Topic: How R can be used for Computational Biology - Continued!

Abigail Meil

Background
I'm a Computer Science & Cognitive Science double major, so I'm always trying to find ways to bring things like coding and biology together. One possible way: Computational Biology! I love the intersection of computer science and biology to more efficiently solve problems in the medical realm, genetic realm, and beyond.

Motivation
Many useful, existing packages in R but one the Bioconductor project (http://www.bioconductor.org/), a set of packages for R to analyze genomic data, is one of the most useful for computational biologists. I touched on some of the broader applications of the bioconductor package in my first post, but for this post, I am solely going to focus on how to represent genomic data in R (version 3.4.1) using RStudio (Version 1.0.153) with the Bioconductor package "Biostrings."

A little background…
First off, what is genomic data exactly? Genomic data refers to the genome and DNA data of an organism. We can use this information in Computational Biology to collect, store and process the DNA of living things. To analyze genomic data, we can use big data processing and analysis techniques, both of which are supported by packages like Bioconductor. To analyze the data, we look at the particular make up and structure of the data, among other genomic parameters. What's the end goal of all of this analysis? We want to determine the functions of specific genes!

Getting Started with Biostrings
Now that we know a little more about what genomic data is and why we want to analyze it, let's learn about how to get started with the Bioconductor package "Biostrings" so that we can start representing DNA or amino acid sequences in R! To start…

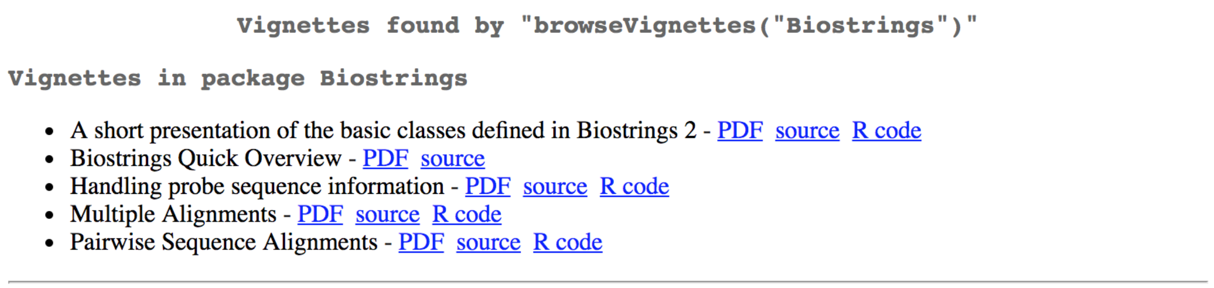
1. Install the Biostrings package (version 3.6)!

```
source("https://bioconductor.org/biocLite.R")
biocLite("Biostrings")
```

2. Now that we have the package installed, let's browse the documentation of the package a bit:

```
browseVignettes("Biostrings")
```

You should see a webpage appear that looks like this:



Vignettes found by "browseVignettes("Biostrings")"

Vignettes in package Biostrings

- A short presentation of the basic classes defined in Biostrings 2 - PDF source R code
- Biostrings Quick Overview - PDF source
- Handling probe sequence information - PDF source R code
- Multiple Alignments - PDF source R code
- Pairwise Sequence Alignments - PDF source R code

This documentation is a great place to get started with the package and a good place to go if you ever need help! Here, you can find a list of all the classes and functions in the Biostrings package, descriptions of each function, how to use them and more!

3. Now that we've looked at the documentation a bit, let's make a DNA sequence! To start:

```
library(Biostrings)
d <- DNAString("TTGAAAA-CTC-N")
```

- Here, we add Biostrings to our library of classes and create a new DNA string of "TTGAAAA-CTC-N" and save it into the variable "d"

4. We have a DNA sequence now - so let's see what we can do with it! Let's start by finding the length:

```
length(d)
```

```
## [1] 13
```

5. We can also convert the sequence we just created back into a string with:

```
str = paste(d, collapse="")
str
```

```
## [1] "TTGAAAA-CTC-N"
```

6. We can look at the metadata of the DNA sequence by using the str() function:

```
str(d)
```

```
## Formal class 'DNAString' [package "Biostrings"] with 5 slots
##   ..@ shared         :Formal class 'SharedRaw' [package "XVector"] with 2 slots
##   .. .. ..@ xp                    :<externalptr>
##   .. .. ..@ .link_to_cached_object:<environment: 0x7fc740b0dae8>
##   ..@ offset         : int 0
##   ..@ length         : int 13
##   ..@ elementMetadata: NULL
##   ..@ metadata       : list()
```

7. We can also calculate the alphabet frequency of our DNA string:

```
alphabetFrequency(d, baseOnly = TRUE, as.prob = TRUE)
```

```
##          A          C          G          T      other
## 0.30769231 0.15384615 0.07692308 0.23076923 0.23076923
```

8. Or just look at a particular alphabet frequency, like the frequency of the letter “A”.

```
letterFrequency(d, "A", as.prob = TRUE)
```

```
##         A
## 0.3076923
```

9. We can even calculate the reverse complement of the DNA sequence!

```
reverseComplement(d)
```

```
##   13-letter "DNAString" instance
## seq: N-GAG-TTTTCAA
```

Indexing into our DNA sequence
We can access individual letters in our sequence using the [] operator:

```
d[3]
```

```
##   1-letter "DNAString" instance
## seq: G
```

If we want a substring instead, we can specify a range of indeces:

```
d[1:5]
```

```
##   5-letter "DNAString" instance
## seq: TTGAA
```

We can also use the function “subseq” to return a subsequence of our DNA string. This is often the preferred method of finding subsequences, especially whwen using very large DNA sequences.

```
subseq(d, start = 1, end = 5)
```

```
##   5-letter "DNAString" instance
## seq: TTGAA
```

We can also easily convert any sequence to a character vector with the toString function:

```
toString(d)
```

```
## [1] "TTGAAAA-CTC-N"
```

In Conclusion…
I hope you enjoyed learning how to represent genomic data in R with the Bioconductor package, and also how apply some cool functions to the data! If you are interested in learning more about Bioconductor and how to use its packages more in depth, here is a course-materials page that has a plethora of informational slides, and open courses from universities: http://www.bioconductor.org/help/course-materials/

References

https://www.techopedia.com/definition/31247/genomic-data
https://bioconductor.org/packages/release/bioc/html/Biostrings.html
https://bioconductor.org/help/workflows/sequencing/
http://127.0.0.1:21769/library/Biostrings/doc/Biostrings2Classes.pdf
http://127.0.0.1:21769/library/Biostrings/doc/BiostringsQuickOverview.pdf
http://127.0.0.1:21769/library/Biostrings/doc/matchprobes.pdf
https://web.stanford.edu/class/bios221/labs/biostrings/lab_1_biostrings.html