Introduction to Biostrings

Maruf Ahmed Bhuiyan

7/25/2020

Table of Contents

setwd("~/Desktop/DataCamp/Bioconductor")

# Exploring the Zika virus sequence

It’s your turn to explore the Zika virus genome, which has been loaded in your workspace as zikaVirus. The genome was downloaded from NCBI and you can apply Biostrings functions to learn more about it.

Start by checking the alphabet of the sequence.

alphabet() # Shows the letters included in the sequence alphabetFrequency() # Shows the counts per letter Remember from the video that each alphabet corresponds to a specific biostring container, and each alphabet usually has extra code letters and symbols.

# Load packages  
library(Biostrings)

## Loading required package: BiocGenerics

## Loading required package: parallel

##   
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':  
##   
## clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,  
## clusterExport, clusterMap, parApply, parCapply, parLapply,  
## parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:stats':  
##   
## IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':  
##   
## anyDuplicated, append, as.data.frame, basename, cbind, colnames,  
## dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,  
## grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,  
## order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,  
## rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,  
## union, unique, unsplit, which, which.max, which.min

## Loading required package: S4Vectors

## Loading required package: stats4

##   
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:base':  
##   
## expand.grid

## Loading required package: IRanges

## Loading required package: XVector

##   
## Attaching package: 'Biostrings'

## The following object is masked from 'package:base':  
##   
## strsplit

# Load the zikavius genome in fasta format,   
# NCBI Reference Sequence: NC\_012532.1  
zikaVirus <- readDNAStringSet('zika.fasta')  
  
# Check the alphabet of the zikaVirus  
alphabet(zikaVirus)

## [1] "A" "C" "G" "T" "M" "R" "W" "S" "Y" "K" "V" "H" "D" "B" "N" "-" "+" "."

# Check the alphabetFrequency of the zikaVirus  
alphabetFrequency(zikaVirus)

## A C G T M R W S Y K V H D B N - + .  
## [1,] 2991 2359 3139 2305 0 0 0 0 0 0 0 0 0 0 0 0 0 0

# Check alphabet of the zikaVirus using baseOnly = TRUE  
alphabet(zikaVirus, baseOnly = TRUE)

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

# Biostrings containers

Now that you know how to check the alphabet of a sequence, can you check what the container class of the zikaVirus object is?

# Check alphabet of the zikaVirus using baseOnly = TRUE  
alphabet(zikaVirus, baseOnly = TRUE)

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

# This is a DNAStringset

# Manipulating Biostrings

Using a short sequence (dna\_seq) from the zikaVirus object, it is your turn to have a go with the two biological processes of transcription and translation.

In the first two parts of this exercise, you will apply these processes separately. In the last part, you’ll apply them in one step.

You’ll be using a very small sequence in this exercise, but remember that the power of Biostrings comes to light when manipulating much larger sequences.

The Biostrings package has already been loaded for you. Using zikaVirus, you will translate the first 21 characters into an AAString.

# Unlist the set, select the first 21 letters, and assign to dna\_seq  
dna\_seq <- subseq(unlist(zikaVirus), end = 21)  
dna\_seq

## 21-letter "DNAString" instance  
## seq: AGTTGTTGATCTGTGTGAGTC

# Transcribe dna\_seq into an RNAString object and print it  
rna\_seq <- RNAString(dna\_seq)   
rna\_seq

## 21-letter "RNAString" instance  
## seq: AGUUGUUGAUCUGUGUGAGUC

# Translate rna\_seq into an AAString object and print it  
aa\_seq <- translate(rna\_seq)  
aa\_seq

## 7-letter "AAString" instance  
## seq: SC\*SV\*V

# Transcribe and translate dna\_seq into an AAString object and print it  
aa\_seq <- translate(dna\_seq)  
aa\_seq

## 7-letter "AAString" instance  
## seq: SC\*SV\*V

# Well done! You were able to get an "AAString" from a "DNAString"!  
# That's excellent. Next time, you can use the shortcut because the  
# function translate() accepts both a "DNAString" and an "RNAString".  
# After this lesson, you might want to go and test the functions you've  
# just learned with sets and see how powerful these tools are!

# From a set to a single sequence

From the video, you know that sequences can be loaded into R using the function readDNAStringSet(). The zikaVirus has been read into your environment using this function.

It is your turn to convert this set into a single sequence, explore the new sequence, and subset it.

# Create zikv with one collated sequence using zikaVirus  
zikv <- unlist(zikaVirus)  
  
# Check the length of zikaVirus and zikv  
length(zikaVirus)

## [1] 1

length(zikv)

## [1] 10794

# Check the width of zikaVirus  
width(zikaVirus)

## [1] 10794

# Subset zikv to only the first 30 bases  
subZikv <- subseq(zikv, end = 30)  
subZikv

## 30-letter "DNAString" instance  
## seq: AGTTGTTGATCTGTGTGAGTCAGACTGCGA

# Subsetting a set

In the previous exercise, you’ve used subseq() to subset a single sequence. Here, you can try subseq() using a set with 3 sequences. The arguments are the same as before: object, start, and end. The last two should be in the form of a vector, for each of the sequences on the set.

subseq(zikaSet, start = c(20, 40, 2), end = c(50, 45, 22) ) What is the width of the sequences after subsetting?

zikaset <- DNAStringSet(zikv,   
 start = c(20, 40, 2),   
 end = c(50, 45, 22))  
zikaset

## A DNAStringSet instance of length 3  
## width seq  
## [1] 31 TCAGACTGCGACAGTTCGAGTCTGAAGCGAG  
## [2] 6 TCTGAA  
## [3] 21 GTTGTTGATCTGTGTGAGTCA

# Common sequence manipulation functions

So far, you’ve learned the the most common sequence manipulation functions:

reverse() complement() reverseComplement() translate() In real life, you can manipulate really large sequences using these functions.

However, for you to visually see the value and the results of these functions in this exercise, you will use a small subset. The object zikv, which you have previously subsetted from the Zika genome, has only 30 bases.

subZikv

## 30-letter "DNAString" instance  
## seq: AGTTGTTGATCTGTGTGAGTCAGACTGCGA

reverse(subZikv)

## 30-letter "DNAString" instance  
## seq: AGCGTCAGACTGAGTGTGTCTAGTTGTTGA

complement(subZikv)

## 30-letter "DNAString" instance  
## seq: TCAACAACTAGACACACTCAGTCTGACGCT

reverseComplement(subZikv)

## 30-letter "DNAString" instance  
## seq: TCGCAGTCTGACTCACACAGATCAACAACT

translate(subZikv)

## 10-letter "AAString" instance  
## seq: SC\*SV\*VRLR

# Searching for a pattern

Let’s find some occurrences of a pattern in the zikaVirus set using vmatchPattern(). Then, let’s try the same pattern search using matchPattern() with a single sequence, zikv.

# For Sets

vmatchPattern(pattern = “ACATGGGCCTACCATGGGAG”, subject = zikaVirus, max.mismatch = 1) # For single sequences matchPattern(pattern = “ACATGGGCCTACCATGGGAG”, subject = zikv, max.mismatch = 1) Both functions should find the same number of occurrences, but you will notice a different output. How many matches do we get when running each pattern search individually?

# For Sets  
vmatchPattern(pattern = "ACATGGGCCTACCATGGGAG",   
 subject = zikaVirus, max.mismatch = 1)

## MIndex object of length 1  
## $`NC\_012532.1 Zika virus, complete genome`  
## IRanges object with 1 range and 0 metadata columns:  
## start end width  
## <integer> <integer> <integer>  
## [1] 8561 8580 20

# For single sequences  
matchPattern(pattern = "ACATGGGCCTACCATGGGAG",   
 subject = zikv, max.mismatch = 1)

## Views on a 10794-letter DNAString subject  
## subject: AGTTGTTGATCTGTGTGAGTCAGACTGCGACAGT...TTCGGCGGCCGGTGTGGGGAAATCCATGGTTTCT  
## views:  
## start end width  
## [1] 8561 8580 20 [ACATGGGCCTACCATGGGAG]

# Finding Palindromes

It is your turn to find palindromic sequences using the zikv1000 sequence. Remember, findPalindromes() searches in a single sequence and does not work with a set.

zikv1000 <- subseq(zikv, end = 1000)  
zikv1000

## 1000-letter "DNAString" instance  
## seq: AGTTGTTGATCTGTGTGAGTCAGACTGCGACAGTTC...CCGGCATACAGTATCAGGTGCATTGGAGTCAGCAAT

# Find palindromes in zikv1000  
findPalindromes(zikv1000)

## Views on a 1000-letter DNAString subject  
## subject: AGTTGTTGATCTGTGTGAGTCAGACTGCGACAGT...GGCATACAGTATCAGGTGCATTGGAGTCAGCAAT  
## views:  
## start end width  
## [1] 18 26 9 [AGTCAGACT]  
## [2] 137 144 8 [ATCCGGAT]  
## [3] 143 151 9 [ATTGTCAAT]  
## [4] 363 370 8 [AAGATCTT]  
## [5] 464 471 8 [GCCATGGC]  
## [6] 503 510 8 [TACATGTA]  
## [7] 599 606 8 [CACATGTG]  
## [8] 835 844 10 [CTTGATCAAG]  
## [9] 866 876 11 [AACCCCGGGTT]  
## [10] 915 925 11 [GCTCGACGAGC]  
## [11] 970 978 9 [ATACAGTAT]

# Finding a conserved region within six frames

Congratulations on getting this far!

Now you will be able to look for the NS5 protein sequence in the Zika virus sequence. The NS5 is a very conserved virus protein. It was downloaded and loaded for you from Uniprot.

The Zika virus DNA sequence has been transcribed into an RNAStringSet, called rnaframesZikaSet. The set has six reading frames (one per sequence) for you to translate into amino acids. When doing the search, you will set the max.mismatch argument in your call of vcountPattern() to add flexibility to your search.

NS5 <- readAAStringSet('NS5.fasta')  
NS5

## A AAStringSet instance of length 1  
## width seq names   
## [1] 325 SRAIWYMWLGARFLEFEALGFLN...LRLMANAICSSVPVDWVPTGRTT tr|A0A0B4ZYS0|A0A...

# Reading Frames available from https://www.ncbi.nlm.nih.gov/orffinder/  
rnaframesZikaSet <-readAAStringSet('rnaframesZikaSet.fasta')  
rnaframesZikaSet

## A AAStringSet instance of length 6  
## width seq names   
## [1] 3598 SC\*SV\*VRLRQFESEARANNSIN...PPRWPPGTDRRTSAAGVGKSMVS lcl|Sequence\_1 Fr...  
## [2] 3597 VVDLCESDCDSSSLKRELTTVST...FHHAGRQAQIAELRRPVWGNPWF lcl|Sequence\_2 Fr...  
## [3] 3597 LLICVSQTATVRV\*SES\*QQYQQ...STTLAARHRSPNFGGRCGEIHGF lcl|Sequence\_3 Fr...  
## [4] 3598 TTSRHSDSQSLELRFRSSVVTDV...WWAPRWACIASSRRGTHPFGHNR lcl|Sequence\_4 Fr...  
## [5] 3597 NNIQTL\*VAVTRTQLSL\*CCY\*C...EVVSAALCLDGFKPPRHPSIWPK lcl|Sequence\_5 Fr...  
## [6] 3597 QQDTHTLSRCNSDSALALLLILL...GGRQGGPVSRRVEAAPTPFDMTE lcl|Sequence\_6 Fr...

# print the rnaframesZikaSet  
rnaframesZikaSet # It's already translated

## A AAStringSet instance of length 6  
## width seq names   
## [1] 3598 SC\*SV\*VRLRQFESEARANNSIN...PPRWPPGTDRRTSAAGVGKSMVS lcl|Sequence\_1 Fr...  
## [2] 3597 VVDLCESDCDSSSLKRELTTVST...FHHAGRQAQIAELRRPVWGNPWF lcl|Sequence\_2 Fr...  
## [3] 3597 LLICVSQTATVRV\*SES\*QQYQQ...STTLAARHRSPNFGGRCGEIHGF lcl|Sequence\_3 Fr...  
## [4] 3598 TTSRHSDSQSLELRFRSSVVTDV...WWAPRWACIASSRRGTHPFGHNR lcl|Sequence\_4 Fr...  
## [5] 3597 NNIQTL\*VAVTRTQLSL\*CCY\*C...EVVSAALCLDGFKPPRHPSIWPK lcl|Sequence\_5 Fr...  
## [6] 3597 QQDTHTLSRCNSDSALALLLILL...GGRQGGPVSRRVEAAPTPFDMTE lcl|Sequence\_6 Fr...

# translate all 6 reading frames   
# AAzika6F <- translate(rnaframesZikaSet)  
# AAzika6F  
  
# Count the matches allowing 15 mistmatches  
#vcountPattern(pattern = NS5, subject = rnaframesZikaSet, max.mismatch = 15)  
  
# Select the frame that contains the match  
# selectedSet <- AAzika6F[3]   
# selectedSet  
  
# Convert this frame into a single sequence  
# selectedSeq <- unlist(selectedSet)  
# selectedSeq

# Looking for a match

From the previous exercise, you have two objects: selectedSet(a set) and selectedSeq (a single sequence).

You have recently discovered that pattern ns5 is on frame 3 of the AAzika6F. Now, you will discover how the match looks like using selectedSet and selectedSeq.

# selectedSeq  
# selectedSet  
  
# Use vmatchPattern() with the set  
# vmatchPattern(pattern = ns5, subject = selectedSet, max.mismatch = 15)  
  
# Use matchPattern() with the single sequence  
# matchPattern(pattern = ns5, subject = selectedSeq, max.mismatch = 15)  
  
# Take your time to see the similarities/differences in the result.

# Genetic Code Table from seqinr package

# install.packages("seqinr")  
library(seqinr)

##   
## Attaching package: 'seqinr'

## The following object is masked from 'package:Biostrings':  
##   
## translate

tablecode()

