

ANÁLISE DOS DADOS COM R

Bioconductor – RNASeq

James R. Hunter, PhD
Retrovirologia, EPM, UNIFESP

NÓTICIA IMPORTANTE

- Aula de 10 de dezembro (semana que vem) vai ser trocado para 8 de janeiro
- Semana que vem Prof. Fernando Antonelli vai dar a 1^a aula dele

INTRODUÇÃO

- Análise de dados da sequência de alto rendimento (“high-throughput sequence data”)
- Investigar vários aspectos de genes e transcripts
- Hoje, olhar na expressão diferencial dos genes
- O processo vai definir um dos muitos **fluxos de trabalho** possíveis de análise bioinformática

PACOTES PARA INSTALAR PARA A AULA

```
1 BiocManager::install(c("airway", "tximeta"))
```

- Instalação desses 2 pacotes vai causar instalação de vários outros necessários para funcionamento

DADOS - AIRWAY

- Um experimento RNA-seq que tratou células musculares lisas das vias aéreas com dexametasona
- 4 linhas de células
 - Cada linha tem uma amostra tratada e outra não-tratada
- Dexametasona - glicocorticoide sintético esteroide com efeitos anti-inflamatórios
- Utilizado no tratamento de doenças respiratórias entre outras coisas
- Dados resumidos no pacote [airway](#)

CARREGAR PACOTES

```
1 library(airway)  
2 library(tximeta)
```

OS PACOTES CARREGADOS

Tximeta & Airway Related Packages		
package	version	date
airway	1.26.0	2024-10-31
Biobase	2.66.0	2024-10-29
BiocGenerics	0.52.0	2024-10-29
GenomeInfoDb	1.42.0	2024-10-29
GenomicRanges	1.58.0	2024-10-29
IRanges	2.40.0	2024-10-29
MatrixGenerics	1.18.0	2024-10-29
matrixStats	1.4.1	2024-09-08
S4Vectors	0.44.0	2024-10-29
SummarizedExperiment	1.36.0	2024-10-29
tximeta	1.24.0	2024-11-08

- Pedir 2, pega 11

PASSO 1 DO FLUXO – QUANTIFICAÇÃO

- Quantificação de expressão dos transcripts
- Contagem feito fora de Bioconductor
- Muitos softwares fazem isso
- Este estudo utilizou **Salmon**
 - <https://combine-lab.github.io/salmon/>
- **airway** contem os arquivos saindo da operação de Salmon

ARQUIVOS DE airway

```
1 dir <- system.file("extdata", package="airway", mustWork=TRUE)
2 list.files(dir)

[1] "GSE52778_series_matrix.txt"          "Homo_sapiens.GRCh37.75_subset.gtf"
[3] "quants"                            "sample_table.csv"
[5] "SraRunInfo_SRP033351.csv"           "SRR1039508_subset.bam"
[7] "SRR1039509_subset.bam"              "SRR1039512_subset.bam"
[9] "SRR1039513_subset.bam"              "SRR1039516_subset.bam"
[11] "SRR1039517_subset.bam"             "SRR1039520_subset.bam"
[13] "SRR1039521_subset.bam"
```

- 8 arquivos BAM
- 1 de referência
- 2 de contagem (na sub-pasta **quants**)

ARQUIVOS COM CONTAGEM

```
1 ff <- list.files(file.path(dir, "quants"))  
[1] "SRR1039508" "SRR1039509"
```

BAM FILE

- *Binary Alignment Map*
- Os dados brutos abrangentes do sequenciamento genómico
- Forma binária de um arquivo SAM (*Sequence Alignment Map*)

TABELA DAS AMOSTRAS

- Próximo passo é criar um arquivo csv com Excel que liga as amostras para os arquivos associados FASTQ e Salmon.
- Aqui já tem ([sample_table.csv](#))

```
1 csvfile <- file.path(dir, "sample_table.csv")
2 coldata <- read_csv(csvfile)
3 coldata[, 1:5]
4 coldata[, c(1, 6:10)]
```

```
# A tibble: 8 × 5
  ...1      SampleName cell    dex   albut
  <chr>     <chr>     <chr>  <chr> <chr>
1 SRR1039508 GSM1275862 N61311 untrt untrt
2 SRR1039509 GSM1275863 N61311 trt   untrt
3 SRR1039512 GSM1275866 N052611 untrt untrt
4 SRR1039513 GSM1275867 N052611 trt   untrt
5 SRR1039516 GSM1275870 N080611 untrt untrt
6 SRR1039517 GSM1275871 N080611 trt   untrt
7 SRR1039520 GSM1275874 N061011 untrt untrt
8 SRR1039521 GSM1275875 N061011 trt   untrt

# A tibble: 8 × 6
  ...1      Run      avgLength Experiment Sample  BioSample
  <chr>     <chr>     <dbl>   <chr>    <chr>  <chr>
1 SRR1039508 SRR1039508       126 SRX384345 SRS508568 SAMN02422669
2 SRR1039509 SRR1039509       126 SRX384346 SRS508567 SAMN02422675
3 SRR1039512 SRR1039512       126 SRX384349 SRS508571 SAMN02422678
4 SRR1039513 SRR1039513        87 SRX384350 SRS508572 SAMN02422670
5 SRR1039516 SRR1039516       120 SRX384353 SRS508575 SAMN02422682
6 SRR1039517 SRR1039517       126 SRX384354 SRS508576 SAMN02422673
7 SRR1039520 SRR1039520       101 SRX384357 SRS508579 SAMN02422683
8 SRR1039521 SRR1039521        98 SRX384358 SRS508580 SAMN02422677
```

TRABALHAR COM SALMON QUANT DATA

- Só as primeiras duas amostras são os dados de Salmon.
 - Vamos trabalhar com eles
- Dá colunas nomes de `names` e `files`

```
1 coldata12 <- coldata[1:2,]
2 coldata12$names <- coldata12$Run
3 coldata12$files <- file.path(dir, "quants", coldata12$names, "quant.sf.gz")
4 file.exists(coldata12$files)
```

```
[1] TRUE TRUE
```

```
1 coldata12$files
```

```
[1]
```

```
"/Users/jameshunter/Library/R/arm64/4.4/library/airway/extdata/quants/SRR103950
```

```
[2]
```

```
"/Users/jameshunter/Library/R/arm64/4.4/library/airway/extdata/quants/SRR103950
```

IMPORTAR OS DADOS PARA PREPARAR ELES

- `tximeta` importa metadata e anotações durante a importação das quantificações dos transcripts
- Importa dados ao nível dos transcripts
- `tximeta()` localiza e faz o download das informações das fontes vários do internet.
- O resultado do trabalho é um objeto da classe `SummarizedExperiment (se)`

Tximeta performs numerous annotation and metadata gathering tasks on behalf of users during the import of transcript quantifications from *Salmon*, *alevin*, or *piscem-infer* into R/Bioconductor. Metadata and transcript ranges are added automatically, facilitating genomic analyses and assisting in computational reproducibility.

```
1 se <- tximeta(coldata12)
```

TXIMETA RETORNA A SEGUINTE INFORMAÇÃO

```
## importing quantifications

## reading in files with read_tsv

## 1 2
## found matching transcriptome:
## [ GENCODE - Homo sapiens - release 29 ]
## useHub=TRUE: checking for TxDb via 'AnnotationHub'
## found matching TxDb via 'AnnotationHub'
## loading from cache
## Loading required package: GenomicFeatures
## Loading required package: AnnotationDbi
## generating transcript ranges
```

TAMANHO DO CONJUNTO DOS DADOS

```
1 dim(se)
```

```
[1] 20580      2
```

```
1 head(rownames(se)) # 1a 6 fileiras
```

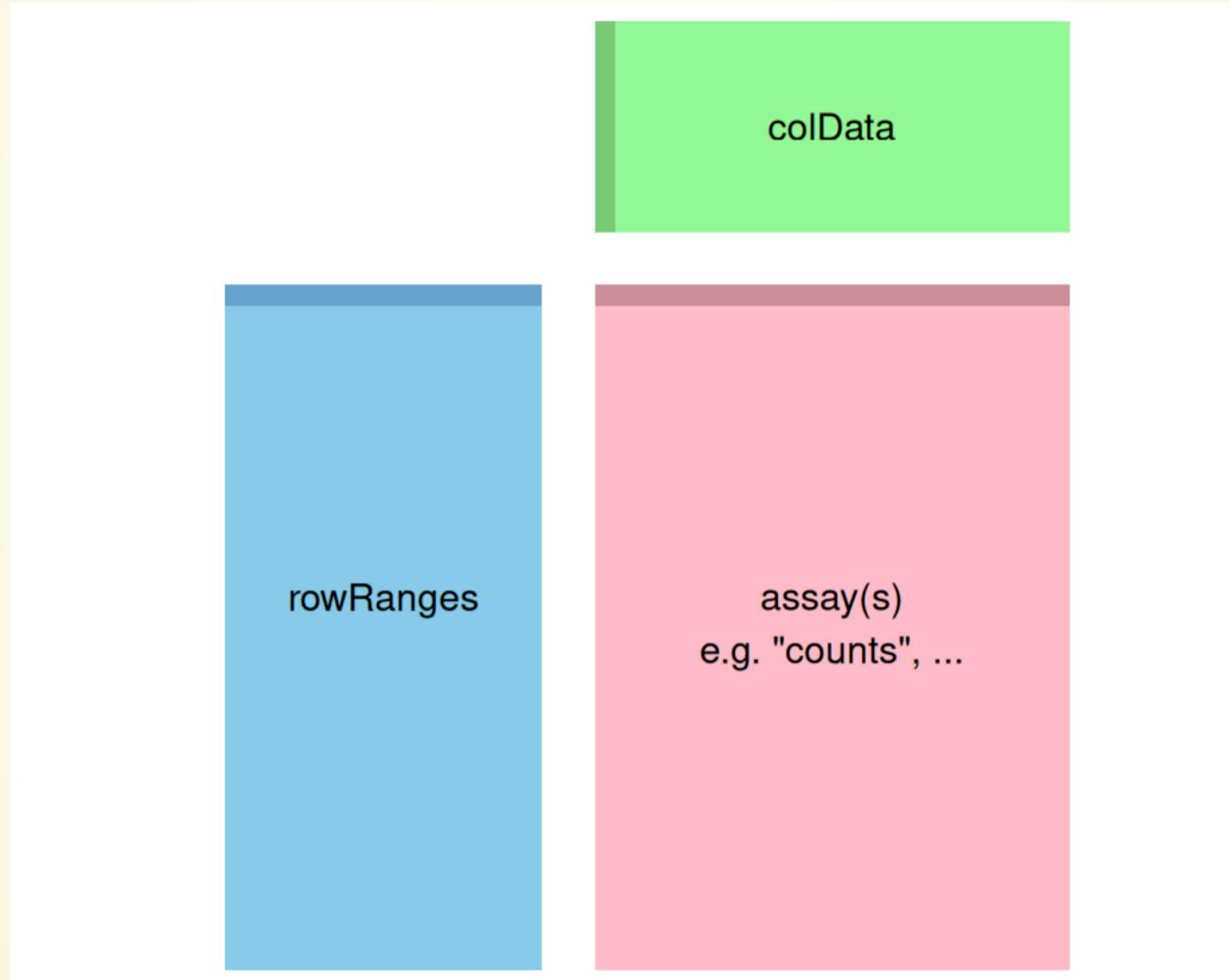
```
[1] "ENST00000456328.2" "ENST00000450305.2" "ENST00000488147.1"
```

```
[4] "ENST00000619216.1" "ENST00000473358.1" "ENST00000469289.1"
```

SUMMARIZEDEXPERIMENT

- Classe geral que facilita a transferência dos dados entre pacotes

SummarizedExperiment - 3 BLOCOS DE INFORMAÇÃO



3 BLOCOS

- rowRanges: info re: os intervalos genômicos
- colData: info re: as amostras
- “assays”/counts matriz da contagem
 - Os dados que vamos usar para a análise

CONVERTER AO UM SummarizedExperiment DOS GENES

```
1 gse <- summarizeToGene(se)
```

```
loading existing TxDb created: 2024-11-27 19:40:51
obtaining transcript-to-gene mapping from database
loading existing gene ranges created: 2024-11-27 20:15:03
assignRanges='range': gene ranges assigned by total range of isoforms
  see details at: ?summarizeToGene,SummarizedExperiment-method
summarizing abundance
summarizing counts
summarizing length
[1] 58294      2
```

VERIFICAR TAMANHO DOS GENES

```
1 dim(gse)
[1] 58294      2
1 head(rownames(gse))
[1] "ENSG0000000003.14" "ENSG0000000005.5"   "ENSG0000000419.12"
[4] "ENSG0000000457.13" "ENSG0000000460.16"  "ENSG0000000938.12"
```

- 58,294 genes invés de 205,870 transcripts e só 53 MB

PRONTO PARA DESEQ2

- Verdadeira análise
- O fluxo de trabalho que usamos é só um. Existem vários

VERSÃO FINAL DE gse

- Vamos usar um outra versão de `gse` que vem do pacote (agora com 38MB)

```
1 data(gse) #por na memoria do pacote
2 gse

class: RangedSummarizedExperiment
dim: 58294 8
metadata(6): tximetaInfo quantInfo ... txomeInfo txdbInfo
assays(3): counts abundance length
rownames(58294): ENSG00000000003.14 ENSG00000000005.5 ...
  ENSG0000285993.1 ENSG0000285994.1
rowData names(1): gene_id
colnames(8): SRR1039508 SRR1039509 ... SRR1039520 SRR1039521
colData names(3): names donor condition
```

OLHAR NA MATRIZ counts

```
1 assayNames(gse)
```

```
[1] "counts"    "abundance" "length"
```

```
1 head(assay(gse), 3) # só 3 fileiras
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG00000000003.14	708.164	467.962	900.992	424.368	1188.295
ENSG00000000005.5	0.000	0.000	0.000	0.000	0.000
ENSG00000000419.12	455.000	510.000	604.000	352.000	583.000
	SRR1039517	SRR1039520	SRR1039521		
ENSG00000000003.14	1090.668	805.929	599.337		
ENSG00000000005.5	0.000	0.000	0.000		
ENSG00000000419.12	773.999	409.999	499.000		

- Os elementos dos dados que são o objetivo do estudo

TOTAIS DAS COLUNAS

- Através de todas as amostras
- Nova função `colSums()`
 - Faz total de todos os elementos de todos os colunas

```
1 colSums(assay(gse))
```

```
SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520  
21100805    19298584    26145537    15688246    25268618    31891456    19683767  
SRR1039521  
21813903
```

ROWRANGES

- Informação sobre os genes (fileiras)
- No formato de GRanges
 - Classe para organizar genes ou transcripts mostrando os dados
 - Extenso do gene (Start - Stop)
 - Metadata
 - Sequências (cromossomos)
- Comando mostra os dados sobre os primeiro 5 e último 5 ranges

```
1 rowRanges(gse)
```

GRanges object with 58294 ranges and 1 metadata column:

	seqnames	ranges	strand	gene_id
	<Rle>	<IRanges>	<Rle>	<character>
ENSG00000000003.14	chrX	100627109–100639991	-	ENSG00000000003.14
ENSG00000000005.5	chrX	100584802–100599885	+	ENSG00000000005.5
ENSG0000000419.12	chr20	50934867–50958555	-	ENSG0000000419.12
ENSG0000000457.13	chr1	169849631–169894267	-	ENSG0000000457.13
ENSG0000000460.16	chr1	169662007–169854080	+	ENSG0000000460.16
...
ENSG0000285990.1	chr14	19244904–19269380	-	ENSG0000285990.1
ENSG0000285991.1	chr6	149817937–149896011	-	ENSG0000285991.1
ENSG0000285992.1	chr8	47129262–47132628	+	ENSG0000285992.1
ENSG0000285993.1	chr18	46409197–46410645	-	ENSG0000285993.1
ENSG0000285994.1	chr10	12563151–12567351	+	ENSG0000285994.1

INFORMAÇÃO SOBRE AS SEQUÊNCIAS

```
1 seqinfo(rowRanges(gse))
```

Seqinfo object with 25 sequences (1 circular) from hg38 genome:

seqnames	seqlengths	isCircular	genome
chr1	248956422	FALSE	hg38
chr2	242193529	FALSE	hg38
chr3	198295559	FALSE	hg38
chr4	190214555	FALSE	hg38
chr5	181538259	FALSE	hg38
...
chr21	46709983	FALSE	hg38
chr22	50818468	FALSE	hg38
chrX	156040895	FALSE	hg38
chrY	57227415	FALSE	hg38
chrM	16569	TRUE	hg38

COLDATA – INFO RE: AMOSTRAS

```
1 colData(gse)
```

DataFrame with 8 rows and 3 columns

	names	donor	condition
	<factor>	<factor>	<factor>
SRR1039508	SRR1039508	N61311	Untreated
SRR1039509	SRR1039509	N61311	Dexamethasone
SRR1039512	SRR1039512	N052611	Untreated
SRR1039513	SRR1039513	N052611	Dexamethasone
SRR1039516	SRR1039516	N080611	Untreated
SRR1039517	SRR1039517	N080611	Dexamethasone
SRR1039520	SRR1039520	N061011	Untreated
SRR1039521	SRR1039521	N061011	Dexamethasone

DESEQ2 - NOSSA FERRAMENTA

- Entre muitas outras

CRIA UMA CLASSE DESeqDataSet

- Construído acima de um `SummarizedExperiment`
- Agora, o slot `assay` será acessado por `counts`
- Tem uma `formula` de design associado
- Podemos acessar os dados das colunas com `$`
 - Aplica igualmente a `SummarizedExperiment` e `DESeqDataSet`
 - Mesmo anotação que os data frames

PODE VER A HIERARQUIA DOS SLOTS COM `View()`

Name	Type	Value
gse	S4 [58294 x 8] (SummarizedExperiment)	S4 object of class RangedSummarizedExperiment
rowRanges	S4 (GenomicRanges::GRanges)	S4 object of class GRanges
colData	S4 [8 x 3] (S4Vectors::DFrame)	S4 object of class DFrame
rownames	character [8]	'SRR1039508' 'SRR1039509' 'SRR1039512' 'SRR1039513' 'SRR1039516' 'SRR1039517' ...
nrows	integer [1]	8
elementType	character [1]	'ANY'
elementMetadata	NULL	Pairlist of length 0
metadata	list [0]	List of length 0
listData	list [3]	List of length 3
names	factor	Factor with 8 levels: "SRR1039508", "SRR1039509", "SRR1039512", "SRR1039513", "S..."
donor	factor	Factor with 4 levels: "N052611", "N061011", "N080611", "N61311"
condition	factor	Factor with 2 levels: "Untreated", "Dexamethasone"
assays	S4 [58294 x 8] (SummarizedExperiment)	S4 object of class SimpleAssays
NAMES	NULL	Pairlist of length 0
elementMetadata	S4 [58294 x 0] (S4Vectors::DF)	S4 object of class DFrame
metadata	list [6]	List of length 6

ALGUMAS COLUNAS DE gse

```
1 gse$donor
```

```
[1] N61311 N61311 N052611 N052611 N080611 N080611 N061011 N061011  
Levels: N052611 N061011 N080611 N61311
```

```
1 gse$condition
```

```
[1] Untreated Dexamethasone Untreated Dexamethasone Untreated  
[6] Dexamethasone Untreated Dexamethasone  
Levels: Untreated Dexamethasone
```

Qual tipo de dado são **donor** e **condition**?

PODE MANIPULAR AS VARIÁVEIS

- `donor` não faz muito sentido em nosso contexto: trocar para `cell`
- Trocar `dex` no lugar de `condition`

```
1 gse$cell <- gse$donor
2 gse$dex <- gse$condition
3
4 # resultado
5 gse@colData@listData

$names
[1] SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520
[8] SRR1039521
8 Levels: SRR1039508 SRR1039509 SRR1039512 SRR1039513 ... SRR1039521

$donor
[1] N61311 N61311 N052611 N052611 N080611 N080611 N061011 N061011
Levels: N052611 N061011 N080611 N61311

$condition
[1] Untreated      Dexamethasone Untreated      Dexamethasone Untreated
[6] Dexamethasone Untreated      Dexamethasone
Levels: Untreated Dexamethasone

$cell
[1] N61311 N61311 N052611 N052611 N080611 N080611 N061011 N061011
Levels: N052611 N061011 N080611 N61311

$dex
[1] Untreated      Dexamethasone Untreated      Dexamethasone Untreated
```

PODEMOS TIRAR AS 2 VARIÁVEIS QUE NÃO PRESTAM MAIS

```
1 gse$condition <- NULL
2 gse$donor <- NULL
3
4 # resultado
5 gse@colData@listData

$names
[1] SRR1039508 SRR1039509 SRR1039512 SRR1039513 SRR1039516 SRR1039517 SRR1039520
[8] SRR1039521
8 Levels: SRR1039508 SRR1039509 SRR1039512 SRR1039513 ... SRR1039521

$cell
[1] N61311 N61311 N052611 N052611 N080611 N080611 N061011 N061011
Levels: N052611 N061011 N080611 N61311

$dex
[1] Untreated Dexamethasone Untreated Dexamethasone Untreated
[6] Dexamethasone Untreated Dexamethasone
Levels: Untreated Dexamethasone
```

PODE MANIPULAR OS NÍVEIS

- Função (base R) levels() relata os níveis dos fatores
 - Tb, pode receber novos valores
- VSS - a ordem dos níveis devem ser preservada
- Para dex
 - Trocar “Untreated” para “untrt”
 - Trocar “Dexamethasone” para “trt”

```
1 levels(gse$dex)
```

```
[1] "Untreated"      "Dexamethasone"
```

```
1 levels(gse$dex) <- c("untrt", "trt")  
2 levels(gse$dex)
```

```
[1] "untrt" "trt"
```

DESIGN FORMULA

- Este elemento usamos para contar para DESeq2 o que queremos testar
- É o tipo de anotação que usamos em muitas funções
 - E.g., `lm()`, a função que usamos para fazer regressão linear
- Queremos saber para quais genes o efeito de tratamento é diferente
- Normalmente tem formato de: ~ `group + treatment`
 - Pode incluir um termo de interação, se for desejado
 - `~ group + treatment + group:treatment`
 - Não em nosso caso

LEMBRE O TAMANHO DOS DADOS

- Salmon mapeou milhões de fragmentos genômicos
 - Abaixo, expresso em milhões de fragmentos

```
1 round( colSums(assay(gse)) / 1e6, 1 )
```

SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516	SRR1039517	SRR1039520
21.1	19.3	26.1	15.7	25.3	31.9	19.7
SRR1039521						
21.8						

CARREGAR DESEQ2

- Chamar a função `DESeqDataSet()` para preparar as operações de algoritmo
- Usando o design formula
- Produz um “Large `DESeqDataSet (58294 elements, 36.1 MB)`”

```
1 library(DESeq2)
2 dds <- DESeqDataSet(gse, design = ~ cell + dex)
```

EXPLORAÇÃO DE dds

- Antes de fazer a análise vamos explorar um pouco nossos dados
- Especialmente as contagens

SÓ QUEREMOS GENES QUE DÁ INFORMAÇÃO

- Filtrar as fileiras dos `counts` que têm suficiente fragmentos para informar nossa análise
 - Muitas genes vão ter um número tão pequeno de fragmentos que são inúteis para a análise
- Vamos olhar um pouco na estrutura de `counts`

ESTRUTURA DE counts

```
1 str(counts(dds))
```

```
int [1:58294, 1:8] 708 0 455 312 88 0 3228 2298 549 427 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:58294] "ENSG0000000003.14" "ENSG0000000005.5" "ENSG0000000419.12" "ENSG0000000457.13" ...
..$ : chr [1:8] "SRR1039508" "SRR1039509" "SRR1039512" "SRR1039513" ...
```

```
1 # olhar no head de counts
2 counts(dds)[1:6, 1:8]
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG0000000003.14	708	468	901	424	1188
ENSG0000000005.5	0	0	0	0	0
ENSG0000000419.12	455	510	604	352	583
ENSG0000000457.13	312	268	363	224	361
ENSG0000000460.16	88	74	52	45	107
ENSG0000000938.12	0	0	2	0	1
	SRR1039517	SRR1039520	SRR1039521		
ENSG0000000003.14	1091	806	599		
ENSG0000000005.5	0	0	0		
ENSG0000000419.12	774	410	499		
ENSG0000000457.13	430	300	289		
ENSG0000000460.16	101	97	83		
ENSG0000000938.12	0	0	0		

PROPOSTA

- Pelo menos, 4 células de cada gene (fileira) deve ter ao menos 10 fragmentos
 - fragmentos = count da célula
 - 4 células porque temos 4 grupos que estamos avaliando
 - 4 linhas de células diferentes
- `rowSums()` calcula o total dos valores dentro de uma fileira, para todas as fileiras.
- Queremos ter então pelo menos metade das colunas (4 de 8) com uma contagem ≥ 10
- Vai fazer o processamento mais rápido e reduzir o tamanho do DESeqDataSet

PARA CHEGAR LÁ . . .

- Quantas fileiras têm uma contagem de 0
 - Nenhuma amostra teve fragmento sobreposto do gene

```
1 zeros <- rowSums(counts(dds)) == 0  
2 ddsz <- dds=zeros, ]  
3 dim(ddsz)
```

```
[1] 22721     8
```

```
1 moreten <- rowSums(counts(dds)) >= 10  
2 ddsmt <- dds[moreten, ]  
3 dim(ddsm)
```

```
[1] 24260     8
```

PARA COMPLETAR O CÁLCULO

- Vamos ver quantos de `ddsmt` tem pelo menos 4 amostras com esta contagem

```
1 keep <- (rowSums(counts(dds) >= 10)) >= 4
2 # parênteses em torno do 1º teste para facilitar a leitura
3 dds_fin <- dds[keep, ]
4 # tamanho
5 dim(dds_fin)
```

```
[1] 16637     8
```

```
1 # mostrar cabeça de novo
2 counts(dds_fin)[1:6, 1:8]
```

	SRR1039508	SRR1039509	SRR1039512	SRR1039513	SRR1039516
ENSG00000000003.14	708	468	901	424	1188
ENSG00000000419.12	455	510	604	352	583
ENSG00000000457.13	312	268	363	224	361
ENSG00000000460.16	88	74	52	45	107
ENSG00000000971.15	3228	3655	6066	4210	6636
ENSG00000001036.13	2298	1630	2682	1357	2209
	SRR1039517	SRR1039520	SRR1039521		
ENSG00000000003.14	1091	806	599		
ENSG00000000419.12	774	410	499		
ENSG00000000457.13	430	300	289		
ENSG00000000460.16	101	97	83		
ENSG00000000971.15	10855	5032	7771		
ENSG00000001036.13	2214	2115	1736		

```
1 dim(dds_fin)
```

```
[1] 16637     8
```

CONCLUSÕES DESTE PASSO

1. Agora chegamos com 16.637 genes que têm informação útil
2. Velho ENSG00000000005.5 não é mais na 2^a fileira – contagem 0

FLUXO PARA EXPRESSÃO DIFERENCIAL

- Especificamos um desenho experimental na *design formula*
 - Quando criamos o `DESeqDataSet` :
“`DESeqDataSet(gse, design = ~ cell + dex)`”
- Utilizando as contagens brutas que já temos, podemos chamar a função `DESeq`
- Retorna um novo `DESeqDataSet` com todos os parâmetros da análise

```
1 dds_out <- DESeq(dds_fin)
```

CONTEÚDO DOS RESULTADOS - `dds_fin`

- Objetivo da função: calcular a expressão diferencial de tratamento ou não-tratamento dos genes
- Produz um serie de resultados num data frame
- A expressão diferencial está diretamente visto na coluna “`log2FoldChange`” que está uma transformação das contagens
 - Controla para diferenças em na profundidade do sequenciamento das amostras e a disperção do cada gene
- Aplica um modelo linear geral com a distribuição “*Negative Binomial*”
- Indicações estatísticas sobre a diferencial para cada gene

TABELA DOS RESULTADOS

```
1 (res <- results(dds_out))

log2 fold change (MLE): dex trt vs untrt
Wald test p-value: dex trt vs untrt
DataFrame with 16637 rows and 6 columns
  baseMean log2FoldChange    lfcSE      stat      pvalue
  <numeric>     <numeric> <numeric> <numeric>     <numeric>
ENSG00000000003.14 740.1093 -0.365327 0.1073385 -3.403501 0.000665281929
ENSG000000000419.12 511.6990  0.202232 0.1278579  1.581694 0.113719502671
ENSG000000000457.13 314.1680  0.033792 0.1552106  0.217717 0.827649766335
ENSG000000000460.16 79.7988 -0.120633 0.3055270 -0.394836 0.692964158474
ENSG000000000971.15 5715.3064 0.442982 0.0904089  4.899766 0.000000959508
...
ENSG0000285953.1    29.5747 -1.920562 0.649661 -2.956253 0.00311401
ENSG0000285967.1    181.1650 -0.325885 0.179340 -1.817132 0.06919688
ENSG0000285976.1    875.4424  0.262132 0.142980  1.833351 0.06675037
ENSG0000285979.1    38.3502  0.338383 0.348445  0.971124 0.33148631
ENSG0000285991.1    11.2772 -0.115472 0.723139 -0.159681 0.87313221
  padj
  <numeric>
ENSG00000000003.14 0.0046355222
ENSG000000000419.12 0.2881253592
```

- Quer saber o que querem dizer essas colunas?

METADATA NA `res`

- Função do pacote `S4Vectors`
- Especificamente acessar as colunas com metadata

```
1 mcols(res, use.names = TRUE)
```

DataFrame with 6 rows and 2 columns

	type	description
	<character>	<character>
baseMean	intermediate mean of normalized c..	
log2FoldChange	results log2 fold change (ML..	
lfcSE	results standard error: dex ..	
stat	results Wald statistic: dex ..	
pvalue	results Wald test p-value: d..	
padj	results BH adjusted p-values	

log2FoldChange

- Estimativa de tamanho de efeito (*effect size*)
- Relata quanto a expressão do gene mudou por causa do tratamento com dexametasona
- Escala logarítmica relativa a base 2

$$\log2FoldChange = 1.5 = 2^{1.5} \approx 2.828$$

- `lfcSE` – erro padrão de *fold change*
- Outros – estatísticas

RESUMO DOS RESULTADOS

- A função generica `summary()` tem método especial aqui para classe `DESeqResults`
- Pode ajustar o valor-p com o argumento `alpha` = e um valor (e.g. 0.05)
- Temos aqui 4381 genes mostrando diferencial dos 16637 genes

```
1 summary(res)
```

```
out of 16637 with nonzero total read count
adjusted p-value < 0.1
LFC > 0 (up)      : 2362, 14%
LFC < 0 (down)    : 2019, 12%
outliers [1]       : 0, 0%
low counts [2]     : 646, 3.9%
(mean count < 12)
[1] see 'cooksCutoff' argument of ?results
[2] see 'independentFiltering' argument of ?results
```

MUDANÇA PARA α DE 0.05

```
1 res05 <- results(dds_out, alpha = 0.05)
2 table(res05$padj < 0.05)
```

```
FALSE  TRUE
12712 3602
```

- Redução do número de genes que mostram a diferencial significativa

GRÁFICOS DOS RESULTADOS

- Uma amostra das possibilidades gráficas

GRÁFICO DAS CONTAGENS BRUTAS

- Queremos saber com o gene mais significativa mudou por causa de tratamento
- Vamos indentificar o gene mais significativa, i.e., o gene com o valor mínimo de `padj`
- Mostrar as contagens brutas relacionados a ele

```
1 (topgene <- rownames(res)[which.min(res$padj)])  
[1] "ENSG00000189221.9"
```

CONSTRUIR GRÁFICO

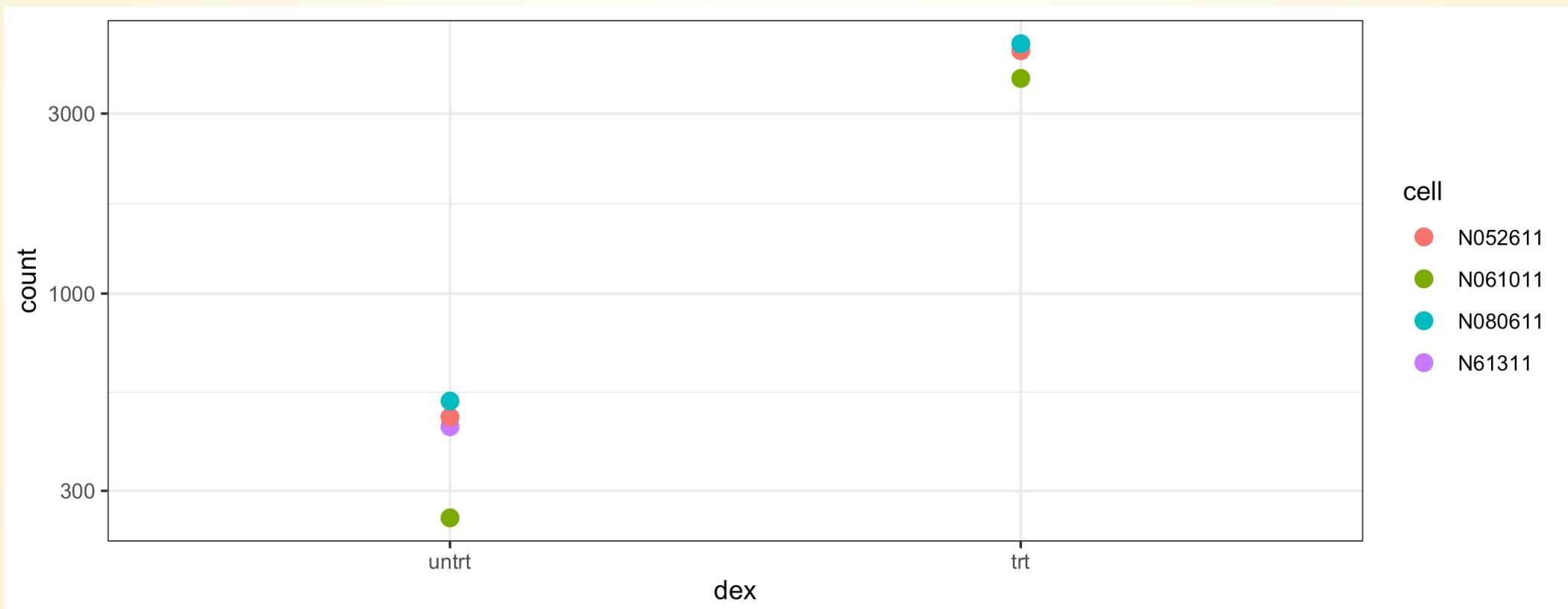
- Precisa `DESeq2::plotCounts()` – função que prepara contagens para gráficos fazendo
 - Normalização
 - Mais um *pseudocontagem* de 0.5 para evitar logs de 0

```
1 library(ggplot2)
2 geneCounts <- plotCounts(dds_out, gene = topgene, intgroup = c("dex", "cell"),
3                           returnData = TRUE)
4 head(geneCounts)
```

	count	dex	cell
SRR1039508	443.7641	untrt	N61311
SRR1039509	4570.3854	trt	N61311
SRR1039512	470.3890	untrt	N052611
SRR1039513	4403.9701	trt	N052611
SRR1039516	518.8819	untrt	N080611
SRR1039517	4594.8870	trt	N080611

O GRÁFICO

```
1 ggplot(geneCounts, aes(x = dex, y = count, color = cell)) +  
2   scale_y_log10() +  
3   geom_point(size = 3) +  
4   theme_bw()
```



PARA FAZER MAIS CLARO, ACRESCENTAR LINHAS

```
1 ggplot(geneCounts, aes(x = dex, y = count, color = cell, group = cell)) +  
2   scale_y_log10() +  
3   geom_point() +  
4   geom_line() +  
5   scale_colour_manual(values = c("#800000FF", "#767676FF", "#155F83FF", "#808080FF")) +  
6   theme_bw()
```

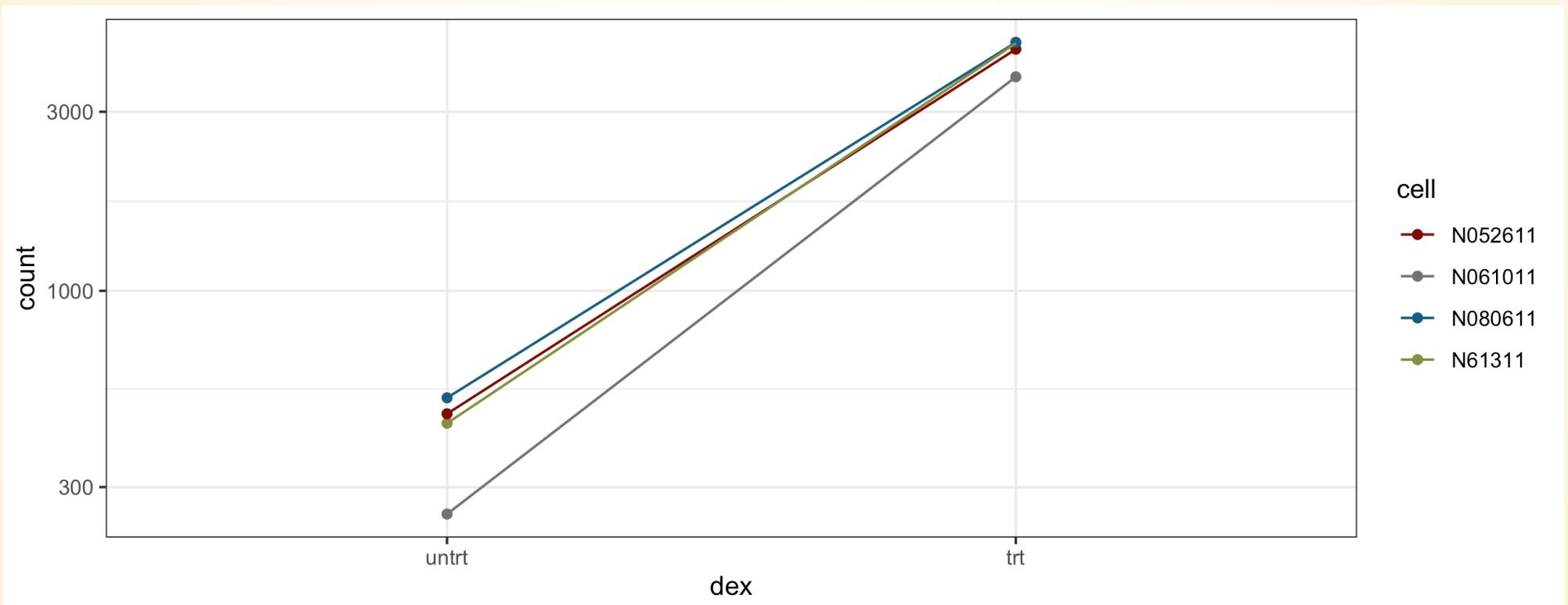


GRÁFICO MA

- MA - “*mean average*” – log2 fold changes (no eixo y) x a média das contagens normalizada
- Mostra quanto variância tem no modelo
- Cores padronizados:
 - Azul - genes com fold change significativa
 - Cinza - genes com fold change não - significativa
- Também chamado gráfico Bland-Altman

PASSO ANTERIOR

- Precisa restritar (“shrink”) os valores dos log2 fold changes para comparison das amostras tratadas contra as amostras não tratadas
- Função `lfcShrink()` usa um dos 3 algoritmos para fazer esta restrição
- Nós vamos usar o algoritmo `apeglm`, que fica no pacote `apeglm`
- Aplicar este algoritmo para a comparação que nos interesse – `dex_trt_vs_untrt`
- Achamos este nome no resultado da função `resultsNames()`

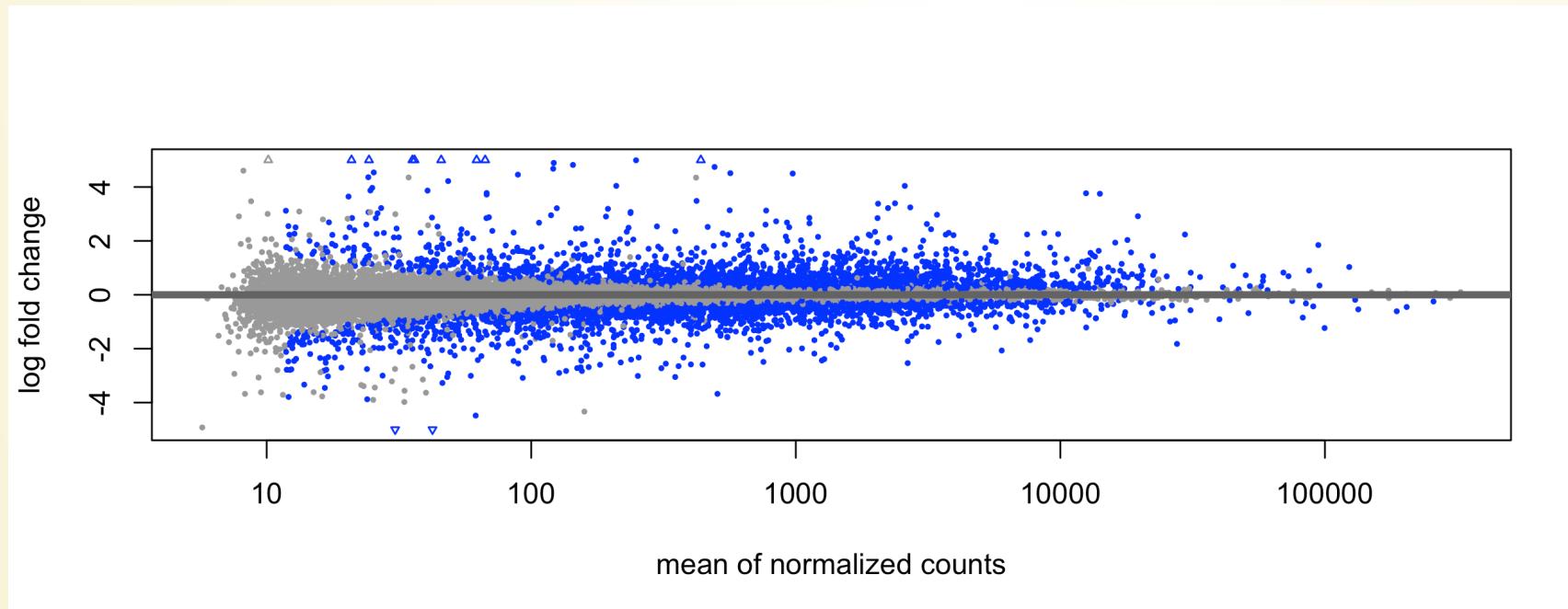
```
1 library(apeglm)
2 resultsNames(dds_out)

[1] "Intercept"           "cell_N061011_vs_N052611"
[3] "cell_N080611_vs_N052611" "cell_N61311_vs_N052611"
[5] "dex_trt_vs_untrt"
```

LFC SHRINK E GRÁFICO

- Agoritmo de shrink utiliza uma procedimento Bayesiano para suavizar contagens muito baixas ou com uma variância alta.
- Gráfico mostra que como os normalized counts aumentam os genes com fold change insignificativa diminuem

```
1 res_shrink <- lfcShrink(dds_out, coef="dex_trt_vs_untrt", type="apeglm")
2 plotMA(res, ylim = c(-5, 5))
```



O QUE FAZER COM ESSES RESULTADOS

RELATÓRIOS E ANOTACÕES

- Até agora, nossos resultados são muitos abstratos
- Não tem como descrever eles em termos biológicos ainda
- O processo de anotação torna esses tabelas dos genes ser mais compreensíveis

Note

Os nomes dos genes que temos agora são ids dos genes [Ensembl](#) :
e.g. ENSG00000285953.1

ANOTAÇÃO EM BIOCONDUCTOR

- Pacote `AnnotationDbi` – relaciona os nomes que já temos para nomes e atributos em uso comum
- Nosso tecido vem de seres humanos
- O pacote que lista os atributos dos genes humanos: `org.Hs.eg.db`
 - organismo/Homo Sapiens/Entrez Gene IDs/base de dados
- Mas, esses têm bastante informação

```
1 library("AnnotationDbi")
2 library("org.Hs.eg.db")
3 columns(org.Hs.eg.db)

[1] "ACCNUM"      "ALIAS"        "ENSEMBL"       "ENSEMLPROT"   "ENSEMLTRANS"
[6] "ENTREZID"     "ENZYME"       "EVIDENCE"      "EVIDENCEALL"  "GENENAME"
[11] "GENETYPE"     "GO"           "GOALL"         "IPI"          "MAP"
[16] "OMIM"         "ONTOLOGY"     "ONTOLOGYALL"  "PATH"         "PFAM"
[21] "PMID"         "PROSITE"      "REFSEQ"        "SYMBOL"       "UCSCKG"
[26] "UNIPROT"
```

CHAVE PARA CRIAR NOVAS COLUNAS DE DADOS:

mapIds ()

- mapIDs () entre na base de dados da espécie (`org.Hs.eg.db`) e deixa você tirar as informações que quiser
- Cada chamada para a função pode criar uma nova coluna em nossos resultados
- Porque os IDs de nossos genes são do tipo ENSEMBL, esse deve ser nosso `keytype = "ENSEMBL"`

CRIAR 2 NOVAS COLUNAS

1. [symbol](#) - ID que é mais comum invés de ENSEMBL
2. [entrez](#) - ID do sistema ENTREZ

ENTREZ

Sistema de NCBI dos NIH que alberga informações sobre os nucleotídeos, proteínas, genes, e mais

PROGRAMAÇÃO DE mapIDs()

- Limitar para primeiros 15 genes em nossos resultados
- Precisa todos os argumentos

```
1 # 1o 15 resultados
2 first_fifteen <- substr(rownames(res), 1, 15)
3 # symbol
4 res$symbol <- mapIds(org.Hs.eg.db,
5                         keys = first_fifteen,
6                         column = "SYMBOL",
7                         keytype = "ENSEMBL",
8                         multiVals = "first") #se tiver valores múltiplas
9 res$entrez <- mapIds(org.Hs.eg.db,
10                       keys = first_fifteen,
11                       column = "ENTREZID",
12                       keytype = "ENSEMBL",
13                       multiVals = "first")
```

VER OS NOVOS DADOS DOS RESULTADOS - EM str()

```
..@ listData      :List of 8
..  ..$ baseMean    : num [1:16637] 740.1 511.7 314.2 79.8 5715.3 ...
..  ..$ log2FoldChange: num [1:16637] -0.3653 0.2022 0.0338 -0.1206 0.443 ...
..  ..$ lfcSE       : num [1:16637] 0.1073 0.1279 0.1552 0.3055 0.0904 ...
..  ..$ stat        : num [1:16637] -3.404 1.582 0.218 -0.395 4.9 ...
..  ..$ pvalue      : num [1:16637] 0.00066528 0.1137195 0.82764977 0.69296416 0.00000096 ...
..  ..$ padj        : num [1:16637] 0.0046355 0.2881254 0.9217179 0.8487431 0.0000136 ...
..  ..$ symbol      : Named chr [1:16637] "TSPAN6" "DPM1" "SCYL3" "FIRRM" ...
..  ..- attr(*, "names")= chr [1:16637] "ENSG00000000003" "ENSG00000000419" "ENSG00000000457" ...
..  ..$ entrez      : Named chr [1:16637] "7105" "8813" "57147" "55732" ...
..  ..- attr(*, "names")= chr [1:16637] "ENSG00000000003" "ENSG00000000419" "ENSG00000000457" ...
```

NOVOS RESULTADOS

- Foco nos 6 mais importantes genes

```
1 resOrdered <- res[order(res$padj), c(1:2, 6:8)]  
2 head(resOrdered, 6)
```

log2 fold change (MLE): dex trt vs untrt

DataFrame with 6 rows and 5 columns

	baseMean	log2FoldChange	padj	symbol
	<numeric>	<numeric>	<numeric>	<character>
ENSG00000189221.9	2371.265	3.39426	3.95806e-134	MAOA
ENSG00000120129.5	3417.255	2.96990	2.75226e-129	DUSP1
ENSG00000101347.9	14106.720	3.74934	8.89591e-124	SAMHD1
ENSG00000152583.12	973.479	4.50022	9.04071e-112	SPARCL1
ENSG00000196136.17	2708.309	3.24329	3.94293e-109	SERPINA3
ENSG00000211445.11	12502.886	3.76804	6.73085e-108	GPX3

	entrez
	<character>
ENSG00000189221.9	4128
ENSG00000120129.5	1843
ENSG00000101347.9	25939
ENSG00000152583.12	8404
ENSG00000196136.17	12
ENSG00000211445.11	2878

SÓ UMA INTRODUÇÃO A RNASEQ E EXPRESSÃO GENÔMICA