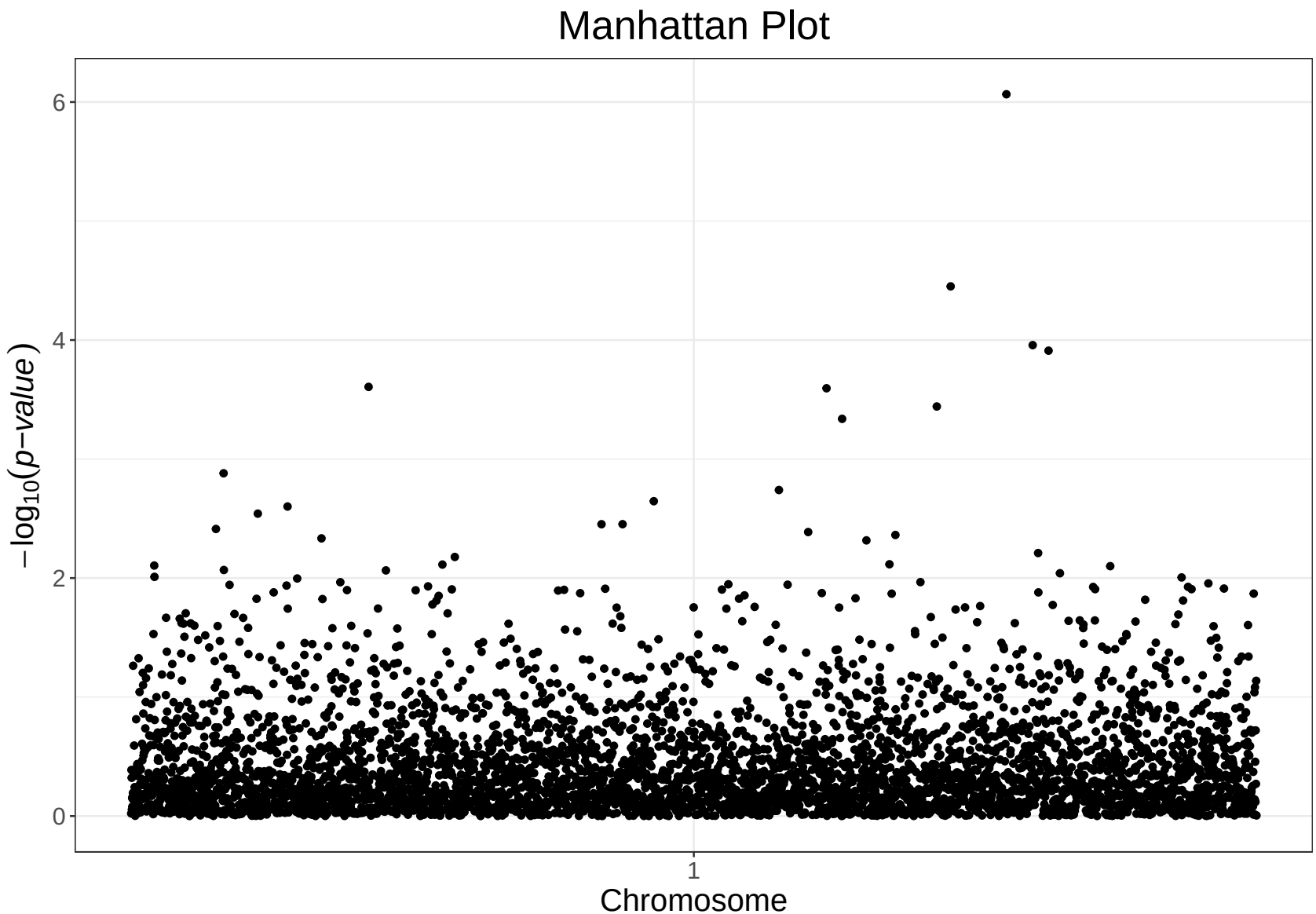
Make sure you know which statistic you are reporting. You can also calculate it without having to view the QQ-plot

```
(lambda.gc <- bigsnpr:::getLambdaGC(obj.gwas))^2
```

```
## [1] 1.341402
```

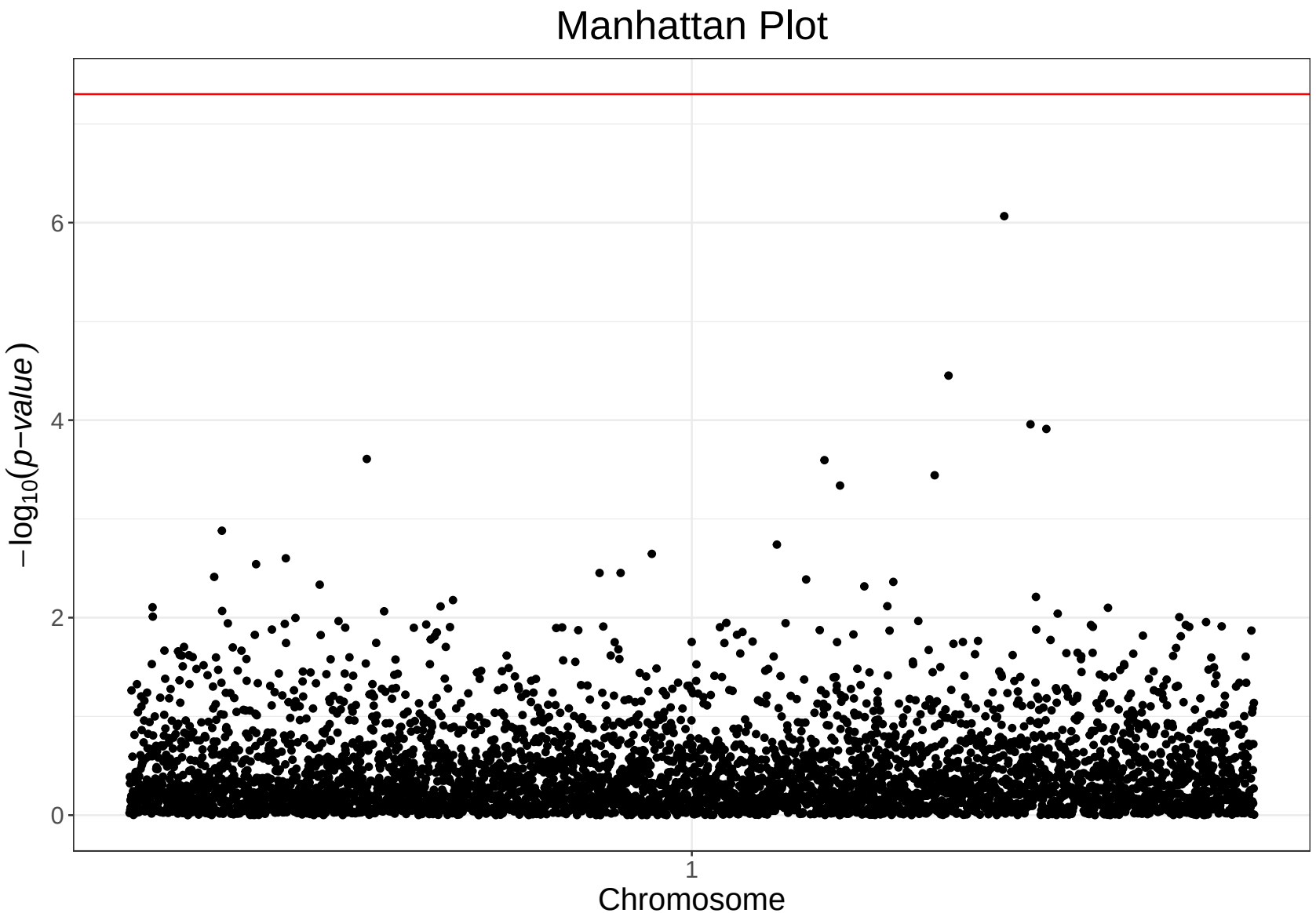
Manhatten plot using bigsnpr

```
plot1 <- snp_manhattan(obj.gwas.gc, infos.chr = CHR, infos.pos = POS)
plot1
```



The plotting functions in bigSNPr use ggplot2

```
library(ggplot2)
plot1 +   geom_hline(yintercept = -log10(5e-8), color = "red")
```



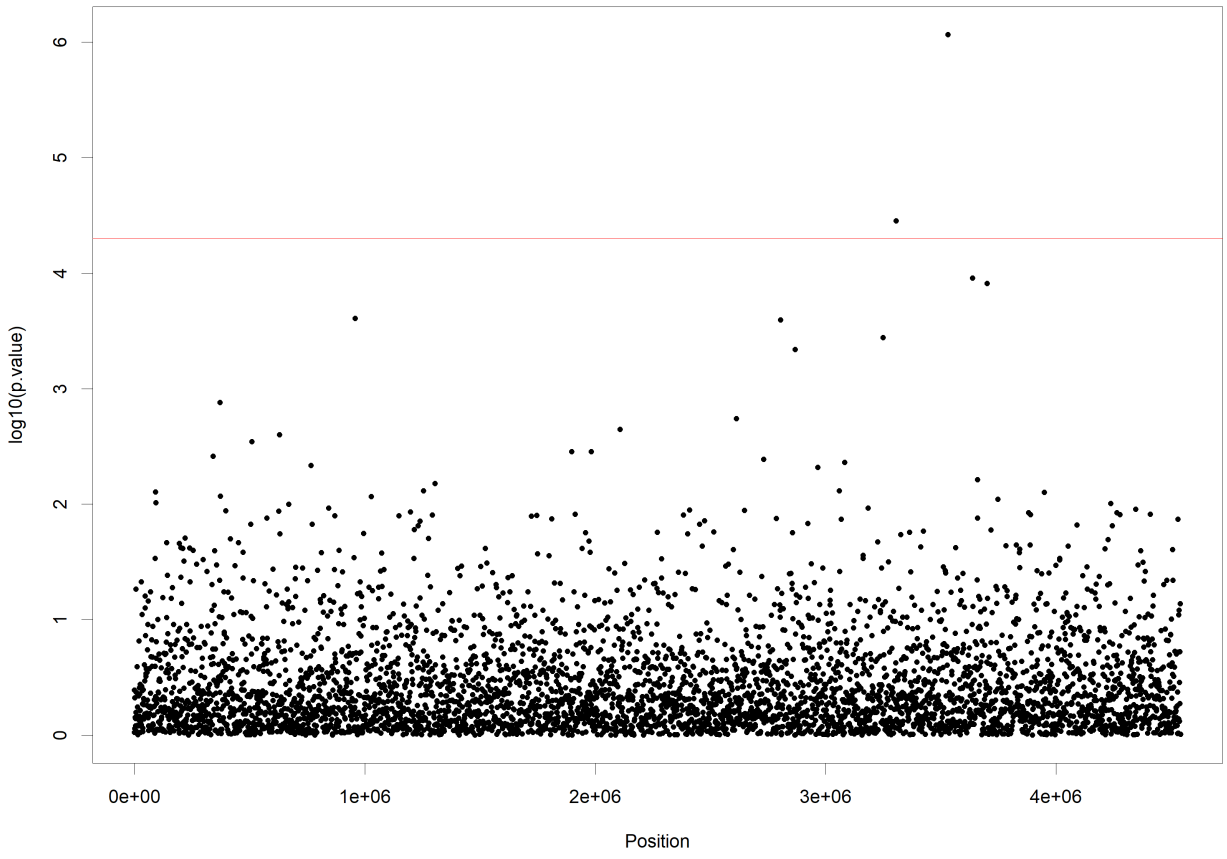
I prefer to use my own plotting functions

- snp_manhatten() is made for multple chromosomes (i.e., x-axis is chromosome, not position)
- I found it difficult to highlight variants, just by rsID. Sometimes I want to use position and the positions are offset depending on the chromosome and the dist.sep.chrs parameter
- I've looked into other options, like qqman library, but also did not like the functionality

Plotting for single chromosomes

```
obj.gwas.gc$p.value <- predict(obj.gwas.gc, log10 = FALSE)
png("Manhatten.png", width=3300,height=2500,
    units="px",pointsize=48)
plot(POS,-log10(obj.gwas.gc$p.value),xlab="Position",
     ylab="log10(p.value)", pch=20)
abline(h=-log10(5E-5),col="red")
dev.off()
```

view the plot



Missing Data

bigSNPr assumes no missing data. However, we don't want to remove participants just because they are missing a few genotypes.

```
class(G)
```

```
## [1] "FBM.code256"
## attr(,"package")
## [1] "bigstatsr"
```

It is not possible to just replace missing values as though G were a vector or a matrix

Missing Data

One way to get around this is to impute missing genotypes using snp_fastImputeSimple()

- mode = most frequent call
- mean0 = rounded mean
- mean2 = rounded mean to 2 decimal places
- random = sampling according to allele frequencies

```
G2 <- snp_fastImputeSimple(G,method="mode")
```