Reproducible Research 1: GC % calculation

Simon Rayner

9/24/2021

Introduction

We want to calculate GC content for a set of sequences loaded from a *FASTA* file

This is a simple text file that has the format

```
>HeaderLine1
seq1
>HeaderLine2
seq2
```

for example

```
>MySeq1
ACGT
>MySeq2
TGCA
```

Filepaths

For simplicity, we have just hardcoded the filepath in the program The following command joins the folder containing the file and the file name.

```
fpMiRBaseFA<-file.path("/Users/simonray/DropboxUiO/teaching2021final/lecture1__ReproducibleResearch/data
paste("using file.path gives us <", fpMiRBaseFA, ">", sep="")
```

```
## [1] "using file.path gives us </Users/simonray/DropboxUiO/teaching2021final/lecture1__ReproducibleRes
```

```
ppMiRBaseFA<-paste0("/Users/simonray/DropboxUiO/teaching2021final/lecture1__ReproducibleResearch/data/"
paste0("using paste gives us <", ppMiRBaseFA, ">")
```

```
## [1] "using paste gives us </Users/simonray/DropboxUiO/teaching2021final/lecture1__ReproducibleResear
```

If you downloaded the file from Canvas, you only need to change the folder name, i.e. (`/Users/simonray/Dropbox/teaching202`
to wherever you saved the file

as a simple demo, we are going to calculate the GC content of *human* precursor hairpin sequences that produce microRNAs, or miRNAs. miRNAs are short RNA sequences that play an important role in post transcriptional gene regulation. They function by binding to the 3'UTR of their targets. There are many different ways to do this in R. Two examples are given:

Using the `seqinr` package

```
suppressPackageStartupMessages(library(seqinr))

# we need this package for string filtering
library(data.table)
```

```r
# (i) load fasta file, (ii) filter human miRNAs (they have 'hsa' in the header line) (iii) calc GC%
allMiRsS <- read.fasta(file = fpMiRBaseFA)
humansMiRsS<-allMiRsS[names(allMiRsS) %like% "hsa"]
gc <- function(x){
GC(x)
}

sprintf("from seqinr, GC average is %.2f%%", 100*mean(sapply(humansMiRsS, gc)))
```

```
## [1] "from seqinr, GC average is 68.69%"
```

Using the `Biostrings` package

```r
# Biostrings version
suppressPackageStartupMessages(library(Biostrings))

# (i) load fasta file, (ii) filter human miRNAs (they have 'hsa' in the header line) (iii) calc GC%
allMiRsB = readRNAStringSet(fpMiRBaseFA)
humanMiRsB<-allMiRsB[names(allMiRsB) %like% "hsa"]
af<-alphabetFrequency(humanMiRsB, baseOnly=TRUE, as.prob=TRUE)
p<-(colSums(af)/nrow(af))["C"] + (colSums(af)/nrow(af))["G"]
sprintf("from Biostrings, GC average is %.2f%%", 100*p)
```

```
## [1] "from Biostrings, GC average is 51.39%"
```

Why are we getting two different results? To figure this out, we should first figure out which calculation is correct

However, since operations are vectorised, this makes things a bit tricky

```r
paste0("we are analyzing <", length(humanMiRsB), "> miRs")
```

```
## [1] "we are analyzing <1917> miRs"
```

i.e., there are too many sequences to calculate by hand. So, let's create a test set

```r
test1 <-DNAStringSet(DNAString("TTAACCGG"))
af1<-alphabetFrequency(test1, baseOnly=TRUE, as.prob=TRUE)
p1<-(colSums(af1)/nrow(af1))["C"] + (colSums(af1)/nrow(af1))["G"]
sprintf("from Biostrings, GC average for test1 is %.2f%%", 100*p1)
```

```
## [1] "from Biostrings, GC average for test1 is 50.00%"
```

```r
test2 <-DNAStringSet(DNAString("GGCCGGCC"))
af2<-alphabetFrequency(test2, baseOnly=TRUE, as.prob=TRUE)
p2<-(colSums(af2)/nrow(af2))["C"] + (colSums(af2)/nrow(af2))["G"]
sprintf("from Biostrings, GC average for test2 is %.2f%%", 100*p2)
```

```
## [1] "from Biostrings, GC average for test2 is 100.00%"
```

So, Biostrings seems to be calculating correctly. Let's try rerunning the original code with a smaller test file

```r
miRBaseFA_050<-file.path("/Users/simonray/DropboxUiO/teaching2021/lecture1_dataintro/data", "GC050.fa")
miRBaseFA_050
```

```
## [1] "/Users/simonray/DropboxUiO/teaching2021/lecture1_dataintro/data/GC050.fa"
```

```r
allMiRsB2 = readRNAStringSet(miRBaseFA_050)
```

```
## Warning in .Call2("fasta_index", filexp_list, nrec, skip, seek.first.rec, :
## reading FASTA file /Users/simonray/DropboxUiO/teaching2021/lecture1_dataintro/
## data/GC050.fa: ignored 2 invalid one-letter sequence codes
```

```
humanMiRsB2<-allMiRsB2[names(allMiRsB2) %like% "hsa"]
af<-alphabetFrequency(humanMiRsB2, baseOnly=TRUE, as.prob=TRUE)
p<-(colSums(af)/nrow(af))["C"] + (colSums(af)/nrow(af))["G"]
sprintf("from Biostrings, GC 50%% average is %.2f%%", 100*p)
```

```
## [1] "from Biostrings, GC 50% average is 66.67%"
```

```
miRBaseFA_100<-file.path("/Users/simonray/DropboxUiO/teaching2021/lecture1_dataintro/data", "GC100.fa")
miRBaseFA_100
```

```
## [1] "/Users/simonray/DropboxUiO/teaching2021/lecture1_dataintro/data/GC100.fa"
```

```
allMiRsB2 = readRNAStringSet(miRBaseFA_100)
humanMiRsB2<-allMiRsB2[names(allMiRsB2) %like% "hsa"]
af<-alphabetFrequency(humanMiRsB2, baseOnly=TRUE, as.prob=TRUE)
p<-(colSums(af)/nrow(af))["C"] + (colSums(af)/nrow(af))["G"]
sprintf("from Biostrings, GC 100%% average is %.2f%%", 100*p)
```

```
## [1] "from Biostrings, GC 100% average is 100.00%"
```

When trying to debug your code, it's better to try and work with a test data set that can investigate the specific error your are trying find.