

Análise de dados de expressão genética – expressão diferencial (RNASeq)

Conceitos e implementação em R/ Bioconductor

RNASeq

Técnicas de *next generation sequencing* permitem sequenciar cDNA e assim ter medidas do mRNA associado

Dados de RNAseq têm algumas etapas de processamento desde os reads gerados pelo sequenciador, até chegar às contagens absolutas de cada sequência (e.g. gene / exon) – estes passos foram já revistos na UC de Algoritmos Avançados e não vão ser revistos aqui

No exemplo seguinte, vamos ver como tratar estes dados assumindo a disponibilidade de uma matriz processada de **contagens**: genes (nas linhas) x condições (colunas)

Package DESeq2

Para tratar dados de contagens provenientes de experiências de RNAseq iremos usar o package **DESeq2** do Bioconductor – implementa testes estatísticos (de Wald) com base num **modelo de regressão generalizada** (distribuição binomial negativa) adaptado a variáveis discretas (contagens)

Documentação completa disponível em:

<https://www.bioconductor.org/packages/3.3/bioc/vignettes/DESeq2/inst/doc/DESeq2.pdf>

```
> BiocManager::install("DESeq2")
```

Instalação do package
DESeq2 do Bioconductor

Publicação com os métodos: Michael I. Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology, 15:550, 2014
DESeq2 vignette:

<https://www.bioconductor.org/packages/3.3/bioc/vignettes/DESeq2/inst/doc/DESeq2.pdf>

Exemplo de dados de RNAseq: *drosophila*

<http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE32038>

```
> c1c2 = read.table("c1c2.allgenes.tab", h=T, row.names=1)
> dim(c1c2)
```

Ler dados

```
> condition = factor(c("C1", "C1", "C1", "C2", "C2", "C2"))
> cd=data.frame(c("C1", "C1", "C1", "C2", "C2", "C2"))
> colnames(cd)[1]="condition"
> rownames(cd)=colnames(c1c2)
> cd
```

Definir condições

```
> c1c2 = c1c2[ rowSums(c1c2) > 1, ]
> dim(c1c2)
```

Filtrar genes
com ≤ 1 cópias

Conjunto de dados pré-processado na aula de Alg. Avançados – vamos usar matriz de contagens resultante do pré-processamento mas aqui para todos os genes

Dados simulados de *Drosophila* com duas condições C1 e C2

Criar os objetos e correr DE

```
dds = DESeqDataSetFromMatrix(countData = c1c2, colData = cd, design = ~ condition)
dds
```

Objeto da classe *DESeqDataSet*
(dados e metadados)

```
> dds = DESeq(dds)
> res = results(dds)
> res
```

Correr testes de expressão diferencial
Ficam guardados como campo **results** do objeto original
Contém p-values e fold changes

```
> resOrdered = res[order(res$padj),]
> summary(res)
out of 10154 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)      : 254, 2.5%
LFC < 0 (down)    : 15, 0.15%
...
> sum(res$padj < 0.1, na.rm=TRUE)
269
```

Exploração dos resultados
269 genes DE dos quais 254 sobre-expressos (log
FC > 0 logo valores C2 maiores do que C1)

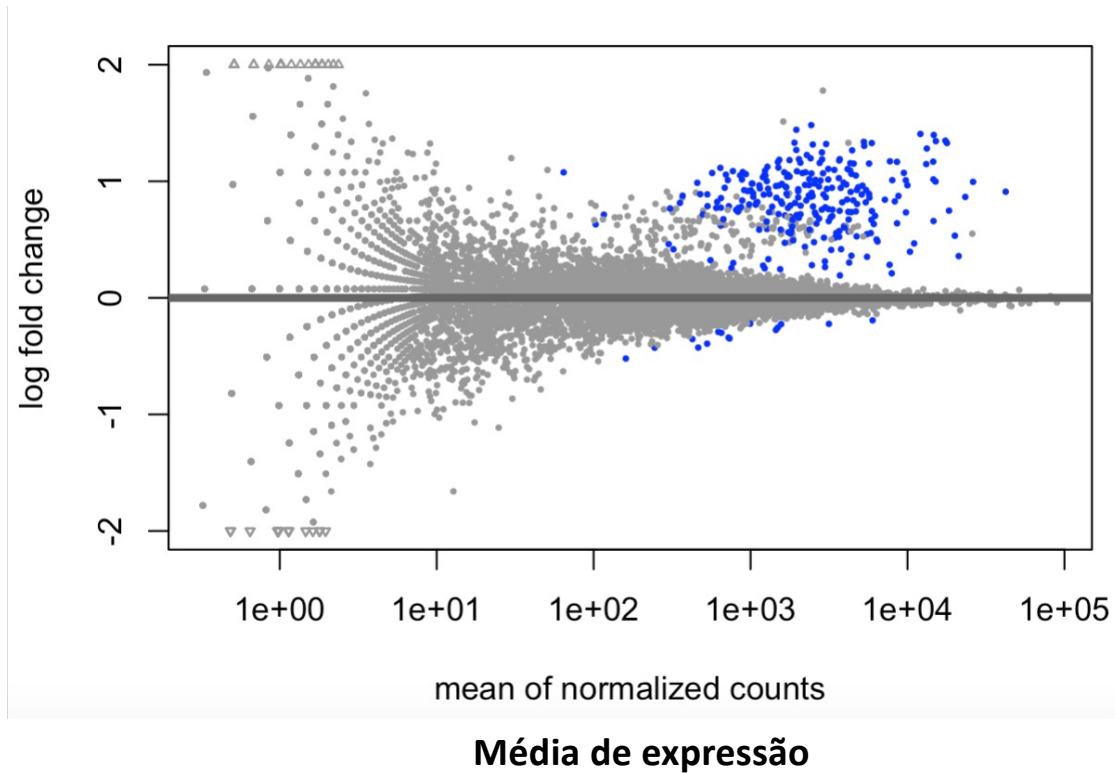
Exploração gráfica: MA plot

```
> DESeq2::plotMA(res, main="DESeq2", ylim=c(-2,2))
```

Over
C2 > C1

Log
Fold
Change

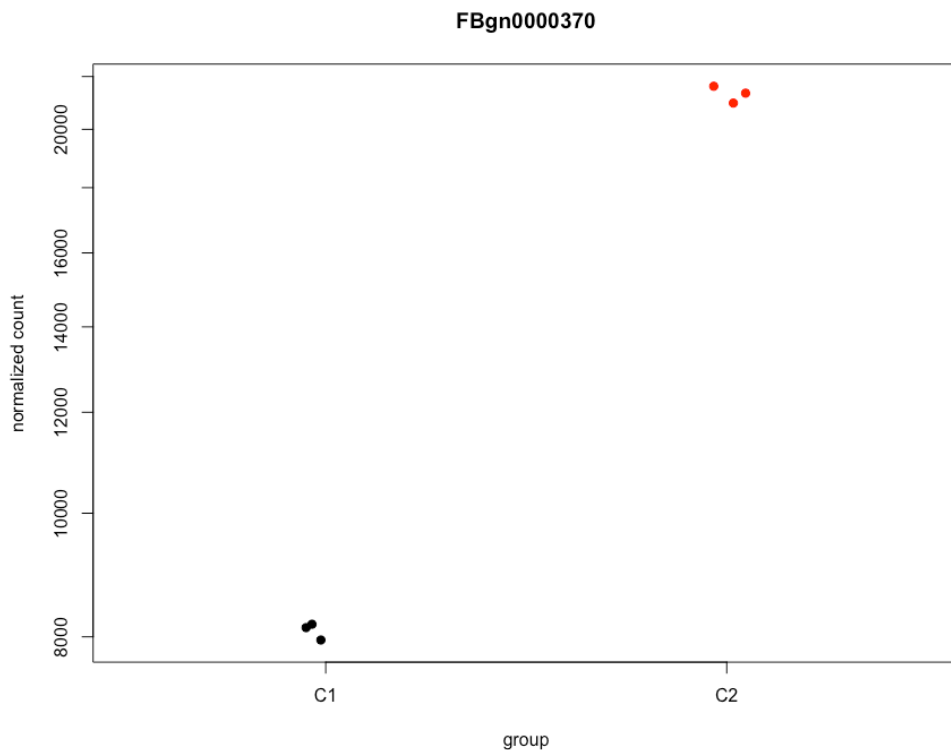
Under
C1 > C2



Pontos a azul – genes DE
Pontos a cinza: outros genes

Exploração gráfica de um gene

```
> plotCounts(dds, gene=which.min(res$padj), intgroup="condition", pch = 19, col = condition)
```




```
> resOrdered[1,]  
log2 fold change (MLE): condition C2 vs C1  
Wald test p-value: condition C2 vs C1  
DataFrame with 1 row and 6 columns  
      baseMean  log2FoldChange  
FBgn0000370 14701.2974720746 1.39742548330701 ...
```

Transformação de dados

- Para realizar os testes de expressão diferencial devem-se utilizar as contagens sem qualquer transformação, utilizando distribuições discretas, como vimos
- Para ajudar a visualização dos dados pode ser útil realizar transformações sobre os dados (a mais usada a transformação logarítmica)
- VST: *Variance Stabilizing Transformation* – mantém a variância independente da média

```
> vsd <- varianceStabilizingTransformation(dds, blind = FALSE)
> head(assay(vsd), 3)
> head(counts(dds), 3)
```

Faz a transformação
usando informação
das condições

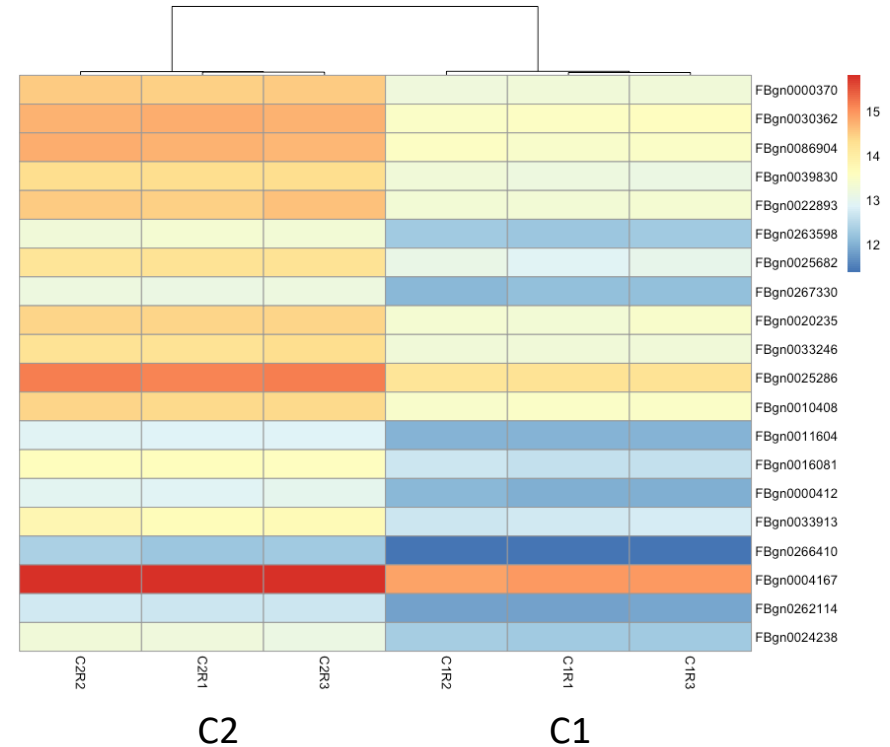


Heatmap dos genes

```
> select <- rownames(head(resOrdered,20))  
> vsd.counts <- assay(vsd)[select,]  
> df <- as.data.frame(colData(dds)[,c("condition")])
```

Selecionar apenas os 20 genes com maior variação (menor p-value)

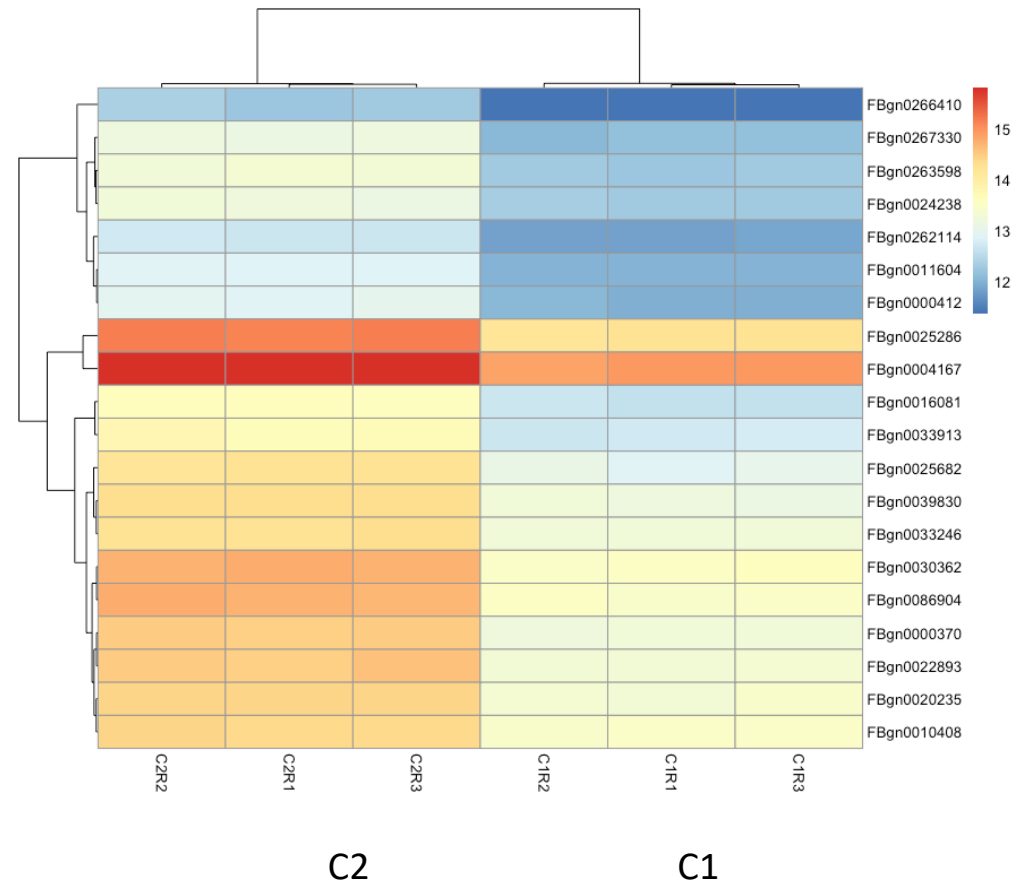
```
> pheatmap(vsd.counts, cluster_rows=FALSE)
```



Heatmap dos genes

```
> pheatmap(vsd.counts)
```

Neste caso, os genes também são agrupados pelo clustering hierárquico



Análise de dados de RNAseq

Um pipeline alternativo com edgeR

Pipeline adaptado de: <https://bioinformatics-core-shared-training.github.io/RNAseq-R>

Carregar os dados de contagens

- Vamos usar os dados do caso do mouse mammary data, mas não vamos usar os dados anteriores (AAB), pois apenas se referem a parte do genoma (cromossoma 1)
- Vamos carregar os dados completos de contagens para todos os genes (para a sub-pasta counts)

Gene counts: <https://figshare.com/s/1d788fd384d33e913a2a>

(ZIP file (Download all) – extrair ZIP para folder MouseData; também disponível no elearning)

Package edgeR

Uma alternativa ao DESeq2 para expressão diferencial e outras ferramentas de análise de dados RNAseq (contagens) é o package **edgeR** também do Bioconductor

```
> BiocManager::install(c("edgeR"))
> BiocManager::install(c("Glimma"))
> BiocManager::install(c("gplots"))
> BiocManager::install(c("org.Mm.eg.db"))

> library(edgeR)
> library(limma)
> library(Glimma)
> library(gplots)
> library(org.Mm.eg.db)
> library(RColorBrewer)
```

Instalação do **edgeR** e outros
packages do Bioconductor
usados no exemplo

Carregar os dados de contagens

```
# dados sobre as amostras
> sampleinfo <- read.delim("./MouseData/SampleInfo_corrected.txt",
  stringsAsFactors = TRUE))
> sampleinfo

# dados sobre as contagens
> seqdata <- read.delim("./MouseData/GSE60450_LactationGenewiseCounts.txt",
  stringsAsFactors = FALSE)
> head(seqdata)
> dim(seqdata)

> countdata <- seqdata[,-(1:2)] # remove duas primeiras colunas
> rownames(countdata) <- seqdata[,1] # IDs dos genes passam a ser nomes das linhas
> colnames(countdata) <- substr(colnames(countdata),1,7) # muda nomes das colunas
> head(countdata)
```

Pré-processamento dos dados de contagens

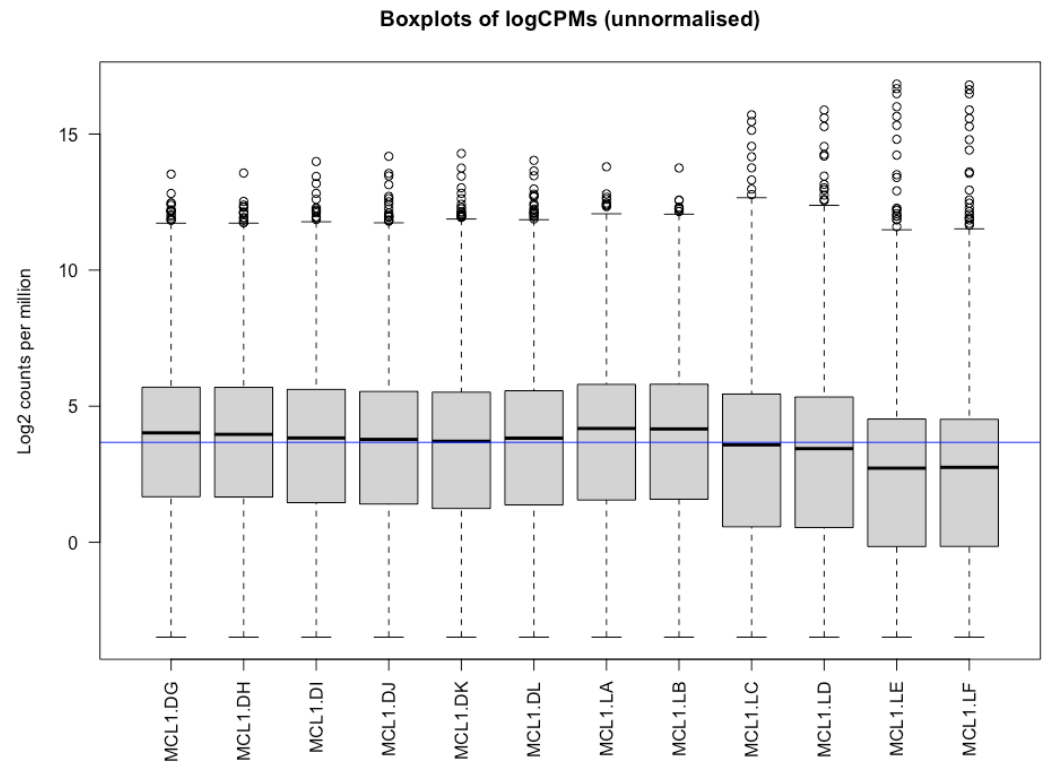
```
# contagens por milhão
> myCPM <- cpm(countdata)
> head(myCPM)

# filtra dados para ter apenas os genes com mais de 0.5 em pelo menos 2 amostras
> thresh <- myCPM > 0.5
> keep <- rowSums(thresh) >= 2
> counts.keep <- countdata[keep,]
> summary(keep)
> dim(counts.keep)

# cria objeto DGEList
> dgeObj <- DGEList(counts.keep)
> dgeObj
> names(dgeObj)
> dgeObj$samples
```

Transformação log e boxplots

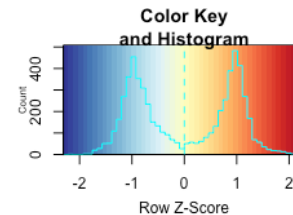
```
> logcounts <- cpm(dgeObj, log=TRUE)
> boxplot(logcounts, xlab="", ylab="Log2 counts per million", las=2)
> abline(h=median(logcounts), col="blue")
> title("Boxplots of logCPMs (unnormalised)")
```



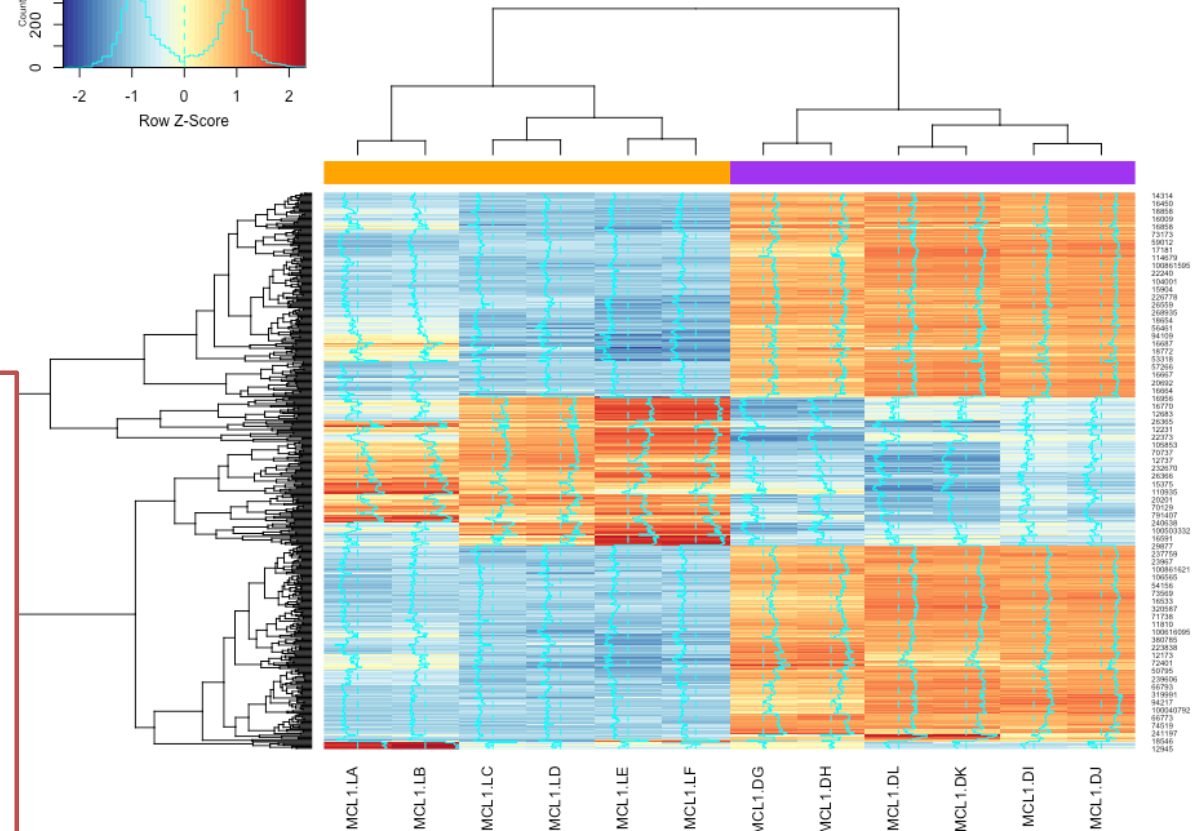
Visualização dos dados - heatmap

Selecionar os 500 genes com mais variabilidade

```
> var_genes <- apply(logcounts, 1, var)
> select_var <- names(sort(var_genes,
decreasing=TRUE))[1:500]
> highly_variable_lcpm <- logcounts[select_var,]
> mypalette <- brewer.pal(11,"RdYlBu")
> morecols <- colorRampPalette(mypalette)
> col.cell <-
c("purple","orange")[sampleinfo$CellType]
> heatmap.2(highly_variable_lcpm,
col=rev(morecols(50)),
trace="column",
main="Top 500 most variable genes
across samples",
colsideColors=col.cell,scale="row")
```



Top 500 most variable genes across samples

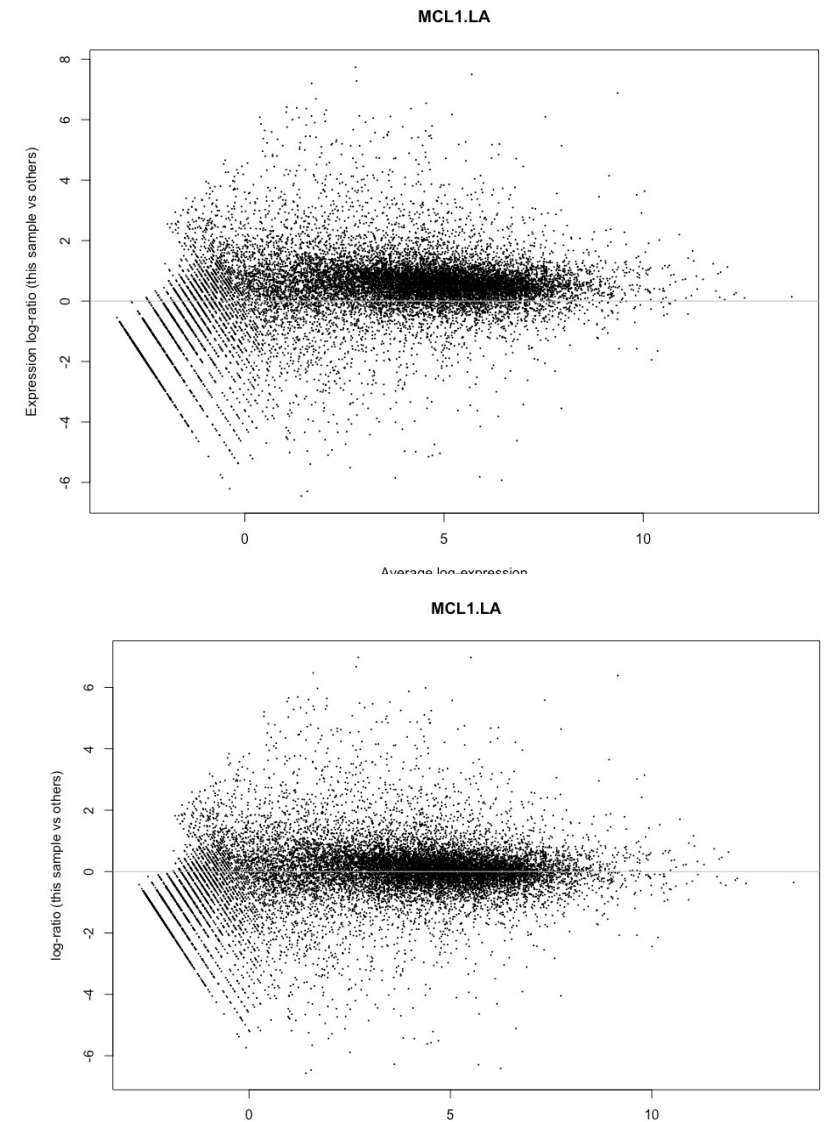


Normalização (*composition bias*)

```
> dgeObj <- calcNormFactors(dgeObj)
> dgeObj$samples

# antes
> plotMD(logcounts, column = 7)
> abline(h=0,col="grey")

# depois
> plotMD(dgeObj, column = 7)
> abline(h=0,col="grey")
```



Genes diferencialmente expressos

Design

```
> group <- paste(sampleinfo$CellType,sampleinfo$Status,sep=".")
> group

# design
> group <- as.character(group)
> type <- sapply(strsplit(group, "."), function(x) x[1])
> status <- sapply(strsplit(group, "."), function(x) x[2])
> design <- model.matrix(~ type + status)
> design
```

Factores:
- Type
- Status

Estimar dispersão

```
> dgeObj <- estimateCommonDisp(dgeObj)
> dgeObj <- estimateGLMTrendedDisp(dgeObj)
> dgeObj <- estimateTagwiseDisp(dgeObj)
```

```
> fit <- glmFit(dgeObj, design)
> names(fit)
> head(coef(fit))
```

Modelos de regressão

Testes

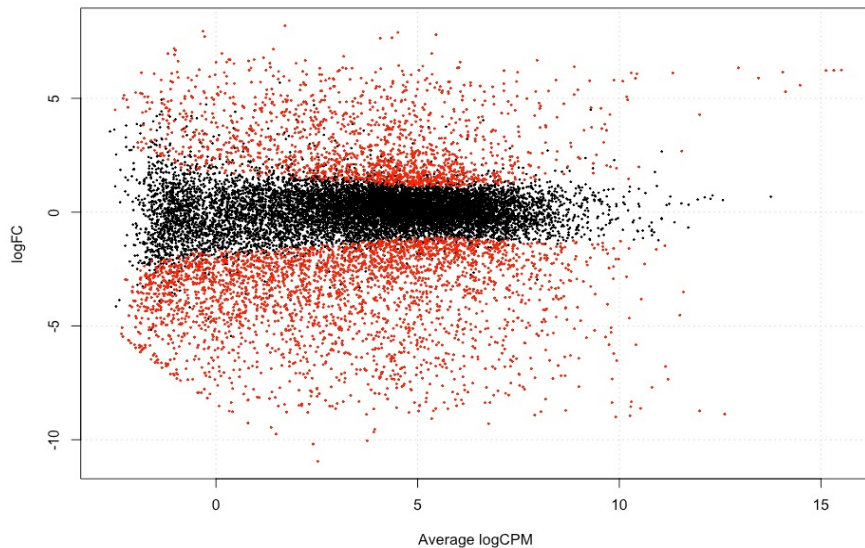
```
> lrt.BvSL <- glmLRT(fit, coef=2)
> topTags(lrt.BvSL)
```

Genes DE: estadísticas

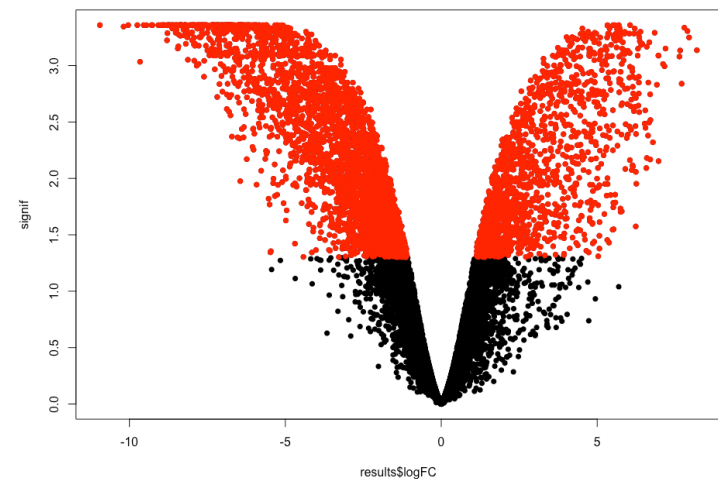
```
> results <- as.data.frame(topTags(lrt.BvsL,n = Inf))
> results
> dim(results)
> summary(de <- decideTestsDGE(lrt.BvsL))
```

Nº de genes DE

```
> detags <- rownames(dgeObj)[as.logical(de)]
> plotSmear(lrt.BvsL, de.tags=detags)
```



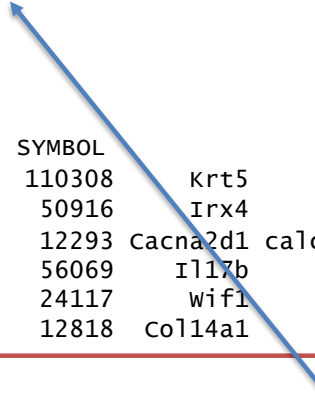
```
> signif <- -log10(results$FDR)
> plot(results$logFC, signif, pch=16)
> points(results[detags, "logFC"], -
log10(results[detags, "FDR"]), pch=16, col="red")
```



Genes DE: anotação

```
> ann <- select(org.Mm.eg.db,keys=rownames(results),columns=c("ENTREZID","SYMBOL","GENENAME"))
> head(ann)
> results.annotated <- cbind(results, ann)
> head(results.annotated)
```

logFC	logCPM	LR	Pvalue	FDR	ENTREZID	SYMBOL	GENENAME
110308	-8.940579	10.264297	24.89789	6.044842e-07	0.0004377963	110308 Krt5	keratin 5
50916	-8.636503	5.749781	24.80038	6.358508e-07	0.0004377963	50916 Irx4	Iroquois homeobox 4
12293	-8.362247	6.794788	24.68527	6.749824e-07	0.0004377963	12293 Cacna2d1	calcium channel, voltage-dependent, alpha2/delta subunit 1
56069	-8.419433	6.124377	24.41532	7.764857e-07	0.0004377963	56069 Il17b	interleukin 17B
24117	-9.290691	6.757163	24.32507	8.137328e-07	0.0004377963	24117 wif1	wnt inhibitory factor 1
12818	-8.216790	8.172247	24.24233	8.494459e-07	0.0004377963	12818 Col14a1	collagen, type XIV, alpha 1



Usa anotação (genoma: mouse) para ir buscar os genes que correspondem a cada transcrito

Genes DE: análise de enriquecimento - GSEA

```
> BiocManager::install(c("fgsea"))
> library(fgsea)

> results.ord <- results[ order(-results[, "logFC"]), ]
> ranks <- results.ord$logFC
> names(ranks) <- rownames(results.ord)

> load("./MouseData/mouse_H_v5.rdata")
> pathways <- Mm.H

> fgseaRes <- fgsea(pathways, ranks, minSize=15, maxSize = 500)
> class(fgseaRes)
> dim(fgseaRes)
> head(fgseaRes[order(padj), ])
```

Carrega grupos de genes do GSEA para cada pathway

Pathways com p-value mais significativo no enriquecimento

Acesso a dados do GDC - TCGA

Materiais úteis:

https://rpubs.com/tiagochst/TCGAbiolinks_to_DESeq2

<https://rpubs.com/tiagochst/TCGAworkshop>

<https://bioconductor.org/packages/release/bioc/vignettes/TCGAbiolinks/inst/doc/index.html>

Package TCGABiolinks

O package TCGABiolinks permite carregar dados do GDC (<https://gdc.cancer.gov/>), incluindo os dados do repositório TCGA, um dos maiores recursos para dados multiômicos sobre cancro

```
> BiocManager::install("TCGABiolinks")  
> BiocManager::install("MultiAssayExperiment")  
  
> browseVignettes("TCGABiolinks")
```

Colaprico, Antonio, et al. "TCGABiolinks: an R/Bioconductor package for integrative analysis of TCGA data." Nucleic acids research 44.8 (2015): e71-e71.

Silva, Tiago C., et al. "TCGA Workflow: Analyze cancer genomics and epigenomics data using Bioconductor packages." F1000Research 5 (2016).

(<https://f1000research.com/articles/5-1542/v2>)

Mounir, Mohamed, et al. "New functionalities in the TCGABiolinks package for the study and integration of cancer data from GDC and GTEx." PLoS computational biology 15.3 (2019): e1006701. (<https://doi.org/10.1371/journal.pcbi.1006701>)

Projetos disponíveis

O package TCGAbiolinks permite carregar dados de diversos projetos:

```
> library(TCGAbiolinks)
> library(MultiAssayExperiment)

> projects <- getGDCprojects()
> projects$id
```

Lista os projetos no portal do GDC

<https://portal.gdc.cancer.gov/projects>

Carregar dados de RNAseq

```
> proj <- "TCGA-GBM"
> query <- GDCquery(
  project = proj,
  data.category = "Transcriptome Profiling",
  data.type = "Gene Expression Quantification",
  workflow.type = "STAR - Counts")

> GDCdownload(query)
> data_rna_gbm <- GDCprepare(query)

> class(data_rna_gbm)
> dim(data_rna_gbm)

> data_rna_gbm$paper_BCR
> data_rna_gbm$paper_Gender
> data_rna_gbm$paper_Grade
> data_rna_gbm$paper_IDH.status

> meta_gbm = colData(data_rna_gbm)
> dim(meta_gbm)
> meta_gbm$patient
> meta_gbm$paper_IDH.status
```

Exemplo para o projeto TCGA-GBM

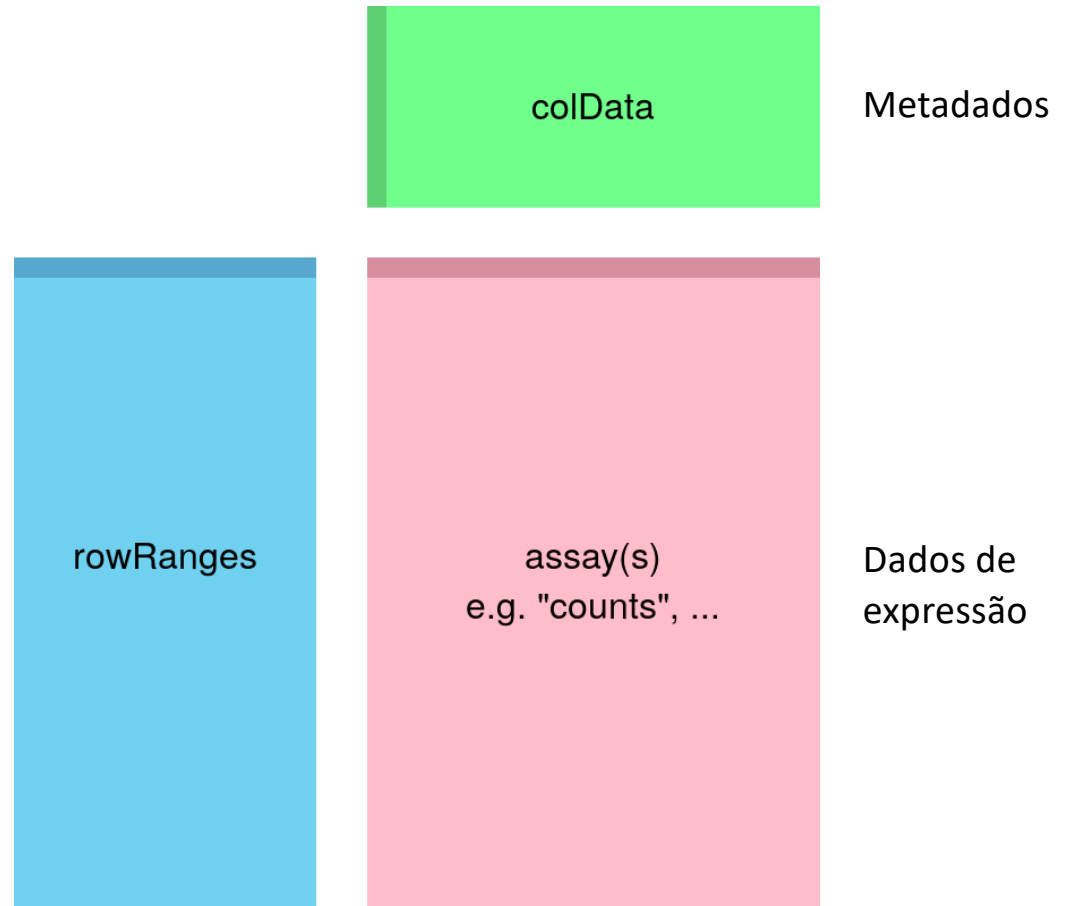
Ficheiros são guardados na pasta definida

Dados clínicos são já carregados em variáveis
data... \$paper_ ...

Dados clínicos globais podem ser obtidos com colData
(resultado data.frame)

Carregar dados de RNAseq

Estrutura de um objeto da
classe ***SummarizedExperiment***



Exemplo: DESeq2 sobre dados do GDC

```
> library(DESeq2)
> data_de = data_rna_gbm[,!is.na(data_rna_gbm$paper_IDH.status)]
> ddsSE <- DESeqDataSet(data_de, design = ~ paper_IDH.status)
> keep <- rowSums(counts(ddsSE)) >= 10
> ddsSE <- ddsSE[keep,]
> ddsSE <- DESeq(ddsSE)

> resultsNames(ddsSE)

> res <- results(ddsSE, name = "paper_IDH.status_WT_vs_Mutant")
> dea <- as.data.frame(res)
> summary(res)
```

Exemplo correndo
pipeline simples de
DE para dados de
RNAseq do GDC

Um exemplo completo com DESeq2

Ver:

<http://master.bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html>

Datasets do cbiportal

Datasets: <https://www.cbiportal.org/datasets>

Datasets provêm de muitos projetos, incluindo os do GDC-TCGA

Vários tipos de dados, incluindo expressão (na maioria dos casos RNAseq)

Dados de RNAseq quantificados usando RSEM e também com versões já standardizadas (Z-scores) - <https://docs.cbiportal.org/1.-general/faq#how-is-tcga-rnaseqv2-processed-what-units-are-used>; para DE usar versão RSEM sem standardização

Interface com R: <https://www.cbiportal.org/rmatlab>

Materiais interessantes: <https://cbiportal.github.io/2020-cbiportal-r-workshop/>

Paper do RSEM: <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-12-323>