Gene Set Analysis

Contents

1 Load and prepare data and gene set **2**

2 Approaches based on association tests **3**

3 Gene set summary statistics **4**

4 Hypothesis testing for gene sets **6**

5 Permutations **8**

1 Load and prepare data and gene set

```
# Import packages
library(rafalib)
library(GSEABase)
library(GSE5859Subset)
library(sva)
library(limma)
library(matrixStats)
library(hgfocus.db)
data(GSE5859Subset)
```

```
# Check pheno data
sampleInfo[1:5, ]
```

```
##     ethnicity       date         filename group
## 107       ASN 2005-06-23 GSM136508.CEL.gz     1
## 122       ASN 2005-06-27 GSM136530.CEL.gz     1
## 113       ASN 2005-06-27 GSM136517.CEL.gz     1
## 163       ASN 2005-10-28 GSM136576.CEL.gz     1
## 153       ASN 2005-10-07 GSM136566.CEL.gz     1
```

```
# Check exprs data
geneExpression[1:5, 1:5]
```

```
##           GSM136508.CEL.gz GSM136530.CEL.gz GSM136517.CEL.gz GSM136576.CEL.gz
## 1007_s_at         6.543954         6.401470         6.298943         6.837899
## 1053_at           7.546708         7.263547         7.201699         7.052761
## 117_at            5.402622         5.050546         5.024917         5.304313
## 121_at            7.892544         7.707754         7.461886         7.558130
## 1255_g_at         3.242779         3.222804         3.185605         3.195363
##           GSM136566.CEL.gz
## 1007_s_at         6.470689
## 1053_at           6.980207
## 117_at            5.214149
## 121_at            7.819013
## 1255_g_at         3.251915
```

```
# Check feature data
geneAnnotation[1:5, ]
```

```
##      PROBEID  CHR     CHRLOC SYMBOL
## 1  1007_s_at chr6   30852327   DDR1
## 30   1053_at chr7  -73645832   RFC2
## 31    117_at chr1  161494036  HSPA6
## 32    121_at chr2 -113973574   PAX8
## 33 1255_g_at chr6   42123144 GUCA1A
```

```
# Correct for batch effects
X = sampleInfo$group
mod <- model.matrix( ~ X)
svafit <- sva(geneExpression, mod)
```

```
## Number of significant surrogate variables is:  5
## Iteration (out of 5 ):1  2  3  4  5
```

```r
svaX <- model.matrix( ~ X + svafit$sv)
lmfit <- lmFit(geneExpression, svaX)
tt <- lmfit$coef[, 2] * sqrt(lmfit$df.residual) / (2 * lmfit$sigma)
pval <- 2 * (1 - pt(abs(tt), lmfit$df.residual[1]))
qval <- p.adjust(pval, "BH")
```

```r
# Load gene set
gsets <- getGmt("Data/c1.all.v7.5.1.entrez.gmt.txt")

# Check some of the genes associated with a specific gene set (i.e., chrYq11)
geneIds(gsets[["chrYq11"]])[1:5]
```

```
## [1] "643034"    "100499416" "84672"     "106480250" "100128190"
```

```r
# Map gene set to Affymetrix
mapGMT2Affy <- function(object, gsets) {
  ann <- annotation(object)
  dbname <- paste(ann, "db", sep = ".")
  require(dbname, character.only = TRUE)
  gns <- featureNames(object)
  map <-
    select(get(dbname),
           keys = gns,
           columns = c("ENTREZID", "PROBEID"))
  map <- split(map[, 1], map[, 2])
  indexes <- sapply(gsets, function(ids) {
    gns2 <- unlist(map[geneIds(ids)])
    match(gns2, gns)
  })
  names(indexes) <- names(gsets)
  return(indexes)
}
rownames(sampleInfo) <- colnames(geneExpression)
e = ExpressionSet(
  assay = geneExpression,
  phenoData = AnnotatedDataFrame(sampleInfo),
  annotation = "hgfocus"
)
gsids <- mapGMT2Affy(e, gsets)
```

2 Approaches based on association tests

```r
# Apply a chi-square test for independence to see if differentially
# expressed genes are enriched in a given gene set
# (for one of the Y chromosome gene sets)
tab <-
  table(ingenset = 1:nrow(e) %in% gsids[["chrYq11"]],
        signif = qval < 0.05)
tab
```

```
##         signif
## ingenset FALSE TRUE
##    FALSE  8768    9
##    TRUE     12    4
```

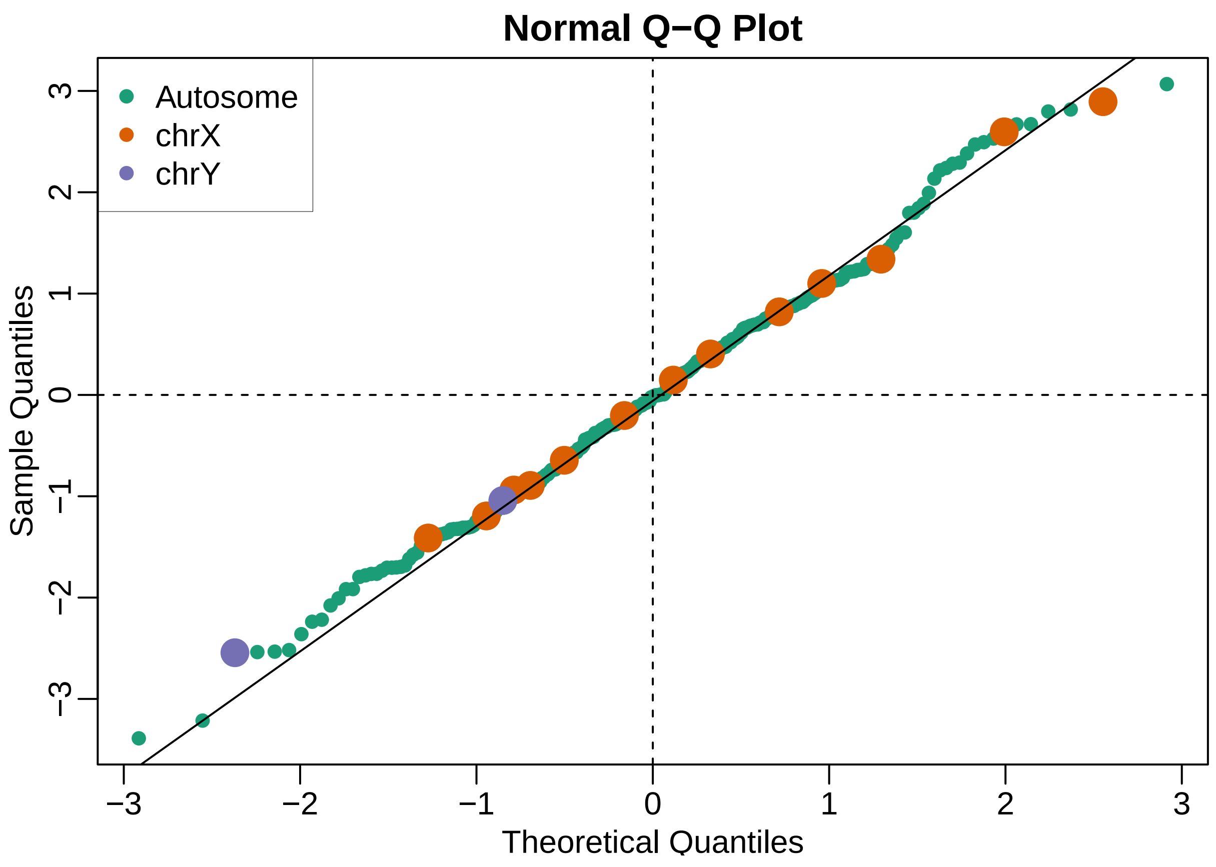
```r
chisq.test(tab)$p.val
```

```
## [1] 1.75031e-113
```

3 Gene set summary statistics

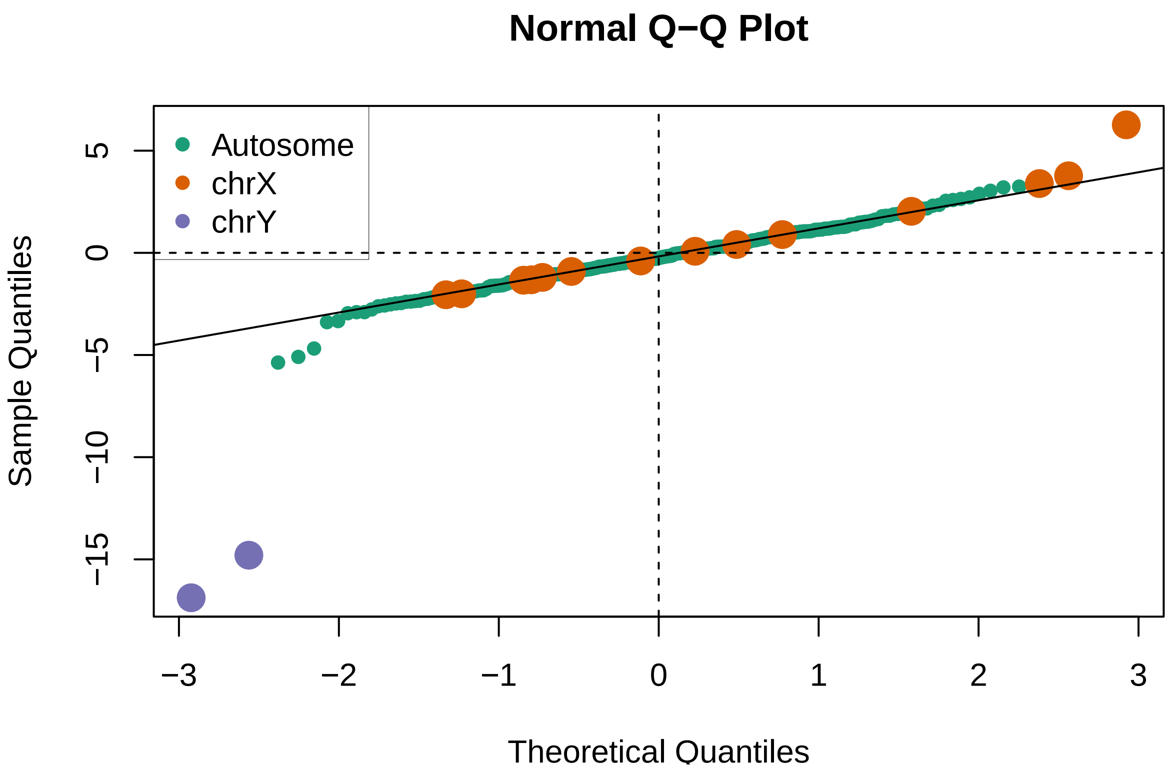
Wilcoxon Mann-Whitney test: The basic idea is to test if the difference between cases and controls is bigger in a given gene set compared to the rest of the genes.

```r
# Compute the Wilcoxon test for each gene set
es <- lmfit$coef[, 2]
wilcox <- function(i) {
  if (length(i) > 2) {
    tmp <- wilcox.test(es[i], es[-i])
    n1 <- length(i)
    n2 <- length(es) - n1
    ## Standardize the stats to have mean 0 & SD 1
    z <- (tmp$stat - n1 * n2 / 2) / sqrt(n1 * n2 * (n1 + n2 + 1) / 12)
    return(c(z, tmp$p.value))
  } else
    return(rep(NA, 2))
}

wilcox_stats <- t(sapply(gsids, wilcox))

## If the gene sets are large enough and the effect size for each gene are
# independent of each other then these statistics will follow
# a normal distribution.

mypar(1, 1)
cols <- rep(1, nrow(wilcox_stats))
cols[grep("chrX", rownames(wilcox_stats))] <- 2
cols[grep("chrY", rownames(wilcox_stats))] <- 3
qqnorm(wilcox_stats[, 1],
       col = cols,
       cex = ifelse(cols == 1, 1, 2),
       pch = 16)
qqline(wilcox_stats[, 1])
abline(h = 0, v = 0, lty = 2)
legend(
  "topleft",
  c("Autosome", "chrX", "chrY"),
  pch = 16,
  col = 1:3,
  box.lwd = 0
)
```



A very simple summary we can construct to test for mean shifts (shift to the left (generally more under expressed genes) or the right (generally more over expressed genes)) is to average the t-statistic in gene set.

```
avgt <- function(i) {
  sqrt(length(i)) * mean(tt[i])
}

avgt_stats <- sapply(gsids, avgt)
qqnorm(avgt_stats,
       col = cols,
       cex = ifelse(cols == 1, 1, 2),
       pch = 16)
qqline(avgt_stats)
abline(h = 0, v = 0, lty = 2)
legend(
  "topleft",
  c("Autosome", "chrX", "chrY"),
  pch = 16,
  col = 1:3,
  box.lwd = 0
)
```



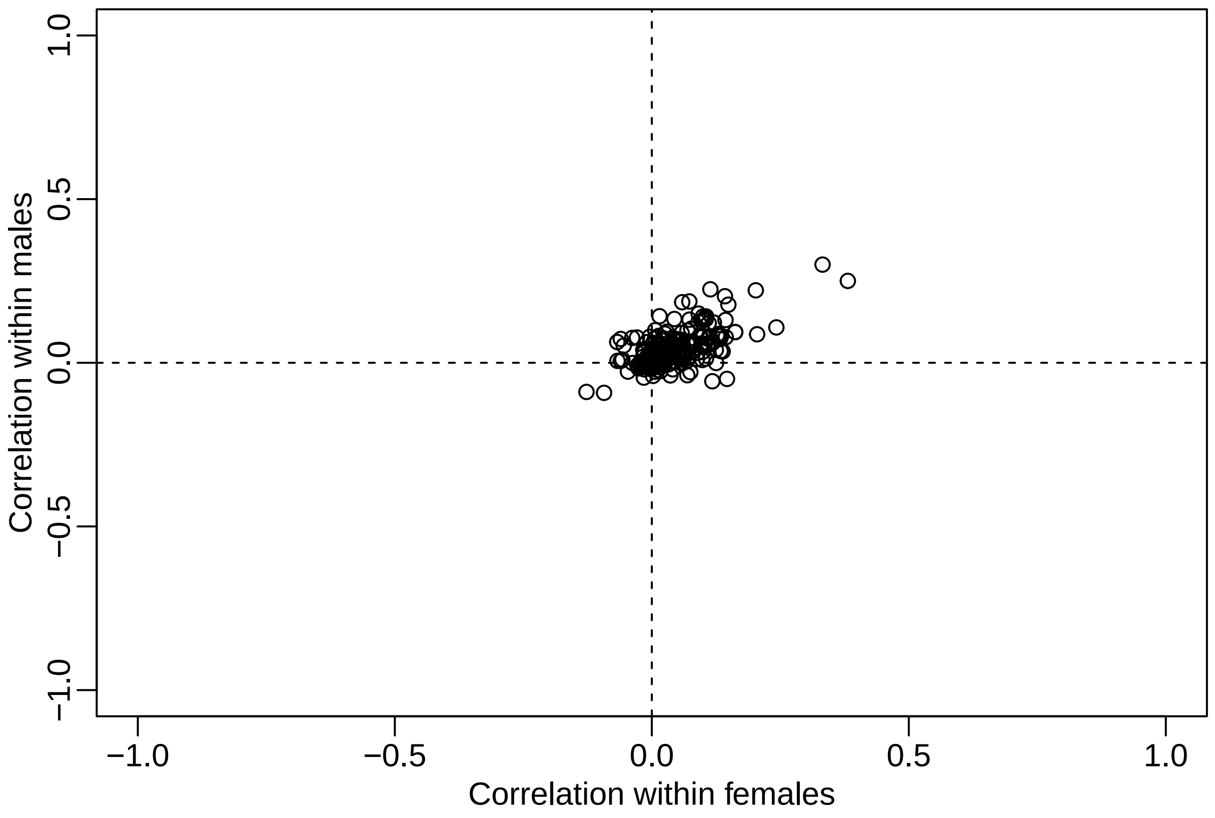
Note that under the null hypothesis that the genes in the gene set are not differentially expressed and t-statistics are independent of each other, the distribution should be approximately normal.

4 Hypothesis testing for gene sets

The above test assumed that t-statistics are independent and identically distributed. However, we can see the pairwise correlations in each gene set as follows:

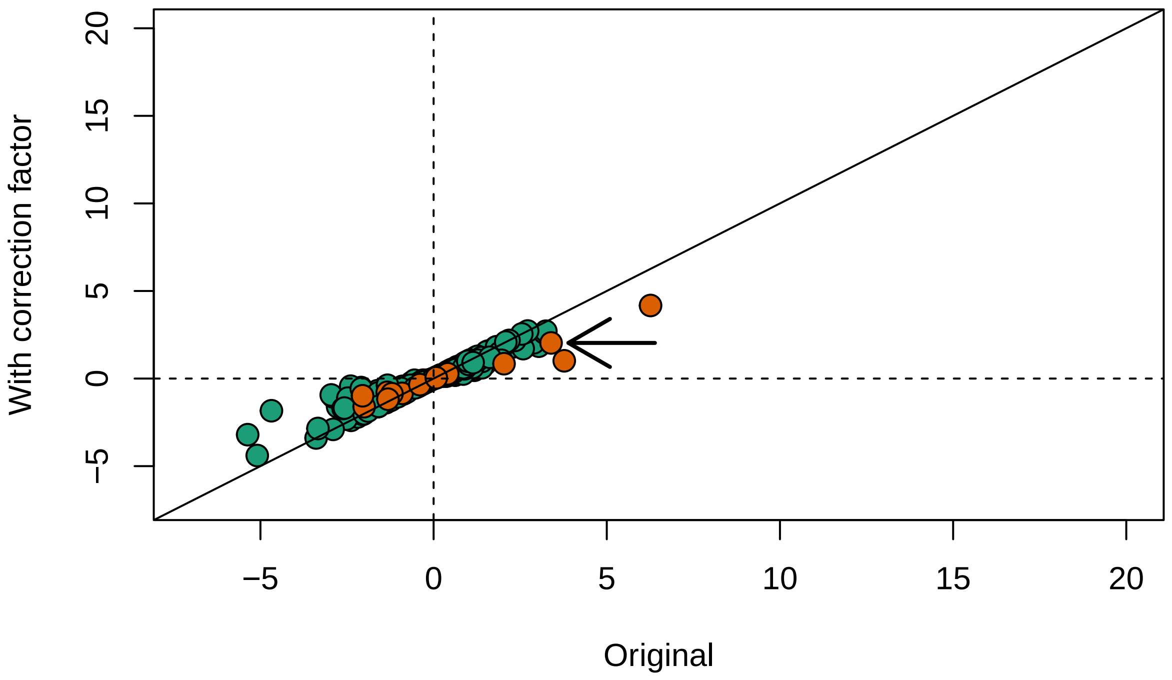
```r
N <- sapply(gsids, length)
ind1 <- which(X == 0)
ind2 <- which(X == 1)
cal_corrs <- function(ind) {
  if (length(ind) >= 2) {
    cc1 <- cor(t(geneExpression[ind, ind1]))
    cc2 <- cor(t(geneExpression[ind, ind2]))
    return(c(median(cc1[lower.tri(cc1)]),
             median(cc2[lower.tri(cc2)])))
  } else
    return(c(NA, NA))
}

corrs <- t(sapply(gsids, cal_corrs))
mypar(1, 1)
plot(
  corrs[N > 10, ],
  xlim = c(-1, 1),
  ylim = c(-1, 1),
  xlab = "Correlation within females",
  ylab = "Correlation within males"
)
```

```
abline(h = 0, v = 0, lty = 2)
```



Using the below method, we can correct the average t-statistic for the correlatiobs in gene sets.

```
avgcorrs <- rowMeans(corrs)
## Correction factor
cf <- (1 + (N - 1) * avgcorrs)
## Ignore negative correlations
cf[cf < 1] <- 1
## Corrected average t
correctedavgt <- avgt_stats / sqrt(cf)
## Plot
plot(
  avgt_stats,
  correctedavgt,
  bg = cols,
  pch = 21,
  xlab = "Original",
  ylab = "With correction factor",
  xlim = c(-7, 20),
  ylim = c(-7, 20),
  cex = 1.5
)
abline(0, 1)
abline(h = 0, v = 0, lty = 2)
thirdhighest <- order(-avgt_stats)[3]
arrows(avgt_stats[thirdhighest] + 3,
       correctedavgt[thirdhighest],
       x1 = avgt_stats[thirdhighest] +
         0.5,
       lwd = 2)
```

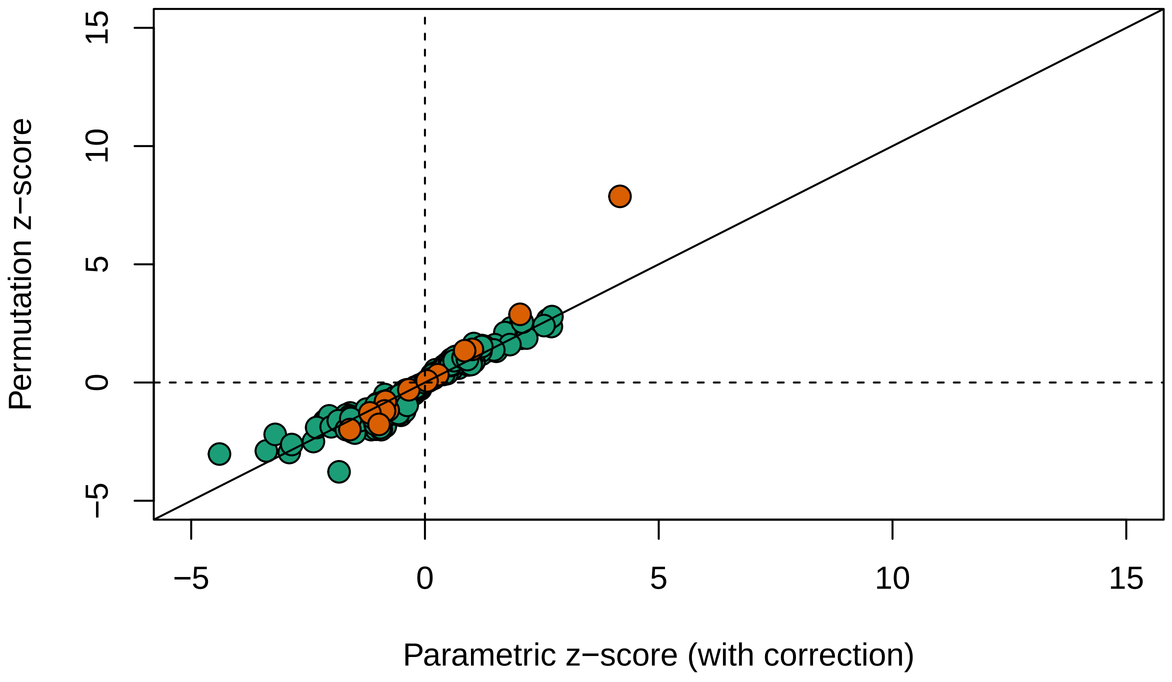


Note in particular, the third highest value (arrow) is now quite close to 0 and is no longer significant.

5 Permutations

Permutations are non-parametric tests.

```
# Create null distributions
null_dist <- function(b) {
  nullX <- sample(X)
  nullsvaX <-
    model.matrix( ~ nullX + svafit$sv)
  nulllmfit <- lmFit(geneExpression, nullsvaX)
  nulltt <-
    nulllmfit$coef[, 2] * sqrt(nulllmfit$df.residual) / (2 * nulllmfit$sigma)
  nullavgt <-
    sapply(gsids, function(i)
      sqrt(length(i)) * mean(nulltt[i]))
  return(nullavgt)
}

null <- sapply(1:400, null_dist)

# Permutation test for average t stats
permavgt <- avgt_stats / rowSds(null)
permpval <- rowMeans(abs(avgt_stats) < abs(null))

# Plot
plot(
  correctedavgt,
  permavgt,
  bg = cols,
  pch = 21,
  xlab = "Parametric z-score (with correction)",
  ylab = "Permutation z-score",
  cex = 1.5,
  ylim = c(-5, 15),
  xlim = c(-5, 15)
)
abline(0, 1)
abline(h = 0, v = 0, lty = 2)
```



In this case the results using permutations are very similar to the results using the parametric approach that uses the correction factor as can be seen in this comparison.

Note: One limitation of the permutation test approach is that to estimate very small p-values we need many permutations.