Gridsemble on Platinum Spike Dataset - Dataset

Jenna Landy

2023-12-21

```
library(tidyverse)
library(tibble)

library(GEOquery)
library(affy)
library(genefilter)

source("PAPER_metrics_helpers.R")
```

Data

Access data with `GEOQuery` R package. This will download some files in a directory named `GSE21344`. The `geo` object contains an `ExpressionSet` object, but we see that it is empty. Instead, we load expression data from the downloaded files.

```
geo <- GEOquery::getGEO("GSE21344")
```

```
Found 1 file(s)
```

```
GSE21344_series_matrix.txt.gz
```

```
class(geo$GSE21344_series_matrix.txt.gz)
```

```
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

```
dim(exprs(geo$GSE21344_series_matrix.txt.gz))
```

```
[1]  0 18
```

Expression Data

Expression data for each sample is in a supplementary file. The `getGEOSuppFiles` function downloads these files and saves them in `GSE21344_RAW.tar` in sub-directory called `GSE21344`. We can decompress this with the `untar` function, and save each contained file into the same sub-directory by setting `exdir = "GSE21344"`.

```
# download supplemental files from GEO browser
dir = "GSE21344"
supp <- getGEOSuppFiles(dir)
untar(
  paste0(dir,"/GSE21344_RAW.tar"),
  exdir = dir
)
```

Each sample has a respective `.cel.gz` file. We can read all these files at once with the `affy::ReadAffy()` function, which returns an `AffyBatch` object.

```
# directory of RAW supplementary files downloaded
# there is a .cel file for each observation
files =  list.files(dir)
files = files[endsWith(files, 'cel.gz')]
Data = affy::ReadAffy(
  filenames = paste(dir, files, sep = '/')
)
class(Data)
```

```
[1] "AffyBatch"
attr(,"package")
[1] "affy"
```

The `affy::rma()` function converts an `AffyBatch` object into an `ExpressionSet` object using the robust multi-array average (RMA) expression measure.

```
eset = affy::rma(Data)
```

```
Warning: replacing previous import 'AnnotationDbi::tail' by 'utils::tail' when
loading 'drosophila2cdf'

Warning: replacing previous import 'AnnotationDbi::head' by 'utils::head' when
loading 'drosophila2cdf'
```

```
Background correcting
Normalizing
Calculating Expression
```

```
class(eset)
```

```
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

Expression data can be accessed with `Biobase::exprs()` function.

```
expression = Biobase::exprs(eset)

# remove .cel.gz from column names
colnames(expression) <- stringr::str_replace(colnames(expression),'.cel.gz','')

expression[1:5, 1:2]
```

```
             GSM533369 GSM533370
1616608_a_at  3.075607  2.987812
1622892_s_at 11.411883 11.415364
1622893_at    3.061561  3.233674
1622894_at    3.930225  3.829407
1622895_at   10.966323 10.857122
```

```r
paste(dim(expression), c("genes", "samples"))
```

```
[1] "18952 genes" "18 samples"
```

Phenotype Data

We can extract phenotype data from the `geo$GSE21344_series_matrix.txt.gz` object with the `Biobase::pData()` function.

```r
pheno <- Biobase::pData(geo$GSE21344_series_matrix.txt.gz)
```

We first check that this phenotype data is in the same order as the expression data.

```r
all(colnames(expression) == rownames(pheno))
```

```
[1] TRUE
```

```r
all(colnames(expression) == pheno$geo_accession)
```

```
[1] TRUE
```

We can extract condition and technical replicate information from the `description.4` column of this data frame.

```r
labels <- pheno %>%
  select(geo_accession, description.4) %>%
  mutate(
    condition = substring(description.4, 1, 1),
    sample = substring(description.4, 2, 2),
    replicate = substring(description.4, 3)
  )
head(labels)
```

```
          geo_accession description.4 condition sample replicate
GSM533369     GSM533369       A1alpha         A      1     alpha
GSM533370     GSM533370        A1beta         A      1      beta
GSM533371     GSM533371       A1gamma         A      1     gamma
GSM533372     GSM533372       A2alpha         A      2     alpha
GSM533373     GSM533373        A2beta         A      2      beta
GSM533374     GSM533374       A2gamma         A      2     gamma
```

We average across technical replicates. The new data frame we create here `expression_per_sample` has one row representing one sample (averaged across three technical replicates). Column names are the conditions (A or B).

```r
labels_grouped = labels %>%
  group_by(condition, sample) %>%
  summarize(
    id = list(geo_accession),
    .groups = "keep"
  ) %>%
  ungroup() %>%
  select(-sample) %>%
  deframe()

expression_per_sample <- lapply(
    labels_grouped,
    function(ids) {
      rowMeans(expression[,ids])
    }
  ) %>%
  do.call(cbind, .)

dim(expression_per_sample)
```

```
[1] 18952     6
```

```r
expression_per_sample[1:2,]
```

```
                     A         A         A         B         B         B
1616608_a_at  3.061806  2.847145  3.042839  3.139057  3.087251  3.010351
1622892_s_at 11.409214 11.767090 11.700351 11.472629 11.852718 11.581252
```

Fold-Change Data

The fold-change data that tells us what probes are differentially expressed is in a supplemental file of the Platinum Spike Paper and can be read directly from the web.

```
fold_change = read.table("https://static-content.springer.com/esm/art%3A10.1186%2F1471-210

colnames(fold_change) = c('affy','FC')

head(fold_change)
```

```
          affy   FC
1 1616608_a_at    0
2 1622892_s_at   MC
3   1622893_at 0.25
4   1622894_at    0
5   1622895_at    1
6   1622896_at  3.5
```

Here we check that each probes in our expression data is also in this fold-change dataset.

```
setequal(
  rownames(expression_per_sample),
  fold_change$affy
)
```

```
[1] TRUE
```

The platinum spike paper notes that empty probe sets were assigned with the value zero. "MC" means the corresponding probe set is assigned to multiple clones. "MF" means the clone assigned to the particular probe set is present in multiple pools and therefore has multiple fold change values. We filter out all probes that fall in one of these three categories.

```
fold_change_FILETERED <- fold_change %>%
  filter(FC != '0' & FC != 'MC' & FC != 'MF') %>%
  mutate(FC = as.numeric(FC))

dim(expression_per_sample)
```

```
[1] 18952     6
```

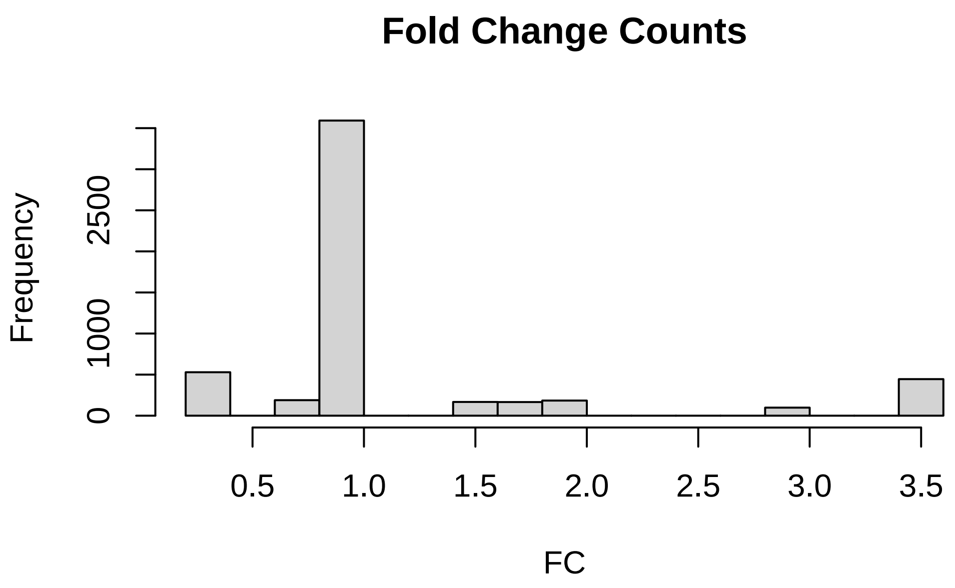
```r
# this also makes expression match order of fold change
expression_per_sample_FILTERED <- expression_per_sample[fold_change_FILETERED$affy,]

dim(expression_per_sample_FILTERED)
```

```
[1] 5370    6
```

Now we can look at the ranges of fold changes, and define "differentially expressed" genes as those with FC $\neq$ 1.

```r
hist(
  fold_change_FILETERED$FC,
  main = "Fold Change Counts",
  xlab = "FC"
)
```



```r
fold_change_FILETERED <- fold_change_FILETERED %>%
  mutate(
    DE = FC != 1
  )
table(fold_change_FILETERED$DE)
```

```
FALSE  TRUE 
 3426  1944 
```

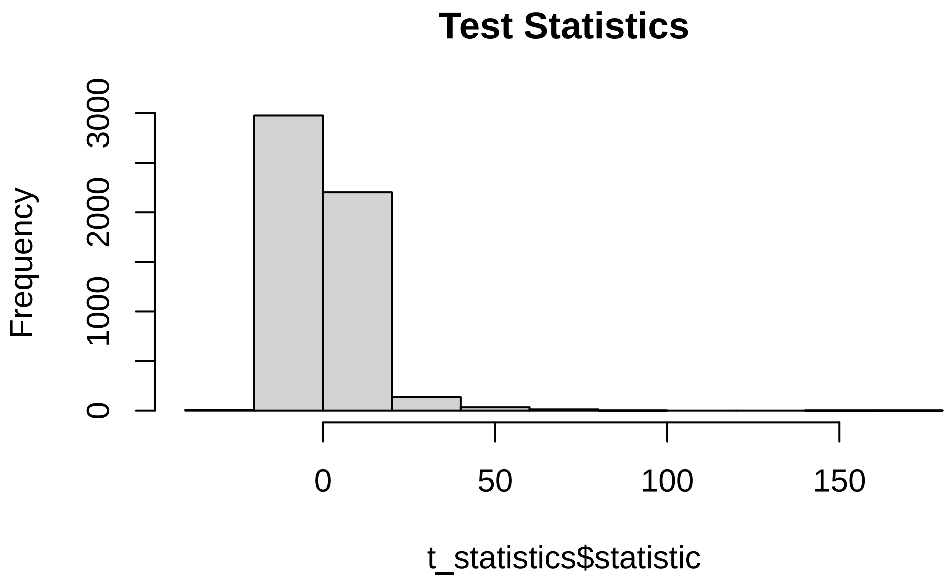
```
pi0 = mean(fold_change_FILETERED$DE == 0)
pi0
```

```
[1] 0.6379888
```

Now we perform two-sample t-tests and plot the resulting test statistics.

```
t_statistics = genefilter::rowttests(
  expression_per_sample_FILTERED,
  as.factor(colnames(expression_per_sample_FILTERED))
)

hist(t_statistics$statistic, main = "Test Statistics")
```



Save

```r
platinum_data <- list(
  expression = expression_per_sample_FILTERED,
  fold_change = fold_change_FILETERED,
  pheno = pheno,
  statistics = t_statistics$statistic,
  Fdr = get_true_Fdr(
    t_statistics$statistic,
    fold_change_FILETERED$DE
  )
)

save(platinum_data, file = "PAPER_platinum_data.RData")
```