Differential expression methods

Learning objectives

- describe main classes of methods used
- understand statistical concepts behind the key scRNA-seq DE methods
- run the key scRNA-seq DE methods

Methods examples

Figure: Software tools for identifying DE genes using scRNAseq data (Wang et al. 2019)

More more examples

And even more examples in the Soneson, Charlotte, and Mark D Robinson (2018)

Main classes

non-parameteric tests

- generic methods
- e.g. Wilcoxon rank-sum test, Kruskal-Wallis, Kolmogorov-Smirnov test
- non-parametric tests generally convert observed expression values to ranks & test whether the distribution of ranks for one group are signficantly different from the distribution of ranks for the other group
- some non-parametric methods fail in the presence of a large number of tied values, such as the case for dropouts (zeros) in single-cell RNA-seq expression data
- if the conditions for a parametric test hold, then it will typically be more powerful than a non-parametric test

bulk RNA-seq based method

- developed for bulk RNA-seq
- e.g. edgeR, DE-seq2
- compare estimates of mean-expression (sample size), based on negative binomial distribution
- can be assessed by datasets where RNA-seq data has beeen validated by RT-qPCR

scRNA-seq specific methods

- developed for scRNA-seq
- e.g. MAST, SCDE, Monocle, Pagoda, D^{3}E etc.
- large number of samples, i.e. cells, for each group we are comparing in single-cell experiments. Thus we can take advantage of the whole distribution of expression values in each group to identify differences between groups -we usually do not have a defined set of experimental conditions; instead we try to identify the cell groups by using an unsupervised clustering approach.

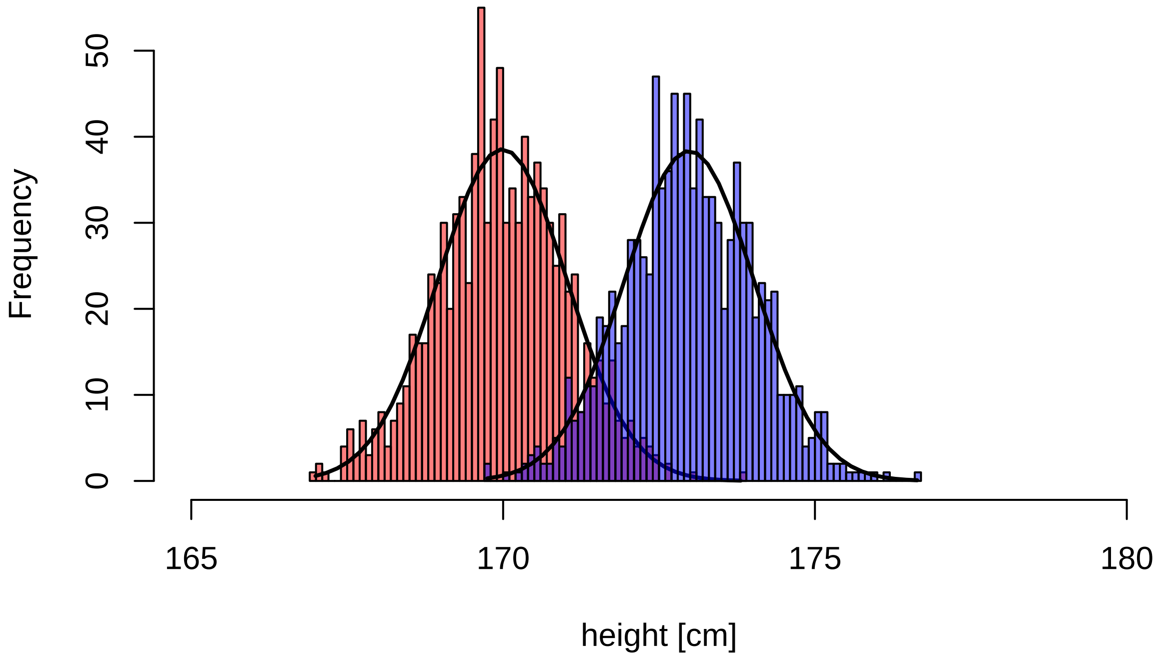
modeling wise

- distribution free (non-parameteric)
- negative binomial
- zero inflated negative binomial
- Poisson and negative binomial
- GLM and GAM
- etc.

Statistical thinking

$$Outcome_i = (Model_i) + error_i$$

Normal distribution example



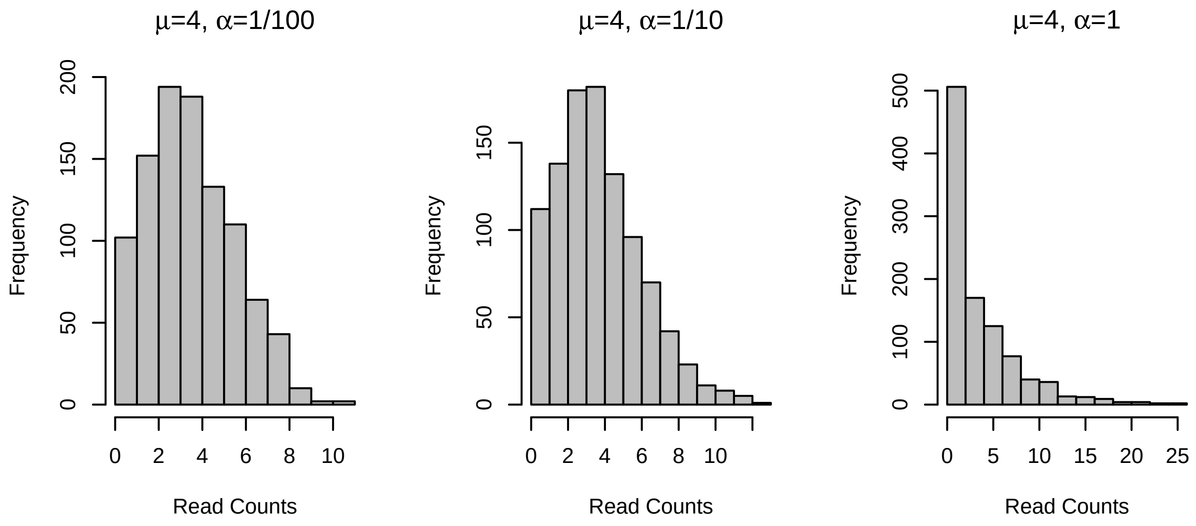
$$t = \frac{x_1 - x_2}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

Generic recipe

- model e.g. gene expression with random error
- fit model to the data and/or data to the model, estimate model parameters
- use model for prediction and/or inference
- the better model fits the data the better statistics \end{block} \end{frame}

Common distributions

Negative binomial (NB)



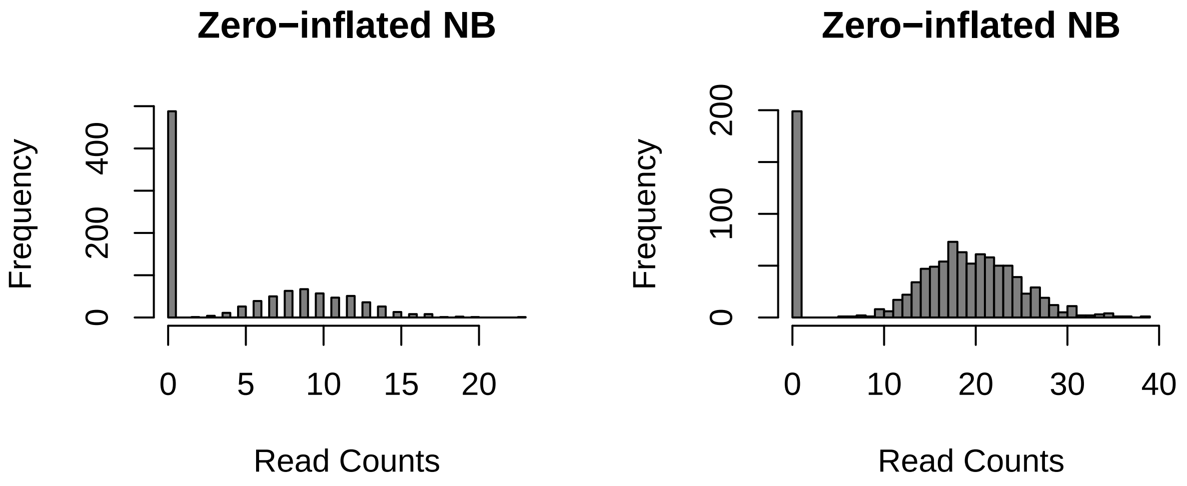
$$Mean = \mu$$

$$Variance = \mu + \mu^2/\alpha$$

α=dispersion parameter (extra variation compared to the Poisson)

NB! fits bulk RNA-seq data very well and it is used for most statistical methods designed for such data. In addition, it has been show to fit the distribution of molecule counts obtained from data tagged by unique molecular identifiers (UMIs) quite well (Grun et al. 2014, Islam et al. 2011).

zero inflated NB

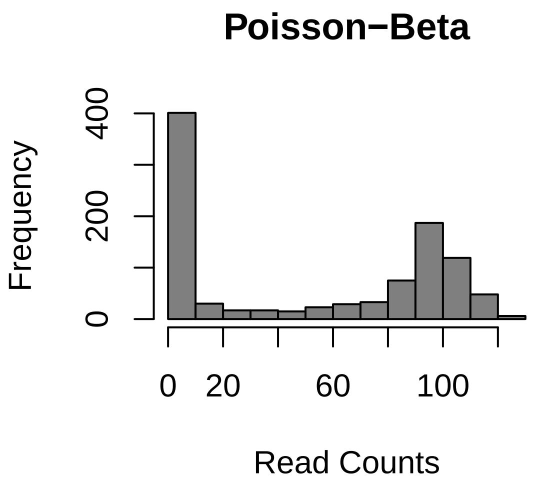


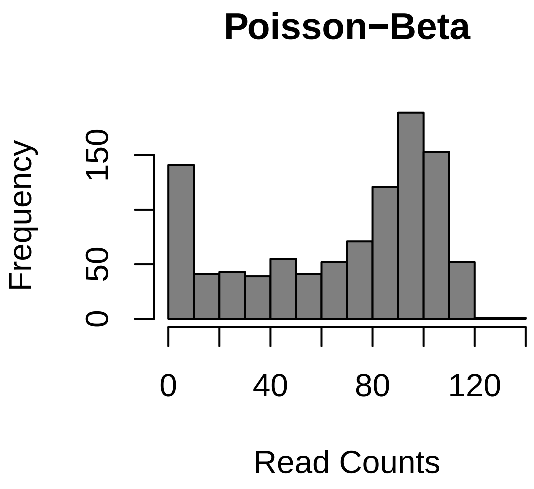
$$\mu = mu * (1 - d)$$

$$\sigma^2 = mu \cdot (1 - d) \cdot (1 + d \cdot \mu + \mu)$$

d, dropout rate; dropout rate of a gene is strongly correlated with the mean expression of the gene. Different zero-inflated negative binomial models use different relationships between mu and d and some may fit mu and d to the expression of each gene independently. Implemented in MAST, SCDE.

Poisson beta





$$\mu = g * a/(a+b)$$

$$\delta^2 = g^2 * a * b/((a+b+1)*(a+b)^2)$$

a: the rate of activation of transcription;

b: the rate of inhibition of transcription;

g: the rate of transcript production while transcription is active at the locus.

Differential expression methods may test each of the parameters for differences across groups or only one (often g). Implemented in BPSC.

May be further expanded to explicitly account for other sources of gene expression differences such as batch-effect or library depth depending on the particular DE algorithm.

Under the hood

SCDE

- *models* the read counts for each gene using a mixture of a NB, negative binomial, and a Poisson distribution
- *NB distribution* models the transcripts that are amplified and detected
- *Poisson distribution* models the unobserved or background-level signal of transcripts that are not amplified (e.g. dropout events)
- subset of robust genes is used to fit, via *EM* algorithm, the parameters to the mixture of models For DE, the posterior probability that the gene shows a fold expression difference between two conditions is computed using a *Bayesian approach*

Loading the data

Data can be downloaded from here

```
# we will read both counts and rpkms as different method are more adopted for different type of data
R<-read.table("data/mouse_embryo/rpkms_Deng2014_preinplantation.txt")
C<-read.table("data/mouse_embryo/counts_Deng2014_preinplantation.txt")
M <- read.table("data/mouse_embryo/Metadata_mouse_embryo.csv",sep=",",header=T)
```

```r
# select only 8 and 16 stage embryos.
selection <- c(grep("8cell",M$Stage),grep("16cell",M$Stage))

# check number of cells
table(M$Stage[selection])

# select those cells only from the data frames
M<-M[selection,]
R <- R[,selection]
C <- C[,selection]
```

Run SCDE

```r
require(scde)

# make a count dataset and clean up
cd <- clean.counts(C, min.lib.size=1000, min.reads = 1, min.detected = 1)

# make factor for stage
stages <- factor(M$Stage,levels=c("X8cell","X16cell"))
names(stages) <- colnames(C)

# fit error models - takes a while to run (~20 mins).
# you can skip this step and load the precomputed file.
savefile<-"data/mouse_embryo/DE/scde_error_models.Rdata"
if (file.exists(savefile)){
  load(savefile)
}else{
  o.ifm <- scde.error.models(counts = cd, groups = stages, n.cores = 1,
                             threshold.segmentation = TRUE, save.crossfit.plots = FALSE,
                             save.model.plots = FALSE, verbose = 0)
  save(o.ifm,file=savefile)
}

# estimate gene expression prior
o.prior <- scde.expression.prior(models = o.ifm, counts = cd, length.out = 400, show.plot = FALSE)

# run differential expression test
ediff <- scde.expression.difference(o.ifm, cd, o.prior, groups  =  stages,
                                    n.randomizations  =  100, n.cores  =  1, verbose  =  1)

# convert Z-score and corrected Z-score to 2-sided p-values
ediff$pvalue <- 2*pnorm(-abs(ediff$Z))
ediff$p.adjust <- 2*pnorm(-abs(ediff$cZ))

# sort and look at top results
ediff <- ediff[order(abs(ediff$Z), decreasing  =  TRUE), ]
head(ediff)

# write output to a file with top DE genes on top.
write.table(ediff,file="data/mouse_embryo/DE/scde_8cell_vs_16_cell.tab",sep="\t",quote=F)
```

Run SCDE with batch info

```
# include also batch information in the test
# in this case we will consider each embryo as a batch just for testing
# OBS! in this case there are no batch that belongs to both groups so the correction may be pointless.

batch <- factor(M$Embryo, levels <- unique(M$Embryo))
ediff.batch <- scde.expression.difference(o.ifm, cd, o.prior, groups  =  stages,
                                          n.randomizations  =  100, n.cores  =  1, batch=batch)

# now scde.expression.difference returns a list with the corrected values as well as the DE results.

de.batch <- ediff.batch$results
de.batch$pvalue <- 2*pnorm(-abs(de.batch$Z))
de.batch$p.adjust <- 2*pnorm(-abs(de.batch$cZ))

# look at top results
head(de.batch[order(abs(de.batch$Z), decreasing  =  TRUE), ])
```

MAST

- uses *generalized linear hurdle model*
- designed to account for stochastic dropouts and bimodal expression distribution in which expression is either strongly non-zero or non-detectable
- the rate of expression Z, and the level of expression Y, are modeled for each gene g, indicating whether gene g is expressed in cell i (i.e.,
$$Z_{ig} = 0$$
if $y_{ig} = 0$ and $z_{ig} = 1$ if $y_{ig} > 0$)
- A *logistic regression model* for the discrete variable Z and a *Gaussian linear model* for the continuous variable (Y|Z=1): $logit(P_r(Z_{ig} = 1)) = X_i\beta_g^D$ $P_r(Y_{ig} = Y|Z_{ig} = 1) = N(X_i\beta_g^C, \sigma_g^2)$, where X_i is a design matrix
- Model parameters are *fitted* using an empirical Bayesian framework
- Allows for a joint estimate of nuisance and treatment effects
- DE is determined using *the likelihood ratio test*

Run MAST

```
library(MAST)

fdata <- data.frame(rownames(R))
rownames(fdata)<-rownames(R)

# Make a single cell assay object
sca <- FromMatrix(log2(as.matrix(R)+1),M,fdata)

# count number of detected genes and scale.
cdr2 <-colSums(assay(sca)>0)
colData(sca)$cngeneson <- scale(cdr2)

# Fit a hurdle model, modeling the condition and (centered) ngeneson factor, thus adjusting for
```

```r
# the cellular detection rate. In this case we set the reference to be 8cell stage using relevel
# of the factor.

# takes a while to run, so save to an file so that you do not have to rerun each time
savefile<-"data/mouse_embryo/DE/mast_data.Rdata"
if (file.exists(savefile)){
  load(savefile)
}else{
  cond<-factor(colData(sca)$Stage)
  cond <- relevel(cond,"X8cell")
  colData(sca)$condition<-cond
  zlmCond <- zlm(~condition + cngeneson, sca)
  #Run likelihood ratio test for the condition coefficient.
  summaryCond <- summary(zlmCond, doLRT='conditionX16cell')
  save(sca,zlmCond,summaryCond,file=savefile)
}
# get the datatable with all outputs
summaryDt <- summaryCond$datatable

# Make a datatable with all the relevant output - combined Hurdle model
fcHurdle <- merge(summaryDt[contrast=='conditionX16cell' & component=='H',.(primerid, `Pr(>Chisq)`)],
     summaryDt[contrast=='conditionX16cell' & component=='logFC',
               .(primerid, coef, ci.hi, ci.lo)], by='primerid')

# change name of Pr(>Chisq) to fdr and sort by fdr
fcHurdle[,fdr:=p.adjust(`Pr(>Chisq)`, 'fdr')]
fcHurdle <- fcHurdle[order(fdr),]
head(fcHurdle)

# write output to a table
write.table(fcHurdle,file="data/mouse_embryo/DE/mast_8cell_vs_16_cell.tab",sep="\t",quote=F)
```

Monocole

- Originally designed for ordering cells by progress through differentiation stages (pseudo-time)
- The mean expression level of each gene is *modeled with a GAM*, generalized additive model, which relates one or more predictor variables to a response variable as $g(E(Y)) = \beta_0 + f_1(x_1) + f_2(x_2) + ... + f_m(x_m)$ where Y is a specific gene expression level, x_i are predictor variables, g is a link function, typically log function, and f_i are non-parametric functions (e.g. cubic splines)
- The observable expression level Y is then modeled using GAM, $E(Y) = s(\varphi_t(b_x, s_i)) + \epsilon$ where $\varphi_t(b_x, s_i)$ is the assigned pseudo-time of a cell and s is a cubic smoothing function with three degrees of freedom. The error term ϵ is normally distributed with a mean of zero
- The DE test is performed using an *approx.* χ^2 *likelihood ratio test*

Run Monocole

```r
# No example, put one in?
```

SC3 Kruskall-Wallis test

In the SC3 package they have implemented non-parametric Kruskal-Wallis test for differential expression.

Run SC3 Kruskall-Wallis test

```r
library(SC3)

# run their DE test using rpkm values and as labels, M$stage
de <- get_de_genes(R,M$Stage)

# output is simply a p-value with no information of directionality.
de.results <- data.frame(gene=rownames(R),p.value=de)
de.results <- de.results[order(de.results$p.value),]

head(de.results)
write.table(de.results,file="data/mouse_embryo/DE/sc3_kwtest_8cell_vs_16_cell.tab",sep="\t",quote=F)
```

Seurat methods

Seurat has several tests for differential expression (DE) which can be set with the test.use parameter in the FindMarkers() function:

- "wilcox" : Wilcoxon rank sum test (default)
- "bimod" : Likelihood-ratio test for single cell gene expression, (McDavid et al., Bioinformatics, 2013)
- "roc" : Standard AUC classifier
- "t" : Student's t-test
- "tobit" : Tobit-test for differential gene expression (Trapnell et al., Nature Biotech, 2014)
- "poisson" : Likelihood ratio test assuming an underlying poisson distribution. Use only for UMI-based datasets
- "negbinom" : Likelihood ratio test assuming an underlying negative binomial distribution. Use only for UMI-based datasets
- "MAST" : GLM-framework that treates cellular detection rate as a covariate (Finak et al, Genome Biology, 2015)
- "DESeq2" : DE based on a model using the negative binomial distribution (Love et al, Genome Biology, 2014)

Run Seurat methods (for Seurat v3)

```r
library(Seurat)

data <- CreateSeuratObject(C)

# Normalize the data
scale.factor <- mean(colSums(C))
data <- NormalizeData(object = data, normalization.method = "LogNormalize",
                      scale.factor = scale.factor)

# Scale the data (needed for PCA etc)
data <- ScaleData(object = data)

# run all DE methods
methods <- c("wilcox","bimod","roc","t","negbinom","poisson","LR","MAST","DESeq2")
DE <- list()
for (m in methods){
 outfile <- paste("data/mouse_embryo/DE/seurat_",m,"_8cell_vs_16_cell.tab", sep='')
 if(!file.exists(outfile)){
   DE[[m]]<- FindMarkers(object = data,ident.1 = "X8cell",
                           ident.2 = "X16cell",test.use = m)
   write.table(DE[[m]],file=outfile,sep="\t",quote=F)
 }
}
```

```r
x2 <- rnorm(1000, 173, 1)
xfit2<-seq(min(x2),max(x2),length=40)
yfit2<-dnorm(xfit2,mean=mean(x2),sd=sd(x2))
yfit2 <- yfit2*diff(h$mids[1:2])*length(x2)
```

Jump to Schedule

Back to Introduction

Next to Methods Evaluation

Soneson, Charlotte, and Mark D Robinson. 2018. "Bias, robustness and scalability in single-cell differential expression analysis." *Nature Methods* 15 (February). Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved.: 255. https://doi.org/10.1038/nmeth.4612.

Wang, Tianyu, Boyang Li, Craig E Nelson, and Sheida Nabavi. 2019. "Comparative analysis of differential gene expression analysis tools for single-cell RNA sequencing data." *BMC Bioinformatics* 20 (1): 40. doi:10.1186/s12859-019-2599-6.