Finding cis-eQTLs

Linear model for eQTL testing

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
e <- read.csv("data/geuv_chr22_100genes_expression.csv", row.names=1) # Gene expressions
head(e)[,1:3] # Rows represent genes
```

```
##                         HG00096     HG00097      HG00099
## ENSG00000249263.2   0.34065568  0.31894179 -0.009145373
## ENSG00000224688.1   4.19482678  3.36943992  2.335469635
## ENSG00000075240.12  3.53180308  3.63554131  1.251434011
## ENSG00000099937.6   0.51905438  0.39921626  0.078965142
## ENSG00000099998.12  0.07363038  0.04110881  0.017493349
## ENSG00000093072.10 13.34694664 13.49828357  2.517851727
```

```
start <- read.csv("data/geuv_chr22_100genes_position.csv", row.names=1) # Start site of genes
head(start)
```

```
##                                  Gene position
## ENSG00000249263.2   ENSG00000249263.2 17134782
## ENSG00000224688.1   ENSG00000224688.1 21495913
## ENSG00000075240.12 ENSG00000075240.12 46971909
## ENSG00000099937.6   ENSG00000099937.6 21128167
## ENSG00000099998.12 ENSG00000099998.12 24615622
## ENSG00000093072.10 ENSG00000093072.10 17660194
```

```
snp <- read.csv("data/chr22_snp.csv", row.names=1) # SNP genotypes
head(snp)[,1:3]
```

```
##          HG00096 HG00097 HG00099
## snp_1590       0       0       0
## snp_1591       0       0       1
## snp_1592       0       0       0
## snp_1593       0       1       1
## snp_1594       0       0       0
## snp_1595       0       0       0
```

```
snp.position <- read.csv("data/chr22_snp_position.csv", row.names=1) # SNP position
head(snp.position)
```

```
##          CHROM      POS
## snp_1590    22 17048696
## snp_1591    22 17049115
## snp_1592    22 17049827
## snp_1593    22 17049907
## snp_1594    22 17050088
## snp_1595    22 17050565
```

For each gene, we'll identify which SNPs are within 1000 bp of the gene start position (either upstream or downstream).

```
eqtl <- list()
for (i in 1:nrow(start)) { # For each gene
  tmp <- vector()
  for (j in 1:nrow(snp.position)) { # For each SNP
    if (snp.position[j,2] >= start[i,2]-1000 & snp.position[j,2] <= start[i,2]+1000) {
      tmp <- append(tmp, rownames(snp.position)[j])
    }
  }
  eqtl[[i]] <- tmp
}

head(eqtl) # For each of the 100 genes, a list of SNPs +-1000 bp of the gene's start site
```

```
## [[1]]
## [1] "snp_1828" "snp_1829" "snp_1830" "snp_1831" "snp_1832" "snp_1833"
##
## [[2]]
## logical(0)
##
## [[3]]
## [1] "snp_81122" "snp_81123" "snp_81124" "snp_81125" "snp_81126" "snp_81127"
## [7] "snp_81128" "snp_81129" "snp_81130"
##
## [[4]]
## [1] "snp_12759" "snp_12761" "snp_12762"
##
## [[5]]
## [1] "snp_23834" "snp_23835" "snp_23836"
##
## [[6]]
##  [1] "snp_3373" "snp_3374" "snp_3375" "snp_3376" "snp_3377" "snp_3378"
##  [7] "snp_3379" "snp_3380" "snp_3381" "snp_3382" "snp_3383" "snp_3384"
## [13] "snp_3385" "snp_3386" "snp_3387" "snp_3388" "snp_3389" "snp_3390"
```

Now, for each SNP within 1000 bp of gene start position, we're going to check if the SNP is associated with expression of the gene using the linear regression model

$$E_i = \mu + \beta_j G_{ij} + \epsilon_{ij}$$

where E_i is expression of gene in individual i and G_{ij} is genotype of SNP j.

We're going to do this **for each gene**.

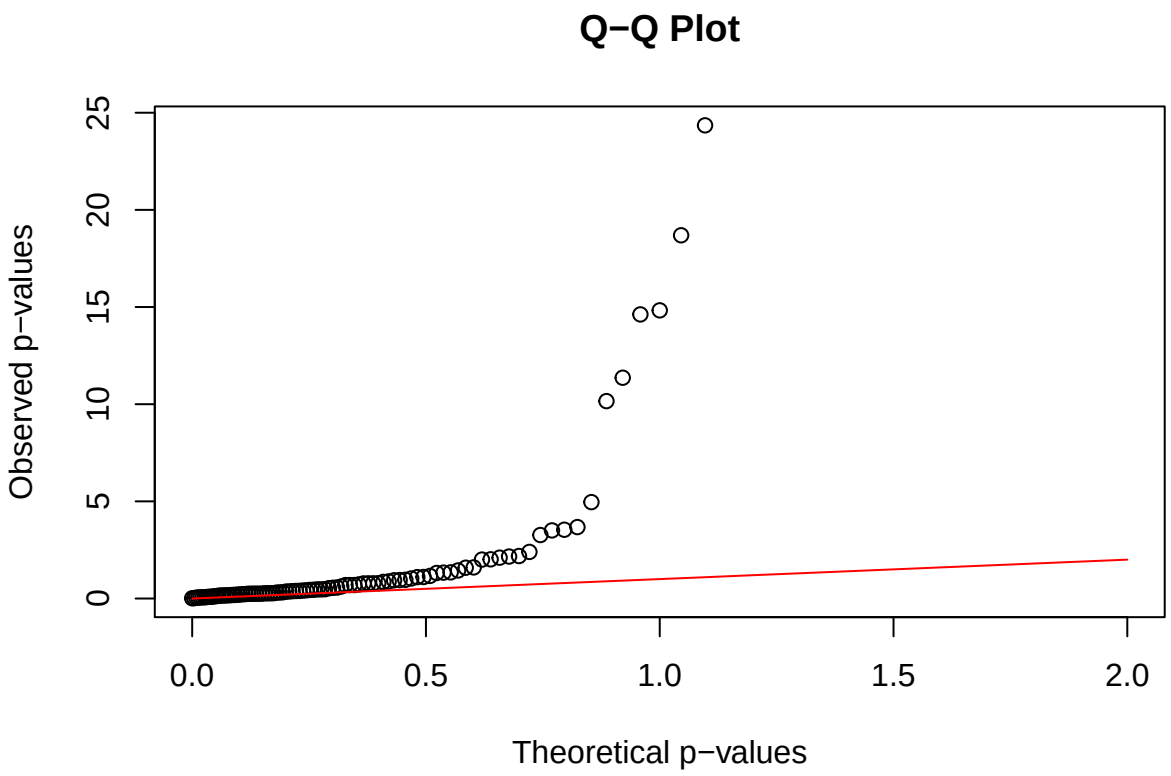
```
# Setup gene expression matrix so that it's an appropriate input for lm()
E <- as.data.frame(t(e)) # Rows now represent individuals

obs_p <- vector()
all.M <- list()
for (i in 1:ncol(E)) { # For each gene
  if (length(eqtl[[i]]) != 0) {
    # Prepare genotype matrix for the given gene
    g <- snp[eqtl[[i]],]
    G <- as.data.frame(t(g))

    M <- lm(as.matrix(E[,i]) ~ as.matrix(G)) # Apply linear regression
    all.M[[i]] <- M # Save result for each gene
    obs_p <- append(obs_p, anova(M)$'Pr(>F)'[1])
  }
  else {
    all.M[[i]] <- NULL
    obs_p <- append(obs_p, 0)
  }
}
# anova(M)$'Pr(>F)'[1]
# summary(M)$coefficients[,4]
```

Q-Q plot of p-values

```
# Since these tests can give very small p-values, we transform by taking -log10(p-value)
obs_p = -log10(sort(obs_p))
# Creating theoretical p-values that follow a uniform distribution
th_p = -log10(seq(1/100,1,1/100))
plot(th_p,obs_p, xlab="Theoretical p-values", ylab="Observed p-values", main="Q-Q Plot")
lines(th_p,th_p,col='red')
```



How many genes have at least one SNP with an association p-value less than 0.01?

```
##       result.genes         result.snps result.pvals
##  [1,] "ENSG00000040608.9"  "snp_11154" "0.00306879873900085"
##  [2,] "ENSG00000131100.8"  "snp_4948"  "0.00578227284560896"
##  [3,] "ENSG00000128185.5"  "snp_11289" "4.42895505988439e-06"
##  [4,] "ENSG00000233995.1"  "snp_1602"  "0.00094072367419325"
##  [5,] "ENSG00000228274.1"  "snp_60228" "0.000837432198472403"
##  [6,] "ENSG00000099940.6"  "snp_12924" "8.3018005042306e-10"
##  [7,] "ENSG00000167074.9"  "snp_65276" "1.85809411106647e-20"
##  [8,] "ENSG00000075234.12" "snp_80092" "2.54963223389296e-15"
##  [9,] "ENSG00000128218.7"  "snp_22399" "0.000307118325005016"
```