Introduction to Scientific Computing for Biologists

ISCB20.09 - R for Bioinformatics

An Introduction to `seqinr` and `Biconductor`

Md. Jubayer Hossain
https://jhossain.me/
jubayer@hdrobd.org

Founder
Health Data Research Organization
Lead Organizer
Scientific Computing for Biologists

10 February 2021

Section-1: Introduction to “seqinr”

seqinr: Package Introduction

- Exploratory data analysis and data visualization for biological sequence (DNA and protein) data.

```
install.packages("seqinr")
```

Reading and Exploring Data

- Reading `FASTA`, `FASTQ` Files.
- Exploring DNA and Protein Sequence.

DNA Sequence Statistics

- Length of a DNA Sequence.
- Nucleotide Frequency/Base Composition
- Nucleotide Percentage
- K-mer Analysis/DNA Words
- Nucleotide Frequency Distribution Plot
- GC Content of DNA
- Local Variation in GC Content / Sliding Window Analysis
- Dot Plot

Section-2: Introduction to Bioconductor

What is Bioconductor?

- Bioconductor is a free, open source and open development software project for the analysis and comprehension of genomic data generated by wet lab experiments in molecular biology.
- Bioconductor is based primarily on the statistical R programming language, but does contain contributions in other programming languages.
- Website: https://www.bioconductor.org/

Installing Bioconductor

Biconductor has its own repository, way to install packages, and each release is designed to work with a speci c version of R.

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(version = "3.12")
```

What do we measure and why?

- Structure: elements, regions, size, order, relationships
- Function: expression, levels, regulation, phenotypes

Useful Online Resources

- https://kasperdanielhansen.github.io/genbioconductor/
- https://kasperdanielhansen.github.io/genbioconductor/html/Online_Resources.html
- https://www.datacamp.com/courses/introduction-to-bioconductor-in-r

Introduction to Biostrings

- Memory efficient to store and manipulate sequence of characters.
- Containers that can be inherited.
- The BString class comes from big string

```
showClass("XString")
showClass("BString")
showClass("BStringSet")
```

Load Biostrings Package

```
# Load Biostrings
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':
```

Biostring Alphabets

```
DNA_BASES # DNA 4 Bases
```

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

```
RNA_BASES # RNA 4 Bases
```

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

```
AA_STANDARD # 20 Amino Acids(AA)
```

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

Biostring Alphabets(Cont..)

```
DNA_ALPHABET # Contains IUPAC_CODE_MAP
```

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

```
RNA_ALPHABET # Contains IUPAC_CODE_MAP
```

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

```
AA_ALPHABET  # Contains AMINO_ACID_CODE
```

```
 [1] "A" "R" "N" "D" "C" "Q" "E" "G" "H" "I" "L" "K" "M" "F" "P" "S" "T" "W
[20] "V" "U" "O" "B" "J" "Z" "X" "*" "-" "+" "."
```

Transcription: DNA to RNA

```
# DNA Single String
dna_seq <- DNAString("ATGATCTCGTAATCCG")
dna_seq
```

```
16-letter DNAString object
seq: ATGATCTCGTAATCCG
```

```
# Transcription DNA to RNA string
rna_seq <- RNAString(dna_seq)
rna_seq
```

```
16-letter RNAString object
seq: AUGAUCUCGUAAUCCG
```

Translation: RNA to Amino Acids(AA)

```
# Translation RNA to Amino Acids(AA)
aa_seq <- translate(rna_seq)
```

```
Warning in .Call2("DNAStringSet_translate", x, skip_code,
dna_codes[codon_alphabet], : last base was ignored
```

```
aa_seq
```

```
5-letter AAString object
seq: MIS*S
```

Shortcut Translation: DNA to Amino Acids(AA)

```
dna_seq <- DNAString("ATGATCTCGTAATCCG")
# Translate
translate(dna_seq)
```

```
Warning in .Call2("DNAStringSet_translate", x, skip_code,
dna_codes[codon_alphabet], : last base was ignored

5-letter AAString object
seq: MIS*S
```

Single vs Set

- XString to store a single sequence
  - BString for any string
  - DNAString for DNA
  - RNAString for RNA
  - AAString for amino acids
- XStringSet for many sequences
  - BStringSet
  - DNAStringSet
  - RNAStringSet
  - AAStringSet

Creating and Collecting stringSet

```
covid19 <- readDNAStringSet("covid19.fasta")
length(covid19) # The set contains only one sequence
```

```
[1] 1
```

```
width(covid19) # Bases
```

```
[1] 29903
```

Creating and Collecting stringSet(Cont..)

```
# To collate the sequence use unlist
covid19_seq <- unlist(covid19)
length(covid19_seq)
```

```
[1] 29903
```

```
# width(covid19_seq)
# Error unable to find width for "DNAString"
```

From a Single Sequence to a Set

```
# To create a new set from a single sequence
single_seq <- DNAStringSet(covid19_seq,
                           start = c(1, 101, 201),
                           end = c(100, 200, 300))
covid19_seq
```

```
29903-letter DNAString object
seq: ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAAC...GACAAAAAAAAAAAAAAAAAAAAAAAAAA
```

```
length(single_seq)
```

```
[1] 3
```

```
width(single_seq)
```

```
[1] 100 100 100
```

Complement Sequence

```
# Create a DNA Sequence
dna_seq <- DNAString("ATGATCTCGTAATTCCGGAA")
dna_seq
```

```
20-letter DNAString object
seq: ATGATCTCGTAATTCCGGAA
```

```
# Complement
complement(dna_seq)
```

```
20-letter DNAString object
seq: TACTAGAGCATTAAGGCCTT
```

Rev a Sequence

```
# Create a DNA Sequence
dna_seq <- DNAString("ATGATCTCGTAATTCCGGAA")
dna_seq
```

```
20-letter DNAString object
seq: ATGATCTCGTAATTCCGGAA
```

```
# Rev
rev(dna_seq)
```

```
20-letter DNAString object
seq: AAGGCCTTAATGCTCTAGTA
```

Reverse a Sequence

```
dna_seq
```

```
20-letter DNAString object
seq: ATGATCTCGTAATTCCGGAA
```

```
# Reverse
reverse(dna_seq)
```

```
20-letter DNAString object
seq: AAGGCCTTAATGCTCTAGTA
```

Reverse Complement

```
# Reverse Complement
reverseComplement(dna_seq)
```

```
20-letter DNAString object
seq: TTCCGGAATTACGAGATCAT
```

```
# Using two functions together
reverse(complement(dna_seq))
```

```
20-letter DNAString object
seq: TTCCGGAATTACGAGATCAT
```

Sequence Alignment

- Pairwise global alignment of DNA sequences using the Needleman-Wunsch algorithm

Loading and Exploring Genome

```
# Length
length()
# Names
names()
# Sequence Info
seqinfo()
# Sequence Level Info
seqlevels()
# Sequence Length
seqlengths()
# Sequence Names
seqnames()
# Sequence Style
seqlevelsStyle()
```