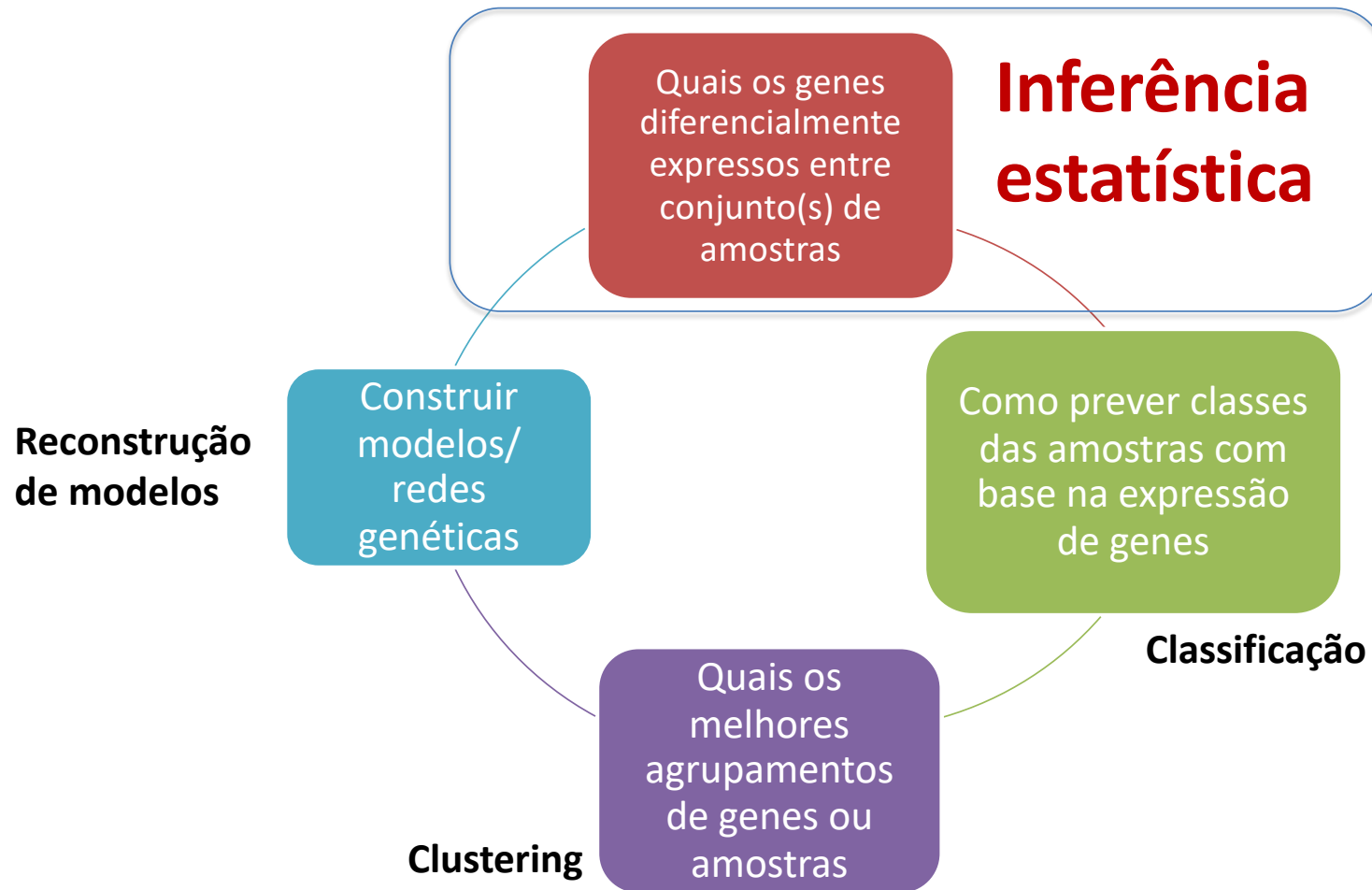


# **Análise de dados de expressão genética – expressão diferencial**

Conceitos e implementação em R/ Bioconductor

# Questões básicas na análise de dados

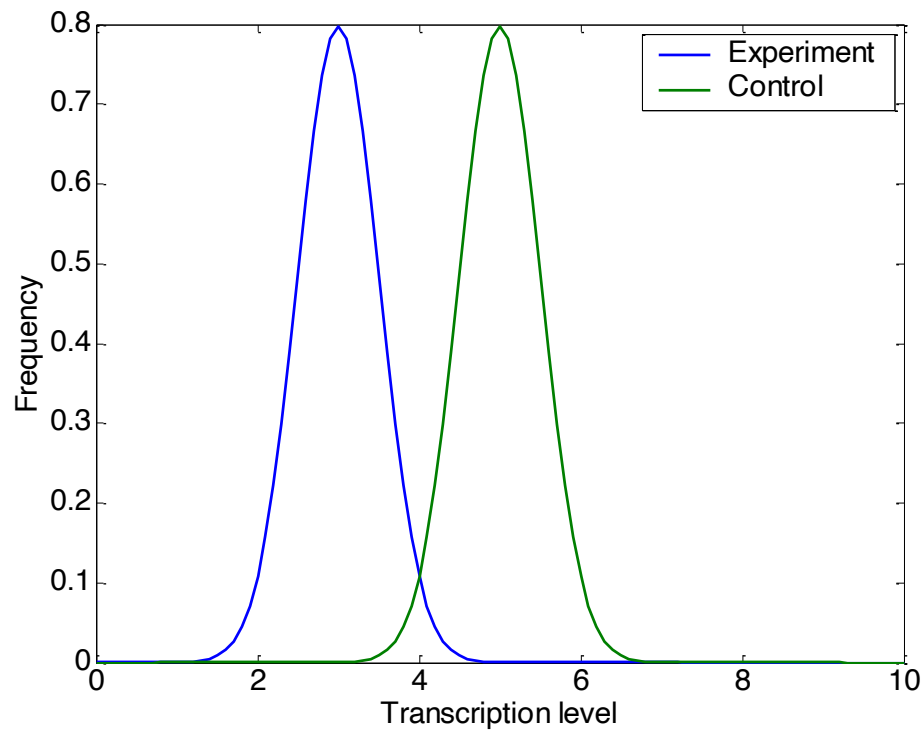


# Expressão diferencial

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)

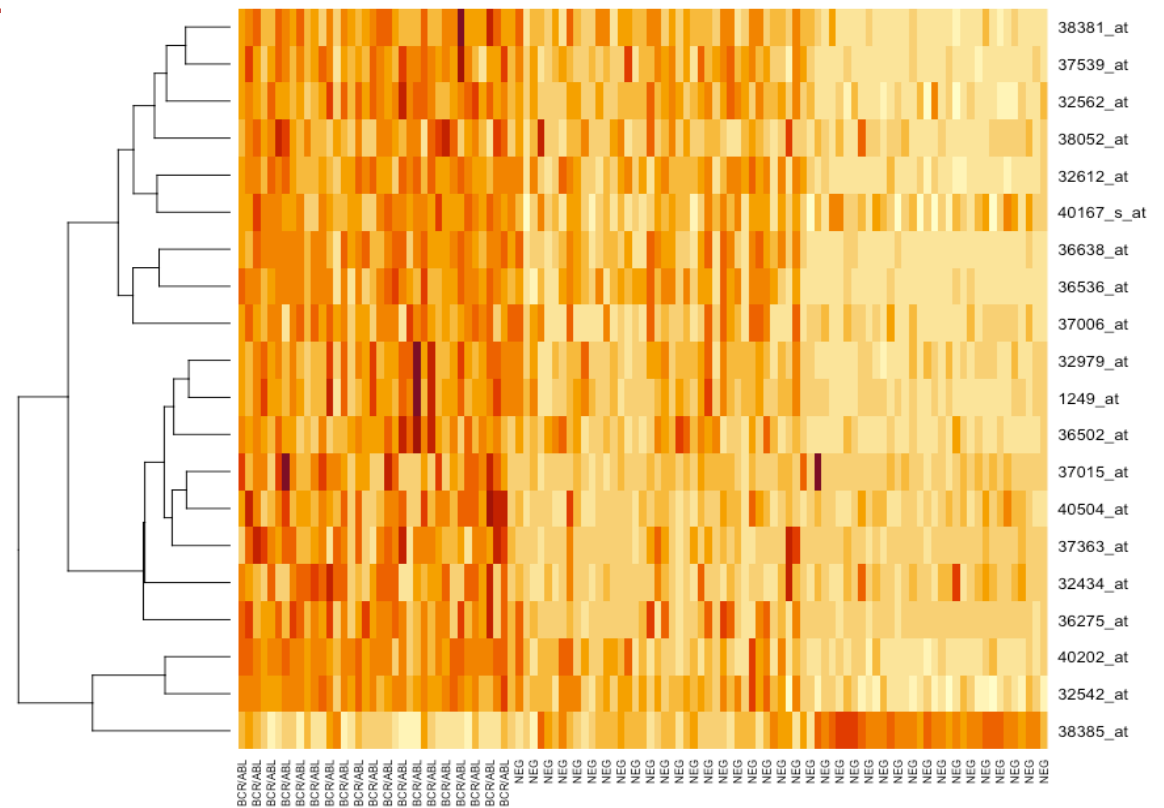
# Expressão diferencial – exemplo

```
> 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  $n^\circ$  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  $n^\circ$  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*

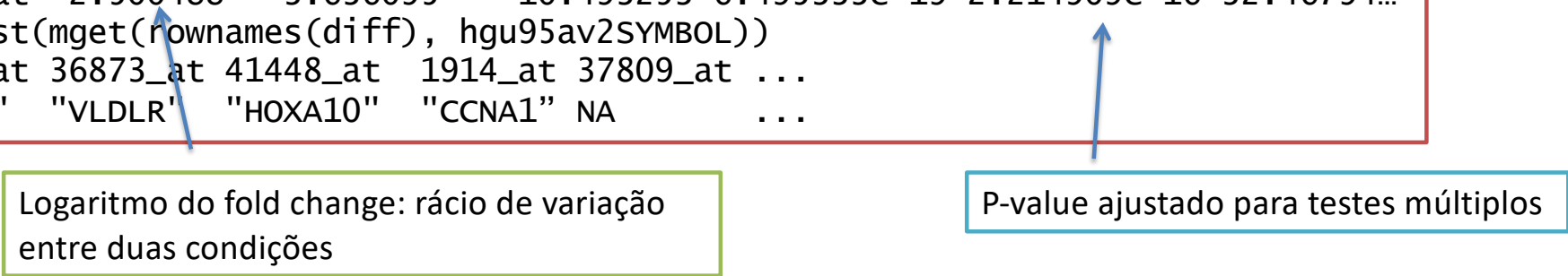
# Exemplo – package limma

```
> 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**)

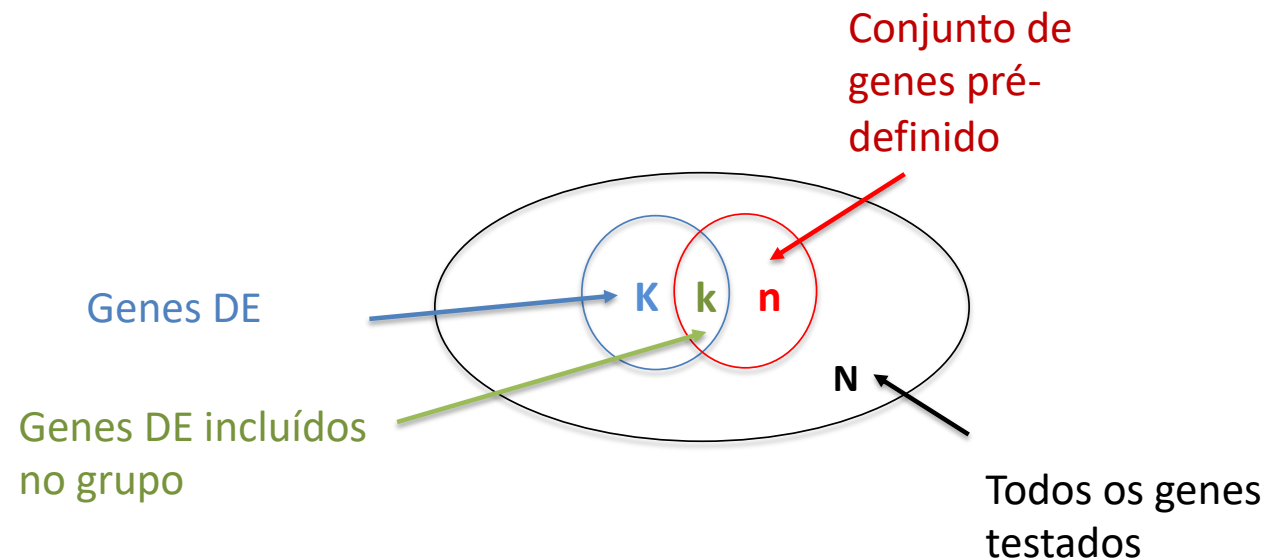
Pode ser realizado com métodos mais elaborados - Gene Set Enrichment Analysis (GSEA)



# 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)
```

	GOMFID	Pvalue	OddsRatio	ExpCount	<span style="color: green;">k</span> Count	<span style="color: red;">n</span> 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)