Análise de dados de expressão genética – expressão diferencial

Conceitos e implementação em R/ Bioconductor

Exercícios com o dataset cachecia

Ex. 1 – distribuição normal ?

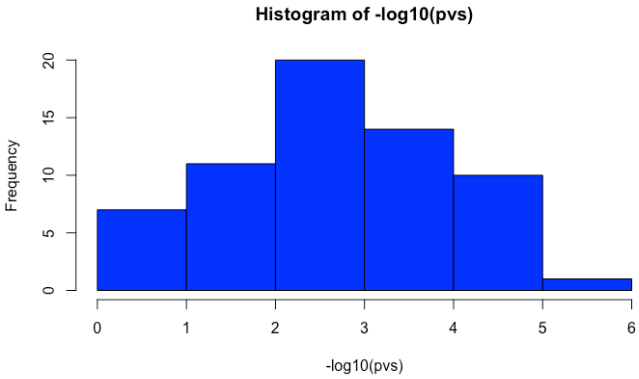
```
> pv.orig = sapply(cachexia[,1:63], function(x)
shapiro.test(x)$p.value)
> sum(pv.orig < 0.01)
[1] 63
> pv.log = sapply(cachexia.log[,1:63], function(x)
shapiro.test(x)$p.value)
> sum(pv.log < 0.01)
[1] 8
> colnames(cachexia.log)[which(pv.log < 0.01)]
```

Qual o efeito da transformação logarítmica ?

Ex. 2 – testes de concentração diferencial

Versão paramétrica

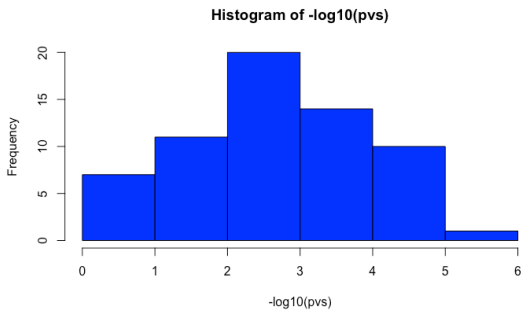
```
testCompound = function(ds, index) ## parametric
{
    g1 = ds[ds$Muscle.loss=="cachexic",index]
    g2 = ds[ds$Muscle.loss=="control",index]  restt = t.test(g1, g2))
    restt$p.value}
}

> pvs = c()
> for (i in 1:63) pvs[i] = testCompound(cachexia.log, i)
> range(pvs)
> hist(-log10(pvs), col = "blue")
[1] 2.563849e-06 3.999921e-01
> rank=order(pvs)[1:10]
> names(cachexia)[rank] # compounds
[1] "Glucose"               "Adipate"              "Quinolinate"
[4] "Leucine"               "Valine"               "myo.Inositol"
[7] "X3.Hydroxyisovalerate" "N.N.Dimethylglycine"  "X3.Hydroxybutyrate
[10] "Succinate"
```

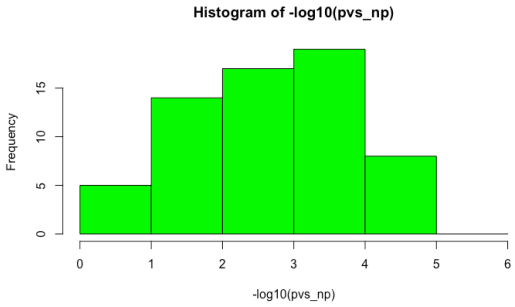


Ex. 2 – testes de concentração diferencial

```
testCompoundNP = function(ds, index) ##  non parametric
{
    g1 = ds[ds$Muscle.loss=="cachexic",index]
    g2 = ds[ds$Muscle.loss=="control",index]
    restt = wilcox.test(g1, g2))
    restt$p.value}
}

> pvs_np = c()
> for (i in 1:63) pvs_np[i] = testCompoundNP(cachexia.log, i)
> range(pvs_np)
> hist(-log10(pvs_np), breaks = 0:6, col = "green")

> rank=order(pvs_np)[1:10]
> names(cachexia)[rank] # compounds
[1] "Quinolinate"           "Glucose"                "Adipate"
[4] "N.N.Dimethylglycine"   "Valine"                 "Leucine"
[7] "X3.Hydroxyisovalerate" "Betaine"                "Creatine"
[10] "myo.Inositol"
```

Versão não paramétrica





Ex. 2 – testes de concentração diferencial

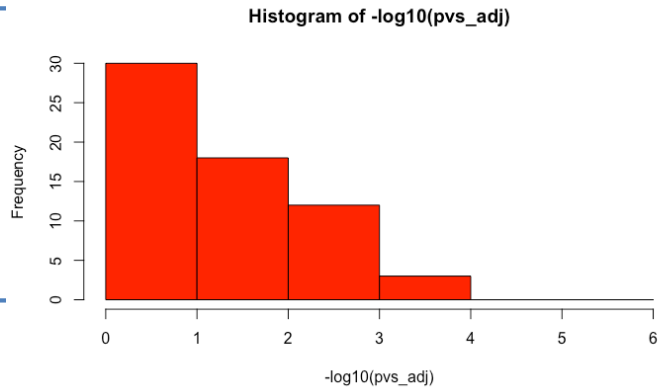
```
testCompoundCheckingNormality = function(ds, index)
{
    g1 = ds[ds$Muscle.loss=="cachexic",index]
    g2 = ds[ds$Muscle.loss=="control",index]
    pv1 = shapiro.test(g1)$p.value
    pv2 = shapiro.test(g2)$p.value
    if (pv1 > 0.01 && pv2 > 0.01)  restt = t.test(g1, g2)
    else restt= wilcox.test(g1, g2)
    restt$p.value
}
```

Versão testando a normalidade dos dados antes

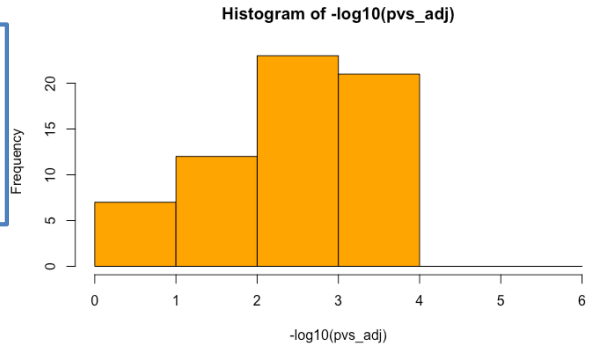
Ex. 2 – testes de concentração diferencial

Testes múltiplos: correções

```
> pvs_adj = p.adjust(pvs, "bonferroni")
> sum(pvs < 0.01)
[1] 45
> sum(pvs_adj < 0.01)
[1] 15
> hist(-log10(pvs_adj), breaks = 0:6, col = "red")
```

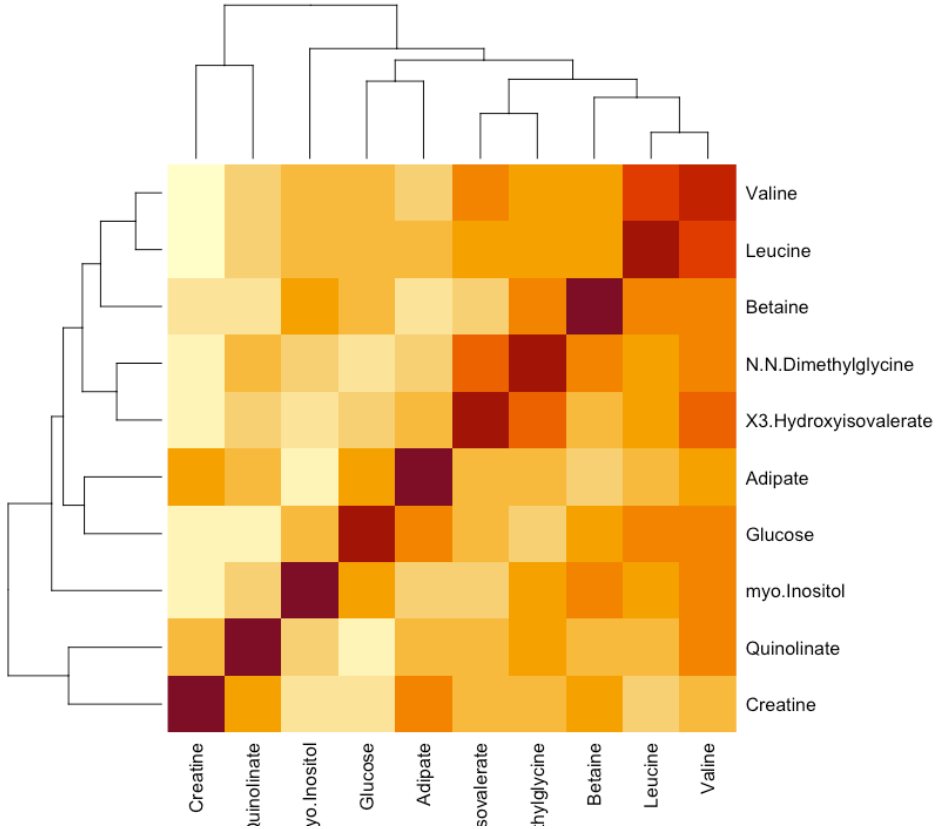



```
> pvs_adj = p.adjust(pvs, "BH")
> sum(pvs_adj < 0.01)
[1] 44
> hist(-log10(pvs_adj), breaks = 0:6, col = "orange")
```



Ex. 3 – regressão

```
> cor(cachexia[,rank])
> heatmap(cor(cachexia[,rank]))
```



Ex. 3 – regressão

```
> lm1 = lm(cachexia$Leucine ~ cachexia$Valine)
> summary(lm1)
Residuals:
    Min      1Q  Median      3Q     Max
 -32.217  -3.583  -0.938   2.810  47.936
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.61316    1.83630   0.878    0.382
cachexia$Valine  0.63786    0.03967  16.079   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 10.27 on 75 degrees of freedom
Multiple R-squared:  0.7751,  Adjusted R-squared:  0.7721
F-statistic: 258.5 on 1 and 75 DF,  p-value: < 2.2e-16
```

Ex. 3 – regressão

```
> lm2 = lm(cachexia$Leucine ~ cachexia$Valine + cachexia$X3.Hydroxybutyrate)
> summary(lm2)
Residuals:
    Min      1Q  Median      3Q     Max
-20.766  -3.929  -0.518   3.324  32.946
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.47446    1.51270   0.975    0.333
cachexia$Valine  0.45681    0.04432  10.306 5.95e-16 ***
cachexia$X3.Hydroxybutyrate  0.30374    0.05024   6.045 5.57e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 8.46 on 74 degrees of freedom
Multiple R-squared:  0.8495,  Adjusted R-squared:  0.8454
F-statistic: 208.8 on 2 and 74 DF,  p-value: < 2.2e-16
```

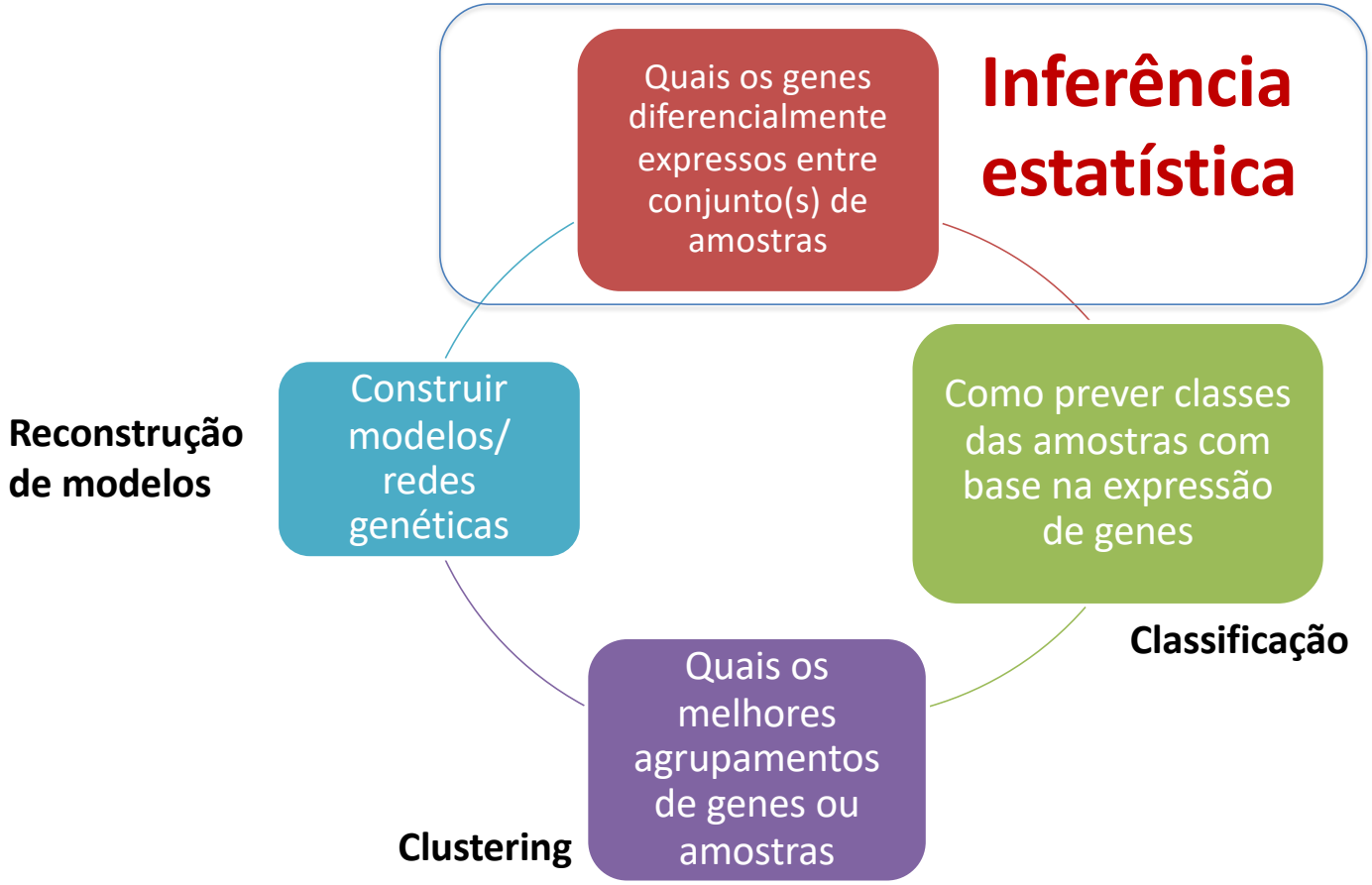
Ex. 3 – regressão

```
> lm3 = lm(cachexia$Leucine ~ cachexia$Valine * cachexia$X3.Hydroxybutyrate)
> summary(lm3)
Residuals:
    Min      1Q  Median      3Q     Max
-19.666  -3.428  -0.656   3.397  33.980
Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      3.255510   1.999935   1.628   0.1079
cachexia$Valine  0.421255   0.051331   8.207 5.76e-12 ***
cachexia$X3.Hydroxybutyrate  0.195836   0.094186   2.079   0.0411 *
Valine:X3.Hydroxybutyrate 0.001415   0.001047   1.351   0.1807
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 8.413 on 73 degrees of freedom
Multiple R-squared:  0.8531,  Adjusted R-squared:  0.8471
F-statistic: 141.4 on 3 and 73 DF,  p-value: < 2.2e-16
```

Ex. 3 – regressão

```
> lm4 = lm(cachexia$Leucine ~ cachexia$Valine * cachexia$Muscle.loss)
> summary(lm4)
Residuals:
Min      1Q      Median      3Q     Max
-32.629  -3.609  -0.822   2.833  47.469
Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                       2.34926    2.62173   0.896    0.373
cachexia$Valine                   0.63429    0.04701  13.494   <2e-16 ***
cachexia$Muscle.losscontrol      -0.07432    4.12604  -0.018    0.986
Valine:Muscle.losscontrol       -0.07392    0.13576  -0.544    0.588

standard error: 10.36 on 73 degrees of freedom
Multiple R-squared:  0.7774,  Adjusted R-squared:  0.7683
F-statistic: 84.98 on 3 and 73 DF,  p-value: < 2.2e-16
```

Questões básicas na análise de dados

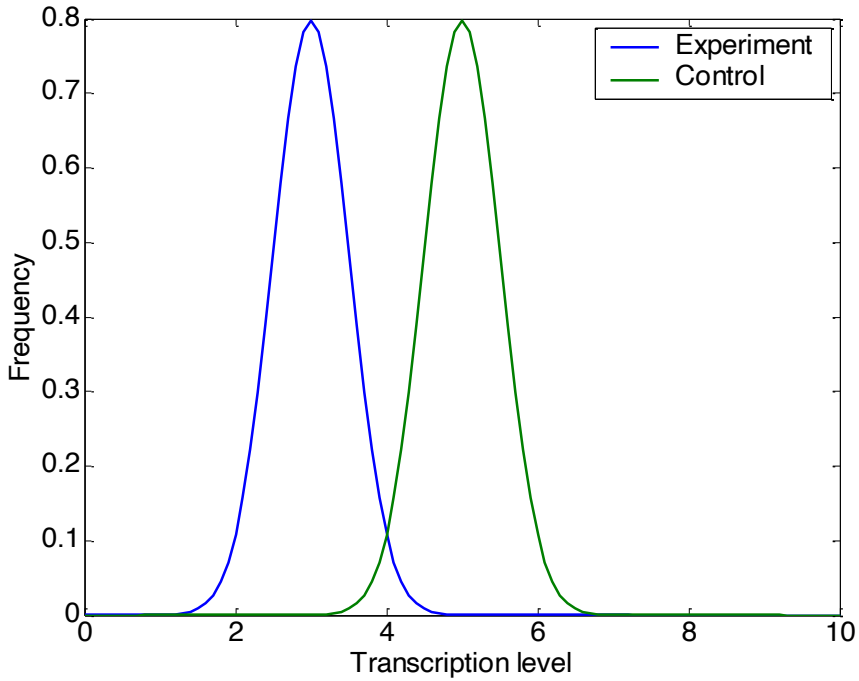


Expressão diferencial

Em termos de investigação biológica é muito importante **identificar o conjunto de genes que têm níveis diferentes de expressão** comparando duas (ou mais) condições experimentais (e.g. célula normal vs cancerígena, wild type vs mutante).

A identificação dos genes que são diferencialmente expressos assenta em **testes estatísticos de hipóteses**, baseados na presunção de que os valores de expressão genética são “gerados” seguindo uma distribuição normal.

Expressão diferencial



Hipótese nula: As médias dos níveis de transcrição das duas situações são idênticas?

Testes estatísticos apropriados

- **Paired t-test**
  - se as amostras são emparelhadas
    (e.g. duas condições por paciente, um com o tratamento A e outro com o tratamento B)
- **Unpaired t-test**
  - as amostras dos dois grupos não são relacionadas

Em ambos os casos, calcula-se a t-statistic e o correspondente p-value;

- valores de p-value mais baixos significam maior confiança que as médias são diferentes

Podem usar-se testes não paramétricos (e.g. Mann-Whitney) se as distribuições dos dados não forem aproximadamente normais

Expressão diferencial - exemplo

Gene	Control 1	Control 2	Control 3	Exp 1	Exp 2	Exp 3	P-value
1	8.41	8.45	8.37	9.29	9.39	9.20	0.003
2	8.82	8.93	9.20	9.84	9.79	9.56	0.004
3	11.70	11.70	11.66	12.03	11.99	12.05	0.005
...	...	...	...	...	...	...	...
N-2	6.05	6.05	6.05	6.05	6.30	5.73	0.423
N-1	8.32	8.26	8.23	8.77	8.08	7.73	0.733
N	10.93	11.14	11.10	10.71	11.59	10.99	0.827

Exemplo de dados de microarrays: ALL (leukemia)

```
> library(ALL)
> data(ALL)
> ALL
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12625 features, 128 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 01005 01010 ... LAL4 (128 total)
  varLabels: cod diagnosis ... date last seen (21 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 14684422 16243790
Annotation: hgu95av2
> dim(ALL)
Features  Samples
   12625      128
```

Carregar o conjunto de dados e verificar o seu conteúdo e dimensão

Filtros flat patterns - exemplos

```
> maximos = apply(exp,1,max)
> minimos = apply(exp,1,min)
> vl = maximos/minimos > 2
> ALLm2=ALL[vl,]
> ALLm2
ExpressionSet (storageMode: lockedEnvironment)
assayData: 1023 features, 128 samples
…
```

Filtra genes cujo rácio do máximo valor sobre o mínimo valor de expressão seja superior a 2

Expressão diferencial – exemplo

```
> s = which(as.character(ALL$mol.biol) %in% c("BCR/ABL", "NEG"))
> ALLs = ALLm2[, s]
> ALLs
assayData: 1023 features, 111 samples
…
> ALLs$mol.biol = factor(ALLs$mol.biol)
> table(ALLs$mol.biol)
BCR/ABL     NEG
     37      74
> library(genefilter)
> tt = rowttests(ALLs, "mol.biol")
> names(tt)
[1] "statistic" "dm"        "p.value"
> tt$p.value
[1] 1.731621e-02 9.113902e-01 4.774039e-01…
> rank = order(tt$p.value)
> p20 = rank[1:20]
> tt$p.value[p20]
 [1] 7.353928e-16 1.696610e-13 …
```

Filtrar amostras de forma a ter apenas dois valores no campo mol.biol

Realizar os t-tests e verificar os p-values

20 genes com menor p-values (maior evidência de expressão diferencial

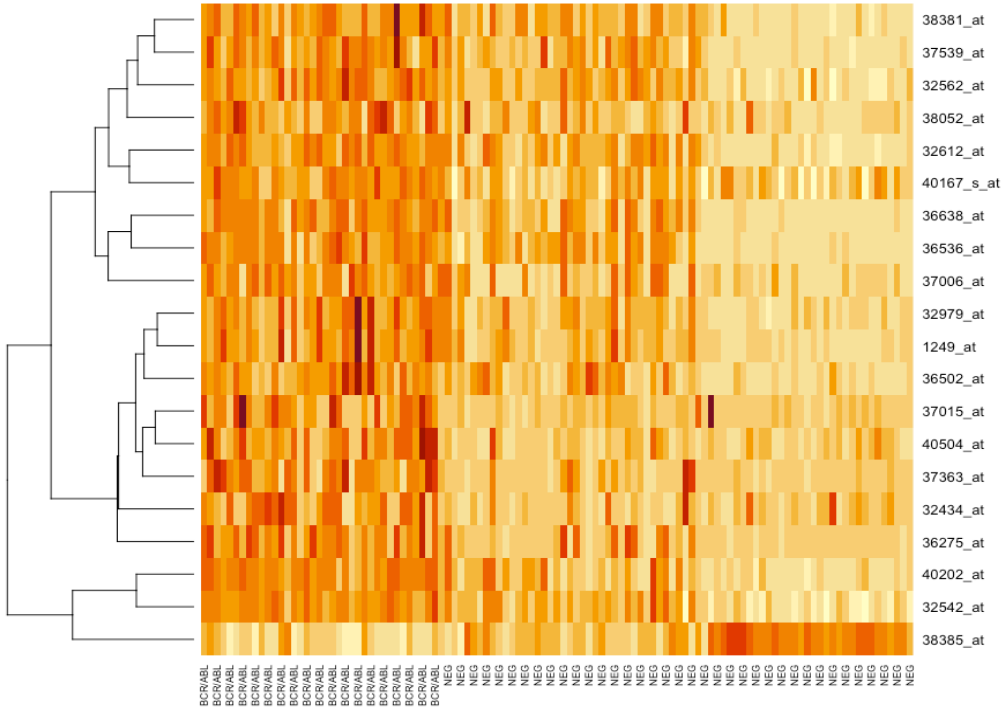
Expressão diferencial – exemplo

```
> g = featureNames(ALLm2[p20])

> g
[1] "40202_at"   "40504_at"   "32979_at"  "37363_at" ….



> annotation(ALL)
[1] "hgu95av2"

> BiocManager::install(c("hgu95av2.db"))
> library(hgu95av2.db)



> unlist(mget(g, hgu95av2SYMBOL))
  40202_at   40504_at   32979_at   37363_at   32542_at ...
  "KLF9"     "PON2"     "GAB1"    "MTSS1"     "FHL1“ ...
```

Usando a anotação recomendada, podem recuperar-se os nomes dos genes que correspondem às probes identificadas

Expressão diferencial – heatmap

```
> ALL20 = ALLs[p20,]
> order_cols = order(ALL20$mol.biol)
> order_cols
> ALL20 = ALL20[,order_cols]
> heatmap(exprs(ALL20), Colv = NA, labCol = ALL20$mol.biol)
```



Bootstrap

Ideia: para cada gene:

- criar muitos conjuntos de valores de expressão retirados aleatoriamente de cada um dos grupos
- em cada conjunto calcular p-value e criar a distribuição dos p-values
- verificar onde se situa o p-value dos valores reais nessa distribuição

Algoritmo com esta abordagem mais conhecido: **SAM - Significance Analysis of Microarrays**

Testes múltiplos

Problema: ao testar R genes em simultâneo, mesmo com uma probabilidade baixa de erro (p) o nº esperado de erros (falsos positivos) é R.p, que pode ser alto pois R pode ser de vários milhares.

Algumas correções têm sido sugeridas (e.g. Bonferroni, Holm) mas todas elas levam a critérios que levam a nº muito alto de falsos negativos.

Solução típica: fixar valor do p-value para garantir uma taxa de falsos positivos aceitável

Análises estatísticas mais elaboradas

One way **ANOVA** (analysis of variance)

- quando temos 3 ou + grupos de condições (amostras)
- Podem usar-se versões não paramétricas

Multifactor ANOVA

- quando temos 2 ou + factores (variáveis)

General linear models – regressão linear

- podem incorporar análise da influência de variáveis contínuas – package ***limma***

Exemplo – package limma

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

> design = model.matrix(~ALLm2$mol.biol)
> fit = lmFit(ALLm2,design)
> fit2 = eBayes(fit)
> diff = topTable(fit2, coef=2, 10)
> diff
             logFC     AveExpr          t      P.Value    adj.P.Val        B
40763_at -3.086992   3.193967   -13.399138 4.969722e-26 5.084026e-23 48.57683
36873_at -3.391662   4.111355   -12.699331 2.489921e-24 1.273595e-21 44.73125
41448_at -2.500488   3.636095   -10.493253 6.495333e-19 2.214909e-16 32.46794…
> unlist(mget(rownames(diff), hgu95av2SYMBOL))
40763_at 36873_at 41448_at  1914_at 37809_at ...
"MEIS1"  "VLDLR"  "HOXA10"  "CCNA1" NA       ...
```

Logaritmo do fold change: rácio de variação entre duas condições

P-value ajustado para testes múltiplos

lmFit – cria os modelos de regressão linear

eBayes – realiza os testes estatísticos e faz a correção de testes múltiplos

Enrichment Analysis

Análise realizada sobre um conjunto de genes alvo, identificados por exemplo por uma análise de expressão diferencial

Conjunto de genes identificados é comparado com outros **conjuntos de genes,** onde cada um destes contém genes biologicamente coerentes (e.g. que partilham funções biológicas semelhantes)

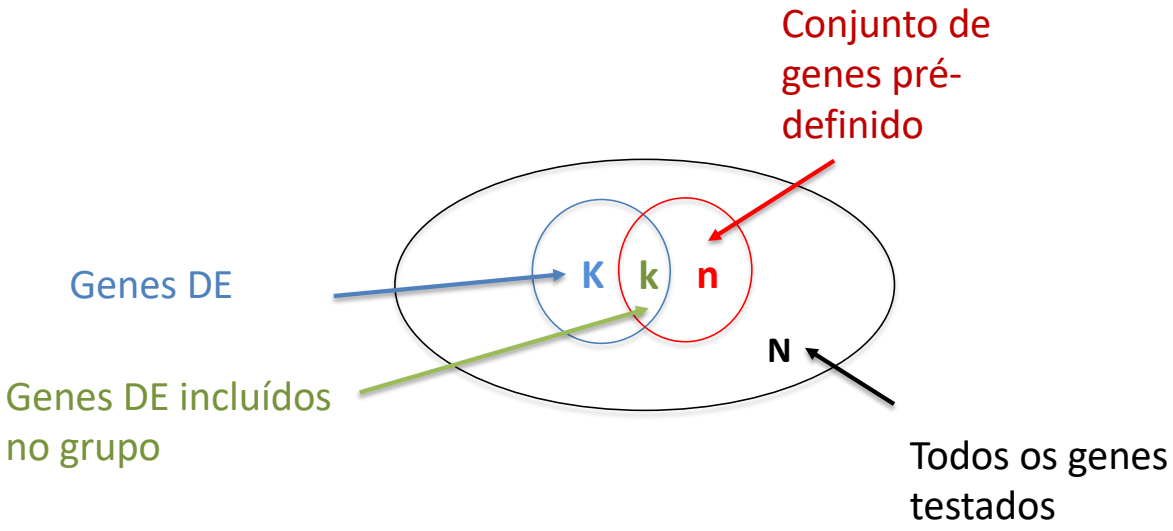
Objectivo: verificar se no nosso conjunto de genes existe "enriquecimento" estatisticamente significativo nos genes de algum (ou vários) destes conjuntos (testes **hipergeométricos**)

Pode ser realizado com métodos mais elaborados - Gene Set Enrichment Analysis (GSEA)

Enrichment Analysis

Teste hipergeométrico realizado com os valores de N, K, n, k

Retorna ***p-value***: probabilidade de termos pelo menos ***k*** genes DE num conjunto de tamanho ***n*** definido de forma **aleatória**, sabendo que há no total ***K*** genes DE num universo de ***N*** genes



Exemplo- enrichment analysis

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

> ALLenr = ALL[, s]
> filt = nsFilter(ALLenr, require.entrez=T, remove.dupEntrez=T,
var.func=IQR, var.cutoff=0.5, feature.exclude="^AFFX")
> ALLf = filt$eset


> affyUniverse = featureNames(ALLf)
> entrezUniverse = unlist(mget(affyUniverse, hgu95av2ENTREZID))
> length(entrezUniverse)

> ttests = rowttests(ALLf, "mol.biol")
> smPV = ttests$p.value < 0.01
> sum(smPV)
> pvalFiltered = ALLf[smPV, ]

> selectedEntrezIds = unlist(mget(featureNames(pvalFiltered),
hgu95av2ENTREZID))
```

Instalação/ carregamento do package

Filtragem de amostras: apenas 2 grupos

Filtragem de genes que não têm anotação no EntrezGene e com pouca variabilidade

Recolhe IDs de todos os genes para a anotação dos dados

Testes t para determinar genes diferencialmente expressos e seus IDs

IDs dos genes selecionados

Exemplo - enrichment analysis

```
> params = new("GOHyperGParams", geneIds=selectedEntrezIds,
universeGeneIds=entrezUniverse, annotation="hgu95av2.db",
ontology="MF", pvalueCutoff= 0.025, testDirection="over")

> hgOver = hyperGTest(params)
> hgOver
Gene to GO MF  test for over-representation
1382 GO MF ids tested (56 have p < 0.025)
Selected gene set size: 713       K
    Gene universe size: 4119      N
    Annotation package: hgu95av2
> summary(hgOver)                                        k    n
       GOMFID       Pvalue OddsRatio   ExpCount Count Size
1  GO:0004888 1.451103e-06  2.442847  26.3112406    50  152
2  GO:0038023 4.700027e-06  2.089302  37.0434571    63  214
3  GO:0060089 1.334826e-05  1.946472  42.0633649    68  243 ...
```

Criar parâmetros para os testes estatísticos hipergeométricos: grupos "target" do Gene ontology, genes a considerar: sobreexpressos

Correr a análise

Resultados da análise

Lista de grupos com menores p-values (ordem crescente; dá contagens de genes no grupo alvo e total de genes no grupo)

Análise de dados de RNAseq

Package DESeq2

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

```
if (!requireNamespace("BiocManager", quietly = TRUE) )
    install.packages("BiocManager")

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

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

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
```

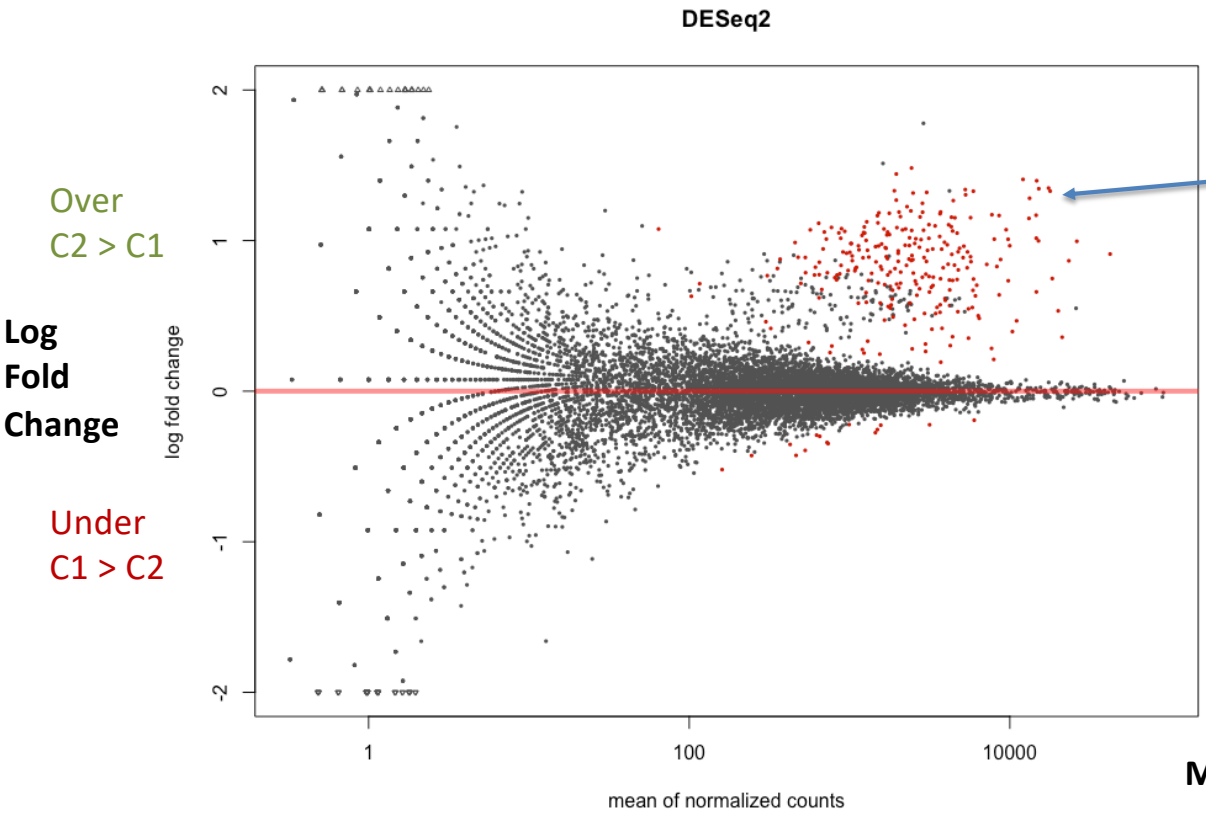
Correr testes de expressão diferencial
Ficam guardados como campo ***results*** do objeto original
Contém p-values e fold changes

```
> 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

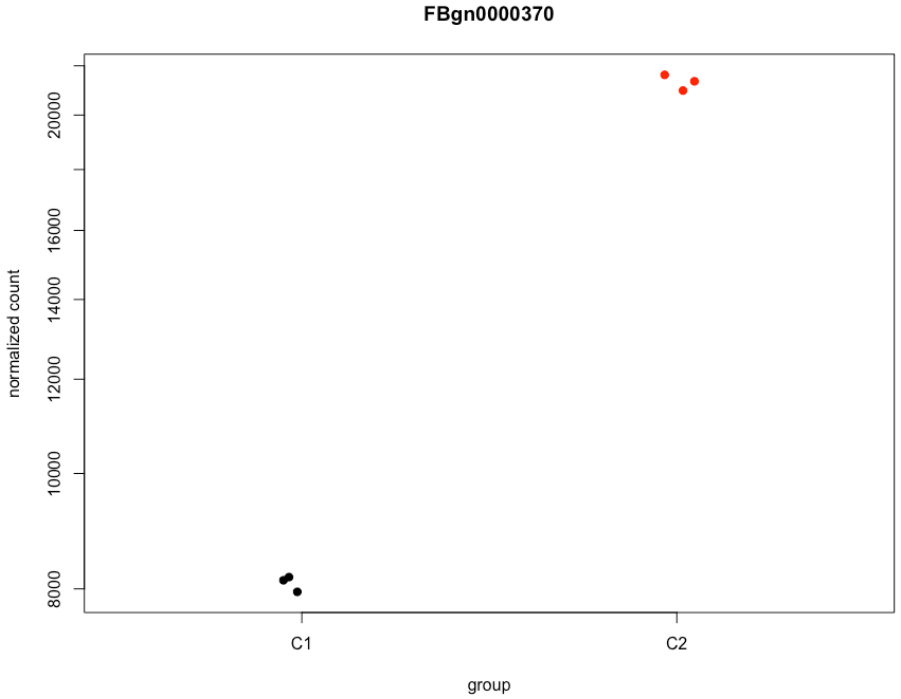
Under
C1 > C2

Pontos a vermelho – genes DE
Pontos a preto: outros genes

Média de expressão

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)
```

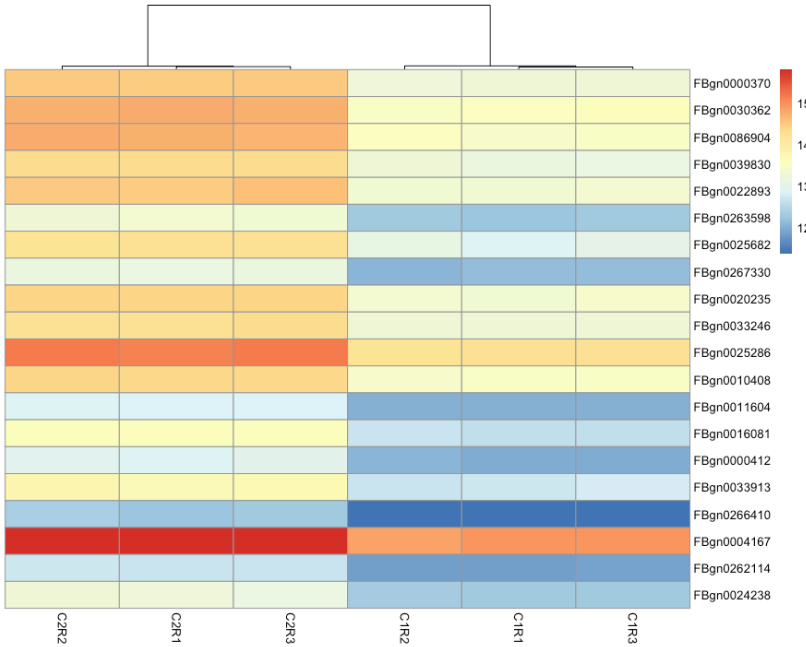
Faz a transformação usando informação das condições

Heatmap dos genes

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

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

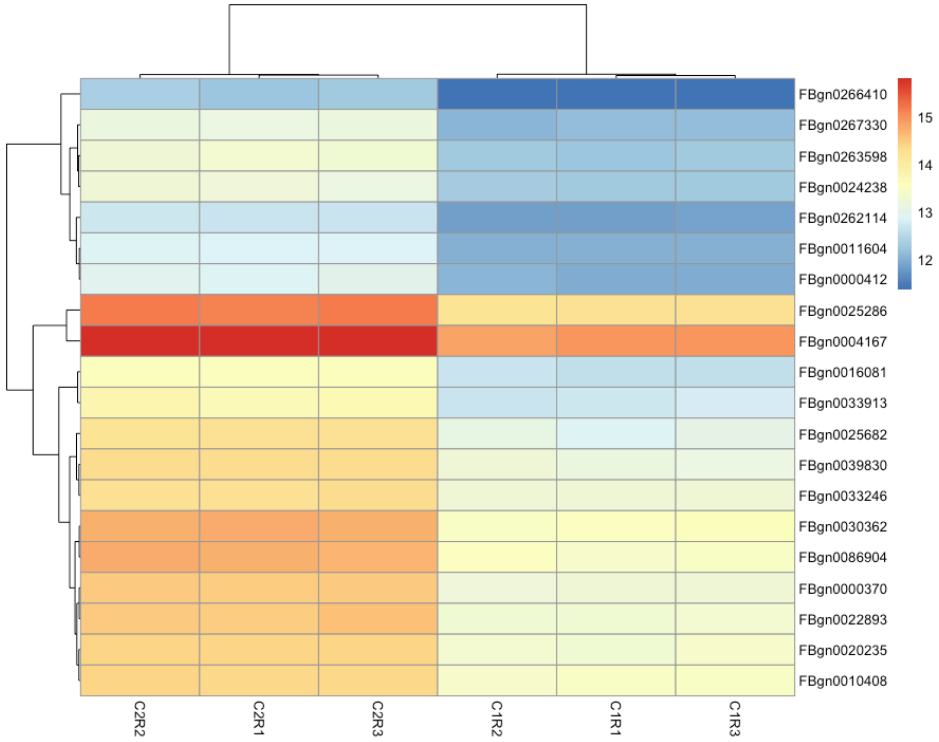
```
> pheatmap(vsd.counts, cluster_rows=FALSE)
```



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 Rsubread e edgeR

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

Package Rsubread

- Uma alternativa para efetuar o processimento inicial dos dados em bruto, que usa só packages R é usar o package ***Rsubread*** do Bioconductor
- Este permite realizar as operações de indexação, alinhamento contra genoma de referência e contagens dos números de reads por gene

```
> if (!requireNamespace("BiocManager", quietly = TRUE) )
    install.packages("BiocManager")

> BiocManager::install(c("Rsubread"))

> library(Rsubread)
```

Conjunto de dados usado

GEO: http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE60450

Transcriptome analysis of luminal and basal cell subpopulations in the lactating versus pregnant mammary gland – Mus musculus

Mouse mammary data:

- FASTQ files (parte 1): https://figshare.com/s/f5d63d8c265a05618137
  (ZIP file (Download all) – unzip para folder data)
- Genoma referência (chr 1) (parte 1): http://hgdownload.soe.ucsc.edu/goldenPath/mm10/chromosomes/ (ficheiro chr1.fa.gz + extrair o gzip)
- Gene counts (parte 2): https://figshare.com/s/1d788fd384d33e913a2a
  (ZIP file (Download all) – extrair ZIP para folder count)

Fu NY, Rios AC, Pal B, Soetanto R et al. EGF-mediated induction of Mcl-1 at the switch to lactation is essential for alveolar cell survival. *Nat Cell Biol* 2015 Apr;17(4):365-75. PMID: 25730472

Alinhamento

Pasta onde se guardam os FASTQ files e onde se geram os BAM

```
# lista de ficheiros FASTQ
> fastq.files = list.files(path = "./Mouse-rnaseq-data/data",
pattern = ".fastq.gz$", full.names = TRUE))
# Construir os indices
> buildindex(basename="./Mouse-rnaseq-data/chr1_mm10",
reference="./Mouse-rnaseq-data/chr1.fa")
# alinhamento
> align(index="./Mouse-rnaseq-data/chr1_mm10",readfile1=fastq.files)
# lista de ficheiros BAM gerados
> bam.files = list.files(path = "./Mouse-rnaseq-data/data", pattern = ".BAM$",
full.names = TRUE)
# proporcao de reads mapeados
> props <- propmapped(files=bam.files)
```

Ficheiros de índices

Nota: foi criada uma pasta base “Mouse-rnaseq-data” que se assume que fica dentro da pasta que é a current “working directory” do R. Aí se guardam todos os ficheiros. Os ficheiros FASTQ ficam numa sub-pasta ”data” dentro dessa pasta base

Controlo de qualidade e contagens

```
# control de qualidade
> qs <- qualityScores(filename="./Mouse-rnaseq-data/data/SRR1552450.fastq.gz",
                      nreads=100)
> dim(qs)
> head(qs)
> boxplot(qs)

# Contagens
> fc <- featureCounts(bam.files, annot.inbuilt="mm10")
> names(fc)
> fc$stat
> dim(fc$counts)
> head(fc$counts)
> head(fc$annotation)
```

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

- Não vamos usar os dados anteriores, pois apenas se referem a parte do genoma
- Vamos carregar os dados completos de contagens para todos os genes (para a sub-pasta counts)

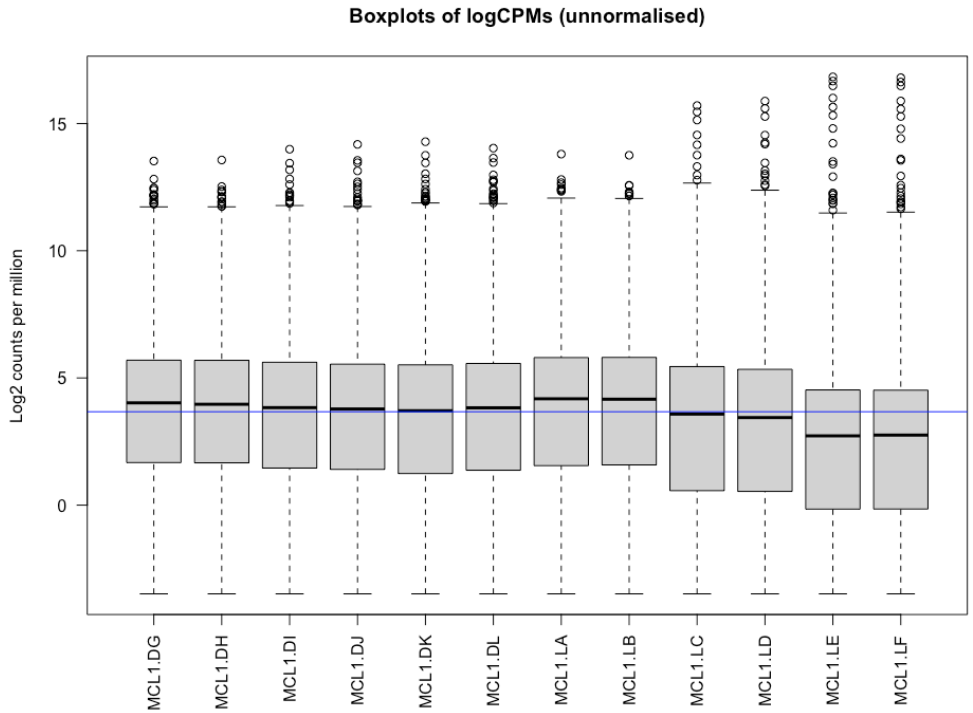
```
# dados sobre as amostras
> sampleinfo <- read.delim("./Mouse-rnaseq-data/counts/SampleInfo_corrected.txt",
                  stringsAsFactors = TRUE))
> sampleinfo

# dados sobre as contagens
> seqdata <- read.delim("./Mouse-rnaseq-data/GSE60450_LactationGenewiseCounts.txt",
                          stringsAsFactors = FALSE)
> head(seqdata)
> View(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
> View(countdata)
```

Pré-processamento dos dados de contagens

```
# contagens por milhao
> 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
```

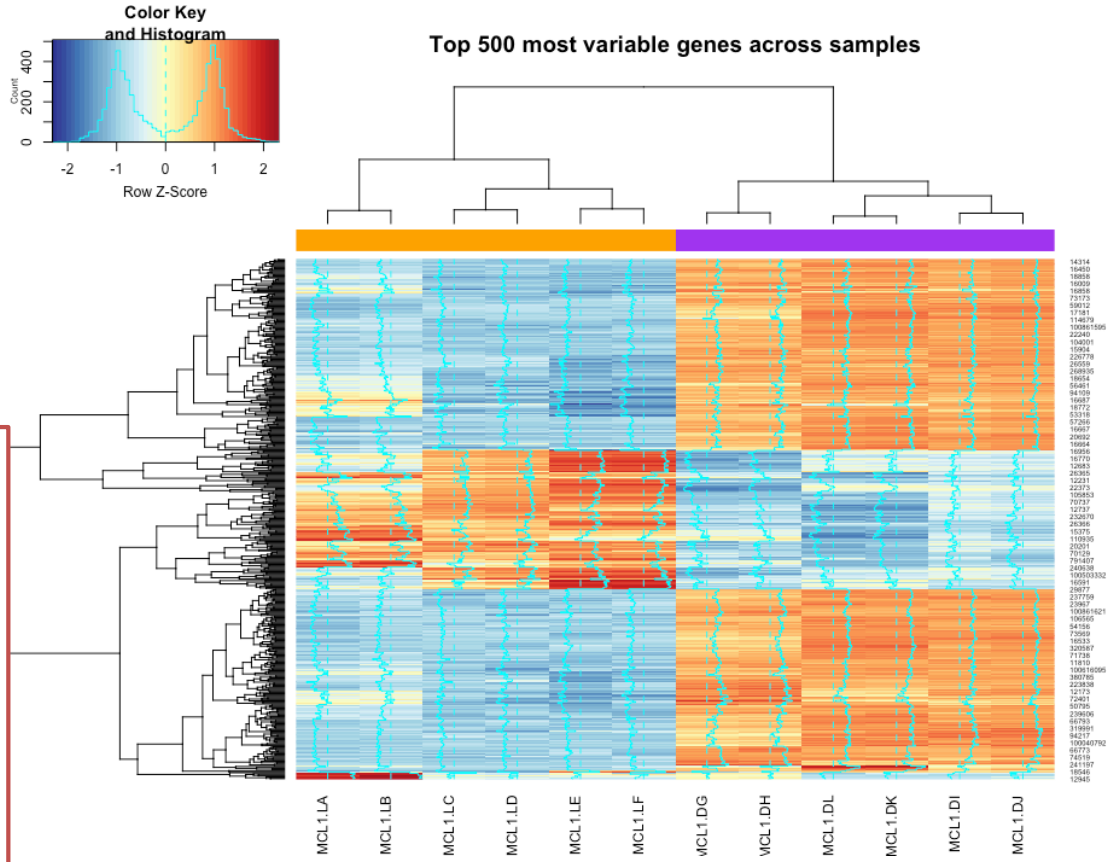
Transformação log e boxplots

```
> 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

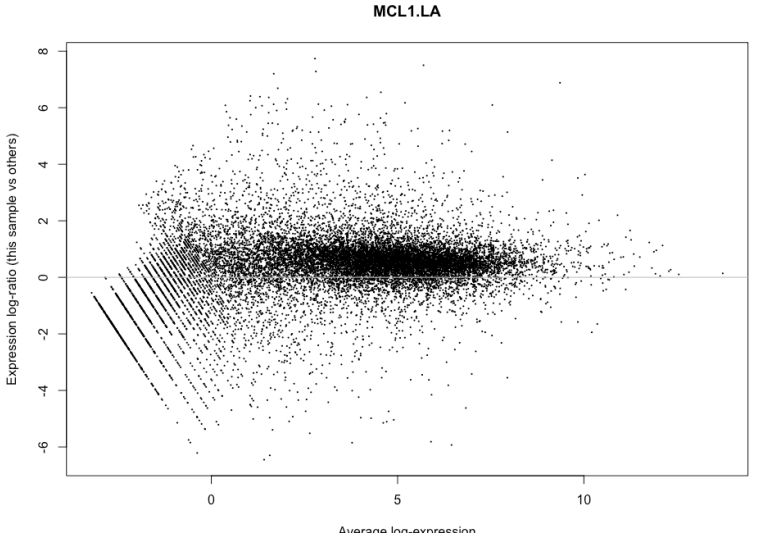
Selecionar os 500 genes com mais variabilidade

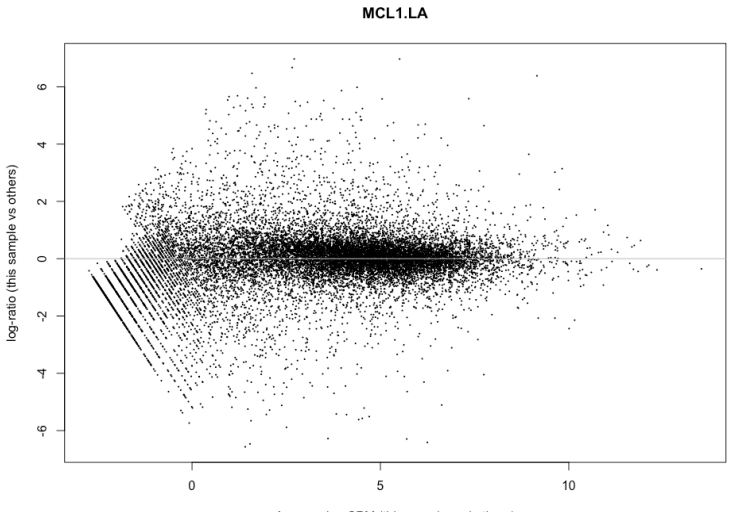



```
> var_genes <- apply(logcounts, 1, var)
> select_var <- names(sort(var_genes,
decreasing=TRUE))[1:500]
> highly_variable_lcpm <- logcounts[select_var,]
> mypalette <- brewer.pal(11,"RdYlBu")
> morecols <- colorRampPalette(mypalette)
> col.cell <-
c("purple","orange")[sampleinfo$CellType]
> heatmap.2(highly_variable_lcpm,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

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

Factores:
- Type
- Status

Estimar dispersão

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

Modelos de regressão

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

Testes

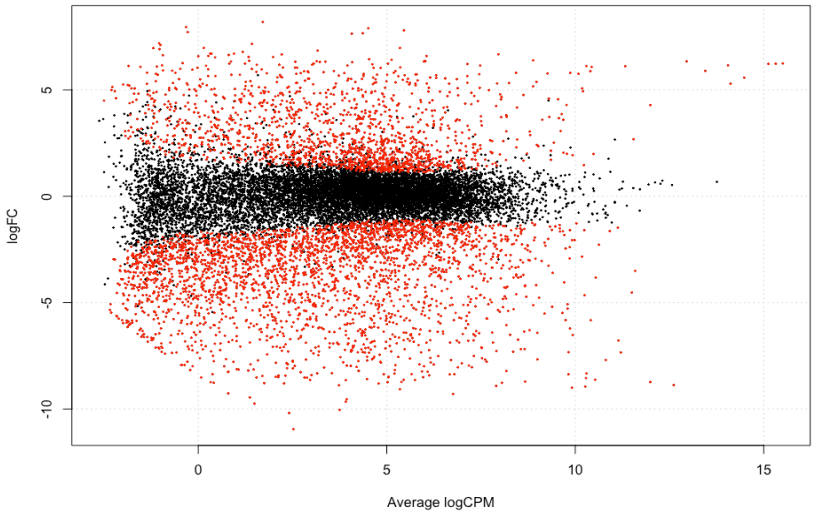
```
> lrt.BvsL <- glmLRT(fit, coef=2)
> topTags(lrt.BvsL)
```

Genes DE: estatísticas

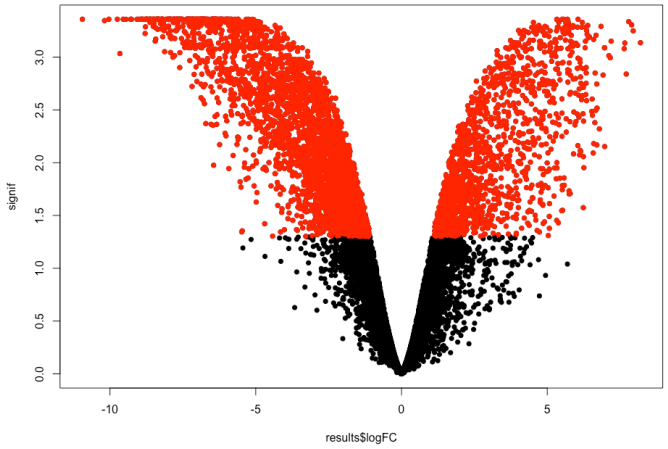
```
> results <- as.data.frame(topTags(lrt.BvsL,n = Inf))
> results
> dim(results)
> summary(de <- decideTestsDGE(lrt.BvsL))
```

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("./Mouse-rnaseq-data/counts/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