Bioconductor II

Maja Kuzman

2018-11-12

Biostrings

BiomaRt

There are handy wrappers: getGene() and getSequence(). Get sequence gor gene EML6.

```
mart = useMart("ensembl", dataset="hsapiens_gene_ensembl")
g <- getSequence( id = "100", type = "entrezgene", seqType = "gene_exon_intron", m
g
```

```
##
## 1 TCCTTTCACTCCCAGCTCCCTGGAGTCTCTCACGTAGAATGTCCTCTCCACCCCCACCCACCCCTGATGAACTCCTGCAGGTTCT
##   entrezgene
## 1        100
```

This representation is not efficient.

```
library(Biostrings)
sequenceExample <- DNAString(g[[1]])
sequence1 <- DNAStringSet(g[,1])

sequence1
```

```
##   A DNAStringSet instance of length 1
##     width seq
## [1] 32712 TCCTTTCACTCCCAGCTCCCTGGAGTCTCTC...CAATAAAGAAGCCCATGGCTGGTGGCATGCA
```

You can get those sequences directly to Biostrings object if you have a genome database for example, BSgenome. This will download the entire genome so it takes space and time, don't do it now!

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("BSgenome", version = "3.8")
library(BSgenome)
available.genomes(splitNameParts=FALSE, type=getOption("pkgType"))
library(BiocManager)
#install("BSgenome.Hsapiens.UCSC.hg38")
#getSeq()
```

Exercise: Get the sequences for IL2, IL2RA and IL2RB genes. Those are hgnc_symbols .Save it in a variable genSeq1

Exercise: Get the sequences for IL2, IL2RA and IL2RB genes. Those are hgnc_symbols .Save it in a variable genSeq1

```
g <- getSequence( id = c("IL2", "IL2RA", "IL2RB"),
                  type = "hgnc_symbol",
                  seqType = "gene_exon_intron",
                  mart = mart)
genSeq1 <- DNAStringSet(g[,1])
```

reverseComplement

There are cool functions you can use on a DNAStringSet object:

```
reverseComplement(genSeq1)
```

```
##   A DNAStringSet instance of length 3
##     width seq
## [1]  5256 TTTATATTTATCAAATTTATTAAATAGTTTT...TTTGTTACATTAGCCCACACTTAGGTGATAG
## [2] 49217 TTTTGTTGCATTGTACTTATTTTGTACAGTT...GCACCTTCACATGCTGGCAGGAGAGCGAAGT
## [3] 51682 TTTTTAAAAAATATCATTTATTCTTTTATAA...TTGGGCTGGCGTGTTCAGCCAGGAAACTGCC
```

```
complement(genSeq1)
```

```
##   A DNAStringSet instance of length 3
##     width seq
## [1]  5256 GATAGTGGATTCACACCCGATTACATTGTTT...TTTTGATAAATTATTTAAACTATTTATATTT
## [2] 49217 TGAAGCGAGAGGACGGTCGTACACTTCCACG...TTGACATGTTTTATTCATGTTACGTTGTTTT
## [3] 51682 CCGTCAAAGGACCGACTTGTGCGGTCGGGTT...AATATTTTCTTATTTACTATAAAAAATTTTT
```

```
reverse(genSeq1)
```

```
##   A DNAStringSet instance of length 3
##     width seq
## [1]  5256 AAATATAAATAGTTTAAATAATTTATCAAAA...AAACAATGTAATCGGGTGTGAATCCACTATC
## [2] 49217 AAAACAACGTAACATGAATAAAACATGTCAA...CGTGGAAGTGTACGACCGTCCTCTCGCTTCA
## [3] 51682 AAAAATTTTTTATAGTAAATAAGAAAATATT...AACCCGACCGCACAAGTCGGTCCTTTGACGG
```

DNAStringSet : subseq

If you want to get subsequences, for example, extract all positions from 1000 - 4000 from all sequences, use subseq:

```
subseq(genSeq1, 1000,4000)
```

```
##   A DNAStringSet instance of length 3
##     width seq
## [1]  3001 ACTAATAGCACAGAGTCTGGGGCCAGATATC...TGACAATATTTAAACAATTATGCTTAAGAGG
## [2]  3001 CCAGGCAAGATCAGAGTCCCACTCACAGGTT...CTTGAACCCGGGAGGCAGAGGTTGCAGGGAG
## [3]  3001 CTCTGTCAGCCAGACTGGAGTGCAGTGGCGC...GCCTTAAATAGGCAGTTTGATAAATCACCTG
```

DNAStringSet -> DNAString

You can extract subsequences from multiple positions in a single sequence. For this, a sequence has to be DNAString instead of DNAStringSet. This can be done easily if you subset DNAStringSet object with [[ ]].

```
genSeq1[1]
```

```
##   A DNAStringSet instance of length 1
##     width seq
## [1]  5256 CTATCACCTAAGTGTGGGCTAATGTAACAAA...AAAACTATTTAATAAATTTGATAAATATAAA
```

```
genSeq_DNAString <- genSeq1[[1]]
genSeq_DNAString
```

```
##   5256-letter "DNAString" instance
## seq: CTATCACCTAAGTGTGGGCTAATGTAACAAAGAG...GTAAAACTATTTAATAAATTTGATAAATATAAA
```

DNAStrings - Views

If you want to get multiple subsequences starting and ending in different positions, you can do it on a single DNAString, and use Views to get this:

```
starts <- seq(1000,4000, by=100)
subsequences <- Views(genSeq_DNAString, start= starts, end= starts+1000-1)
subsequences
```

```
##   Views on a 5256-letter DNAString subject
## subject: CTATCACCTAAGTGTGGGCTAATGTAACAAAG...AAAACTATTTAATAAATTTGATAAATATAAA
## views:
##       start  end width
##  [1]  1000 1999  1000 [ACTAATAGCACAGAGTCTGGGGCC...AGAAACTAATAAAAATATTTGATT]
##  [2]  1100 2099  1000 [GATTCAGTTTCATGTCTACTTAAA...TTCTGAGATTTAGTGTGCTTATTT]
##  [3]  1200 2199  1000 [ATAAGGTAAATACCATACAAGCAT...TTCTTTTAAATTGTAATATACCAT]
##  [4]  1300 2299  1000 [CCAATAGAACTTGAGATTTATAAT...ATTTCACTGGGACAAACATTTTTA]
##  [5]  1400 2399  1000 [TCCAAGCTCCTAGGCTACATTAGG...TATTGAGAGCCACTACTACATGAT]
##  ...   ...  ...   ... ...
## [27]  3600 4599  1000 [AAAGAGATTCACTTTTGTCTTTTT...TTTTTATTTAAATCTTTATTTGCA]
## [28]  3700 4699  1000 [GACATTCCTAAAAGTAACTCCAGT...TCATTTGGTATCATAACAAAATAT]
## [29]  3800 4799  1000 [ACATTGACAGATTCAGTTCCTTAT...CAACAGGCCTATAAGACTTCAATT]
## [30]  3900 4899  1000 [AACCATGCAAAAATCTGAAAACTG...ACATTCATGTGTGAATATGCTGAT]
## [31]  4000 4999  1000 [GATACAGAACACTGCAACAGTTTT...AATTAAGTGCTTCCCACTTAAAAC]
```

Exercise: Get sequences for all exons in IL2RA gene.

2 Points for an idea how to do this!

Nucleotide Frequency

Sometimes we are interested in nucleotide frequency of a sequence. There are nice functions for this:

alphabetFrequency:

```
alphabetFrequency(subsequences)[1:6,]
```

```
##        A   C   G   T M R W S Y K V H D B N - + .
## [1,] 346 162 182 310 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [2,] 352 153 181 314 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [3,] 355 152 181 312 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [4,] 353 156 181 310 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [5,] 357 155 182 306 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [6,] 353 153 173 321 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Nucleotide Frequency

Sometimes we are interested in nucleotide frequency of a sequence. There are nice functions for this:

trinucleotideFrequency:

```
trinucleotideFrequency(subsequences, as.prob = T)[1:6,1:9]
```

```
##              AAA        AAC        AAG        AAT        ACA          ACC
## [1,] 0.04609218 0.01302605 0.02705411 0.03707415 0.01503006 0.011022044
## [2,] 0.04809619 0.01302605 0.02805611 0.04008016 0.01402806 0.011022044
## [3,] 0.05210421 0.01102204 0.02905812 0.03907816 0.01402806 0.011022044
## [4,] 0.05010020 0.01202405 0.02805611 0.04008016 0.01402806 0.009018036
## [5,] 0.05811623 0.01102204 0.02805611 0.04108216 0.01603206 0.009018036
## [6,] 0.05711423 0.01202405 0.02505010 0.04108216 0.01503006 0.009018036
##      ACG        ACT        AGA
## [1,]   0 0.01903808 0.03607214
## [2,]   0 0.01803607 0.03707415
## [3,]   0 0.01803607 0.03907816
## [4,]   0 0.01703407 0.04008016
## [5,]   0 0.01703407 0.03807615
## [6,]   0 0.01903808 0.03607214
```

Nucleotide Frequency

Sometimes we are interested in nucleotide frequency of a sequence. There are nice functions for this:

oligonucleotideFrequency:

```
oligonucleotideFrequency(subsequences, as.prob = T, width = 5)[1:6,1:6]
```

```
##             AAAAA       AAAAC       AAAAG       AAAAT       AAACA
## [1,] 0.002008032 0.001004016 0.003012048 0.006024096 0.000000000
## [2,] 0.002008032 0.001004016 0.003012048 0.006024096 0.000000000
## [3,] 0.002008032 0.001004016 0.004016064 0.007028112 0.000000000
## [4,] 0.002008032 0.001004016 0.005020080 0.005020080 0.001004016
## [5,] 0.004016064 0.001004016 0.005020080 0.008032129 0.001004016
## [6,] 0.004016064 0.001004016 0.005020080 0.008032129 0.001004016
##             AAACC
## [1,] 0.002008032
## [2,] 0.002008032
## [3,] 0.002008032
## [4,] 0.002008032
## [5,] 0.002008032
## [6,] 0.002008032
```

consensusMatrix

It is easy to get consensus for many sequences of some length:

```
consensusMatrix(subsequences)[,1:6]
```

```
##   [,1] [,2] [,3] [,4] [,5] [,6]
## A   15    9   12   14   13    9
## C    5    9    5    1    4    4
## G    5    4    2    5    7   11
## T    6    9   12   11    7    7
## M    0    0    0    0    0    0
## R    0    0    0    0    0    0
## W    0    0    0    0    0    0
## S    0    0    0    0    0    0
## Y    0    0    0    0    0    0
## K    0    0    0    0    0    0
## V    0    0    0    0    0    0
## H    0    0    0    0    0    0
## D    0    0    0    0    0    0
## B    0    0    0    0    0    0
## N    0    0    0    0    0    0
## -    0    0    0    0    0    0
## +    0    0    0    0    0    0
## .    0    0    0    0    0    0
```

```
rowSums(consensusMatrix(subsequences))
```

```
##     A     C     G     T     M     R     W     S     Y     K     V     H
## 11121  5076  5059  9744     0     0     0     0     0     0     0     0
```

Alignments

We can align sequences as well:

```
matchPattern("CATGCATGGGCCATGTCTGACACAGTCTGCATTTGTAAGTAAAG",genSeq_DNAString)
```

```
##   Views on a 5256-letter DNAString subject
## subject: CTATCACCTAAGTGTGGGCTAATGTAACAAAG...AAAACTATTTAATAAATTTGATAAATATAAA
## views: NONE
```

```
pairwiseAlignment("CATGCATGGGCCATGTCTGACACAGTCTGCATTTGTAAGTAAAG",genSeq_DNAString)
```

```
## Global PairwiseAlignmentsSingleSubject (1 of 1)
## pattern: C-----------------------------...----------------------------AAG
## subject: CTATCACCTAAGTGTGGGCTAATGTAACAAAG...AAAACTATTTAATAAATTTGATAAATATAAA
## score: -20846.55
```

```
pairwiseAlignment("CATGCATGGGCCATGTCTGACACAGTCTNNNNTTGTAAGTAAAG",genSeq_DNAString)
```

```
## Global PairwiseAlignmentsSingleSubject (1 of 1)
## pattern: C-----------------------------...----------------------------AAG
## subject: CTATCACCTAAGTGTGGGCTAATGTAACAAAG...AAAACTATTTAATAAATTTGATAAATATAAA
## score: -20870.2
```

stringDist

If you want to calculate distance between two sequences, use fast optimized function: stringDist:

```
stringDist(c("hazy", "lazy", "crazy"))
```

```
##   1 2
## 2 1
## 3 2 2
```

stringDist

If you want to calculate distance between two sequences, use fast optimized function: stringDist:

```
stringDist(c("hazy", "lazy", "crazy"))
```

```
##   1 2
## 2 1
## 3 2 2
```

```
as.matrix(stringDist(c("hazy", "lazy", "crazy")))
```

```
##   1 2 3
## 1 0 1 2
## 2 1 0 2
## 3 2 2 0
```

Exercise: dotplot

Make a dotplot of the following sequence with itself. Use window size 70 and threshold 34.

```
SpongeSequence
```

```
##   3371-letter "DNAString" instance
## seq: AAAGAATGCTTGATGTAAGTTTTATGAAATCACA...GATTGAATCCACAATATATTGCTTTATTCGTTT
```

2 points for the idea !!

Solution - show in app!

matchPattern

If you have a pattern you wish to search for, there is optimized function to do this:

```
pattern <- c("TGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTGTG
matchPattern(pattern, SpongeSequence, max.mismatch = 15, with.indels = T)
```

```
##   Views on a 3371-letter DNAString subject
## subject: AAAGAATGCTTGATGTAAGTTTTATGAAATCA...TTGAATCCACAATATATTGCTTTATTCGTTT
## views:
##       start end width
##  [1]   782 850    69 [TGGTTTAAAACTTCTAATGTGTGCA...TGTGTGTGTGTGTGTGTGTGTGTG]
##  [2]   784 852    69 [GTTTAAAACTTCTAATGTGTGCATT...TGTGTGTGTGTGTGTGTGTGTGTG]
##  [3]   786 854    69 [TTAAAACTTCTAATGTGTGCATTTG...TGTGTGTGTGTGTGTGTGTGTGTG]
##  [4]   787 856    70 [TAAAACTTCTAATGTGTGCATTTGA...TGTGTGTGTGTGTGTGTGTGTGTG]
##  [5]   793 862    70 [TTCTAATGTGTGCATTTGAGTGTGT...TGTGTGTGTGTGTGTGTGTGTGTG]
##  ...   ... ...   ... ...
## [29]   843 912    70 [TGTGTGTGTGTGTGTGTGTGTGTGT...TGTGTGTGTGTGTGTGTGTGTGTG]
## [30]   845 914    70 [TGTGTGTGTGTGTGTGTGTGTGTGT...TGTGTGTGTGTGTGTGTGTGTGTG]
## [31]   847 915    69 [TGTGTGTGTGTGTGTGTGTGTGTGT...GTGTGTGTGTGTGTGTGTGTGTGT]
## [32]   851 918    68 [TGTGTGTGTGTGTGTGTGTGTGTGT...TGTGTGTGTGTGTGTGTGTGTACT]
## [33]   859 927    69 [TGTGTGTGTGTGTGTGTGTGTGTGT...GTGTGTGTGTGTACTACACACACG]
```

Another example:

1. Search for pattern p2 in SpongeSequence, allowing 30 mismatches with indels.
2. Extract only those matches
3. Make a logo of them using the seqLogo library and the makePWM function. Plot the logo with seqLogo function.

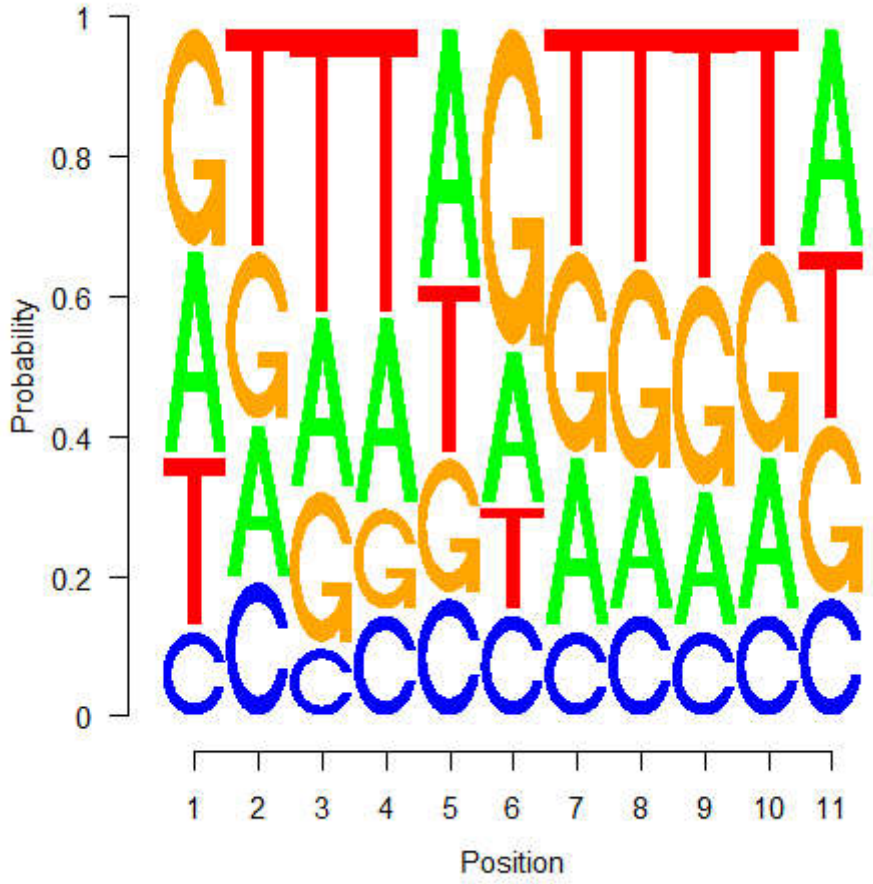
Solution

```
p2 <- "CGACGTGTAACTGCGGTTTAATTGTTTGTCGAGAGACAGTGCATGGAATGCGTGGAGCGAGTGGACCACC"
matchedPattern <- matchPattern(p2, SpongeSequence, max.mismatch = 30, with.indels
ss<-subseq(DNAStringSet(matchedPattern),1,40)
tt <- t(oligonucleotideFrequency(ss,1))
```

```
library(seqLogo)
```

```
## Loading required package: grid
```

```
p <- makePWM(tt/40)
seqLogo(p, ic.scale = F)
```



```
moreQueries <- PDict(DNAStringSet(c(pattern, p2)))
matchPDict(moreQueries, SpongeSequence)
```

```
## MIndex object of length 2
## [[1]]
## IRanges object with 17 ranges and 0 metadata columns:
##            start       end     width
##        <integer> <integer> <integer>
##    [1]       813       882        70
##    [2]       815       884        70
##    [3]       817       886        70
##    [4]       819       888        70
##    [5]       821       890        70
##    ...       ...       ...       ...
##   [13]       837       906        70
##   [14]       839       908        70
##   [15]       841       910        70
##   [16]       843       912        70
##   [17]       845       914        70
##
## [[2]]
## IRanges object with 1 range and 0 metadata columns:
##           start       end     width
##       <integer> <integer> <integer>
##   [1]      2811      2880        70
```

More useful functions

vmatchPattern

matchPattern, countPattern, vmatchPattern, vcountPattern, neditStartingAt, neditEndingAt, isMatchingStartingAt, isMatchingEndingAt, which.isMatchingStartingAt, which.isMatchingEndingAt

pairwiseAlignment

pairwiseAlignment, stringDist

matchPWM:

matchPWM, countPWM

OTHER

matchLRPatterns, trimLRPatterns, matchProbePair, findPalindromes, findComplementedPalindromes