R basic and GWAS

GBIO0009

Kridsadakorn Chaichoompu
University of Liege

Basic commands

• q()	To quit R environment
• x = 5	Assignment operator
• y <- 5	Assignment operator
• ls()	To list objects in R environment
• ?ls()	To check how to use a function
• getwd()	To get a working directory
• setwd("New/Directory")	To set a new working directory
• save(x,y,file="mydata.RData")	To save objects as the R data file
• save.image(file="alldata.RData")	To save all objects as the R data file
• load("mydata.RData")	To load the R data file to the working space

Arithmetic operators

5+7	Addition
8-3	Subtraction
5*2	Multiplication
9/2	Division
(8+3)*4	Parentheses
2^4	Power
exp(4)	Exponential function
log(8)	Natural Logarithm
log10(8)	Logarithm in base 10
pi	Pi number

Logical operators

The values can be T, TRUE, F, FALSE

• 5<6	less than
• 5<=6	less than or equal to
• 5>6	greater than
• 5>=6	greater than or equal to
• 5==6	exactly equal to
• 5!=6	not equal to
• !a	NOT a
• a|b	a OR b
• a&b	a AND b
• xor(a,b)	a XOR b
• isTRUE(a)	test if X is TRUE

Expression statement

- if (a == 5 && b > 5)
- if (a == 5 || b > 5)

Basic data types

class() - to check class of object

- Logical TRUE, T, FALSE, F
```
class(TRUE)
```
- Numeric 2.4, 10, 200
```
class(6.5)
```
- Integer 1L, 0L, -7L
```
class(-8L)
```
- Complex 6 + 3i
```
class(6 + 3i)
```
- Character 'hello', "I", "like", 'R'
```
class('hello')
```
- Factor
```
a = as.factor(1)
a = as.factor('hello')
class(a)
```

Vector

To create vectors

- `a = c(1,2,0,6.6,-2.5)`
- `b = c("a","b","c")`
- `c = c(F,T,TRUE,FALSE)`

Vectors and operators

- `a + 5`
- `a * 2`
- `c & TRUE`
- `c | FALSE`
- `1:5            Vector of 1 to 5`
- `c(a,1:5)       Concatenate 2 vectors`

Matrix

To create matrices

matrix(vector, nrow=r, ncol=c, byrow=FALSE)

- `a = matrix(1:12, nrow=3, byrow=F)`
- `b = matrix(1:12, nrow=3, byrow=T)`
- `c = matrix(runif(12,min=0,max=1), nrow=3, byrow=T)`
- `d = matrix(sample(c(TRUE,FALSE),12,replace=TRUE), nrow=3, byrow=T)`

Matrices and operators

```
• a + 5
• a + b
• t(b)              Transpose of matrix
• a * b             Element-wise multiplication
• a %*% t(b)        Matrix multiplication
```

Matrix (2)

To access elements of matrix

- `a[1,1]`
- `a[,1]`
- `a[1,]`
- `a[,2:3]`

To name row and columns

- `colnames(a) = c("a","b","c","d")`
- `rownames(a) = c("1","2","3")`

To combine 2 matrices

- `cbind(a,b)Combine by column`
- `rbind(a,b)Combine by row`

Data frame

“data.frame” is the collections of variables which share many of the properties of matrices and of lists

To create data.frame

- `x = c("Kris", "Jack", "Steve", NA)`
- `y = c(50,20,60,40)`
- `z = c(FALSE,TRUE,TRUE,FALSE)`
- `df = data.frame(x,y,z)`
- `colnames(df) <- c("name","paid","registered")`

Useful functions

- `df$name`
- `is.na(df$name) Check all elements if they are NA?`
- `anyNA(df$name) Is there any NA?`
- `df$paid * 1.21`
- `dim(df)         Check dimension`
- `df[which(df$name=="Kris"),]    Get specific row`

Data frame (2)

To name row and columns

- `colnames(df) = c("1","2","3”)`
- `rownames(df) = c("a","b","c","d")`

To combine 2 matrices

- `cbind(df,df)  Combine by column`
- `rbind(df,df)  Combine by row`

List

A collection of objects which can be in different length

- `m = list(car=c("Toyota","Honda","Nissan"), age=c(23,67),single=TRUE)`

To access objects

- `m$car`
- `m$age`
- `m[[1]]`
- `m[[2]]`

Conversion functions

- `as.matrix(df)`
- `as.data.frame(a)`
- `as.list(1:5)`
- `as.integer(1:5)`
- `as.logical(c(0,1,1,0))`
- `as.factor(1:5)`

Concatenation functions

- `c()` To combine vectors
- `list()` To combine lists
- `cbind()` To combine matrices and data frames by column
- `rbind()` To combine matrices and data frames by row
- `paste("Hello","my","name","is","Kris")`
  To combine strings
- `paste0("Hello","my","name","is","Kris")`
  To combine strings without space

Trick to display text on screen

- `str = paste("Hello","my","name","is","Kris","\n")`
- `cat(str)` To display text
- `print(str)` To display all values as they are

Control Flow

- if(condition) ...
- if(condition) ... else ...

- for(variable in sequence) ...
- while(condition) ...

• break	To stop iteration
• next	To skip to next iteration

IF

Examples:

```
age = 10
if (age > 18){
   cat("Old\n")
}else{
   cat("Young\n")
}

age = 20
if ((age>18) && (age<25)){
   cat("Teenager\n")
}else{
   cat("Other type\n")
}
```

FOR

Examples:

```
for (i in 1:10){
  cat(paste(i,"\n"))
}

name =
c("Hello","my","name","is","Kris")
for (i in name)
  cat(paste0(i," "))
```

WHILE

Examples:

```
i = 0
while (i<5){
   print(i)
   i = i+1
}

i = 0
while (i<10){
   if (i>5) next
   print(i)
   i = i+1
}
```

Import delimited text file

- The formatted text files can be imported to R by these functions:
  - Read.table()
  - read.csv(), read.csv2()
  - read.delim(), read.delim2()
- Important parameters:
  - file : the name of input file
  - header : to indicate whether the first line contains the names of the variables or not
  - sep = the separator character

- Try to import *orange.csv*

  Download from the course website:

  http://www.montefiore.ulg.ac.be/~chaichoompu
- Example:

```
mydata=read.table(file="orange.csv",sep=",",header=TRUE)
head(mydata)
```

Export as delimited text file

- You can use these functions to export to file
  - write.table(x, file = "")
  - write.csv()
- Important parameters:
  - file : the name of input file
  - row.names : to indicate whether row names will be exported or not
  - col.names : to indicate whether column names will be exported or not
  - sep: the separator character
  - quote: to indicate whether text will be quoted (“hello”)

- Example:

```
write.table(mydata,file="newfile.csv",quote=T,sep="\t",
row.name=T,col.name=T)
```

Text display

To display text on screen

- `print(x, ...)`
- `cat(...)`

Concatenate variables

- `paste (...)`
- `paste0(...)`

Example:

- `dd <- 28`
- `mm <- "October"`
- `yy <- 2016`
- `cat(paste0(dd,mm,yy))`
- `cat(paste(dd,mm,yy,sep="-"))`

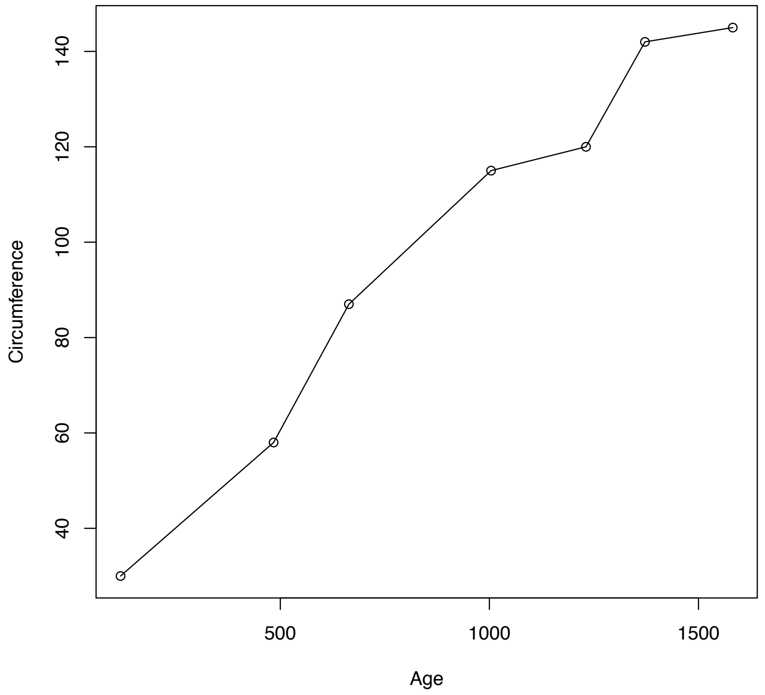
Plots

- Use plot() to create a simple XY plot
  - plot(rnorm(10))
- In the computing servers, we need to save plots as files and transfer to a local computer to view
  - pdf(file="./xyplot.pdf") ➔ create a pdf file in the current working directory
  - plot(rnorm(10))
  - points(rnorm(2),col="red") ➔ add 2 red dots to the plot
  - dev.off() ➔ close the graphical session, all graphical functions called before *dev.off()* will be saved to pdf file
- R also supports the other types of graphical files
  - Check: jpeg(), tiff(), png(), bmp()

Plotting for multiple data series

Single line:

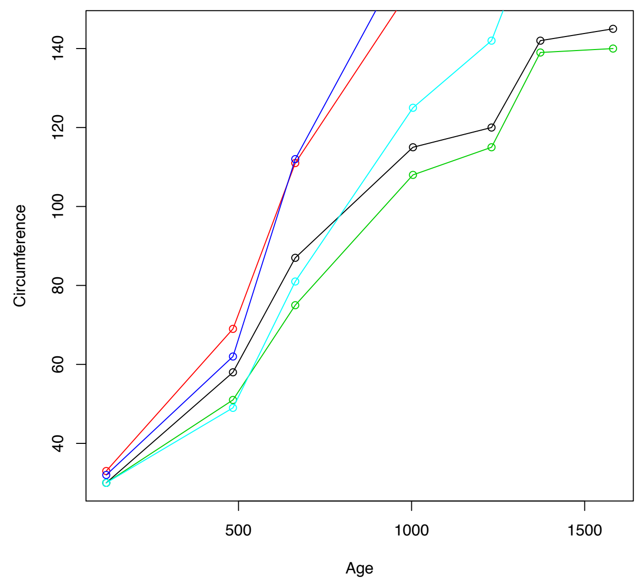
```
age=mydata$age[which(mydata$Tree==1)]
cir=mydata$circumference[which(mydata$Tree==1)]
plot(age,cir,type="o",xlab="Age",ylab="Circumference",
col=1)
```



Plotting for multiple data series (2)

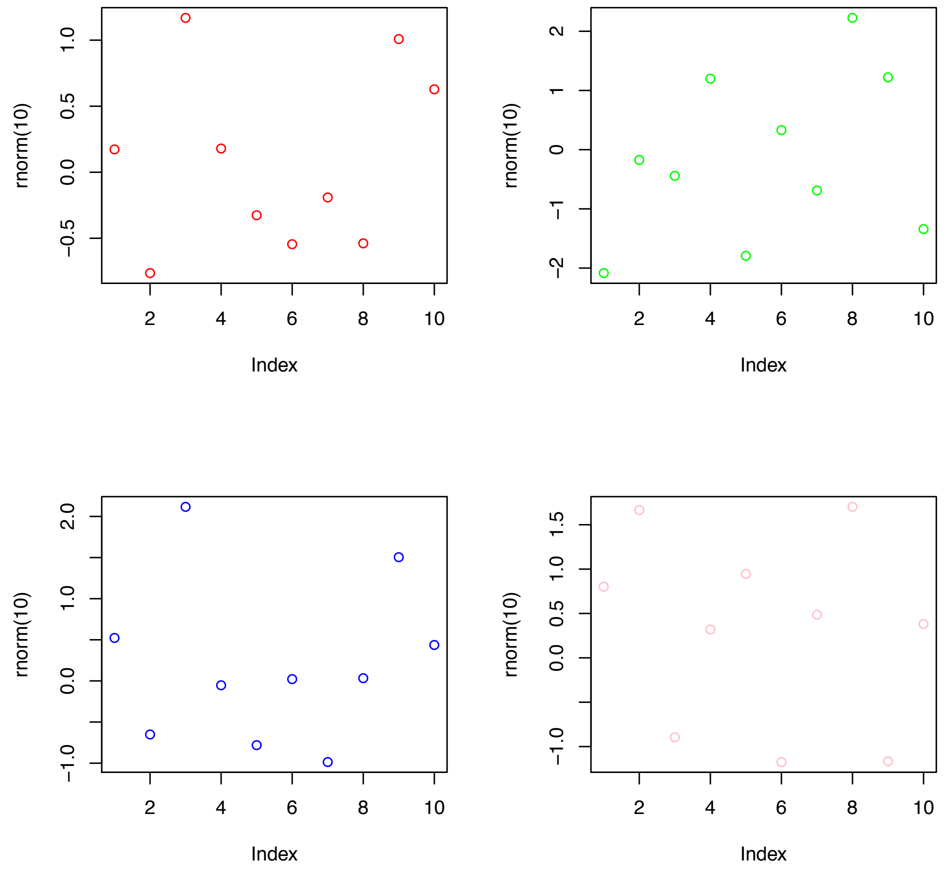
Add more lines:

```
trees=sort(unique(mydata$Tree))
subtrees=trees[-1]
for (item in subtrees){
  age=mydata$age[which(mydata$Tree==item)]
  cir=mydata$circumference[which(mydata$Tree==item)]
  lines(age,cir,col=item,type="o")
}
```



Multiple plots

```
par(mfrow=c(2,2))
plot(rnorm(10),col="red")
plot(rnorm(10),col="green")
plot(rnorm(10),col="blue")
plot(rnorm(10),col="pink")
```



Writing your own function

To define function:

```
f1 <- function(param1, param2, ... ){
   print(param1)
   return(param2)
}
```

Nested Function:

```
f2 <- function(p2,...){
   f1 <- function(p1,...){
      var1 <- log10(p1)
      return(var1)
   }
   var2 <- f1(p2)
   return(var2)
}
```

SNP genotyping

- From DNA to Protein
- Single-nucleotide polymorphism (SNP)
- BeadArray Microarray Technology

Quality Control Processes

- Hardy–Weinberg equilibrium test
- Linkage disequilibrium pruning
- Missing genotype filtering
- Minor Allele Frequency filtering
- Batch effect correction
- Population correction

Chi-square test

Single SNP data

	#Control	#Case	sum
AA	50	30	80
AT	30	20	50
TT	20	40	60
sum	100	90	190

Allele counts

	#Control	#Case	Sum
A	50*2+30=130	30*2+20=80	210
T	20*2+30=70	40*2+20=100	170
Sum	200	180	380

Chi-square test (2)

Observed allele counts

	#Control	#Case	Sum
A	130	80	210
T	70	100	170
Sum	200	180	380

Expected allele counts

	#Control	#Case
A	210*200/380=110.53	210*180/380=99.47
T	170*200/380=89.47	170*180/380=80.53

$$X^2 = \sum \frac{\left(Obs - Exp\right)^2}{Exp} \qquad X^2 = \frac{\left(130-110.53\right)^2}{110.53} + \frac{\left(80-99.47\right)^2}{99.47} + \frac{\left(70-89.47\right)^2}{89.47} + \frac{\left(100-80.53\right)^2}{80.53}$$

Degree of freedom = (#row-1) X (#col-1) = 1

R packages

- GenABEL package
  - Designed for GWA analysis
  - Graphical results of GWA analysis
- To install:

```
install.packages("GenABEL")
```

- genetics package
  - Classes and methods for handling genetic data
- To install:

```
install.packages("genetics")
```

- HardyWeinberg package
  - Tools for exploring Hardy-Weinberg equilibrium for diallelic genetic marker data
- To install:

```
install.packages("HardyWeinberg")
```

Bioconductor packages

- GWASTools package
  - Tools for Genome Wide Association Studies
- To install:

```
source("https://bioconductor.org/biocLite.R")
biocLite("GWASTools")
```

- SNPRelate package
  - Parallel Computing Toolset for Relatedness and Principal Component Analysis of SNP Data
- To install:

```
source("https://bioconductor.org/biocLite.R")
biocLite("SNPRelate")
```

File conversion with SNPRelate

- `library(SNPRelate)`
- `bed.fn = "genotype_850.bed"`
- `fam.fn = "genotype_850.fam"`
- `bim.fn = "genotype_850.bim"`
- `gdsfile = "genotype_850.gds"`
- `snpgdsBED2GDS(bed.fn, fam.fn, bim.fn, gdsfile, family=TRUE, cvt.chr="int", cvt.snpid="int", verbose=FALSE)`

Association test with GWASTools

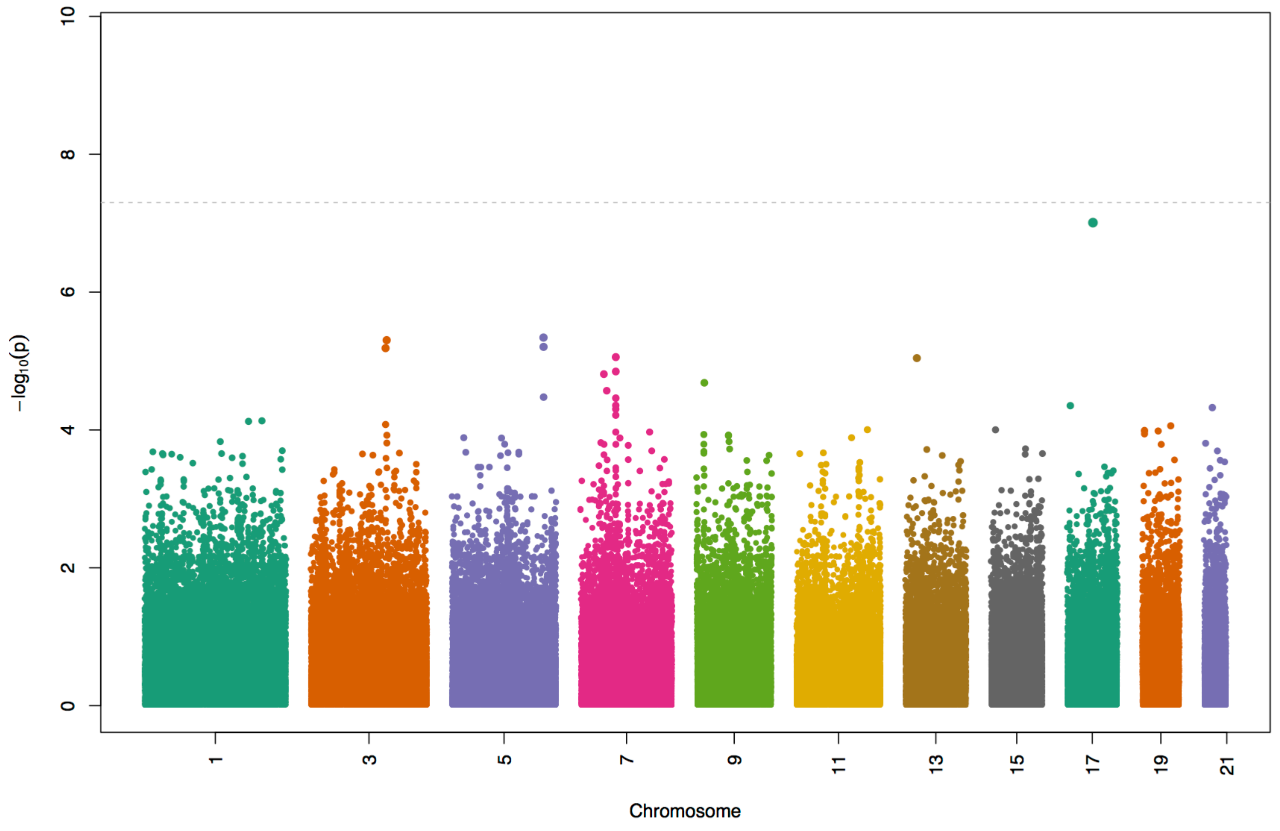
- `library(GWASTools)`
- `gdsfile = "genotype_850.gds"`
- `genoData = GdsGenotypeReader(gdsfile, YchromCode=24L, XYchromCode=25L)`

```
#Try to test the first 50 SNPs
```

- `assoc = assocRegression(genoData, outcome="status", model.type="logistic", snpStart=1, snpEnd=50)`
- `close(genoData)`

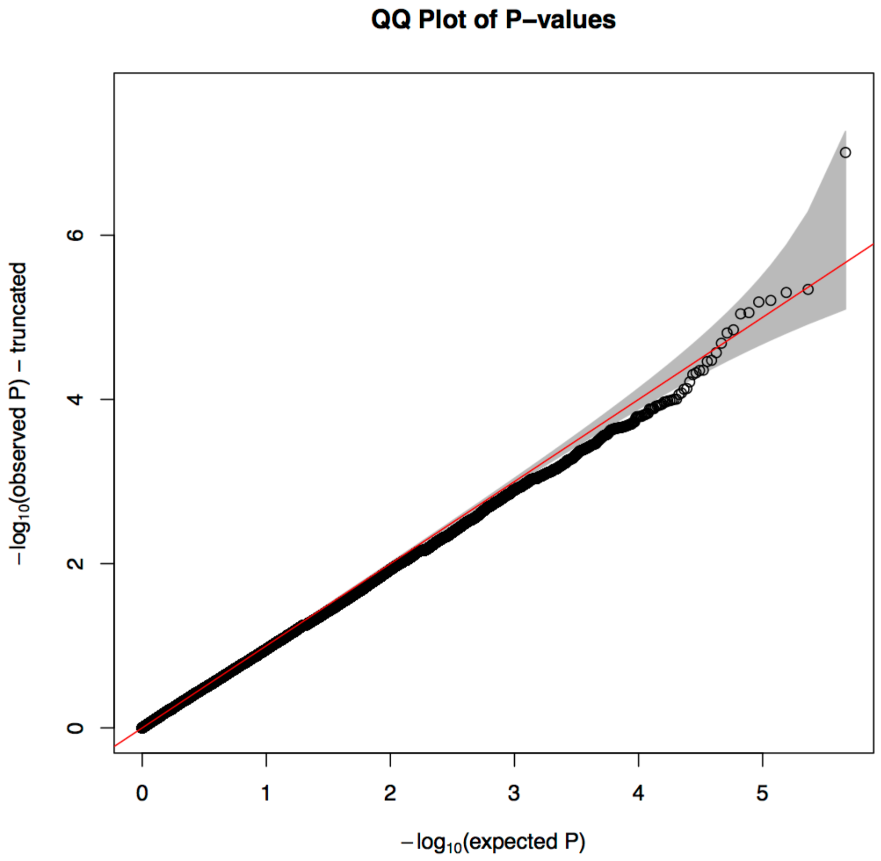
Manhattan plot

```
manhattanPlot(assoc$P,chromosome=assoc$CHR)
```

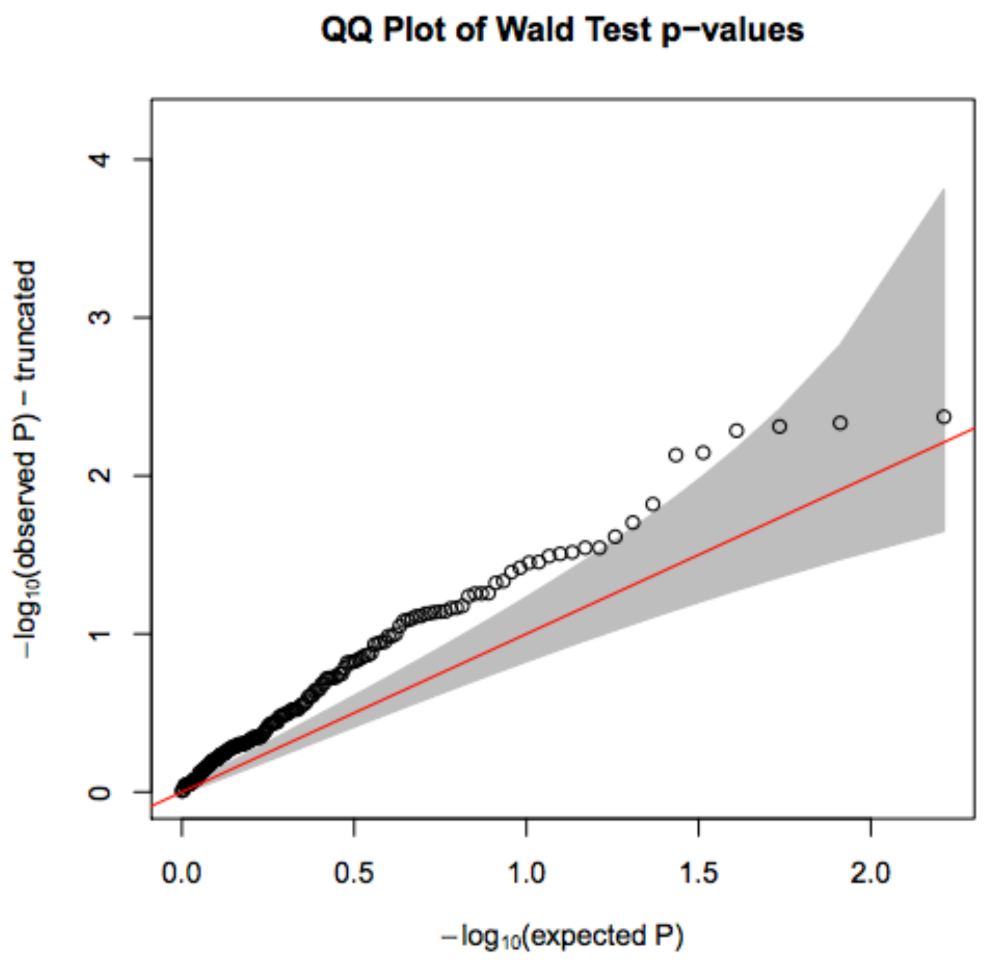


QQ plot

```
qqPlot(pval=assoc$P,truncate=TRUE, main="QQ
Plot of P-values")
```



Poor quality result



Good test statistic was used?

Ref: GWASTools

Alternative tests

- Fisher's exact test (GWASTools, PLINK)
  - similar to the chi-square test, but we use the fisher's exact test when the sample sizes are small.
- Linear Model (GWASTools, PLINK)
- Logistic Model (GWASTools, PLINK)
- Poisson Model (GWASTools)