Lab 03: Breast cancer methylation analysis

Sean Cho

Overview

```
library(GEOquery)
library(minfi)
library(limma)

## source custom functions for the workshop
## heatmap
source('https://raw.githubusercontent.com/sean-cho/biotrac2018_epigenetics/master/code/custom_fxn.R')
## hypergeometric test
source('https://raw.githubusercontent.com/sean-cho/biotrac2018_epigenetics/master/code/sig_overlap.R')
## readGMT
source('https://raw.githubusercontent.com/sean-cho/biotrac2018_epigenetics/master/code/readgmt.R')

## other functions
factor2char <- function(x){
  if(is.factor(x)){
    return(as.character(x))
  } else {
    return(x)
  }
}
```

For this task, we will be analysing a public dataset on the Gene Expression Omnibus (GEO), GSE20713. This GSE contains two datasets, an expression and a methylation dataset. We will be working with the latter.

The methylation dataset contains 273 samples; 236 tumors, 12 normals, and 25 cell lines. There are 4 batches of data.

We will be doing the following:

1. Download the data
2. Process the metadata
3. Perform QC analysis using our own script
4. Normalize the data and create analysis ready data
5. Exploratory analyses: cluster analysis
6. Identify differentially methylated probes (DMPs)
7. Run gene set enrichment analysis

Download dataset

We will be using the `GEOquery` package to download the GSE20713 dataset. The raw data in the form of IDAT is unavailable for this dataset, but the authors have uploaded the raw, unprocessed beta-values onto GEO. While it is not ideal, we can still perform QC and normalization on the data using methods that do not depend on probe level intensities.

We will first download the data, in the format of raw beta-values.

```
## get phenodata
if(!file.exists('rda/breast_gse.rds')){
```

```
  breast <- getGEO('GSE20713')
  saveRDS(breast, file = 'rda/breast_gse.rds')
} else {
  breast <- readRDS('rda/breast_gse.rds')
}

## assign the expression and methylation datasets
## to appropriate objects
breast_expr <- breast[[1]]
breast_meth <- breast[[2]]

class(breast_meth)
```

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

Also, note that the `class` for these datasets are `ExpressionSets`, which are data classes that are a little more processed compared to `RGChannelSets`. `ExpressionSets` store:

1. `exprs`: the data from the array
2. `pData`: phenotype data about the samples
3. `fData`: feature data about the probes in the array

Explore annotation and add annotations

Before we dive into the data, let's first look at the feature annotations, or probe annotations, of this microarray. There is some information here, including the gene the probe maps to, whether the probe is in a CpG island, and where it is located in the genome.

We will add a new annotation called `Promoter`, which flags a probe if it is within 1500bp to a transcriptional start site (TSS). This annotation will be used later on in visualizing our data and making sense of our data.

```
annots <- fData(breast_meth)
str(annots, vec.len = 2, nchar.max = 20, max.level = 2)
```

```
## 'data.frame':    27578 obs. of  38 variables:
##  $ ID                 : chr  "cg00000292" "cg00002426" ...
##  $ Name               : chr  "cg00000292" "cg00002426" ...
##  $ IlmnStrand         : chr  "TOP" "TOP" ...
##  $ AddressA_ID        : int  99037| __truncated__ ...
##  $ AlleleA_ProbeSeq   : chr  "AAA"| __truncated__ "AAT"| __truncated__ ...
##  $ AddressB_ID        : int  66606| __truncated__ ...
##  $ AlleleB_ProbeSeq   : chr  "AAA"| __truncated__ "AAT"| __truncated__ ...
##  $ GenomeBuild        : int  36 36 36 36 36 ...
##  $ Chr                : chr  "16" "3" ...
##  $ MapInfo            : int  28797| __truncated__ ...
##  $ Ploidy             : chr  "diploid" "diploid" ...
##  $ Species            : chr  "Homo sapiens" "Homo sapiens" ...
##  $ Source             : chr  "NCBI:RefSeq" "NCBI:RefSeq" ...
##  $ SourceVersion      : num  36.1 | __truncated__ ...
##  $ SourceStrand       : chr  "TOP" "TOP" ...
##  $ SourceSeq          : chr  "CGG"| __truncated__ "CGC"| __truncated__ ...
##  $ TopGenomicSeq      : chr  "TGG"| __truncated__ "CCG"| __truncated__ ...
##  $ Next_Base          : chr  "T" "T" ...
```

```
##  $ Color_Channel       : chr  "Red" "Red" ...
##  $ TSS_Coordinate      : int  28797| __truncated__ ...
##  $ Gene_Strand         : chr  "+" "+" ...
##  $ Gene_ID             : chr  "GeneID:487" "GeneID:7871" ...
##  $ Symbol              : chr  "ATP2A1" "SLMAP" ...
##  $ Synonym             : chr  "ATP2A; SERCA1;" "SLAP; KIAA1601;" ...
##  $ Accession           : chr  "NM_173201.2" "NM_007159.2" ...
##  $ GID                 : chr  "GI:47132613" "GI:56550042" ...
##  $ Annotation          : chr  "iso"| __truncated__ "Sar"| __truncated__ ...
##  $ Product             : chr  "ATP"| __truncated__ "sar"| __truncated__ ...
##  $ Distance_to_TSS     : int  291 369 432 268 671 ...
##  $ CPG_ISLAND          : logi  TRUE TRUE TRUE ...
##  $ CPG_ISLAND_LOCATIONS: chr  "16:"| __truncated__ "3:5"| __truncated__ ...
##  $ MIR_CPG_ISLAND      : chr  "" "" ...
##  $ RANGE_GB            : chr  "NC_000016.8" "NC_000003.10" ...
##  $ RANGE_START         : int  28797| __truncated__ ...
##  $ RANGE_END           : int  28797| __truncated__ ...
##  $ RANGE_STRAND        : chr  "+" "+" ...
##  $ GB_ACC              : chr  "NM_173201.2" "NM_007159.2" ...
##  $ ORF                 : int  487 7| __truncated__ ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 38
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

```
annots$Promoter <- ifelse(is.na(annots$Distance_to_TSS),
                          'no','yes')
```

Process metadata

Here, we will wrangle the phenodata.

```
str(pData(breast_meth), vec.len = 2)
```

```
## 'data.frame':    273 obs. of  77 variables:
##  $ title                  : Factor w/ 273 levels "Breast cancer cell line BT20 5 AZA (methylation da
##  $ geo_accession          : chr  "GSM519817" "GSM519818" ...
##  $ status                 : Factor w/ 1 level "Public on Oct 19 2011": 1 1 1 1 1 ...
##  $ submission_date        : Factor w/ 4 levels "Jun 09 2010",..: 3 3 3 3 3 ...
##  $ last_update_date       : Factor w/ 1 level "Oct 19 2011": 1 1 1 1 1 ...
##  $ type                   : Factor w/ 1 level "genomic": 1 1 1 1 1 ...
##  $ channel_count          : Factor w/ 1 level "1": 1 1 1 1 1 ...
##  $ source_name_ch1        : Factor w/ 9 levels "Breast cancer cell line",..: 2 2 2 2 2 ...
##  $ organism_ch1           : Factor w/ 1 level "Homo sapiens": 1 1 1 1 1 ...
##  $ characteristics_ch1    : Factor w/ 13 levels "conditions: 5-AZA",..: 11 11 11 11 11 ...
##  $ characteristics_ch1.1  : Factor w/ 132 levels "cell type: HCT116 cell line",..: 5 27 84 97 98 ...
##  $ characteristics_ch1.2  : Factor w/ 150 levels "genome/variation: DNMT1 and DNMT3B double knock ou
##  $ characteristics_ch1.3  : Factor w/ 22 levels "","cell line: BT20",..: 16 16 14 16 15 ...
##  $ characteristics_ch1.4  : Factor w/ 21 levels "","agent: 5-AZA",..: 20 20 19 20 20 ...
##  $ characteristics_ch1.5  : Factor w/ 14 levels "","methylation barcode: 5324215023_A",..: 13 13 12
##  $ characteristics_ch1.6  : Factor w/ 7 levels "","agebin: 0",..: 3 3 3 3 2 ...
##  $ characteristics_ch1.7  : Factor w/ 7 levels "","age bin: 0",..: 5 5 6 5 5 ...
##  $ characteristics_ch1.8  : Factor w/ 7 levels "","er: 0","er: 1",..: 6 6 5 6 5 ...
```

```
##  $ characteristics_ch1.9  : Factor w/ 9 levels "","her2: 0","her2: 1",..: 6 6 7 5 7 ...
##  $ characteristics_ch1.10 : Factor w/ 4 levels "","e.rfs: 0",..: 3 2 2 2 2 ...
##  $ characteristics_ch1.11 : Factor w/ 205 levels "","t.rfs: 0",..: 24 67 182 57 55 ...
##  $ characteristics_ch1.12 : Factor w/ 4 levels "","e.os: 0","e.os: 1",..: 2 3 2 2 2 ...
##  $ characteristics_ch1.13 : Factor w/ 210 levels "","t.os: 0","t.os: 0.03",..: 191 69 184 53 50 ...
##  $ treatment_protocol_ch1 : Factor w/ 2 levels "","Cells were harvested after 5 or 6 passages. They w
##  $ growth_protocol_ch1    : Factor w/ 2 levels "","HCT116 cells were maintained in McCoy's 5A medium
##  $ molecule_ch1           : Factor w/ 1 level "genomic DNA": 1 1 1 1 1 ...
##  $ extract_protocol_ch1   : Factor w/ 3 levels "Genomic DNA from frozen samples was extracted from tw
##  $ extract_protocol_ch1.1 : Factor w/ 2 levels "","Genomic DNA from  cells was extracted using the Q
##  $ label_ch1              : Factor w/ 2 levels "biotin and DNP",..: 1 1 1 1 1 ...
##  $ label_protocol_ch1     : Factor w/ 2 levels "Labelled nucleotides were added to extend the primers
##  $ taxid_ch1              : Factor w/ 1 level "9606": 1 1 1 1 1 ...
##  $ hyb_protocol           : Factor w/ 1 level "DNA samples were hybridized onto the HumanMethylation1
##  $ hyb_protocol.1         : Factor w/ 2 levels "","Labelled nucleotides were added to extend the prim
##  $ scan_protocol          : Factor w/ 2 levels "Labelled nucleotides were added to extend the primers
##  $ description            : Factor w/ 2 levels "","no additional information": 2 2 2 2 2 ...
##  $ data_processing        : Factor w/ 3 levels "Data analysis was performed with the Methylation modu
##  $ data_processing.1      : Factor w/ 1 level "": 1 1 1 1 1 ...
##  $ data_processing.2      : Factor w/ 2 levels "","Average beta value is defined as the ratio of sign
##  $ platform_id            : Factor w/ 1 level "GPL8490": 1 1 1 1 1 ...
##  $ contact_name           : Factor w/ 1 level "Benjamin,,Haibe-Kains": 1 1 1 1 1 ...
##  $ contact_email          : Factor w/ 1 level "benjamin.haibe.kains@utoronto.ca": 1 1 1 1 1 ...
##  $ contact_phone          : Factor w/ 1 level "+14165818626": 1 1 1 1 1 ...
##  $ contact_laboratory     : Factor w/ 1 level "Bioinformatics and Computational Genomics": 1 1 1 1 1
##  $ contact_department     : Factor w/ 1 level "Princess Margaret Research": 1 1 1 1 1 ...
##  $ contact_institute      : Factor w/ 1 level "Princess Margaret Cancer Centre": 1 1 1 1 1 ...
##  $ contact_address        : Factor w/ 1 level "610 University Avenue": 1 1 1 1 1 ...
##  $ contact_city           : Factor w/ 1 level "Toronto": 1 1 1 1 1 ...
##  $ contact_state          : Factor w/ 1 level "Ontario": 1 1 1 1 1 ...
##  $ contact_zip/postal_code: Factor w/ 1 level "M5G 2M9": 1 1 1 1 1 ...
##  $ contact_country        : Factor w/ 1 level "Canada": 1 1 1 1 1 ...
##  $ supplementary_file     : Factor w/ 1 level "NONE": 1 1 1 1 1 ...
##  $ data_row_count         : Factor w/ 1 level "27578": 1 1 1 1 1 ...
##  $ age bin:ch1            : chr  NA NA ...
##  $ agebin:ch1             : chr  "1" "1" ...
##  $ agent:ch1              : chr  NA NA ...
##  $ cell line:ch1          : chr  NA NA ...
##  $ cell type:ch1          : chr  NA NA ...
##  $ conditions:ch1         : chr  NA NA ...
##  $ e.os:ch1               : chr  "0" "1" ...
##  $ e.rfs:ch1              : chr  "1" "0" ...
##  $ er:ch1                 : chr  "0" "0" ...
##  $ genome/variation:ch1   : chr  NA NA ...
##  $ grade:ch1              : chr  "3" "3" ...
##  $ growth protocol:ch1    : chr  NA NA ...
##  $ her2:ch1               : chr  "1" "1" ...
##  $ methylation barcode:ch1: chr  "4690141010_D" "4690141013_D" ...
##  $ node:ch1               : chr  "1" "1" ...
##  $ quality control:ch1    : chr  "1" "1" ...
##  $ size bin:ch1           : chr  NA NA ...
##  $ sizebin:ch1            : chr  "1" "1" ...
##  $ subtype ihc:ch1        : chr  NA NA ...
##  $ subtypege:ch1          : chr  "HER2+" "HER2+" ...
```

```
##  $ subtypeihc:ch1         : chr  "HER2" "HER2" ...
##  $ t.os:ch1               : chr  "8.28" "2.52" ...
##  $ t.rfs:ch1              : chr  "0.93" "2.12" ...
##  $ tissue:ch1             : chr  NA NA ...
##  $ treatment protocol:ch1 : chr  NA NA ...
```

Here, the rows are samples and columns are phenotype data.

Notice that there is data regarding the experimental protocols and author information that is the same for all the samples, and not really required for our downstream analysis. We should read it to understand the experiment, but there is no need to keep these variables for downstream analysis.

There are 4 submission dates, which appears to correspond to 4 different batches. We can confirm this by looking at the naming convention of the samples.

```
head(subset(pData(breast_meth), submission_date == 'Mar 09 2010', select = title))
```

```
##                                                      title
## GSM519817   Breast tumor from patient P_1 (methylation data)
## GSM519818  Breast tumor from patient P_10 (methylation data)
## GSM519819 Breast tumor from patient P_101 (methylation data)
## GSM519820 Breast tumor from patient P_102 (methylation data)
## GSM519821 Breast tumor from patient P_103 (methylation data)
## GSM519822 Breast tumor from patient P_104 (methylation data)
```

```
head(subset(pData(breast_meth), submission_date == 'Jun 09 2010', select = title))
```

```
##                                                       title
## GSM553748 Breast tumor from patient P2_104 (methylation data)
## GSM553749 Breast tumor from patient P2_103 (methylation data)
## GSM553750 Breast tumor from patient P2_105 (methylation data)
## GSM553751 Breast tumor from patient P2_101 (methylation data)
## GSM553752 Breast tumor from patient P2_102 (methylation data)
## GSM553753  Breast tumor from patient P2_87 (methylation data)
```

We can see that the ones submitted in Mar 09 2010 has the prefix P while the ones submitted on Jun 09 2010 has the prefix P2. The other batches correspond to cell lines. We will capture this information in a separate field.

There are also repeated measures with the characteristic label appended in front of the actual value. These are conveniently parsed by `GEOquery` to remove the labels. For example, , `characteristics_ch1.6` was parsed into `agebin:ch1`.

Also, many of the fields are `factors`, that might be difficult to wrangle and we should convert those to `character` fields.

Finally, there are repeated measures that we should combine and some features that are useful that we can engineer into the data. For example, `characteristics_ch1.6` and `characteristics_ch1.7` both describe the age, but for the different batches. We need to combine those into a single column.

```
#### Wrangle phenodata
bmpheno <- data.frame(lapply(pData(breast_meth), factor2char), stringsAsFactors = FALSE)

bmpheno$Samplename <- gsub('.*(?:patient|line) (.*)(?: \\(meth.*|$)', '\\1',
                           sapply(bmpheno$title, function(x) strsplit(x, ' \\(')[[1]][1]))
bmpheno$Batch <- ifelse(grepl('^P_', bmpheno$Samplename), 'P1',
       ifelse(grepl('^P2_', bmpheno$Samplename), 'P2',
              ifelse(grepl('^HCT', bmpheno$Samplename), 'C1', 'C2')))
bmpheno$Tissue <- ifelse(grepl('^C', bmpheno$Batch),'cell_line',
```

```
                        ifelse(grepl('_N', bmpheno$Samplename),
                               'normal', 'tumor'))
bmpheno$IHC <- ifelse(is.na(bmpheno$subtype.ihc.ch1),
                      bmpheno$subtypeihc.ch1, bmpheno$subtype.ihc.ch1)
bmpheno$IHC <- ifelse(is.na(bmpheno$IHC), 'cell_line', bmpheno$IHC)
bmpheno$Age <- ifelse(is.na(bmpheno$age.bin.ch1), bmpheno$agebin.ch1, bmpheno$age.bin.ch1)
rownames(bmpheno) <- bmpheno$Samplename

bmpheno <- bmpheno[,c(1:2,53:ncol(bmpheno))]
str(bmpheno)
```

```
## 'data.frame':    273 obs. of  32 variables:
##  $ title                   : chr  "Breast tumor from patient P_1 (methylation data)" "Breast tumor fr
##  $ geo_accession           : chr  "GSM519817" "GSM519818" "GSM519819" "GSM519820" ...
##  $ age.bin.ch1             : chr  NA NA NA NA ...
##  $ agebin.ch1              : chr  "1" "1" "1" "1" ...
##  $ agent.ch1               : chr  NA NA NA NA ...
##  $ cell.line.ch1           : chr  NA NA NA NA ...
##  $ cell.type.ch1           : chr  NA NA NA NA ...
##  $ conditions.ch1          : chr  NA NA NA NA ...
##  $ e.os.ch1                : chr  "0" "1" "0" "0" ...
##  $ e.rfs.ch1               : chr  "1" "0" "0" "0" ...
##  $ er.ch1                  : chr  "0" "0" "1" "0" ...
##  $ genome.variation.ch1    : chr  NA NA NA NA ...
##  $ grade.ch1               : chr  "3" "3" "1" "3" ...
##  $ growth.protocol.ch1     : chr  NA NA NA NA ...
##  $ her2.ch1                : chr  "1" "1" "0" "1" ...
##  $ methylation.barcode.ch1: chr  "4690141010_D" "4690141013_D" "4690141074_D" "4690141075_F" ...
##  $ node.ch1                : chr  "1" "1" "0" "1" ...
##  $ quality.control.ch1     : chr  "1" "1" "1" "1" ...
##  $ size.bin.ch1            : chr  NA NA NA NA ...
##  $ sizebin.ch1             : chr  "1" "1" "0" "1" ...
##  $ subtype.ihc.ch1         : chr  NA NA NA NA ...
##  $ subtypege.ch1           : chr  "HER2+" "HER2+" "LumA" "Basal" ...
##  $ subtypeihc.ch1          : chr  "HER2" "HER2" "LumA" "HER2" ...
##  $ t.os.ch1                : chr  "8.28" "2.52" "7.9" "13.42" ...
##  $ t.rfs.ch1               : chr  "0.93" "2.12" "7.9" "13.42" ...
##  $ tissue.ch1              : chr  NA NA NA NA ...
##  $ treatment.protocol.ch1  : chr  NA NA NA NA ...
##  $ Samplename              : chr  "P_1" "P_10" "P_101" "P_102" ...
##  $ Batch                   : chr  "P1" "P1" "P1" "P1" ...
##  $ Tissue                  : chr  "tumor" "tumor" "tumor" "tumor" ...
##  $ IHC                     : chr  "HER2" "HER2" "LumA" "HER2" ...
##  $ Age                     : chr  "1" "1" "1" "1" ...
```

Get beta-values and match with phenodata

Now, we extract the beta-values using `exprs`. We can see that the data is organized with columns as samples and rows as probes.

Since `bmpheno` contains the phenotypic data for our methylation dataset, we should verify that the sample names are matching. For that, we use `==` to compare the colnames of the methylation dataset to the rownames of the phenodata in order. The function `all` assesses if all the elements of a logical vector are `TRUE`. Once we

verify that match, we can then rename the samples from their GEO identifiers (`GSMNN...NN`) to the sample IDs provided by the study. These are shorter, sometimes informative on the biology of the sample, and will allow us to identify and match samples with expression data.

```
## Get beta value
bm_beta_raw <- exprs(breast_meth)
head(bm_beta_raw[,1:5])
```

```
##            GSM519817 GSM519818 GSM519819 GSM519820 GSM519821
## cg00000292      0.67      0.80      0.56      0.85      0.52
## cg00002426      0.42      0.37      0.20      0.41      0.14
## cg00003994      0.05      0.08      0.25      0.16      0.25
## cg00005847      0.46      0.67      0.57      0.50      0.49
## cg00006414      0.02      0.04      0.05      0.04      0.03
## cg00007981      0.03      0.04      0.07      0.07      0.05
```

```
## all checks a logical vector and returns TRUE if
## all entries in the vector is true
all(colnames(bm_beta_raw) == bmpheno$geo_accession)
```

```
## [1] TRUE
```

```
colnames(bm_beta_raw) <- bmpheno$Samplename

## Make colors
# Batch
mapper_batch <- c('P1'='orange','P2'='skyblue3',
                  'C1'='grey','C2'='forestgreen')
colors_batch <- mapper_batch[bmpheno$Batch]
```

QC

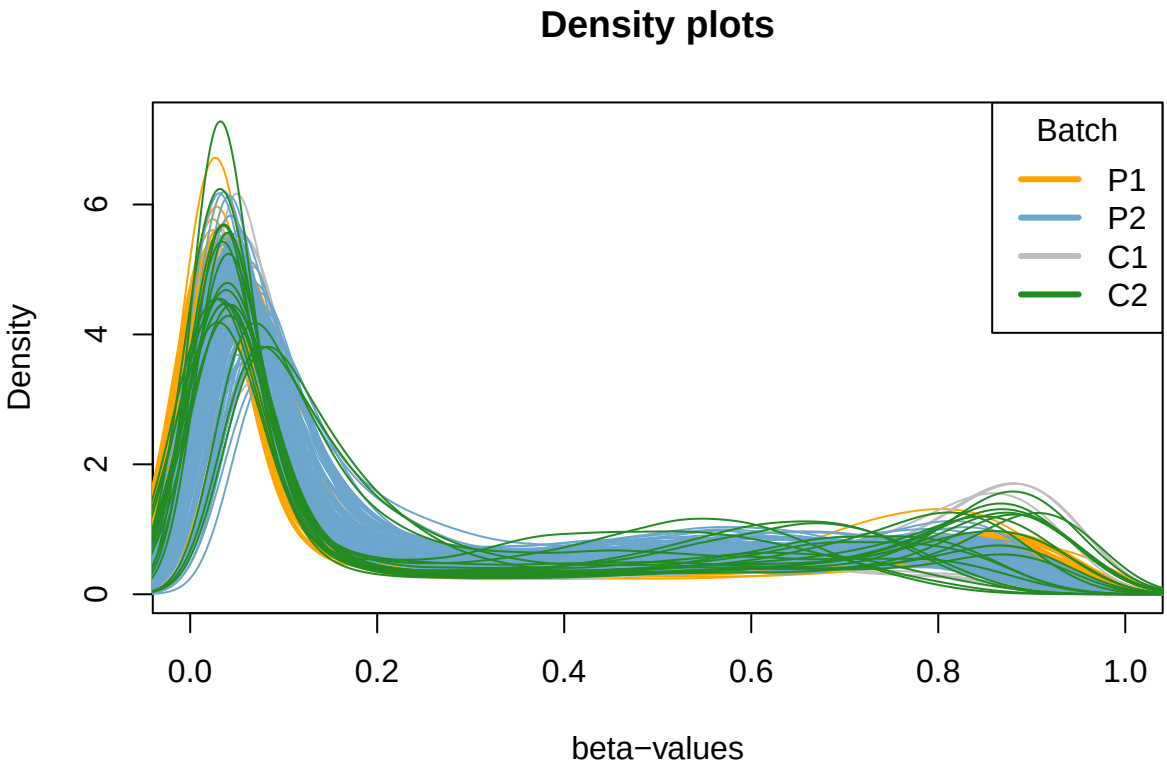
`qcReport` only works on `RGChannelSet`, and since we don't have the raw data, we will not be able to generate one. Here, we will write our own custom function to make beta-value density plots. We first extract the densities using `density` and use `apply` to loop through every column.

The `density` output returns a named list with `$x` and `$y` which describes the plotting coordinates of a density curve. We know that the beta-values fall between 0 and 1, but we have to find the limits of y using `range` and `lapply` to extract the information.

```
## Get density information
bm_density <- apply(bm_beta_raw, 2, density, na.rm = TRUE)
ylimits <- range(unlist(lapply(bm_density, '[[', 'y')))
```

Now that we have all the densities calculated and the y-limits, let's make our plot.

```
## Make plot
# Batch
plot(0, type = 'n', ylim = ylimits, xlim = c(0,1),
     main = 'Density plots',  xlab = 'beta-values',
     ylab = 'Density')
for(i in 1:length(bm_density)){
  lines(bm_density[[i]], col = colors_batch[i])
}
legend('topright', legend = names(mapper_batch),
       col = mapper_batch, lwd = 3,
       title = 'Batch')
```



While we don't observe a bimodal distribution with modes around 0.1 (for unmethylated loci) and 0.9 (for methylated loci), this doesn't immediately mean that there is something wrong with this array. These density curves suggest that for the our samples are largely unmethylated in many of the probes that are measured by this microarray. A poorly performing sample tends to have density curves with a unimodal distribution at 0.5. Without additional information from the raw IDAT files, it will be difficult to evaluate further.

Another observation is that the C1 and C2 batches appear to have different distributions that the P1 and P2 distribution. This could either be due to these being the batches with cell lines treated under various conditions or an effect associated with batch. Unfortunately, since there are no tumor samples in the C batches and cel line samples in the P batches, we will not be able to distinguish these effects. Fortunately for us, we won't be analysing the cell line data today. However, if we have to, we should normalize and analyse the cell line data separately, and make comparisons based on diffential methylation statistics over actually comparing beta-values of tumors vs. cell lines.

Subset only tissue data

Given the density curves above, we will filter away the cell line samples using logical indexing.

```
## logical indexing
idx <- bmpheno$Tissue != 'cell_line'
tm_beta_raw <- bm_beta_raw[,idx]
tmpheno <- bmpheno[idx,]
```

Make colors

To aid visualizing that data, we will make color vectors correspondong to the batch, IHC status, and tissue of the samples. The concept here is to use a named list and subsetting to map colors onto a vector.

For example, let's say we have a vector called `students` with elements of either a student living in the dorms

or off campus. We can first make a vector with names `dorms` and `offcampus` and the corresponding colors. We can then subset that using the `students` vector and that will pull out the color assigned to the name since this just pulls the element matching the name regardless if the name has been matched previously.

```
students <- c('dorms','offcampus','dorms','dorms','offcampus')
mapper <- c('dorms' = 'skyblue3', 'offcampus' = 'orange')
mapper['dorms']
```

```
##      dorms
## "skyblue3"
```

```
mapper[c('dorms','dorms')]
```

```
##      dorms      dorms
## "skyblue3" "skyblue3"
```

```
mapper[students]
```

```
##      dorms  offcampus      dorms      dorms  offcampus
## "skyblue3"   "orange" "skyblue3" "skyblue3"   "orange"
```

Using the same scheme, we will make the necessary colors for this dataset.

```
## Make colors
# Batch
mapper_batch <- c('P1'='orange','P2'='skyblue3',
                  'C1'='grey','C2'='forestgreen')
colors_batch <- mapper_batch[tmpheno$Batch]
# Tissue
mapper_tissue <- c('tumor'='orange',
                   'normal'='skyblue3',
                   'cell_line'='grey')
colors_tissue <- mapper_tissue[tmpheno$Tissue]
# IHC
mapper_ihc <- c('Basal'='orange','HER2'='forestgreen',
                'LumA'='skyblue2', 'LumB'='blue',
                'Normal'='grey70', 'cell_line'='grey20')
colors_ihc <- mapper_ihc[tmpheno$IHC]
```

Normalize the data

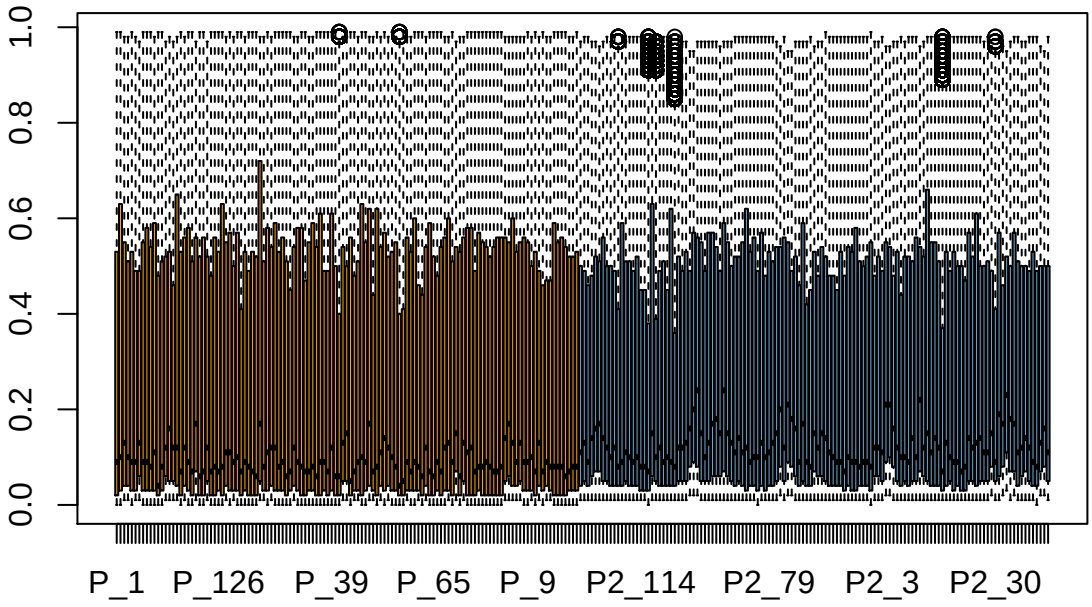
Then, we remove probes with missing information and use the `normalizeQuantlies` function to perform quantile normalization to our data. We cannot use `funnorm` because that requires information available in the raw IDAT files or an `RGChannelSet`. Quantiles will work for our purposes, especially for such a large sample size.

```
#### Methylation data
tm_beta_raw <- tm_beta_raw[complete.cases(tm_beta_raw),]
tm_beta <- normalizeQuantiles(tm_beta_raw)
```

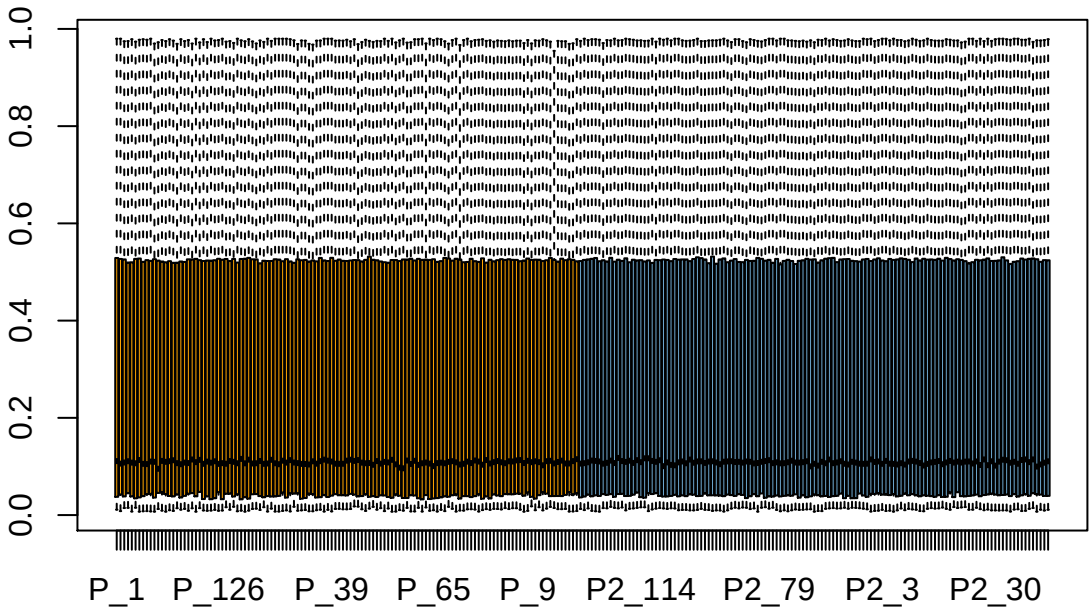
Compare

Let's visualize the distributions before and after normalization.

```
boxplot(tm_beta_raw, col = colors_batch)
```




```
boxplot(tm_beta, col = colors_batch)
```



As observed, the quantile normalization normalized the data such that the quantiles of each sample lines up with the other. Dr. Wayne Yu will go into greater detail regarding normalization methods tomorrow. We now have analysis ready data.

Exploratory analysis

With this high dimensional data, we can start performing unsupervised analyses to explore structures within the data. We can draw correlations between these structures and biological phenotypes of interest, perhaps observing new relationships that we can generate hypotheses for.

Curse of dimensionality

With high dimensional data, there are many features (probes in this case) and few samples. This leads to unique problems that the field has to address.

A common problem is that high dimensionality allows us to overfit the data to answer the questions that we may have. This is most evident in machine learning and classification problems, where with enough data points about every single subject, one can train a classifier to perfectly classify each sample. However, when the same classifier is used in a different cohort, the classifier underperforms, because the model learned incorrect relationships between the features and the phenotype of interest.

It is also more difficult to represent or visualize the data so that we can start exploring structures within the data itself. Beyond that, features can be colinear, which offers us an opportunity to summarise these dimensions into more tractable datasets through a process called dimensionality reduction.

Here, we will use two major types of dimensionality reduction, (1) feature projection and (2) feature selection, to visualize and explore our data.

We will only be describing the concepts briefly and introducing the functions that perform these analyses, as a deeper discussion is beyond the goal of this lab session.

Classical multidimensional scaling

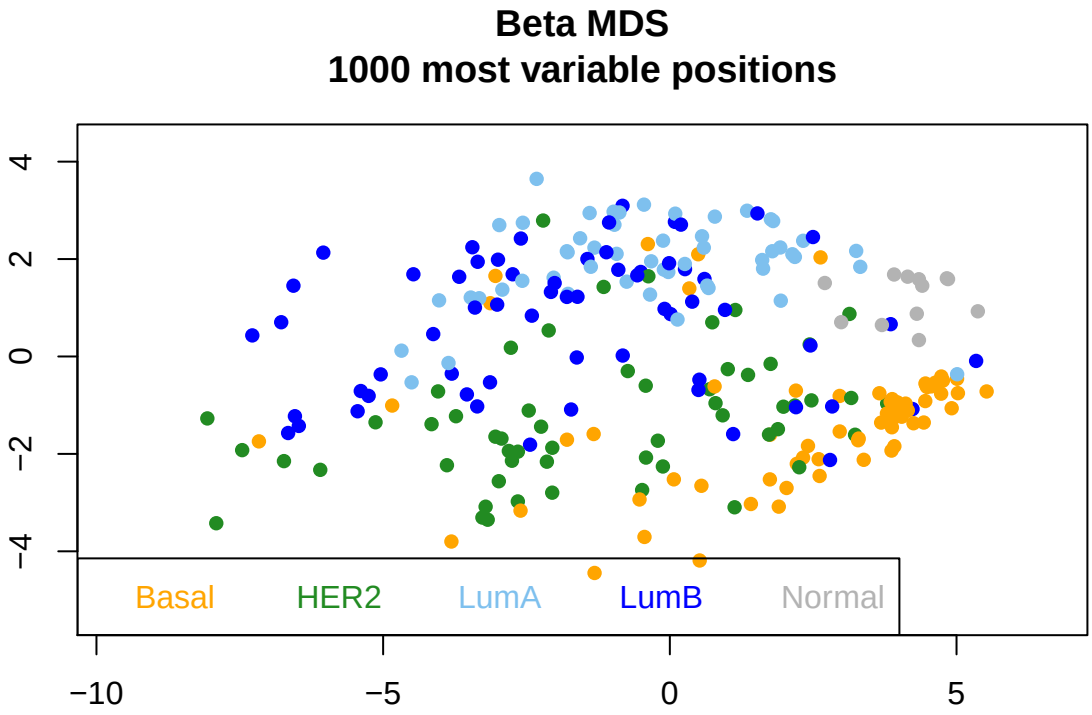
Note that MDS represents a class of feature projection methods, but for the purposes of this document, it will be used to refer to classical MDS as performed by the R `cmdscale` function.

The first approach is called classical multidimensional scaling (MDS), which is a feature projection technique. MDS tries to preserve the pairwise distances between all samples in a complex data space as they are projected into lower dimensions. Imagine flattening a 3D object to a 2D object across a single plane and choosing the angle to flatten to maximize distances of all the elements up to their original distances.

This has been implemented in the `minfi` package using the function `mdsPlot`. We will be providing 3 arguments to the function:

1. The beta-values matrix, `tm_beta`.
2. The phenotype of interest as a group. In this case, IHC status, `tmpheno$IHC`
3. A palette of how we want the plot colored.

```
mdsPlot(tm_beta, sampGroups = tmpheno$IHC, pch = 16, pal = mapper_ihc, numPositions = 1000)
```



The orientation of the x- and y-axes are arbitrary, and the scales are relative. It is unintuitive to relate how different a pair of samples are if they are one unit apart in the x axis vs another pair which is one unit apart in y. However, this preserves the structure of the data and we can start identifying which samples are more similar to each other.

We observe that `normal` samples tend to cluster together. `Luminals`, `LumA` and `LumB`, co-cluster away from the HER2 and `basal` samples. `Basals` are the most distant from the luminals. In this dataset, we observe a higher degree structure or clustering of the samples to a known biological phenotype relevant to disease, suggesting that this dataset and approach is capturing variation of interest.

Most variable probes

Note that in the MDS exercise, the `numPositions` argument was set to 1000 (the default). Here, we are making an assumption and performing an unsupervised feature selection.

Often, a small number of probes or features capture most of the variation in the data set. A majority of the probes could have no variability in the tissue that we are studying. In other words, if we select the top X%, let's say 5 or 10% of the probes, we can start observing structures between the samples. This reduces our search space and makes our calculations faster (recall that MDS works on pairwise distances, so the computation time increases exponentially with n).

In the next analysis, we will also simplify our analysis by selecting most variable probes. Here, we will use standard deviation of each probe to do so. The function `rowSds` calculates the standard deviations of each row (probes). We will extract the top 1000 probes and subset our beta-values matrix as `tm_mvp`. The probe names are stored as `mvp_names`.

```
tm_sd <- rowSds(tm_beta)
top1000q <- (nrow(tm_beta) - 1000)/nrow(tm_beta)
mvp_idx <- tm_sd >= quantile(tm_sd, top1000q)
tm_mvp <- tm_beta[mvp_idx,]
mvp_names <- rownames(tm_mvp)
```

Principal component analysis

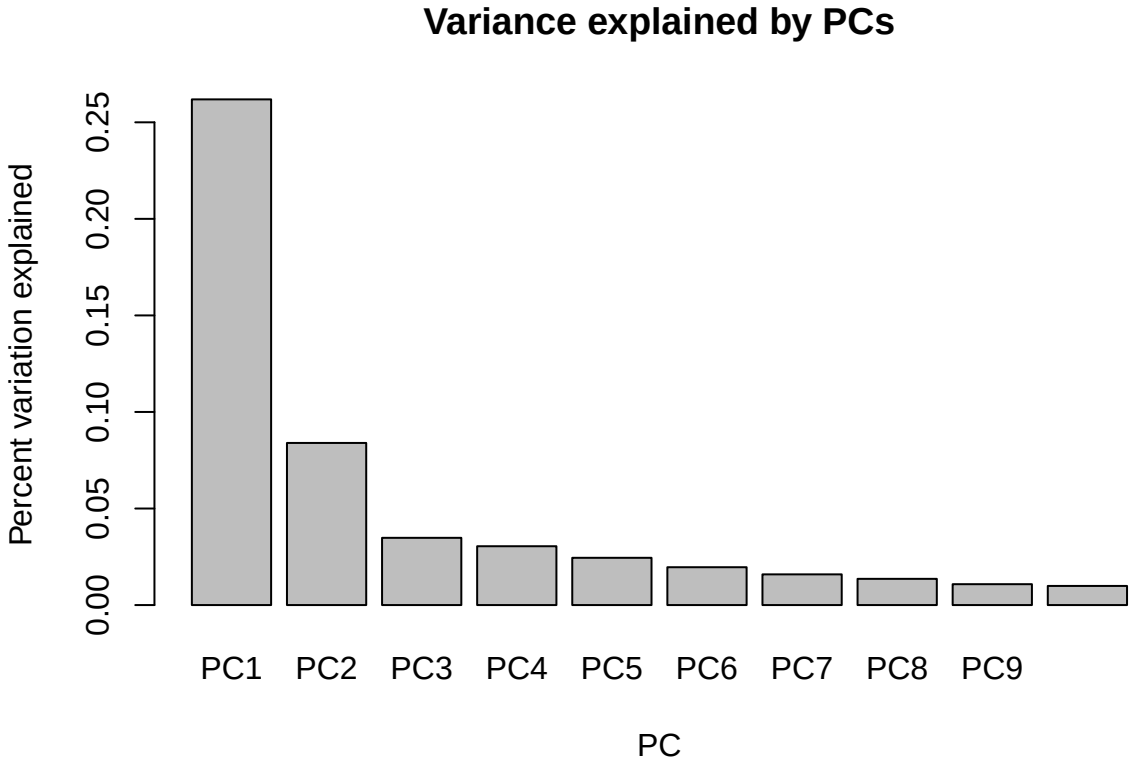
Principal component analysis (PCA) is also a feature projection method with similar goals as the MDS described above. The approaches used are slightly different. PCA attempts to maximizing variation by looking for orthogonal components in the data called principal components (PCs). We will use `prcomp`. With the default arguments, `prcomp` returns similar projections as `mdsPlot` as observed later.

We will estimate the PCs and investigate at the proportion of variance captured by the first 10 PCs.

```
tmpca <- prcomp(t(tm_mvp))
summary(tmpca)$importance[,1:10]
```

```
##                             PC1      PC2      PC3      PC4       PC5
## Standard deviation     3.223347 1.825062 1.175249 1.099454 0.9853985
## Proportion of Variance 0.261870 0.083950 0.034810 0.030470 0.0244700
## Cumulative Proportion  0.261870 0.345820 0.380630 0.411090 0.4355700
##                             PC6      PC7       PC8       PC9      PC10
## Standard deviation     0.8817906 0.794307 0.7340653 0.6551676 0.6268633
## Proportion of Variance 0.0196000 0.015900 0.0135800 0.0108200 0.0099000
## Cumulative Proportion  0.4551600 0.471070 0.4846500 0.4954700 0.5053700
```

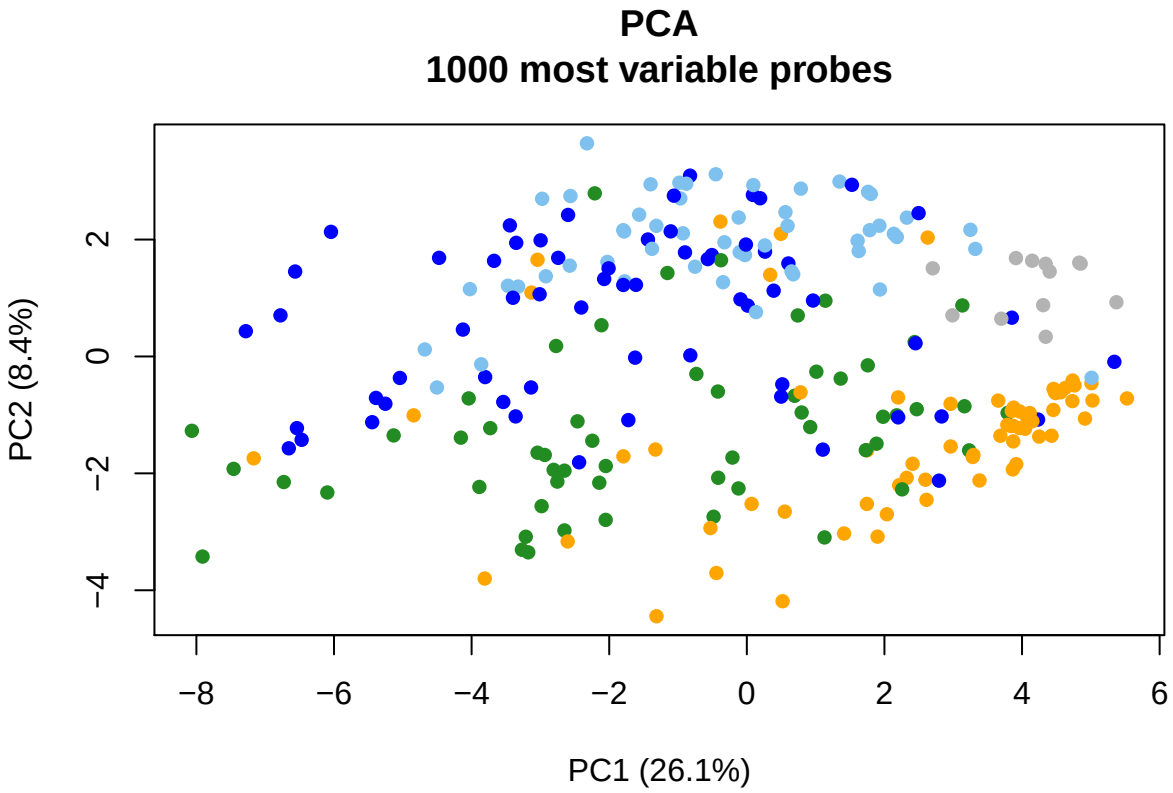
```
barplot(summary(tmpca)$importance[2,1:10], ylab = 'Percent variation explained', xlab = 'PC', main = 'Va
```



The first row shows the standard deviations of the PCs, the second shows the proportion of variance of the original data that it captures, and finally the cumulative proportion with increasing PCs. We see here that PC1 captures 26.19% of the variation in `tm_mvp`, PC2 captures 8.4%, and so on.

We can take the rotated data, or projections, stored in `$x` and plot that.

```
plot(tmpca$x[,1], tmpca$x[,2], pch = 16, col = colors_ihc,
     main = 'PCA\n1000 most variable probes',
     xlab = 'PC1 (26.1%)', ylab = 'PC2 (8.4%)')
```



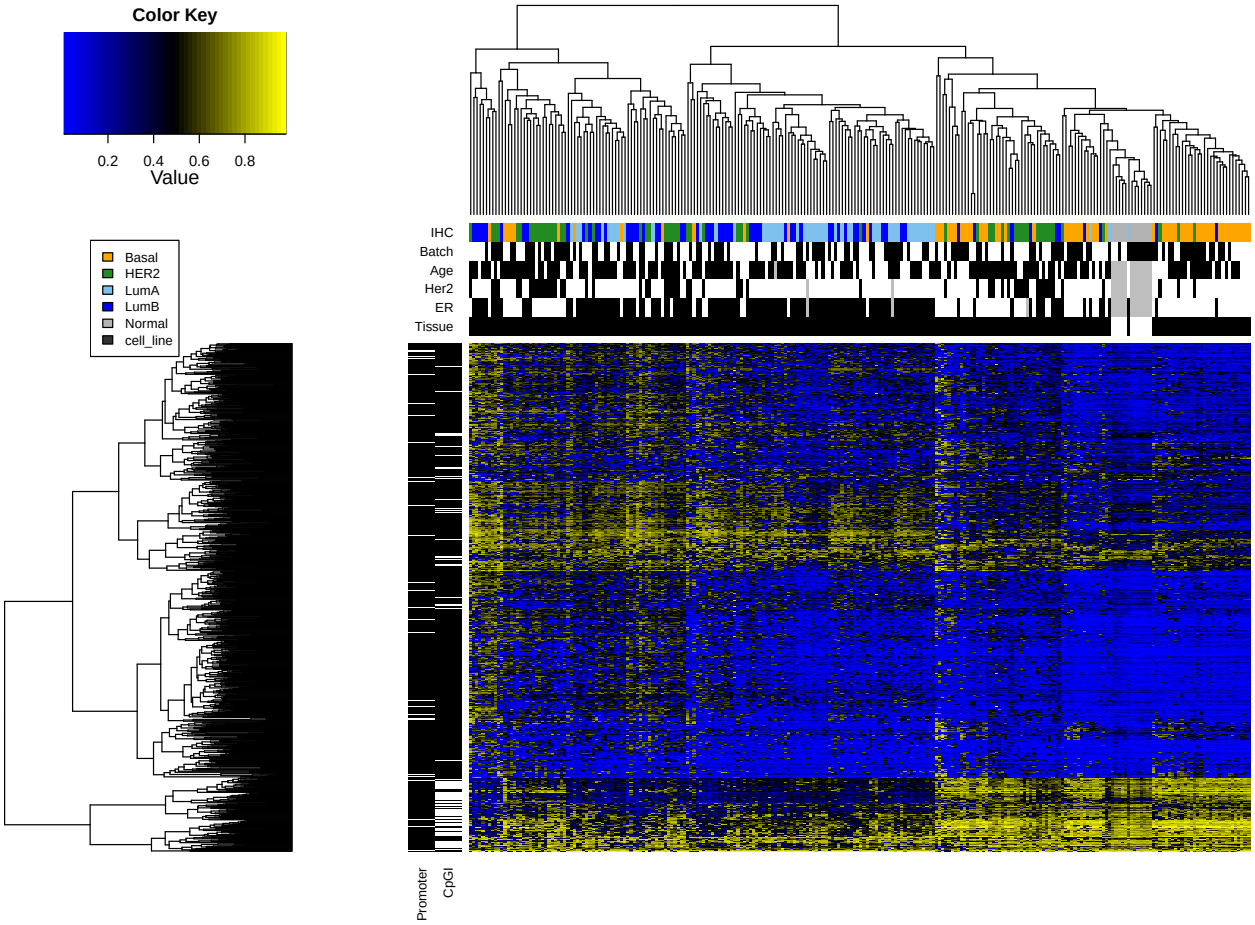
Clustering analysis

We see some structure in the dataset, now let's visualize that with the beta-values using a heatmap. We will be using a custom function sourced earlier called `myheatmap3`. This function will (1) perform clustering analysis on the samples and the genes, (2) plots the heatmap, and (3) allow us to add column side and row side color matrices to annotate our samples and probes respectively.

First, we make the color matrices using the same approach we did as before. We will then visualize `tm_mvp`.

```
## make column side colors
# tissue type
hmcolor_tissue <- c('normal' = 'white', 'tumor' = 'black', 'NA' = 'grey')[tmpheno$Tissue]
# er status
hmcolor_er <- c('0' = 'white', '1' = 'black', 'NA' = 'grey')[tmpheno$er.ch1]
# her2 status
hmcolor_her2 <- c('0' = 'white', '1' = 'black', 'NA' = 'grey')[tmpheno$her2.ch1]
# age
hmcolor_age <- c('0' = 'white', '1' = 'black', 'NA' = 'grey')[tmpheno$Age]
# batch
hmcolor_batch <- c('P1' = 'white', 'P2' = 'black', 'NA' = 'grey')[tmpheno$Batch]
# ihc
hmcolor_ihc <- colors_ihc
# make the matrix
column_colors <- cbind(Tissue = hmcolor_tissue, ER = hmcolor_er, Her2 = hmcolor_her2,
                       Age = hmcolor_age, Batch = hmcolor_batch, IHC = hmcolor_ihc)
## make row side colors
# promoter
hmrow_promoter <- ifelse(annots[mvp_names,]$Promoter == 'yes', 'black', 'white')
```

```
# cpg island
hmrow_cpgi <- ifelse(annots[mvp_names,]$CPG_ISLAND, 'black', 'white')
# make the matrix
row_colors <- rbind(NA, NA, NA, NA,
                    Promoter = hmrow_promoter,
                    CpGI = hmrow_cpgi)
## heatmap
myheatmap3(tm_mvp, ColSideCol = column_colors,
           RowSideCol = row_colors,
           side.height.fraction=0.8, labRow = NA, labCol = NA,
           margins = c(6,3))
legend(0, 0.8, legend = names(mapper_ihc), fill = mapper_ihc, cex = 0.7)
```



Here, the samples separate into two major clusters of `Basal` vs `Luminal` tumors. The probes cluster into CpG island and non-CpG island clusters. We can infer additional subcluster information from this plot and very the number of probes to discover additional relationships if we want to.

These three methods are commonly used methods to first explore the data, describe and annotate relationships between the samples and features.

Identify DMPs using limma

As with the squamous cell carcinoma (SCC) analysis, we will identify differentially methylated probes using `limma`. To keep things simple for the analysis and visualization, we won't be using batch as a covariate. If we

have time, we will come back to this and add batch as a covariate.

Get M-values, design matrix and contrasts

Since we did not begin with a `GenomicRatioSet`, we will have to transform our data from beta-values to M-values manually using the convenient `logit2` function from `minfi`.

We will create a design matrix with `tumor` and `normal` and create contrasts against them.

```
## get M-values
tm_m <- logit2(tm_beta)
## make model matrix and contrasts
tissue_model <- model.matrix(~ 0 + tmpheno$Tissue)
colnames(tissue_model) <- c('tumor','normal')
tissue_contrasts <- makeContrasts(tumor - normal, levels = tissue_model)
```

Fit the model

Again, we will use `lmFit`, `contrasts.fit`, and `eBayes` to identify the DMPs. Finally, we will use `top.table` to extract the DMPs.

```
## fit model
tissue_fit1 <- lmFit(tm_m, design = tissue_model)
tissue_fit2 <- contrasts.fit(tissue_fit1, tissue_contrasts)
tissue_eb <- eBayes(tissue_fit2)
tissue_res <- topTable(tissue_eb, coef = 1, number = nrow(tm_m))

## do the same for beta-values
tissue_fit1b <- lmFit(tm_beta, design = tissue_model)
tissue_fit2b <- contrasts.fit(tissue_fit1b, tissue_contrasts)
tissue_ebb <- eBayes(tissue_fit2b)
tissue_resb <- topTable(tissue_ebb, coef = 1, number = nrow(tissue_ebb))

## flag differentially methylated probes
dmps <- rownames(tissue_res)[tissue_res$adj.P.Val <= 0.05] ## FDR < 0.05 in M-value
dmps <- dmps[abs(tissue_resb[dmps,]$logFC) >= 0.3] ## delta beta >= 0.3
tissue_resb$mfdr <- tissue_res[rownames(tissue_resb),]$adj.P.Val
tissue_resb$DMP <- rownames(tissue_resb) %in% dmps

head(tissue_resb)
```

```
##                logFC    AveExpr         t      P.Value    adj.P.Val
## cg11172423 -0.2422180 0.83886305 -19.98112 1.643521e-53 4.532503e-49
## cg00476577  0.1268219 0.07672218  16.27533 5.721433e-41 7.889284e-37
## cg07651914 -0.2459941 0.74817627 -15.70996 4.947213e-39 4.547808e-35
## cg13631913  0.2858157 0.08544349  15.60844 1.102227e-38 7.599302e-35
## cg10221736 -0.2136802 0.83614332 -15.11914 5.234328e-37 2.887046e-33
## cg27433088  0.1964011 0.15867715  14.26983 4.207895e-34 1.934089e-30
##                    B         mfdr   DMP
## cg11172423 110.95745 9.141622e-33 FALSE
## cg00476577  82.43398 1.361305e-20 FALSE
## cg07651914  78.02512 1.406453e-26 FALSE
## cg13631913  77.23311 4.846959e-19 FALSE
## cg10221736  73.41616 1.181547e-23 FALSE
```

```
## cg27433088  66.80162 7.657036e-18 FALSE
```

Here, we can see that there are only 16 DMPs identified between tumor and normal using our relatively stringent criteria.

```
sum(tissue_resb$DMP)
```

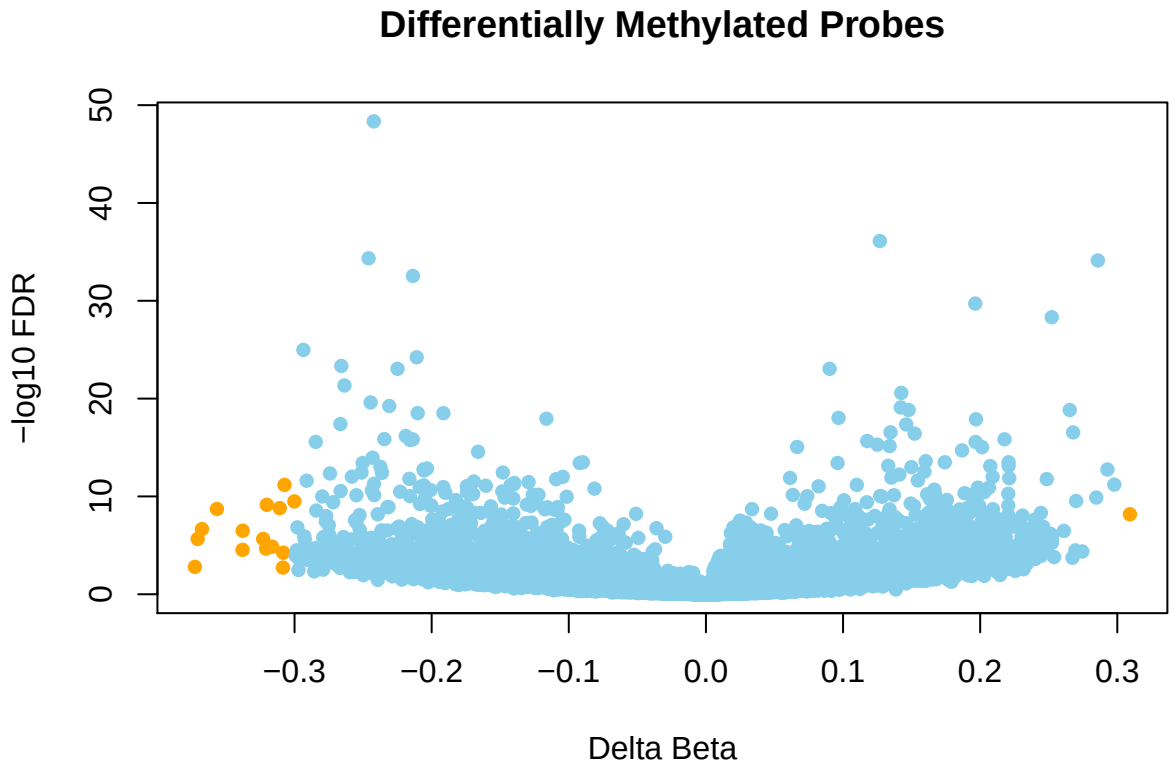
```
## [1] 16
```

```
head(subset(tissue_resb, DMP))
```

```
##                 logFC   AveExpr         t      P.Value    adj.P.Val
## cg25902889 -0.3073020 0.6960034 -8.132821 2.041019e-14 6.622026e-12
## cg04456238 -0.3001156 0.5323670 -7.448008 1.580950e-12 3.253690e-10
## cg01009664 -0.3202005 0.6292375 -7.309264 3.712436e-12 7.209969e-10
## cg13288195 -0.3107772 0.6294138 -7.162597 9.055828e-12 1.570702e-09
## cg02164046 -0.3565839 0.4826387 -7.125866 1.130194e-11 1.912178e-09
## cg02989940  0.3092370 0.4457428  6.886337 4.709808e-11 6.695210e-09
##                   B         mfdr  DMP
## cg25902889 22.05692 6.798652e-10 TRUE
## cg04456238 17.78850 2.163530e-10 TRUE
## cg01009664 16.95211 2.115743e-08 TRUE
## cg13288195 16.07897 1.114382e-08 TRUE
## cg02164046 15.86212 7.926072e-11 TRUE
## cg02989940 14.46612 1.650380e-08 TRUE
```

Visualizing the DMPs using a volcano plot, we observe that a majority of the probes were filtered away using the 0.3 threshold. There are technological differences between 27k and 450k probes, and we can be less stringent in our decision of a delta-beta threshold.

```
plot(x = tissue_resb$logFC, y = -log10(tissue_resb$adj.P.Val),
     col = ifelse(tissue_resb$DMP, 'orange', 'skyblue'), pch = 16,
     ylab = '-log10 FDR', xlab = 'Delta Beta', main = 'Differentially Methylated Probes')
```



Annotate result

Next, we annotate our `top.table` with the `annots` data.frame from the beginning of the analysis. We do so by using the `merge` function. `merge` returns a `data.frame` sorted by the column that is used to merge the data.frames. We will sort the output using the `order` function.

```
toannotate <- annots[,c('Symbol', 'CPG_ISLAND', 'Distance_to_TSS', 'Promoter')]
## merge table 1 with table 2 by column. If by = 0, by row names
tissue_annotated <- merge(tissue_resb, toannotate, by = 0)
rownames(tissue_annotated) <- tissue_annotated$Row.names
tissue_annotated$Row.names <- NULL
## rearrange by pvalue and logfc
tissue_annotated <- tissue_annotated[order(
  -tissue_annotated$DMP, ## first by inverse DMP status
  tissue_annotated$mfdr, ## then by p-values (low to high)
  -1*abs(tissue_annotated$logFC)),] ## then by inverse delta-beta
tissue_annotated$dmp0.3 <- tissue_annotated$mfdr <= 0.05 & abs(tissue_annotated$logFC) >= 0.3
tissue_annotated$dmp0.2 <- tissue_annotated$mfdr <= 0.05 & abs(tissue_annotated$logFC) >= 0.2
head(tissue_annotated)
```

```
##                logFC   AveExpr         t      P.Value    adj.P.Val
## cg02164046 -0.3565839 0.4826387 -7.125866 1.130194e-11 1.912178e-09
## cg04456238 -0.3001156 0.5323670 -7.448008 1.580950e-12 3.253690e-10
## cg25902889 -0.3073020 0.6960034 -8.132821 2.041019e-14 6.622026e-12
## cg13288195 -0.3107772 0.6294138 -7.162597 9.055828e-12 1.570702e-09
## cg02989940  0.3092370 0.4457428  6.886337 4.709808e-11 6.695210e-09
## cg01009664 -0.3202005 0.6292375 -7.309264 3.712436e-12 7.209969e-10
##                   B         mfdr  DMP Symbol CPG_ISLAND Distance_to_TSS
## cg02164046 15.86212 7.926072e-11 TRUE    SST       TRUE              53
```

```
## cg04456238 17.78850 2.163530e-10 TRUE    WT1       TRUE             NA
## cg25902889 22.05692 6.798652e-10 TRUE   FSD1      FALSE            399
## cg13288195 16.07897 1.114382e-08 TRUE FBXL22      FALSE            115
## cg02989940 14.46612 1.650380e-08 TRUE   ERAF      FALSE             31
## cg01009664 16.95211 2.115743e-08 TRUE    TRH       TRUE             50
##            Promoter dmp0.3 dmp0.2
## cg02164046      yes   TRUE   TRUE
## cg04456238       no   TRUE   TRUE
## cg25902889      yes   TRUE   TRUE
## cg13288195      yes   TRUE   TRUE
## cg02989940      yes   TRUE   TRUE
## cg01009664      yes   TRUE   TRUE
```

Gene set analysis

Since we have information on which probes map to which genes, we can consider using gene set analysis to identify potential sets of genes or pathways that are associated with our phenotype of interest. We will explore two classes of analysis, over-representation analysis (ORAs) and functional class scoring (FCS).

We will download the Hallmark gene sets from the Molecular Signatures Database. Hallmark gene sets are meant to summarise specific biological processes and states derived from gene expression data.

```
## get gene sets
hallmarks <- readgmt('https://raw.githubusercontent.com/sean-cho/datasets/master/datasets/h.all.v6.2.sy
names(hallmarks) <- gsub('HALLMARK_', '', names(hallmarks))
names(hallmarks)
```

```
##  [1] "TNFA_SIGNALING_VIA_NFKB"
##  [2] "HYPOXIA"
##  [3] "CHOLESTEROL_HOMEOSTASIS"
##  [4] "MITOTIC_SPINDLE"
##  [5] "WNT_BETA_CATENIN_SIGNALING"
##  [6] "TGF_BETA_SIGNALING"
##  [7] "IL6_JAK_STAT3_SIGNALING"
##  [8] "DNA_REPAIR"
##  [9] "G2M_CHECKPOINT"
## [10] "APOPTOSIS"
## [11] "NOTCH_SIGNALING"
## [12] "ADIPOGENESIS"
## [13] "ESTROGEN_RESPONSE_EARLY"
## [14] "ESTROGEN_RESPONSE_LATE"
## [15] "ANDROGEN_RESPONSE"
## [16] "MYOGENESIS"
## [17] "PROTEIN_SECRETION"
## [18] "INTERFERON_ALPHA_RESPONSE"
## [19] "INTERFERON_GAMMA_RESPONSE"
## [20] "APICAL_JUNCTION"
## [21] "APICAL_SURFACE"
## [22] "HEDGEHOG_SIGNALING"
## [23] "COMPLEMENT"
## [24] "UNFOLDED_PROTEIN_RESPONSE"
## [25] "PI3K_AKT_MTOR_SIGNALING"
## [26] "MTORC1_SIGNALING"
## [27] "E2F_TARGETS"
```

```
## [28] "MYC_TARGETS_V1"
## [29] "MYC_TARGETS_V2"
## [30] "EPITHELIAL_MESENCHYMAL_TRANSITION"
## [31] "INFLAMMATORY_RESPONSE"
## [32] "XENOBIOTIC_METABOLISM"
## [33] "FATTY_ACID_METABOLISM"
## [34] "OXIDATIVE_PHOSPHORYLATION"
## [35] "GLYCOLYSIS"
## [36] "REACTIVE_OXIGEN_SPECIES_PATHWAY"
## [37] "P53_PATHWAY"
## [38] "UV_RESPONSE_UP"
## [39] "UV_RESPONSE_DN"
## [40] "ANGIOGENESIS"
## [41] "HEME_METABOLISM"
## [42] "COAGULATION"
## [43] "IL2_STAT5_SIGNALING"
## [44] "BILE_ACID_METABOLISM"
## [45] "PEROXISOME"
## [46] "ALLOGRAFT_REJECTION"
## [47] "SPERMATOGENESIS"
## [48] "KRAS_SIGNALING_UP"
## [49] "KRAS_SIGNALING_DN"
## [50] "PANCREAS_BETA_CELLS"
```

Preparing vectors for comparison

Here, we will flag two levels of DMPs for gene set analysis:

1. DMP0.3
    1. FDR <= 0.05 in the M-value analysis
    2. abs(delta beta) >= 0.3 in the beta-value analysis
2. DMP0.2
    1. FDR <= 0.05 in the M-value analysis
    2. abs(delta beta) >= 0.2 from beta-value analysis

We will also write a convenience function that will summarise our data from probes to genes. Finally, we will make a vector called `gene_statistics` that will be used for functional class scoring analysis.

```
## 0.3
genes0.3 <- names(which(tapply(tissue_annotated$dmp0.3,
                               tissue_annotated$Symbol, max) == 1))
length(genes0.3)
```

```
## [1] 15
```

```
## 0.2
genes0.2 <- names(which(tapply(tissue_annotated$dmp0.2,
                               tissue_annotated$Symbol, max) == 1))
length(genes0.2)
```

```
## [1] 390
```

```
## get entry with the highest difference
getbest <- function(x) x[which.max(abs(x))]

## moderated t-statistic
```

```
genesall <- unique(tissue_annotated$Symbol)
gene_statistics <- tapply(tissue_annotated$t, tissue_annotated$Symbol, getbest)
```

Gene sets

GSEA

Over-representation analysis

ORA methods test for over-representation of genes in a particular gene set amongst genes that are defined as differentially methylated. Therefore, the investigator has to choose a threshold to classify their probes, and subsequently genes, as differentially methylated. This threshold is subjective (more sensitive or specific), and different thresholds from the same analysis could lead to differences in ORA results.

Hypergeometric test

One major implementation of ORA is the hypergeometric test. Conceptually, it considers the following scenario.

1. We are given an urn containing `m` colored balls and `n` non-colored balls.
2. Randomly draw `k` balls out of the urn without replacement.
3. What is the probability observing `j` or more colored balls?

In our scenario,

1. `m` = number of genes in gene set
2. `n` = number of genes in the gene universe not in the gene set
3. `k` = number of differentially expressed genes
4. `j` = number of differentially expressed genes within the gene set

I have written a function called `sig_overlap` that we have sourced previously to perform a hypergeometric test against the Hallmark gene sets.

We will be using a `for` loop to run this analysis.

```
hg0.3_pval <- c()
hg0.2_pval <- c()
hg0.3_n <- c()
hg0.2_n <- c()
for(i in 1:length(hallmarks)){
  geneset <- hallmarks[[i]]
  univ <- union(genesall,geneset)
  hg0.2_n[i] <- sum(genes0.2 %in% geneset)
  hg0.3_n[i] <- sum(genes0.3 %in% geneset)
  hg0.2_pval[i] <- sig_overlap(genes0.2,geneset,univ)
  hg0.3_pval[i] <- sig_overlap(genes0.3,geneset,univ)
}
hg_res <- data.frame(
  genesets = names(hallmarks),
  hg0.2p = hg0.2_pval,
  hg0.2n = hg0.2_n,
  hg0.3p = hg0.3_pval,
  hg0.3n = hg0.3_n
```

```
  )
hg_res
```

```
##                              genesets     hg0.2p hg0.2n     hg0.3p hg0.3n
## 1            TNFA_SIGNALING_VIA_NFKB 0.97277929      2 0.18817217      0
## 2                           HYPOXIA 0.90852293      3 0.18818393      0
## 3            CHOLESTEROL_HOMEOSTASIS 0.86801128      1 0.07398976      0
## 4                    MITOTIC_SPINDLE 0.99583901      1 0.18786690      0
## 5         WNT_BETA_CATENIN_SIGNALING 0.68284910      1 0.04265612      0
## 6                 TGF_BETA_SIGNALING 0.77165563      1 0.05452061      0
## 7            IL6_JAK_STAT3_SIGNALING 0.68383175      2 0.08644224      0
## 8                         DNA_REPAIR 0.98363058      0 0.14452607      0
## 9                      G2M_CHECKPOINT 0.99586142      0 0.18803115      0
## 10                          APOPTOSIS 0.43767325      5 0.15445524      0
## 11                    NOTCH_SIGNALING 0.58286321      1 0.03264569      0
## 12                       ADIPOGENESIS 0.90825349      3 0.18805464      0
## 13           ESTROGEN_RESPONSE_EARLY 0.99587579      1 0.18813690      1
## 14            ESTROGEN_RESPONSE_LATE 0.90862083      3 0.18823099      1
## 15                 ANDROGEN_RESPONSE 0.28869411      4 0.09962395      0
## 16                       MYOGENESIS 0.62859844      5 0.18827808      0
## 17                 PROTEIN_SECRETION 0.73477313      2 0.09495601      0
## 18         INTERFERON_ALPHA_RESPONSE 0.73967621      2 0.09584249      0
## 19         INTERFERON_GAMMA_RESPONSE 0.45115000      6 0.18806639      0
## 20                   APICAL_JUNCTION 0.62780793      5 0.18812514      0
## 21                    APICAL_SURFACE 0.69965076      1 0.04463229      0
## 22                HEDGEHOG_SIGNALING 0.62625647      1 0.03667065      0
## 23                       COMPLEMENT 0.62823353      5 0.18820746      0
## 24         UNFOLDED_PROTEIN_RESPONSE 0.95465667      0 0.11079512      0
## 25           PI3K_AKT_MTOR_SIGNALING 0.94365710      1 0.10343291      0
## 26                 MTORC1_SIGNALING 0.99587579      1 0.18813690      0
## 27                      E2F_TARGETS 0.99584222      1 0.18789034      0
## 28                    MYC_TARGETS_V1 0.99586302      0 0.18804290      0
## 29                    MYC_TARGETS_V2 0.79518200      0 0.05841536      0
## 30 EPITHELIAL_MESENCHYMAL_TRANSITION 0.17232729      8 0.18827808      1
## 31             INFLAMMATORY_RESPONSE 0.17219374      8 0.18824276      0
## 32             XENOBIOTIC_METABOLISM 0.17223824      8 0.18825453      0
## 33             FATTY_ACID_METABOLISM 0.80187004      3 0.15164777      0
## 34         OXIDATIVE_PHOSPHORYLATION 0.99587419      0 0.18812514      0
## 35                       GLYCOLYSIS 0.90835150      3 0.18810164      0
## 36   REACTIVE_OXIGEN_SPECIES_PATHWAY 0.73808290      0 0.04958477      0
## 37                       P53_PATHWAY 0.90825349      3 0.18805464      0
## 38                    UV_RESPONSE_UP 0.61862150      4 0.15171558      0
## 39                    UV_RESPONSE_DN 0.98073585      1 0.13922142      0
## 40                      ANGIOGENESIS 0.07204393      3 0.03667065      0
## 41                    HEME_METABOLISM 0.90805735      3 0.18796072      0
## 42                       COAGULATION 0.31482669      5 0.13383622      0
## 43               IL2_STAT5_SIGNALING 0.17143890      8 0.18804290      0
## 44              BILE_ACID_METABOLISM 0.95347340      1 0.10992444      1
## 45                        PEROXISOME 0.94199381      0 0.10244298      0
## 46              ALLOGRAFT_REJECTION 0.09147092      9 0.18819570      0
## 47                  SPERMATOGENESIS 0.29781821      5 0.13091067      0
## 48                KRAS_SIGNALING_UP 0.01958836     11 0.18813690      0
## 49                KRAS_SIGNALING_DN 0.09112014      9 0.18805464      0
## 50              PANCREAS_BETA_CELLS 0.29292453      2 0.04065023      1
```

Fisher's exact test

Fisher's exact test can also be used to perform ORA. To run this analysis in R, we have to first set up a contingency (2 x 2) table. We will test for independence of the variables use to set up the contingency table.

Given a contingency table, how many times can we observe the values distributed at the observed extreme or more if we maintained the same total for each row and column?

```
geneset1 <- hallmarks[[1]]
ctab <- table(genesall %in% genes0.3, genesall %in% geneset1, dnn = c('DMP','geneset'))
ctab
```

```
##        geneset
## DMP     FALSE  TRUE
##   FALSE 14283   179
##   TRUE     15     0
```

```
fisher.test(ctab)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  ctab
## p-value = 1
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   0.00000 22.44428
## sample estimates:
## odds ratio
##          0
```

Again, we will use a `for` loop to analyse the data for every Hallmark gene set.

```
fe0.2_pval <- c()
fe0.2_n <- c()
fe0.3_pval <- c()
fe0.3_n <- c()
for(i in 1:length(hallmarks)){
  geneset <- intersect(hallmarks[[i]], genesall)
  ctab0.2 <- table(genesall %in% genes0.2,genesall %in% geneset)
  ctab0.3 <- table(genesall %in% genes0.3,genesall %in% geneset)
    table(genesall %in% genes0.2,genesall %in% geneset)
  fe0.2_pval[i] <- fisher.test(ctab0.2,
                               alternative = 'greater')$p.value
  fe0.2_n[i] <- ctab0.2[2,2]
  fe0.3_pval[i] <- fisher.test(ctab0.3,
                               alternative = 'greater')$p.value
  fe0.3_n[i] <- ctab0.3[2,2]
}
# names(fe0.2_pval) <- names(hallmarks)
# names(fe0.3_pval) <- names(hallmarks)
fisher_res <- data.frame(
  genesets = names(hallmarks),
  fe0.2p = fe0.2_pval,
  fe0.2n = fe0.2_n,
  fe0.3p = fe0.3_pval,
  fe0.3n = fe0.3_n
```

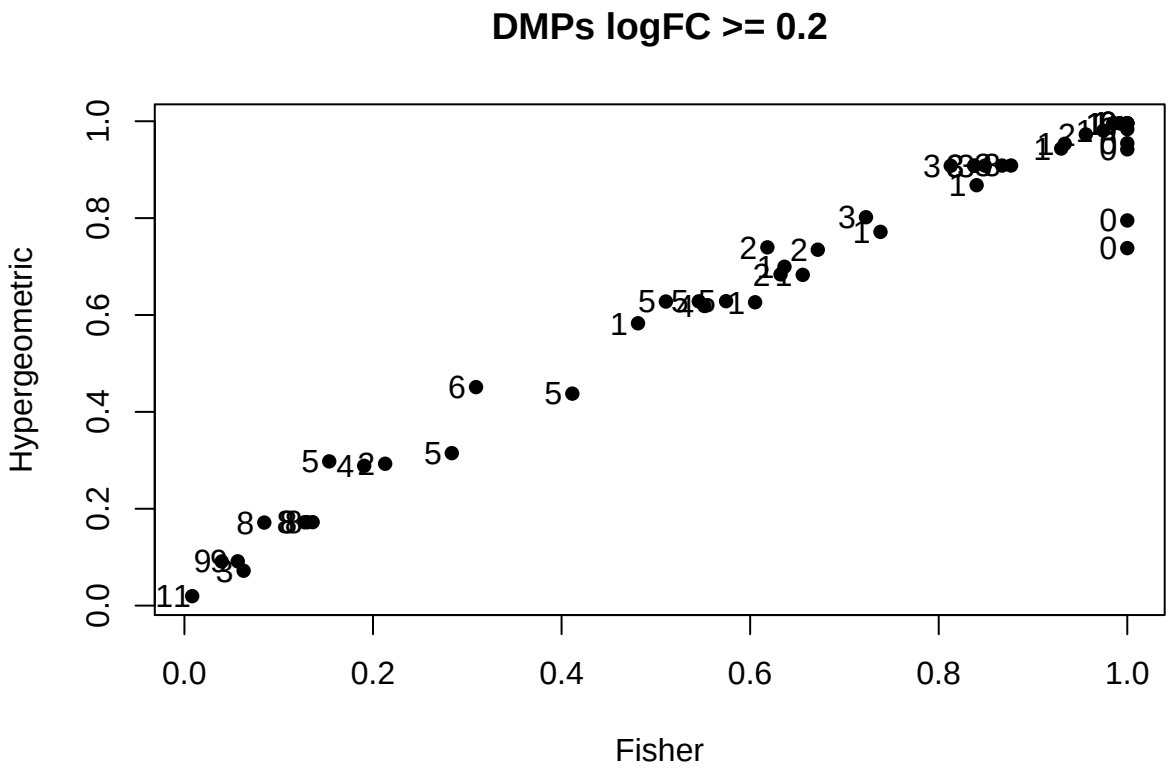
```
  )
fisher_res
```

```
##                              genesets      fe0.2p fe0.2n     fe0.3p fe0.3n
## 1            TNFA_SIGNALING_VIA_NFKB 0.956022043      2 1.00000000      0
## 2                           HYPOXIA 0.867226132      3 1.00000000      0
## 3            CHOLESTEROL_HOMEOSTASIS 0.840215744      1 1.00000000      0
## 4                    MITOTIC_SPINDLE 0.985014052      1 1.00000000      0
## 5          WNT_BETA_CATENIN_SIGNALING 0.655776004      1 1.00000000      0
## 6                TGF_BETA_SIGNALING 0.738255143      1 1.00000000      0
## 7            IL6_JAK_STAT3_SIGNALING 0.632259008      2 1.00000000      0
## 8                         DNA_REPAIR 1.000000000      0 1.00000000      0
## 9                      G2M_CHECKPOINT 1.000000000      0 1.00000000      0
## 10                          APOPTOSIS 0.411380480      5 1.00000000      0
## 11                    NOTCH_SIGNALING 0.481045731      1 1.00000000      0
## 12                       ADIPOGENESIS 0.837673983      3 1.00000000      0
## 13           ESTROGEN_RESPONSE_EARLY 0.992061588      1 0.16770230      1
## 14            ESTROGEN_RESPONSE_LATE 0.876738709      3 0.17466224      1
## 15                 ANDROGEN_RESPONSE 0.190651959      4 1.00000000      0
## 16                       MYOGENESIS 0.574477026      5 1.00000000      0
## 17                 PROTEIN_SECRETION 0.671748711      2 1.00000000      0
## 18         INTERFERON_ALPHA_RESPONSE 0.618297848      2 1.00000000      0
## 19         INTERFERON_GAMMA_RESPONSE 0.309267067      6 1.00000000      0
## 20                   APICAL_JUNCTION 0.510629737      5 1.00000000      0
## 21                    APICAL_SURFACE 0.636400134      1 1.00000000      0
## 22                HEDGEHOG_SIGNALING 0.605278018      1 1.00000000      0
## 23                       COMPLEMENT 0.545527336      5 1.00000000      0
## 24         UNFOLDED_PROTEIN_RESPONSE 1.000000000      0 1.00000000      0
## 25           PI3K_AKT_MTOR_SIGNALING 0.929904559      1 1.00000000      0
## 26                 MTORC1_SIGNALING 0.992061588      1 1.00000000      0
## 27                      E2F_TARGETS 0.985819017      1 1.00000000      0
## 28                    MYC_TARGETS_V1 1.000000000      0 1.00000000      0
## 29                    MYC_TARGETS_V2 1.000000000      0 1.00000000      0
## 30 EPITHELIAL_MESENCHYMAL_TRANSITION 0.135981946      8 0.17812181      1
## 31             INFLAMMATORY_RESPONSE 0.127448108      8 1.00000000      0
## 32             XENOBIOTIC_METABOLISM 0.130260931      8 1.00000000      0
## 33             FATTY_ACID_METABOLISM 0.722840547      3 1.00000000      0
## 34         OXIDATIVE_PHOSPHORYLATION 1.000000000      0 1.00000000      0
## 35                        GLYCOLYSIS 0.849021293      3 1.00000000      0
## 36   REACTIVE_OXIGEN_SPECIES_PATHWAY 1.000000000      0 1.00000000      0
## 37                       P53_PATHWAY 0.837673983      3 1.00000000      0
## 38                    UV_RESPONSE_UP 0.551364665      4 1.00000000      0
## 39                    UV_RESPONSE_DN 0.974689266      1 1.00000000      0
## 40                      ANGIOGENESIS 0.062782529      3 1.00000000      0
## 41                   HEME_METABOLISM 0.812768896      3 1.00000000      0
## 42                       COAGULATION 0.283574653      5 1.00000000      0
## 43               IL2_STAT5_SIGNALING 0.084695397      8 1.00000000      0
## 44              BILE_ACID_METABOLISM 0.933655246      1 0.09785391      1
## 45                        PEROXISOME 1.000000000      0 1.00000000      0
## 46               ALLOGRAFT_REJECTION 0.056561055      9 1.00000000      0
## 47                   SPERMATOGENESIS 0.153572880      5 1.00000000      0
## 48                 KRAS_SIGNALING_UP 0.008211894     11 1.00000000      0
## 49                 KRAS_SIGNALING_DN 0.039547973      9 1.00000000      0
## 50               PANCREAS_BETA_CELLS 0.212883077      2 0.03266346      1
```

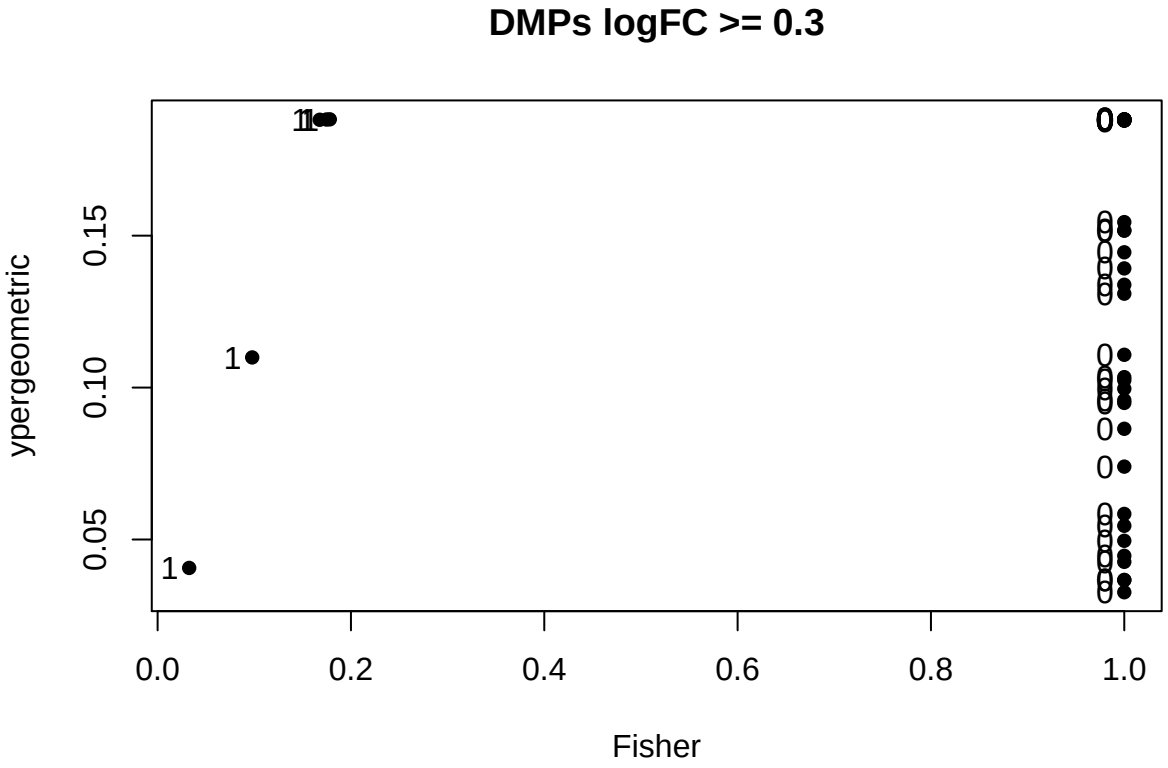
Compare

Let us compare the results between the hypergeometric test and Fisher's exact test. We will plot the p-values of each test, and the number of genes overlapping the gene set to the left of the point in the scatter plot.

```
plot(fisher_res$fe0.2p, hg_res$hg0.2p, pch = 16,
     xlab = 'Fisher', ylab = 'Hypergeometric',
     main = 'DMPs logFC >= 0.2')
text(fisher_res$fe0.2p - 0.02, hg_res$hg0.2p, hg_res$hg0.2n)
```




```
plot(fisher_res$fe0.3p, hg_res$hg0.3p, pch = 16,
     xlab = 'Fisher', ylab = 'ypergeometric',
     main = 'DMPs logFC >= 0.3')
text(fisher_res$fe0.3p - 0.02, hg_res$hg0.3p, hg_res$hg0.3n)
```



We observe that for the `beta >= 0.2` analysis, the p-values from the hypergeometric test and the Fisher's exact test are largely correlated. This is because Fisher's exact test, especially in our scenario of comparing only one-tail, calculates its probabilities assuming a hypergeometric distribution.

Interestingly, we see that relationship disappear when there are no differentially methylated genes in the gene set. This violates an assumption made by the hypergeometric test, and therefore is unreliable. Always take into account the number of overlapping genes.

Functional class scoring (FCS)

FCS approaches hypothesizes that both large changes in individual genes and concerted smaller changes in a set of functionally related genes can contribute to a biological phenotype. Therefore, the test considers the complete set of feature changes derived from a given experiment.

Briefly, these methods:

1. Calculates differential methylation gene-wise statistics.
2. Calculates a pathway/gene set score based on (1).
3. Tests for significance by permuting a null hypothesis by shuffling gene or sample labels.

We will be using the mean-rank gene set enrichment (MR-GSE) test, implemented using `wilcoxGST` in `limma`. We will write a convenience function that performs the analysis and plots the results. We will use `apply` to run the analysis on all gene sets and perform p-value adjustment using Benjamini-Hochberg's FDR method. This will take a few moments.

```
## convenience functions: gene set test wrapper and plotting
gstestwrapper <- function(gs, alternative = 'mixed'){
  cat('|')
  gs_idx <- names(gene_statistics) %in% gs
  n <- sum(gs_idx)
  pval <- wilcoxGST(gs_idx, gene_statistics, alternative = alternative)
```

```
  return(c(N = n, pval = pval))
}

plotgse <- function(gs, ...){
  gs_idx <- names(gene_statistics) %in% gs
  barcodeplot(statistics = as.numeric(gene_statistics),
              index = gs_idx, ...)
}

tissue_gse <- data.frame(t(sapply(hallmarks, gstestwrapper, alternative = 'less')),
                         stringsAsFactors = FALSE)
```

```
## ||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
tissue_gse <- tissue_gse[order(tissue_gse$pval),]
tissue_gse$qval <- p.adjust(tissue_gse$pval, method = 'fdr')
```

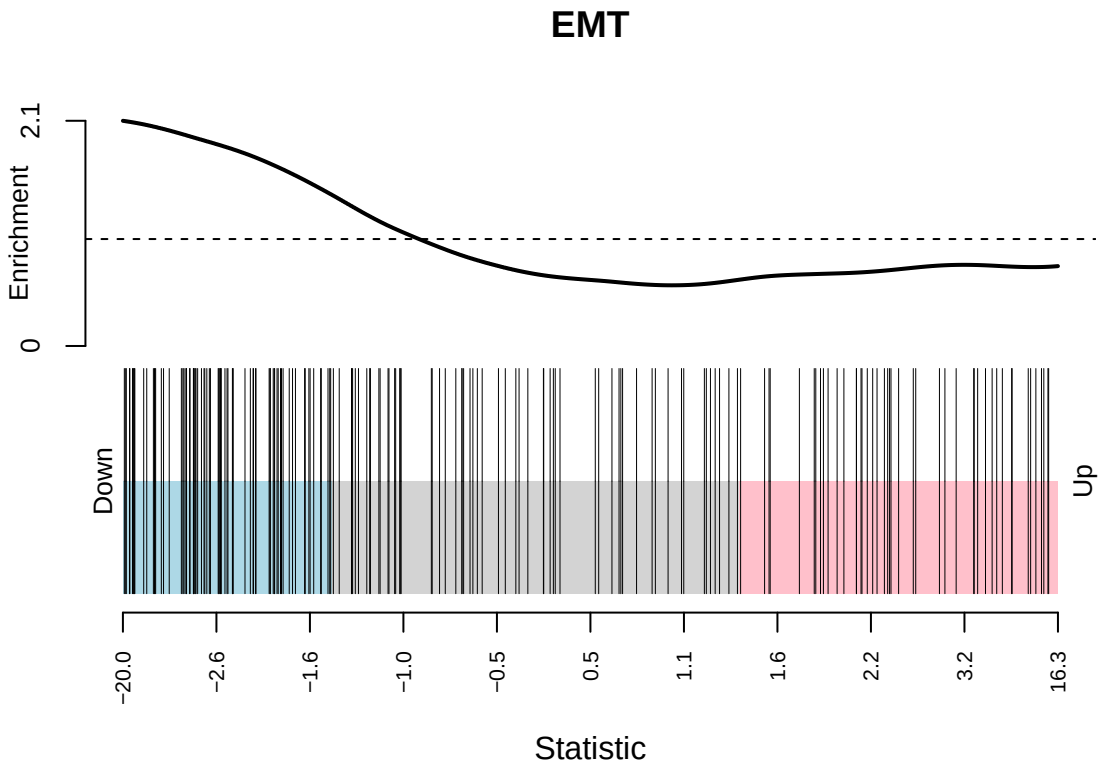
Let's subset the results for gene sets that are significant with an FDR <= 0.05.

```
subset(tissue_gse, qval <= 0.05)
```

```
##                                      N         pval         qval
## EPITHELIAL_MESENCHYMAL_TRANSITION 188 7.729898e-09 3.864949e-07
## SPERMATOGENESIS                   105 1.316100e-03 3.290250e-02
## PANCREAS_BETA_CELLS                32 3.853707e-03 4.967218e-02
## MYOGENESIS                        188 4.112013e-03 4.967218e-02
## ADIPOGENESIS                      169 4.967218e-03 4.967218e-02
```

The most significant gene set is EMT with 188 genes inside the gene set. Let's visualize that relationship.

```
plotgse(hallmarks$EPITHELIAL_MESENCHYMAL_TRANSITION, main = 'EMT')
```



We can see that genes in the EMT gene set tends to be hypomethylated in our tumor vs. normal analysis. We can hypothesize that this hypomethylation leads to over-expression of EMT genes, a hallmark of cancer. However, we cannot yet conclude that, as we have to show that the expression of these genes are indeed upregulated and perform orthogonal experiments to validate some of these findings. We now have a lead hypothesis that we can pursue and test.

For fun: Repeat this analysis comparing ER+ vs Basal tumors.

session info

```
sessionInfo()
```

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] limma_3.36.1                minfi_1.26.2
##  [3] bumphunter_1.22.0           locfit_1.5-9.1
##  [5] iterators_1.0.9             foreach_1.4.4
##  [7] Biostrings_2.48.0           XVector_0.20.0
##  [9] SummarizedExperiment_1.10.1 DelayedArray_0.6.0
## [11] BiocParallel_1.14.1         matrixStats_0.53.1
## [13] GenomicRanges_1.32.3        GenomeInfoDb_1.16.0
## [15] IRanges_2.14.10             S4Vectors_0.18.2
## [17] GEOquery_2.48.0             Biobase_2.40.0
## [19] BiocGenerics_0.26.0
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-137             bitops_1.0-6
##  [3] bit64_0.9-7              RColorBrewer_1.1-2
##  [5] progress_1.1.2           httr_1.3.1
##  [7] tools_3.5.0              doRNG_1.6.6
##  [9] nor1mix_1.2-3            R6_2.2.2
## [11] HDF5Array_1.8.1          DBI_1.0.0
## [13] withr_2.1.2              tidyselect_0.2.4
## [15] prettyunits_1.0.2        base64_2.0
## [17] preprocessCore_1.42.0    bit_1.1-14
## [19] compiler_3.5.0           xml2_1.2.0
## [21] pkgmaker_0.27            rtracklayer_1.40.2
## [23] quadprog_1.5-5           genefilter_1.62.0
## [25] readr_1.1.1              stringr_1.3.1
## [27] digest_0.6.15            Rsamtools_1.32.0
## [29] illuminaio_0.22.0        rmarkdown_1.10.10
```

```
## [31] siggenes_1.54.0          pkgconfig_2.0.1
## [33] htmltools_0.3.6          bibtex_0.4.2
## [35] rlang_0.2.1              htmldeps_0.1.1
## [37] RSQLite_2.1.1            DelayedMatrixStats_1.2.0
## [39] bindr_0.1.1              mclust_5.4.1
## [41] dplyr_0.7.5              RCurl_1.95-4.10
## [43] magrittr_1.5             GenomeInfoDbData_1.1.0
## [45] Matrix_1.2-14            Rcpp_0.12.17
## [47] Rhdf5lib_1.2.1           stringi_1.2.2
## [49] yaml_2.2.0               MASS_7.3-49
## [51] zlibbioc_1.26.0          rhdf5_2.24.0
## [53] plyr_1.8.4               grid_3.5.0
## [55] blob_1.1.1               lattice_0.20-35
## [57] splines_3.5.0            annotate_1.58.0
## [59] multtest_2.36.0          GenomicFeatures_1.32.0
## [61] hms_0.4.2                knitr_1.20
## [63] beanplot_1.2             pillar_1.2.3
## [65] rngtools_1.3.1           codetools_0.2-15
## [67] biomaRt_2.36.1           XML_3.98-1.11
## [69] glue_1.3.0               evaluate_0.11
## [71] data.table_1.11.4        openssl_1.0.1
## [73] purrr_0.2.5              tidyr_0.8.1
## [75] reshape_0.8.7            assertthat_0.2.0
## [77] xtable_1.8-2             survival_2.41-3
## [79] tibble_1.4.2             GenomicAlignments_1.16.0
## [81] AnnotationDbi_1.42.1     registry_0.5
## [83] memoise_1.1.0            bindrcpp_0.2.2
```