

# Sequences

Sonali Arora \*

27-28 February 2014

## Contents

---

|   |          |
|---|----------|
| <b>1 Introduction to Bioconductor Classes and objects for String manipulation</b> | <b>1</b> |
| 1.1 Containers for sequences  | 1        |
| 1.2 Basic Manipulations   | 3        |
| 1.3 Pre-defined constants   | 3        |
| 1.4 Frequencies   | 4        |
| 1.5 Bioconductor has data packages for your favourite organism                    | 5        |
| <b>2 Session Info</b>   | <b>6</b> |

## 1 Introduction to Bioconductor Classes and objects for String manipulation

---

Aim of this section

- get familiar with various containers for sequences
- read and display sequences from a FASTA file
- simple manipulations on sequences stores in a FASTA file such as `reverse()`, `reverseComplement()`, `translate()`
- calculate gc content

```
library(Biostrings)
library(ShortRead)
```

### 1.1 Containers for sequences

Bioconductor has various classes for storing sequences. You can find out the possible containers using:

```
showMethods(complement)
## Function: complement (package Biostrings)
## x="DNASTring"
## x="DNASTringSet"
## x="MaskedDNASTring"
## x="MaskedRNASTring"
```

---

\*[sarora@fhcrc.org](mailto:sarora@fhcrc.org)

```
## x="RNAString"
## x="RNAStringSet"
## x="XStringViews"
```

A quick example for each of them is:

```
b <- BString("I store any set of characters!" )
d <- DNASTring("GCATAT-TAC") # Creates DNASTring object.
r <- RNAString("GCAUAU-UAC") # Creates RNAString object.
r <- RNAString(d) # Converts d into RNAString object.
p <- AAString("HCWYHH")
```

Lets look at how you can use these on a daily basis:

You're studying Breast Cancer genes -BRCA1 and BRCA2 - Inherited mutations in BRCA1 and BRCA2, confer increased lifetime risk of developing breast or ovarian cancer

We want to do the following:

For BRCA1: We can learn more about the gene at - <http://www.ncbi.nlm.nih.gov/gene/672> We can download the FASTA sequence as a file at: [http://www.ncbi.nlm.nih.gov/nuccore/NC\\_000017.11?report=fasta&from=43032116&to=43137660&strand=true](http://www.ncbi.nlm.nih.gov/nuccore/NC_000017.11?report=fasta&from=43032116&to=43137660&strand=true)

Similarly, for BRCA2 , We can learn more about the gene at <http://www.ncbi.nlm.nih.gov/gene/675> we can get the FASTA sequence from : [http://www.ncbi.nlm.nih.gov/nuccore/NC\\_000013.11?report=fasta&from=32302850&to=32412300](http://www.ncbi.nlm.nih.gov/nuccore/NC_000013.11?report=fasta&from=32302850&to=32412300)

These files are already saved for you and you can access them using

```
fls <- list.files(system.file("extdata", package="BiocIntro"),
                 pattern = ".txt",full=TRUE)
fls

## [1] "/tmp/RtmpKcoIsX/Rinst2338456f7a12/BiocIntro/extdata/brca1_cds.txt"
## [2] "/tmp/RtmpKcoIsX/Rinst2338456f7a12/BiocIntro/extdata/brca2_cds.txt"
```

So lets begin by reading them into an R session using DNASTringSet, a container for storing DNASTring objects. For this we use the readFASTA from the ShortReads package in Bioconductor. readFasta reads all FASTA-formated files stored in fls. It returns a DNASTringSet containing sequences and qualities contained in the given file. We will then use sread from the ShortRead package to create a DNASTringSet and diaply the sequence for one of the genes in a nice, user fiendly format.

```
#Approach- 1
# read in the FATSFA file
seq <- readFasta(flS)

##Lets create a DNASTringSet which is a container for storing a set of DNASTring
dna <- sread(seq)

#Approach-2
dna <- readDNASTringSet(flS)

# let us look at the first DNASTring, brca1 stored at [1]
# This [[]] operation converts a DNASTringSet to DNASTring
brca1 <- dna[[1]]
brca2 <- dna[[2]]

# making the output more understandable.
```

```
# A sequence in FASTA format is represented as a series of lines, each of which
# usually do not exceed 80 characters.
successiveViews(brca1, width=rep(50,length(dna[[1]])/50+1))

## Views on a 5592-letter DNASTring subject
## subject: ATGGATTTATCTGCTCTTCGCGTTGAAGAAGTACAAAAT...ACCTACCTGATACCCAGATCCCCACAGCCACTACTGA
## views:
##      start  end width
## [1]      1   50   50 [ATGGATTTATCTGCTCTTCGCGTTGAAGAAGTACAAAATGTCATTAATGC]
## [2]     51  100   50 [TATGCAGAAAATCTTAGAGTGTCCCATCTGTCTGGAGTTGATCAAGGAAC]
## [3]    101  150   50 [CTGTCTCCACAAAGTGTGACCACATATTTTGCAAATTTTGCATGCTGAAA]
## [4]    151  200   50 [CTTCTCAACCAGAAGAAAGGGCCTTACAGTGTCTTTATGTAAGAATGA]
## [5]    201  250   50 [TATAACCAAAGGAGCCTACAAGAAAGTACGAGATTTAGTCAACTTGTTG]
## ...    ...   ...   ...
## [108] 5351 5400   50 [TACAGCTGTGTGGTGCTTCTGTGGTGAAGGAGCTTTCATCATTACCCTT]
## [109] 5401 5450   50 [GGCACAGGTGTCCACCCAATTGTGGTTGTGCAGCCAGATGCCTGGACAGA]
## [110] 5451 5500   50 [GGACAATGGCTTCCATGCAATTGGGCAGATGTGTGAGGCACCTGTGGTGA]
## [111] 5501 5550   50 [CCCGAGAGTGGGTGTTGGACAGTGTAGCACTCTACCAGTGCCAGGAGCTG]
## [112] 5551 5600   50 [GACACCTACCTGATACCCAGATCCCCACAGCCACTACTGA      ]
```

By default we show only the top 5 and last 5 in a given View. But we can set options(showHeadLines=Inf) to display everything

```
options(showHeadLines=Inf)
successiveViews(brca1, width=rep(50,length(dna[[1]])/50+1))
```

## 1.2 Basic Manipulations

### Exercise:1

a) Create the complement, the reverse and the reverse and complement sequences for brca1

```
reverse(brca1)
## 5592-letter "DNASTring" instance
## seq: AGTCATCACCGACACCCCTAGACCCCATAGTCCATCCACA...TGTA AACATGAAGAAGTTGCGCTTCTCGTCTATTTAGGTA

complement(brca1)
## 5592-letter "DNASTring" instance
## seq: TACCTAAATAGACGAGAAGCGCAACTTCTTCATGTTTTACA...TGTGGATGACTATGGGGTCTAGGGGTGTCGGTGATGACT

reverseComplement(brca1)
## 5592-letter "DNASTring" instance
## seq: TCAGTAGTGGCTGTGGGGATCTGGGGTATCAGGTAGGTGT...ACATTTTGTACTTCTTCAACGCGAAGAGCAGATAAATCCAT
```

b) Translate your random DNA sequences into proteins.

```
translate(brca1)
## 1864-letter "AAString" instance
## seq: MDLSALRVEEVQNVINAMQKILECPICLELIKEPVSTKCDH...IGQMCEAPVVTREWLDSVALYQCQELDTYLIPQIPHSHY*
```

## 1.3 Pre-defined constants

Bioconductor also has some predefined constants which you can use.

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

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

IUPAC_CODE_MAP
##      A      C      G      T      M      R      W      S      Y      K      V      H
##    "A"    "C"    "G"    "T"    "AC"   "AG"   "AT"   "CG"   "CT"   "GT"   "ACG"  "ACT"
##      D      B      N
##    "AGT"  "CGT"  "ACGT"
```

## 1.4 Frequencies

We can also find out various frequencies for our given GOI, lets look at brca2:

```
# what are the unique letter ?
uniqueLetters(brca2)
## [1] "A" "C" "G" "T"

alphabetFrequency(brca2)
##      A      C      G      T      M      R      W      S      Y      K      V      H      D      B      N      -      +      .
## 3769 1784 1882 2822      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0

alphabetFrequency(brca2, baseOnly=TRUE)
##      A      C      G      T other
## 3769 1784 1882 2822      0

dinucleotideFrequency(brca2)
##      AA      AC      AG      AT      CA      CC      CG      CT      GA      GC      GG      GT      TA      TC      TG      TT
## 1562  554  805  847  774  339   74  597  765  333  313  471  667  558  690  907

trinucleotideFrequency(brca2)
## AAA AAC AAG AAT ACA ACC ACG ACT AGA AGC AGG AGT ATA ATC ATG ATT CAA CAC CAG CAT CCA CCC
## 687 206 334 334 233 110  31 180 336 146 133 190 225 143 216 263 278 106 229 161 161  64
## CCG CCT CGA CGC CGG CGT CTA CTC CTG CTT GAA GAC GAG GAT GCA GCC GCG GCT GGA GGC GGG GGT
##  12 102  22  15  12  25 116 107 184 190 369 102 122 172 146  71  12 104 152  59  36  66
## GTA GTC GTG GTT TAA TAC TAG TAT TCA TCC TCG TCT TGA TGC TGG TGT TTA TTC TTG TTT
##  124  90 109 148 228 140 120 179 234  94  19 211 255 113 132 190 202 218 181 306
```

you can also have oligonucleotideFrequency()

**Exercise:2** Can you find the GC content for BRCA1 and BRCA2? Hint: use alphabetFrequency

Solution:

```
gcContent <-
  function(x)
  {
    alf <- alphabetFrequency(x, as.prob=TRUE)
    sum(alf[c("G","C")])
  }

gcContent(brca1)
```

```
## [1] 0.4122
gcContent(brca2)
## [1] 0.3574
```

## 1.5 Bioconductor has data packages for your favourite organism

### BSgenome Data Packages

- Full genomes stored in Biostrings containers
- Currently 16 organisms supported (Human, Mouse, Worm, Yeast, etc...)
- facilities for supporting new genomes (BSgenomeForge)

**Exercise:3** a. Can you find out the gc content for chromosome 17 ( home of BRCA1) and the gc content for chromosome 13 ( home of BRCA2)

```
library(BSgenome.Hsapiens.UCSC.hg19)
```

```
gcContent(Hsapiens[["chr17"]])
## [1] 0.4363
gcContent(Hsapiens[["chr13"]])
## [1] 0.3198
```

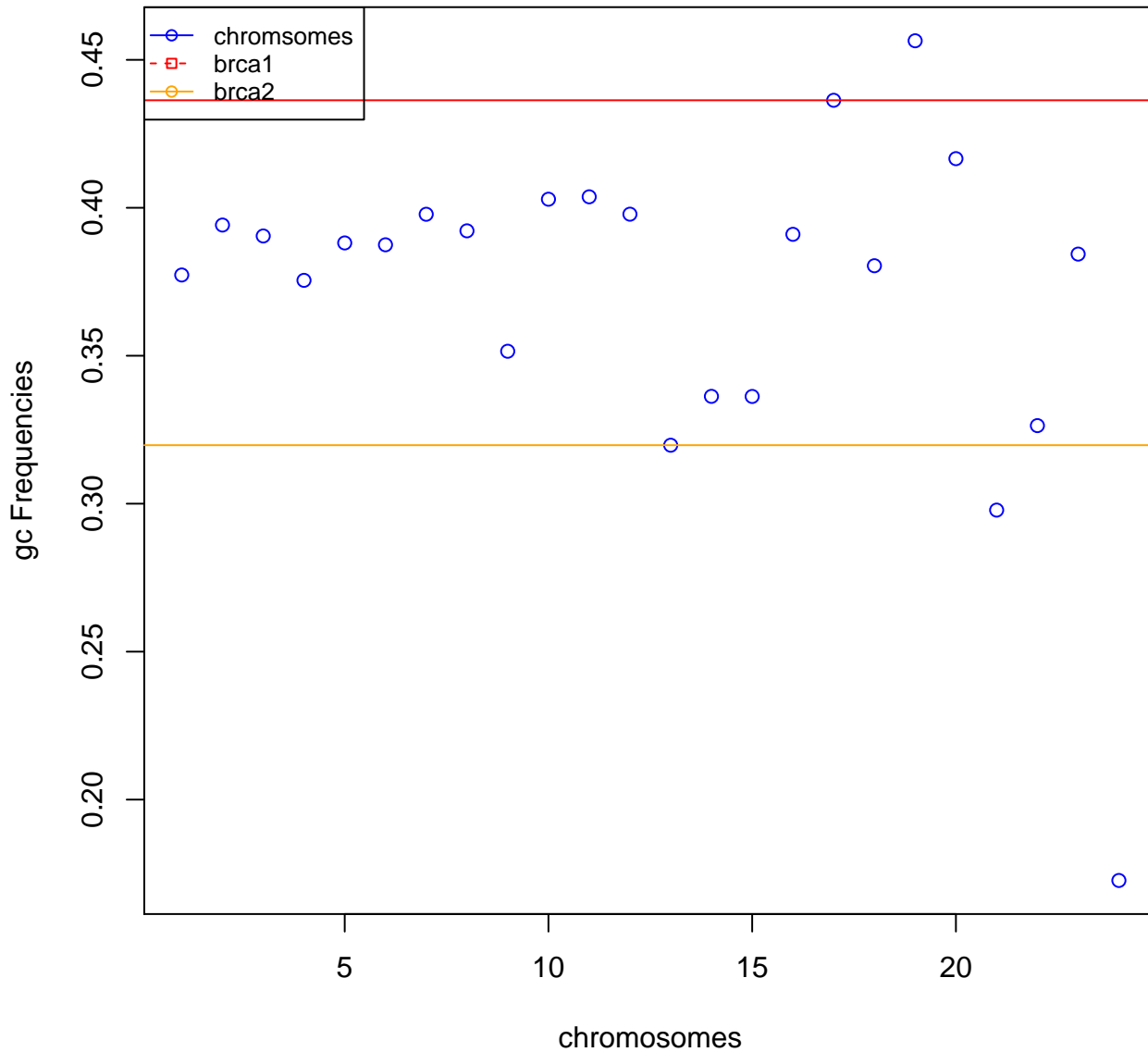
b. Please create a plot of the GC frequencies for all the primary chromosomes Please superimpose the frequencies of BRCA1 and BRCA2 on this plot Add title , legend and appropriate labels for the axes.

```
chrs <- paste0("chr", c(1:22,"X","Y"))
data <- sapply(chrs, function(x) gcContent(Hsapiens[[x]]))
names(data) <- chrs

plot(data,
      xlab="chromosomes",ylab="gc Frequencies",
      xlim=c(1,24),
      col="blue")
abline(h=gcContent(Hsapiens[["chr17"]]),col="red")
abline(h=gcContent(Hsapiens[["chr13"]]),col="orange")
title(main="gc Frequencies across Human Chromosomes", col.main="blue",
      font.main=4)

legend("topleft",c("chromosomes","brca1","brca2"), cex=0.8,
      col=c("blue","red","orange"), pch=21:22, lty=1:2)
```

### *gc Frequencies across Human Chromosomes*



## 2 Session Info

---

Here is the output of `sessionInfo` on the system on which this document was compiled:

```
sessionInfo()
## R Under development (unstable) (2014-02-23 r65064)
## Platform: x86_64-unknown-linux-gnu (64-bit)
##
## locale:
```

```

## [1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C          LC_TIME=en_US.UTF-8
## [4] LC_COLLATE=C                LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
## [7] LC_PAPER=en_US.UTF-8       LC_NAME=C             LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel stats      graphics grDevices utils      datasets methods base
##
## other attached packages:
## [1] quantreg_5.05           SparseM_1.03
## [3] BiocIntro_0.0.4        rtracklayer_1.23.12
## [5] AnnotationHub_1.3.18   BSgenome.Hsapiens.UCSC.hg19_1.3.99
## [7] BSgenome_1.31.11      TxDb.Hsapiens.UCSC.hg19.knownGene_2.10.1
## [9] GenomicFeatures_1.15.7 org.Hs.eg.db_2.10.1
## [11] RSQLite_0.11.4        DBI_0.2-7
## [13] AnnotationDbi_1.25.9   Biobase_2.23.5
## [15] RNAseqData.HNRNPC.bam.chr14_0.1.6 ggplot2_0.9.3.1
## [17] VariantAnnotation_1.9.41 ShortRead_1.21.14
## [19] GenomicAlignments_0.99.24 Rsamtools_1.15.29
## [21] GenomicRanges_1.15.31 Biostrings_2.31.14
## [23] XVector_0.3.7         IRanges_1.21.32
## [25] BiocParallel_0.5.14   BiocGenerics_0.9.3
## [27] knitr_1.5
##
## loaded via a namespace (and not attached):
## [1] BBmisc_1.5             BatchJobs_1.2         BiocInstaller_1.13.3
## [4] BiocStyle_1.1.17      Category_2.29.1       GSEABase_1.25.2
## [7] MASS_7.3-29          Matrix_1.1-2         RBGL_1.39.1
## [10] RColorBrewer_1.0-5    RCurl_1.95-4.1       RJSONIO_1.0-3
## [13] Rcpp_0.11.0           XML_3.98-1.1         annotate_1.41.1
## [16] biomaRt_2.19.3       bitops_1.0-6         brew_1.0-6
## [19] codetools_0.2-8      colorspace_1.2-4     dichromat_2.0-0
## [22] digest_0.6.4         evaluate_0.5.1       fail_1.2
## [25] foreach_1.4.1        formatR_0.10         genefilter_1.45.1
## [28] graph_1.41.2         grid_3.1.0           gridSVG_1.4-0
## [31] gtable_0.1.2         highr_0.3            httpuv_1.2.3
## [34] httr_0.2             hwriter_1.3          interactiveDisplay_1.0.30
## [37] iterators_1.0.6      labeling_0.2         lattice_0.20-24
## [40] latticeExtra_0.6-26  munsell_0.4.2        plyr_1.8
## [43] proto_0.3-10         reshape2_1.2.2       rjson_0.2.13
## [46] scales_0.2.3         sendmailR_1.1-2      shiny_0.8.0.99
## [49] splines_3.1.0        stats4_3.1.0         stringr_0.6.2
## [52] survival_2.37-7     tools_3.1.0          xtable_1.7-1
## [55] zlibbioc_1.9.0

```