ANÁLISE DOS DADOS COM R

Intro – Bioconductor

James R. Hunter, PhD

Retrovirologia, EPM, UNIFESP

BIOCONDUCTOR

INSTALAÇÃO DE BIOCONDUCTOR

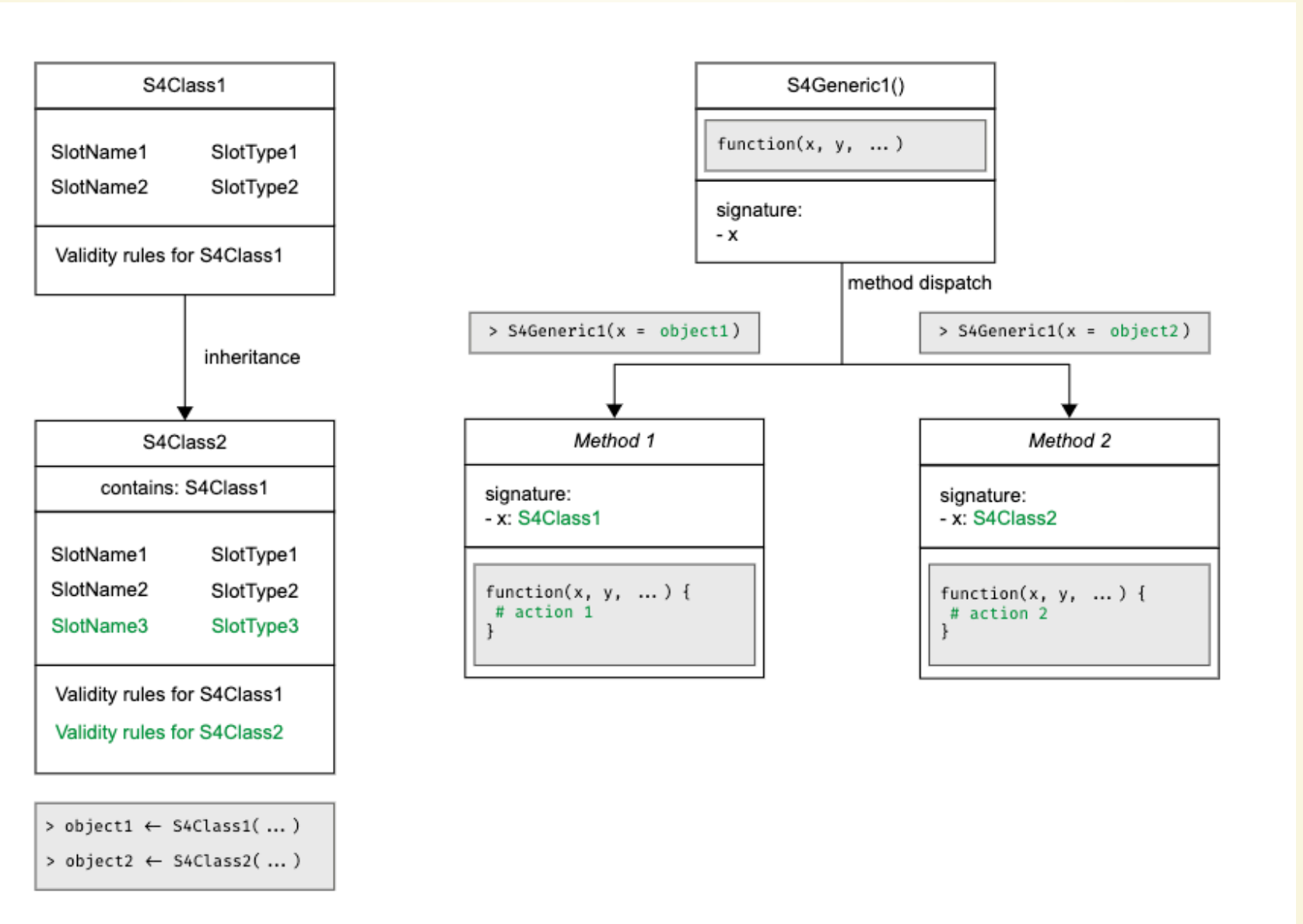
```r
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install(c("Biostrings", "Biobase", "S4Vectors", "rtracklayer",
```

- Amostra dos quase 2.300 pacotes disponíveis

FERRAMENTA `Biostrings`

- Pacote com as funções básicas para a manipulação das sequências
- Sequências são aqui tratadas como objetos de S4 classes:
  - `DNAString`, `RNAString`, `AAString`
  - Estruturas eficientes de memoria
  - Fazem verificações automáticas da validade dos *strings*
  - Utilizem métodos para utilidades da manipulação dos `strings`

S4 - REVISÃO


S4Class1
SlotName1 SlotType1
SlotName2 SlotType2
Validity rules for S4Class1
inheritance
S4Class2
contains: S4Class1
SlotName1 SlotType1
SlotName2 SlotType2
SlotName3 SlotType3
Validity rules for S4Class1
Validity rules for S4Class2
> object1 ← S4Class1(...)
> object2 ← S4Class2(...)
S4Generic1()
function(x, y, ...)
signature:
- x
method dispatch
> S4Generic1(x = object1)
> S4Generic1(x = object2)
Method 1
signature:
- x: S4Class1
function(x, y, ...) {
# action 1
}
Method 2
signature:
- x: S4Class2
function(x, y, ...) {
# action 2
}

CONSTRUIR UMA SEQUÊNCIA DE DNA

- Em S4, a criação de um objeto está chamada **construção** e o método (`DNAString()`) é um **construtor**

```
1  seq_dna1 <- DNAString("ACTTTGGCTAAG")
2  seq_dna1
```

```
12-letter DNAString object
seq: ACTTTGGCTAAG
```

- Até agora, fácil de entender

DENTRO DA MEMORIA

- Agora organizada como uma classe com 5 “slots” denotados por “@”

```
1 str(seq_dna1)
```

```
Formal class 'DNAString' [package "Biostrings"] with 5 slots
  ..@ shared         :Formal class 'SharedRaw' [package "XVector"] with 2
slots
  .. .. ..@ xp                    :<externalptr>
  .. .. ..@ .link_to_cached_object:<environment: 0x1448f9b50>
  ..@ offset         : int 0
  ..@ length         : int 12
  ..@ elementMetadata: NULL
  ..@ metadata       : list()
```

VERIFICAÇÃO AUTOMÁTICA

- Se a sequência realmente é uma sequência de DNA, simplesmente a função aceita e você vê no `Environment` de Rstudio
- Mesmo com caracteres ambíguos, vai ser a mesma coisa
- Diferente com caracteres que não são do alfabeto IUPAC

```r
# string ok
seq_dna1 <- DNAString("ACTTTGGCTAAG")

# string com ambiguous
seq_dna2 <- DNAString("TGATTGCTTGGTTGMTT")
```

```
> seq_dna3 <- DNAString("ACT55GGCTRRG")
Error in .Call2("new_XString_from_CHARACTER", class(x0), string, start,  :
  key 53 (char '5') not in lookup table
```

STRINGSETS

- Pode criar grupos (sets) de sequências em um objeto, um `DNAStringSet`
- Esse comando é um contstrutor
- Elementos devem ser *strings*
- Muito outras maneiras de construir stringsets, mas avançado

```
1  seqs_set1 <- c("ACTTTGGCTAAG", "TGATTGCTTGGTTGMTT")
```

MESMO PROCESSO PARA RNA E AA'S

FERRAMENTAS PARA XSTRINGS

CORRESPONDÊNCIA DE PADRÕES (*PATTERN MATCHING*)

- Mostra as opções mas não faz a desambiguação

```
IUPAC_CODE_MAP[5:15]
```

```
   M      R      W      S      Y      K      V      H      D      B      N
"AC"   "AG"   "AT"   "CG"   "CT"   "GT"  "ACG"  "ACT"  "AGT"  "CGT" "ACGT"
```

```
seq_dna2
```

```
17-letter DNAString object
seq: TGATTGCTTGGTTGMTT
```

```
# se fixed = FALSE, codigos ambiguous podem ser extensos
matchPattern("MT", seq_dna2, fixed = FALSE)
```

```
Views on a 17-letter DNAString subject
subject: TGATTGCTTGGTTGMTT
views:
      start end width
  [1]     3   4     2 [AT]
  [2]     7   8     2 [CT]
  [3]    15  16     2 [MT]
```

```
matchPattern("MT", seq_dna2, fixed = TRUE)
```

```
Views on a 17-letter DNAString subject
subject: TGATTGCTTGGTTGMTT
views:
      start end width
  [1]    15  16     2 [MT]
```

CONTAR BASES

- `alphabetFrequency()` - equivalente de `seq_table()`

```
1 alphabetFrequency(seq_dna2)
```

```
A C G T M R W S Y K V H D B N - + .
1 1 5 9 1 0 0 0 0 0 0 0 0 0 0 0 0 0
```

- Quer saber quais bases a sequência tem - `uniqueLetters()`

```
1 uniqueLetters(seq_dna2)
```

```
[1] "A" "C" "G" "T" "M"
```

- Só interessado em bases específicas - `letterFrequency()`
  - E com proporção (argumento `as.prob = TRUE`)

```
1 letterFrequency(seq_dna2, letters = c("CG"))
```

```
C|G
  6
```

```
1 letterFrequency(seq_dna2, letters = c("CG"), as.prob = TRUE)
```

```
      C|G
0.3529412
```

ESSAS FUNÇÕES SERVEM BEM PARA SEQUÊNCIAS GRANDES

- Importação feito com `readxxxStringSet()` utilizando um arquivo fasta

```
1 gag_dna <- readDNAStringSet(here("gag_sequence.fasta"))
2 gag_dna
```

```
DNAStringSet object of length 1:
    width seq                                               names
[1]  1291 GCCTGTTAGAAACAGCAGAGGGC...AACAATTCCCTCTCAGAAGCAGG  Grp6_2R_GAG
```

```
1 alphabetFrequency(gag_dna)
```

```
       A   C   G   T M R W S Y K V H D B N - + .
[1,] 482 251 314 244 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
1 uniqueLetters(gag_dna)
```

```
[1] "A" "C" "G" "T"
```

```
1 letterFrequency(gag_dna, letters = "GC")
```

```
     G|C
[1,] 565
```

VSS - BIOCONDUCTOR IMPORTAR

- Só Importar com `readxxxStringSet()`

CONSIDERAÇÕES ESPECIAIS PARA AMINOÁCIDOS

- Existem 2 alfabetos para AAs
  - Padrão - `AA_STANDARD` – 20 AA's que todos conhecem
  - Proteinogenic – `AA_PROTEINOGENIC` - mais 2 AAs ("U" e "O")
    - Em **algumas espécies**, codons normalmente interpretados como codons *stop*

```
1 AA_STANDARD
```

```
 [1] "A" "R" "N" "D" "C" "Q" "E" "G" "H" "I" "L" "K" "M" "F" "P" "S" "T" "W" "Y"
[20] "V"
```

```
1 AA_PROTEINOGENIC
```

```
 [1] "A" "R" "N" "D" "C" "Q" "E" "G" "H" "I" "L" "K" "M" "F" "P" "S" "T" "W" "Y"
[20] "V" "U" "O"
```

```
1 setdiff(AA_PROTEINOGENIC, AA_STANDARD)
```

```
[1] "U" "O"
```

TRANSCRIÇÃO E TRANSCRIÇÃO REVERSA

- Chamar a função cujo resultado que você quer e usar como argumento o `String` ou `StringSet` apropriado

```
1  (seq_rnax <- RNAString(seq_dna1))
```

```
12-letter RNAString object
seq: ACUUUGGCUAAG
```

```
1  (seq_dnax <- DNAString(seq_rnax)) # reverter ao original
```

```
12-letter DNAString object
seq: ACTTTGGCTAAG
```

```
1  (seq_gag_rna <- RNAStringSet(gag_dna))
```

```
RNAStringSet object of length 1:
    width seq                                               names
[1]  1291 GCCUGUUAGAAACAGCAGAGGGC...AACAAUUCCCUCUCAGAAGCAGG  Grp6_2R_GAG
```

CODONS

- Pode olhar diretamente nos codons no seu DNA string

```
1 codons(seq_dna1)
```

```
Views on a 12-letter DNAString subject
subject: ACTTTGGCTAAG
views:
      start end width
  [1]     1   3     3 [ACT]
  [2]     4   6     3 [TTG]
  [3]     7   9     3 [GCT]
  [4]    10  12     3 [AAG]
```

- Para vertente (*strand*) reversa, útil de conhecer o complemento reversa

```
1 reverseComplement(seq_dna1)
```

```
12-letter DNAString object
seq: CTTAGCCAAAGT
```

```
1 codons(reverseComplement(seq_dna1))
```

```
Views on a 12-letter DNAString subject
subject: CTTAGCCAAAGT
views:
      start end width
  [1]     1   3     3 [CTT]
  [2]     4   6     3 [AGC]
  [3]     7   9     3 [CAA]
  [4]    10  12     3 [AGT]
```

TRADUÇÃO

- Pode traduzir diretamente de DNA para AA
- Para reading frames outros que 1, precisa fazer um subset do `DNAString`
  - Utilizando `base::substr()` e não `stringr::str_sub()`
  - E.g. `substr(seq_dna2, start = 2, stop = 17)`

```
1  (seq_aa1 <- translate(seq_dna1))
```

```
4-letter AAString object
seq: TLAK
```

TRADUÇÃO COM CODONS AMBÍGUOS

- Argumento `if.fuzzy.codon =` oferece opções
  - = “X” – substituir “X” para qualquer codon com caracteres ambíguos
  - = “solve” – tentar de resolver a tradução
    - E.g., se for um codon sinônimo, vai achar uma solução

```
1 fuzzy_dna <- DNAString("HTGATHTGRCCCYTRTRA")
2 (seq_aa2 <- translate(seq_dna2, if.fuzzy.codon = "solve"))
```

```
5-letter AAString object
seq: *LLGX
```

```
1 (seq_aa2 <- translate(seq_dna2, if.fuzzy.codon = "X"))
```

```
5-letter AAString object
seq: *LLGX
```

ALINHAMENTO

INÍCIO

- Bioconductor: onde vai para achar funções de alinhamento de sequências
- Alinhamento leva 2 ou mais sequências divergentes de um único tipo
  - Produz um resultado que capta alguma qualidade de interesse
- Algoritmo está treinado para maximizar um sistema de pontuação
- Existem 2 pacotes que focam no alinhamento:
  - `DECIPHER`
  - `msa`
- Vamos usar msa só porque tenho mais experiência com ele.

INSTALAÇÃO DE `msa`

- Como os outros pacotes que precisamos

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

BiocManager::install("msa")
```

VANTAGEM SINAL DE msa

EXEMPLO – DE PACOTE msa

```r
library(msa)
msa_seqs_file <- system.file("examples", "exampleAA.fasta", package = "msa")
msa_seqs <- readAAStringSet(msa_seqs_file)
msa_seqs
```

```
AAStringSet object of length 9:
    width seq                                               names
[1]   452 MSTAVLENPGLGRKLSDFGQETS...LKILADSINSEIGILCSALQKIK PH4H_Homo_sapiens
[2]   453 MAAVVLENGVLSRKLSDFGQETS...KILADSINSEVGILCNALQKIKS PH4H_Rattus_norve...
[3]   453 MAAVVLENGVLSRKLSDFGQETS...KILADSINSEVGILCHALQKIKS PH4H_Mus_musculus
[4]   297 MNDRADFVVPDITTRKNVGLSHD...DVAPDDLVLNAGDRQGWADTEDV PH4H_Chromobacter...
[5]   262 MKTTQYVARQPDDNGFIHYPETE...MALVHEAMRLGLHAPLFPPKQAA PH4H_Pseudomonas_...
[6]   451 MSALVLESRALGRKLSDFGQETS...LKILADSISSEVEILCSALQKLK PH4H_Bos_taurus
[7]   313 MAIATPTSAAPTPAPAGFTGTLT...DVVDGDAVLNAGTREGWADTADI PH4H_Ralstonia_so...
[8]   294 MSGDGLSNGPPPGARPDWTIDQG...AVLTRGTQAYATAGGRLAGAAAG PH4H_Caulobacter_...
[9]   275 MSVAEYARDCAAQGLRGDYSVCR...QTADFEAIVARRKDQKALDPATV PH4H_Rhizobium_loti
```

O ALINHAMENTO DAS SEQUÊNCIAS

```
1 (test_align <- msa(msa_seqs, method = "ClustalW"))
```

```
use default substitution matrix

CLUSTAL 2.1

Call:
   msa(msa_seqs, method = "ClustalW")

MsaAAMultipleAlignment with 9 rows and 456 columns
    aln                                                  names
[1] MAAVVLENGVLSRKLSDFGQETSYIE...QLKILADSINSEVGILCNALQKIKS PH4H_Rattus_norve...
[2] MAAVVLENGVLSRKLSDFGQETSYIE...QLKILADSINSEVGILCHALQKIKS PH4H_Mus_musculus
[3] MSTAVLENPGLGRKLSDFGQETSYIE...QLKILADSINSEIGILCSALQKIK- PH4H_Homo_sapiens
[4] MSALVLESRALGRKLSDFGQETSYIE...QLKILADSISSEVEILCSALQKLK- PH4H_Bos_taurus
[5] -------------------------...LNAGDRQGWADTEDV---------- PH4H_Chromobacter...
[6] -------------------------...LNAGTREGWADTADI---------- PH4H_Ralstonia_so...
[7] -------------------------...LTRGT-QAYATAGGRLAGAAAG--- PH4H_Caulobacter_...
[8] -------------------------...------------------------- PH4H_Pseudomonas_...
[9] -------------------------...------------------------- PH4H_Rhizobium_loti
Con -------------------------...??????????????IL??A???--- Consensus
```

VISÃO MAIS CLARA DO RESULTADO

```
1 print(test_align, show = "alignment")
```

```
MsaAAMultipleAlignment with 9 rows and 456 columns
    aln                                                  names
[1] MAAVVLENGVLSRKLSDFGQETSYIE...QLKILADSINSEVGILCNALQKIKS PH4H_Rattus_norve...
[2] MAAVVLENGVLSRKLSDFGQETSYIE...QLKILADSINSEVGILCHALQKIKS PH4H_Mus_musculus
[3] MSTAVLENPGLGRKLSDFGQETSYIE...QLKILADSINSEIGILCSALQKIK- PH4H_Homo_sapiens
[4] MSALVLESRALGRKLSDFGQETSYIE...QLKILADSISSEVEILCSALQKLK- PH4H_Bos_taurus
[5] -------------------------...LNAGDRQGWADTEDV---------- PH4H_Chromobacter...
[6] -------------------------...LNAGTREGWADTADI---------- PH4H_Ralstonia_so...
[7] -------------------------...LTRGT-QAYATAGGRLAGAAAG--- PH4H_Caulobacter_...
[8] -------------------------...------------------------- PH4H_Pseudomonas_...
[9] -------------------------...------------------------- PH4H_Rhizobium_loti
Con -------------------------...????????????IL??A???--- Consensus
```

FUNÇÃO PRETTY PRINT

- Para quem tem opção de imprimir em pdf através de Latex
- Função `msa::msaPrettyPrint()` dá muitas opções para formatação do alinhamento
- Este exemplo mostra regiões conservadas entre as sequências

CALCULAR COM OUTROS PACOTES

- Mesmo pacotes fora de Bioconductor
- Vamos computar um matriz de distâncias entre as sequências de teste.
- Distância será expresso como o raiz quadrado de distância entre cada par das sequências
- `seqinr::dist.alignment()`
  - Pacote é um velho que tem ferramentas para sequências genómicas
  - Mas, `msa` tem um conversor que permite aplicação desta função

COMO FUNCIONAR

- 1º - converter o alinhamento para o tipo `seqinr::alignment`

```
test_align_dist <- msaConvert(test_align, type = "seqinr::alignment")
```

- 2º – computar a matriz de distâncias em `seqinr`

```
library(seqinr)
d <- dist.alignment(test_align_dist, matrix = "identity" )
as.matrix(d)[1:6, 1:4]
```

```
                              PH4H_Rattus_norvegicus PH4H_Mus_musculus
PH4H_Rattus_norvegicus                     0.0000000         0.1328911
PH4H_Mus_musculus                          0.1328911         0.0000000
PH4H_Homo_sapiens                          0.2782690         0.2742649
PH4H_Bos_taurus                            0.2978117         0.2864261
PH4H_Chromobacterium_violaceum             0.8655360         0.8655360
PH4H_Ralstonia_solanacearum                0.8766349         0.8766349
                              PH4H_Homo_sapiens PH4H_Bos_taurus
PH4H_Rattus_norvegicus                0.2782690       0.2978117
PH4H_Mus_musculus                     0.2742649       0.2864261
PH4H_Homo_sapiens                     0.0000000       0.2705009
PH4H_Bos_taurus                       0.2705009       0.0000000
PH4H_Chromobacterium_violaceum        0.8655360       0.8635755
PH4H_Ralstonia_solanacearum           0.8802952       0.8766349
```

2º EXEMPLO DE ALINHAMENTO COM `msa`

- Sequências de 17 replicatas/clones do gene gag de um paciente num estudo de nosso lab
- Arquivo fasta "`XXX_translation_result.fasta`"

```
aa_xxx <- readAAStringSet(here("XXX_translation_result.fasta"))

align_xxx <- msa(aa_xxx, method = "ClustalOmega")
```

```
using Gonnet
```

SEQUÊNCIA DE CONSENSO

- Esse o que precisamos para fazer a próxima fase da análise

```
1 align_xxx
```

```
ClustalOmega 1.2.0

Call:
   msa(aa_xxx, method = "ClustalOmega")

MsaAAMultipleAlignment with 17 rows and 189 columns
     aln                                                 names
 [1] ELERFAVNPGLLETAEGCKQILXQL...------------------------ LMC_clone16_ate_n...
 [2] ELERFAVNPGLLETAEGCKQILXQL...ETINEXAAEWXRIHPVHAGPVAPGX LMC_clone06_ate_n...
 [3] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone11_ate_n...
 [4] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone13_ate_n...
 [5] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone10_ate_n...
 [6] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone08_ate_n...
 [7] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone07_ate_n...
 [8] ELERFAVNPGLLXTAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone17_ate_n...
 [9] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone19_ate_n...
[10] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone02_ate_n...
[11] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone18_ate_n...
[12] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone03_ate_n...
[13] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone15_ate_n...
[14] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone14_ate_n...
[15] ELERFAVNPGLLETAEGCKQILXQL...ETINXXAAEWXRIHPVHAGPVAPGX LMC_clone05_ate_n...
```

```
1 msaConsensusSequence(align_xxx)
```

```
[1]
"ELERFAVNPGLLETAEGCKQILXQLQPALRTGSEELKSVYNTVATLYCVHRKIXVQXTKEALDKIXEEQNKSQKKVQQAAAATXNSSSQVSQNYPIVQNLQGQMVHQAISPR
```

SEMANA QUE VEM EM BIOCONDUCTOR

- Filogenética
- Expressão dos Genes