Introduction to Programming in R: genetic sequences

Erik Spence

SciNet HPC Consortium

12 April 2023

Today's slides

Today's slides can be found here. Go to the "Introduction to programming in R" page, under Lectures, "Genetic sequences".

`https://scinet.courses/1277`

Today's Lecture

Today's class will explore the wild wild world of file input and output in R.

- Using Biostrings to hold sequence data,
- FASTA,
- Genomic databases (Entrez),
- rentrez.

Introducing Bioconductor

Not surprisingly, R comes with many packages for dealing with genetic sequence data.

- The majority of these are stored at Bioconductor (`https://bioconductor.org`), rather than CRAN.
- If "install.packages" doesn't work for a package that you're pretty sure exists, it's probably hosted at Bioconductor.
- To get easy access to Bioconductor packages, install the 'BiocManager' package from CRAN.

```
>
> install.packages('BiocManager')
>
```

Dealing with genetic sequences

We often want to work with genetic sequences. There are several packages available to do so.

- We will start by using the 'Biostrings' package.
- It can handle DNA, RNA and Amino Acid sequences.
- It can contain groups of strings in a 'Set'.
- It can also transcribe DNA to RNA or protein sequences.

```
> library(BiocManager)
>
> BiocManager::install('Biostrings')
> library(Biostrings)
>
> seq <- DNAString("AGTACACTGTAG")
>
> class(seq)
[1] "DNAString"
attr(,"package")
[1] "Biostrings"
>
> print(seq)
12-letter DNAString object
seq: AGTACACTGTAG
>
```

Dealing with genetic sequences, continued

Biostrings has pretty much all the functionality you might want.

- It can give you subsets of the sequence.
- It can give you the complement and reverse complement.

```
>
> complement(seq)
12-letter DNAString object
seq: TCATGTGACATC
>
> reverseComplement(seq)
12-letter DNAString object
seq: CTACAGTGTACT
>
> seq[3:5]
3-letter DNAString object
seq: TAC
>
> subseq(seq, 6, 9)
4-letter DNAString object
seq: ACTG
>
```

Transcription and translation

Transcription and translation are builtin as well.

- Conversion from DNAString to RNAString, or vice versa, is done using the appropriate function.
- Translation to amino acids is done using the 'translate' function.
- The single-letter representation of the proteins is used.
- You can also search for substrings within sequences.

```
>
> RNAString(seq)
12-letter RNAString object
seq: AGUACACUGUAG
>
> translate(seq)
4-letter AAString object
seq: STL*
>
> matchPattern("ACA", seq)
Views on a 12-letter DNAString subject
subject: AGTACACTGTAG
views:
    start end width
[1]     4   6     3 [ACA]
>
```

String Sets

There are two basic containers for holding genetic sequences:

- The functions for single strings are DNAString, RNAString or AAString, as we've already seen.
- Containers for sets of strings are built using DNAStringSet, RNAStringSet and AAStringSet.

The DNAString uses an extended alphabet: ('-', and 'N' are allowed). You can see the full alphabet in the 'IUPAC_CODE_MAP' variable.

```
>
> dna1 <- DNAString("ACGT-N")
>
> dna2 <- DNAStringSet(list(ch1 = seq,
+                    ch2 = dna1,
+                    ch3 = DNAString("ACGTACG")))
>
> dna2
DNAStringSet object of length 1:
    width seq          names
[1]    12 AGTACACTGTAG ch1
[2]     6 ACGT-N       ch2
[3]     7 ACGTACG      ch3
>
```

Genomic Data Formats

Obviously, data is usually stored in files. There are many different types that are associated with sequence data and clinical research:

- Raw sequencing data: BAM, FASTQ, WIGGLE, FASTA
- Nucleotide variation: TSV, MAF, VCF
- Microarray, gene expression data: TSV
- Structural rearrangement: VCF, FASTA

And others. Obviously the file format needed for your genetic data will depend specifically on what you're doing.

These data were stolen from `https://gdc.cancer.gov/node/265`.

Genomic Data Formats: FASTA

The format we're most interested in is FASTA.

```
>1 dna:chromosome chromosome:Galgal4:1:1:195276750:1 REF CCGACCAGTTGTAACTCAAAAACCAAAA-
GAAACGCAGGACAGGCCAGCGGGGCTGCCCCC GCAGGAGCTGGAGAGAGTAGGGATTATTAGACCTGCACACAGCCCATACAACTCCCCCAT
...
```

FASTA is a common genetic file format.

- One line starts with '>'. Following the '>' is the identifier, the rest is software-dependent.
- The next lines that do not start with '>' are part of the sequence, which can be split over several lines.
- Several sequences are allowed in one file.
- Files extensions: .fa ,.fasta, .fna

Reading from a file

Not surprisingly, we can use Biostrings to read data directly from a file:

- The readDNAStringSet function supports FASTA and FASTQ formats.
- Direct reading of gzipped files is supported.
- If your file format is not supported by Biostrings, the odds are that another package will read your data.

```
>
> chick <- readDNAStringSet('chromosome1.fa')
>
> chick
DNAStringSet object of length 1:
       width seq                                names
[1] 19307913 CCGACCAGTTGT...ATGGGATGGA 1 dna:chromosome ...
>
```

Writing sequences to a file

You can also write your string set to file.

- We use the writeXStringSet to write sequence sets to file.
- You can't write a single string to file, only sets.

```
>
> dna2
DNAStringSet object of length 2:
    width seq           names
[1]    12 AGTACACTGTAG ch1
[2]     6 ACGT-N       ch2
[3]     7 ACGTACG      ch3
>
> writeXStringSet(dna2, 'mydata.fasta')
>
> dir()
[1] "chromosome1.fa" "mydata.fasta"
>
```

Online genomic databases with R

A lot of genomic data is available in online databases on the internet.

- Databases are structured collections of data.
- They have a well-define way (an "API") to get data out.
- It would be too much data to download all of it on your own computer.
- Specific bits of data can be requested from these online databases.
- With certain R packages, we can do this directly from within R.
- A lot of genomic data still comes as text.
- That's convenient in an online setting, as the internet is text-based.

With Biostrings you often do not need to know the file details as long as you know the file format. However, to interact with online databases, we need to look at what kind of data we can find in online genomic databases and how the data will be delivered.

Public online genomic databases

What do online genetic databases offer?

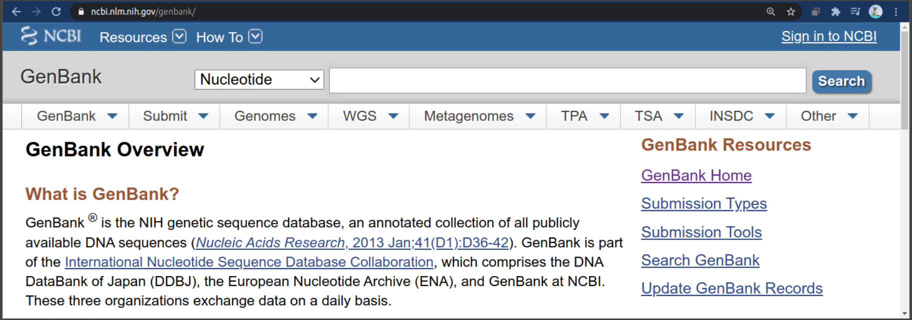
- Complete reference genome sequences
- Protein structures
- Genotypes
- SNPs
- Databases (tables)

These typically have a web GUI and an API. The more-commonly used ones can be access directly through R.

Genbank and Entrez

Genbank is an open access database of all publicly available nucleotide sequences and their protein translations.

- Maintained by NCBI.
- www.ncbi.nlm.nih.gov/genbank
- Entrez is its search and retrieval tool



GenBank Overview

What is GenBank?

GenBank ® is the NIH genetic sequence database, an annotated collection of all publicly available DNA sequences (*Nucleic Acids Research*, 2013 Jan;41(D1):D36-42). GenBank is part of the International Nucleotide Sequence Database Collaboration, which comprises the DNA DataBank of Japan (DDBJ), the European Nucleotide Archive (ENA), and GenBank at NCBI. These three organizations exchange data on a daily basis.

GenBank Resources

GenBank Home

Submission Types

Submission Tools

Search GenBank

Update GenBank Records

Accessing Entrez using R

There are several packages you can use to access GenBank. We will use 'rentrez'.

- Hosted on CRAN, rather than Bioconductor.
- The 'entrez_dbs' function lists all of the available databases.
- The functions 'entrez_db_summary', 'entrez_db_searchable', 'entrez_db_links' give more information about the individual databases.

```
> install.packages('rentrez')
>
> library(rentrez)
>
> entrez_dbs()
[1] "pubmed" "protein" "nuccore" "ipg"
[5] "nucleotide" "structure" "genome" "annotinfo''
[9] "assembly" "bioproject" "biosample" "blastdbinfo"
[13] "books" "cdd" "clinvar" "gap"
[17] "gapplus" "grasp" "dbvar" "gene"
[21] "gds" "geoprofiles" "homologene" "medgen"
[25] "mesh" "nlmcatalog" "omim" "orgtrack"
[29] "pmc" "popset" "proteinclusters" "pcassay"
[33] "protfam" "pccompound" "pcsubstance" "seqannot"
[37] "snp" "sra" "taxonomy" "biocollections"
[41] "gtr"
>
```

Accessing Entrez using R, continued

We use the 'entrez_search' function to search a database. It returns

- the list of record ids.
- the total number of ids available.
- what was actually searched for.
- other information.

Notice how the server understood what was meant by "E. coli", and gave the proper name as well.

```
>
> output <- entrez_search(db = 'nucleotide',
+     term = 'E. coli', retmax = 13)
>
> str(output)
List of 5
$ids : chr [1:13] "2430506574" "2430506518" ...
$count : int 20002297
$retmax : int 13
$QueryTranslation: chr "Escherichia coli [Organism]
OR E. coli[All Fields]"
$file :Classes 'XMLInternalDocument', 'XMLAbstract-
Document' <externalptr>
- attr(*, "class")= chr [1:2] "esearch" "list"
>
```

Refining our search

We can refine our search by filtering by some of the many search terms.

- We can see what's available using 'entrez_db_searchable'.
- The resulting terms can be used as part of the search string.

```
> entrez_db_searchable('nucleotide')
Searchable fields for database 'nuccore'
ALL All terms from all searchable fields
UID Unique number assigned to each sequence
FILT Limits the records
WORD Free text associated with record
TITL Words in definition line
KYWD Nonstandardized terms provided by submitter
:
DIV Division
STRN Strain
ISOL Isolate
CULT Cultivar
BRD Breed
BIOS BioSample
>
```

Refining our search, continued

We can use the search terms to refine our search.

- We put search particular search strings in square brackets.
- We can use boolean operators and parentheses to control the search.

We now have far fewer hits.

```
>
> output <- entrez_search(db = 'nucleotide',
+    term = "(E. coli [ORGN]) AND (public [ACS])
+    AND (shotgun [TITL])", retmax = 13)
>
> str(output)
List of 5
$ids : chr [1:13] "2428765475" "2428765474" ...
$count : int 683163
$retmax : int 13
$QueryTranslation: chr ""Escherichia coli" [Organ-
ism] AND public[All Fields] AND shotgun[TITL]"
$file :Classes 'XMLInternalDocument', 'XMLAbstract-
Document' <externalptr>
- attr(*, "class")= chr [1:2] "esearch" "list"
>
```

Refining our search, continued more

Suppose we don't want the genetic sequence, but are interested in other information.

- Request the formal to be 'xml'.
- The result is XML, which is a mess to read. Instead, convert the XML into a standard R list.
- All manner of information about the associated genetic sequence is now available.

This allows you to collect the meta-data associated with the sequence in question.

```
> result2 = entrez_fetch(db = 'nucleotide',
+   id = output$ids[1], rettype = 'xml',
+   parsed = TRUE)
>
> xml.result = XML::xmlToList(result2)
>
> xml.result$GBSeq$GBSeq_source
[1] "Escherichia coli"
>
> xml.result$GBSeq$GBSeq_definition
[1] "Escherichia coli strain 1815942, whole
genome shotgun sequencing project"
>
> xml.result$GBSeq$GBSeq_taxonomy
[1] "Bacteria; Proteobacteria; Gammapro-
teobacteria; Enterobacterales; Enterobacteri-
aceae; Escherichia"
```

Downloading sequences

We use 'entrez_fetch' to get the sequence associated with a specific database entry. We must specify

- the database the record is coming from,
- the database id,
- the format we want.

The result is the entire complete genome for some strain of E. coli, in FASTA format.

```
>
> output <- entrez_search(db = 'nucleotide',
+     term = "E. coli", retmax = 13)
>
> result <- entrez_fetch(db = 'nucleotide',
+     id = output$ids[1], rettype = 'fasta')
>
> str(result)
chr ">NZ_JAPQWA010000002.1 Citrobacter freundii
strain BAU_132-2 2, whole genome shotgun se-
quence\nACGAAGTCAAGCATTCA"| __truncated__
>
```

Downloading sequences, continued

The string that we get back is the full FASTA string, including the header. We're generally not interested in the header, so here is a quick function to remove it.

This string can then be dropped into the DNAString function to put it into the correct format. Note that this will only work for a single sequence within the string.

```
>
> clean.fasta <- function(full.fasta.string) {
+    temp <- strsplit(full.fasta.string, '\n')[[1]]
+    temp2 <- temp[-1]
+    return(paste(temp2, collapse = ''))
+ }
>
> str(result)
chr ">NZ_JAPQWA010000002.1 Citrobacter freundii
strain BAU_132-2 2, whole genome shotgun se-
quence\nACGAAGTCAAGCATTCA"| __truncated__
>
> str(clean.fasta(result))
chr "ACGAAGTCAAGCATTCAATTTCGCTTTCTTCGCCGGAGTTCT
CAGTGGAACCCCGCTGACCCGGCGGCTTGTAATCCGTTGTTCCGTGT
CAGTGGAGGCGCATTATAGGGA"| __truncated__
>
```

Downloading sequences, a null result

Occasionally, you may notice that you don't get a result when you request it.

I haven't yet figured out the cause of this failure. It seems to be more-common with sequences that have only been uploaded in the past week or so.

```
>
> entrez_fetch(db = "nucleotide",
+     id = "2449582413", rettype="fasta")
[1] "\n"
>
```

Alphabet frequency

We can get a frequency analysis of the letters in the sequence.

The 'as.prob' flag indicates that we're interested in the fraction that each letter is.

```
>
> dna <- DNAString(clean.fasta(result))
>
> dna
761731-letter DNAString object
seq:  ACGAAGTCAAGCATTCAATTTCGCTTTCTTCGC-
CGG...GTGCGCTCTACCAACTGAGCCATATCAGCACACTAA
>
> alphabetFrequency(dna, as.prob = TRUE)
        A         C         G         T M R
0.2429558 0.2715998 0.2497968 0.2356475 0 0
        W         S         Y         K V H
        0         0         0         0 0 0
        D         B         N         - + .
        0         0         0         0 0 0
>
```

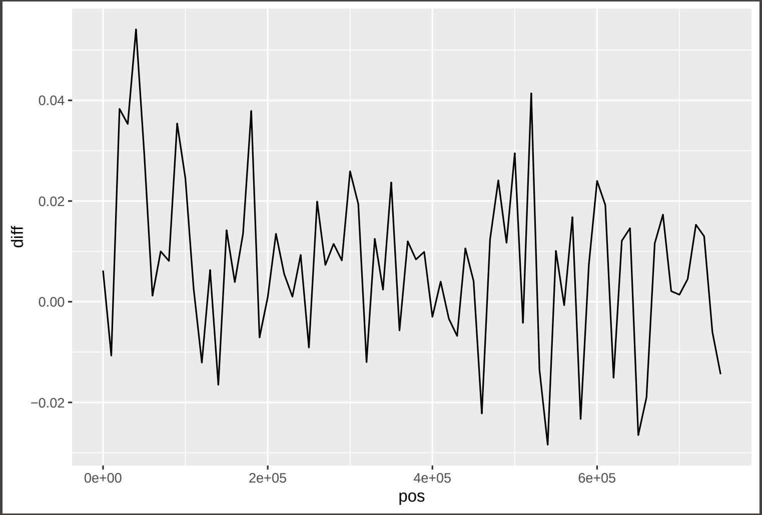
Alphabet sliding frequency

We can get a frequency analysis of the letters in a sliding window as well.

Here we get the proportion of A and T in sliding windows of length 10000.

```
>
> sliding <- letterFrequencyInSlidingView(dna,
+   view.width = 10000, letters = c('A', 'T'),
+   as.prob = T)
>
> ind <- seq(1, nrow(sliding), 10000)
>
> sliding.df <- as.data.frame(sliding[ind,])
> sliding.df$pos <- ind
> sliding.df$diff <- sliding.df$A - sliding.df$T
>
> library(ggplot2)
> ggplot(sliding2) + geom_path(aes(x = pos, y = diff))
>
```

Alphabet sliding frequency, continued



Searching for patterns

We can search for specific patterns within the sequence. This can be useful for searching for start codons, and other important sequences.

You can also specify partial matches, using the 'max.mismatch' and 'with.indels' flags.

The countPattern function is used to get just the counts.

```
>
> matchPattern("AAAAA", dna)
Views on a 761731-letter DNAString subject
subject:  ACGAAGTCAAGCATTCAATT...AGCCATATCAGCACACTAA
views:
start end width
[1] 133 137 5 [AAAAA]
[2] 141 145 5 [AAAAA]
[3] 142 146 5 [AAAAA]
[4] 202 206 5 [AAAAA]
[5] 5027 5031 5 [AAAAA]
...  ...  ...  ...  ...
[1838] 760607 760611 5 [AAAAA]
[1839] 760733 760737 5 [AAAAA]
[1840] 761009 761013 5 [AAAAA]
[1841] 761010 761014 5 [AAAAA]
[1842] 761643 761647 5 [AAAAA]
>
```

Summary

Some of the topics we covered today.

- If install.packages doesn't work for you, the package you are looking for might be hosted at bioconductor.org.
- The Biostrings package contains much of the functionality you need to so sequence analysis and manipulation.
- We can use the rentrez package to directly access the Genbank (NCBI) genetic sequence database.
- This allows us to perform searches and download sequences of interest, directly from R.