BIO214 Project1

Table of Contents

## 1. Project Description.

### Assessment for BIO214 in General:

Due to the change of the school’s returning arrangement, the assessments for BIO214 is altered into a contingency plan. Currently, BIO214 will be assessed by 5 programming projects, and each one will contribute to 20% of the final mark.

All of the programming projects in BIO214 will be assessed by the following two criteria:

1. The correctness of the computational output & figures (50%)
2. The quality of the written answers for the short answer question (50%)

If a question demands a short written answer, it will be clearly labeled in the assignment document by SAQ + number, and you should type your answer on the line below it. For all the coding and biological questions raised in the assignments, you are encouraged to solve them with active online searching.

The projects are expected to be submitted in the form of knitted HTML files of the project Rmarkdown document (end with .Rmd).

For example, after finishing the project on this Rmd file, you can click the knit button on the upper right corner of Rstudio. If your code can run through without any error, you will then get the HTML document under your project directory on the server.

When the project deadline is reached, I will check your project directories on the server for the HTML files, and I will treat the one named by ProjectX.html as your final submission.

Submission using word file is also OK, but please remember to include all of the codes and programming outputs in the word file. If you decide to use word files, please send your finished project document to my email: zhen.wei01@xjtlu.edu.cn

### The Aim of Project1:

In project1, your target is to screen for the thermal stable secondary structures on the transcript of human genome assembly hg19. After the predictions are made by RNAfold, you will map the predicted RNA structure from the transcript coordinate into the genome coordinate. Further analysis and statistical tests will be conducted to compare the sequence content of different RNA structure regions on the genome. Please refer to the instructions at the beginning of each question for the detailed project information.

## 2. Load the hg19 Genome and TxDb:

The first step of this project is to load the transcript annotation and genome sequence for human genome assembly hg19.

To achieve this, load the TxDb and BSgenome libraries for hg19 with library(). Then, rename the BSgenome and TxDb object into the R variables: genome and txdb.

Please write your code in the specified area in the code chunk below.

Furthermore, you could choose to refer to the hint comments above the code region, particularly when you find this question is difficult for you.

## === Hint for code, fill the xs ============= ##  
#library(BSgenome.xxxxx.UCSC.xxx)  
#library(TxDb.xxxxx.UCSC.xxx.knownGene)  
#genome <- BSgenome.xxxxxx.UCSC.xxx  
#txdb <- TxDb.xxxxxx.UCSC.xxx.knownGene  
  
## ===== Enter Your code below =============== ##  
library(BSgenome.Hsapiens.UCSC.hg19)  
library(TxDb.Hsapiens.UCSC.hg19.knownGene)  
genome <- BSgenome.Hsapiens.UCSC.hg19  
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene

Check that if ‘txdb’ and ‘genome’ are based on the hg19 assembly:

txdb

## TxDb object:  
## # Db type: TxDb  
## # Supporting package: GenomicFeatures  
## # Data source: UCSC  
## # Genome: hg19  
## # Organism: Homo sapiens  
## # Taxonomy ID: 9606  
## # UCSC Table: knownGene  
## # Resource URL: http://genome.ucsc.edu/  
## # Type of Gene ID: Entrez Gene ID  
## # Full dataset: yes  
## # miRBase build ID: GRCh37  
## # transcript\_nrow: 82960  
## # exon\_nrow: 289969  
## # cds\_nrow: 237533  
## # Db created by: GenomicFeatures package from Bioconductor  
## # Creation time: 2015-10-07 18:11:28 +0000 (Wed, 07 Oct 2015)  
## # GenomicFeatures version at creation time: 1.21.30  
## # RSQLite version at creation time: 1.0.0  
## # DBSCHEMAVERSION: 1.1

genome

## Human genome:  
## # organism: Homo sapiens (Human)  
## # provider: UCSC  
## # provider version: hg19  
## # release date: Feb. 2009  
## # release name: Genome Reference Consortium GRCh37  
## # 93 sequences:  
## # chr1 chr2 chr3   
## # chr4 chr5 chr6   
## # chr7 chr8 chr9   
## # chr10 chr11 chr12   
## # chr13 chr14 chr15   
## # ... ... ...   
## # chrUn\_gl000235 chrUn\_gl000236 chrUn\_gl000237   
## # chrUn\_gl000238 chrUn\_gl000239 chrUn\_gl000240   
## # chrUn\_gl000241 chrUn\_gl000242 chrUn\_gl000243   
## # chrUn\_gl000244 chrUn\_gl000245 chrUn\_gl000246   
## # chrUn\_gl000247 chrUn\_gl000248 chrUn\_gl000249   
## # (use 'seqnames()' to see all the sequence names, use the '$' or '[[' operator  
## # to access a given sequence)

## 3. Extract the Exon Ranges Grouped by Transcript from ‘txdb’:

The next step is to extract the exon regions for the RNA transcripts encoded in the hg19 transcript annotation.

You should use function exonsBy(), and set the argument by = "tx", also, please set the argument use.names = TRUE.

The input for exonsBy() should be the TxDb object for transcript annotation, the output for exonsBy() is a GRangesList of exons grouped by the levels defined by by = Store the output into the R variable named with transcripts.

## === Hint code, fill the xs ============= ##  
#transcripts <- exonsBy(xxx,xxxx,xxxxx)  
## ===== Enter Your code below ============ ##  
transcripts <- exonsBy(txdb, by="tx", use.names =TRUE)

Check the layout of the resulting GRangesList object:

transcripts

## GRangesList object of length 82960:  
## $uc001aaa.3  
## GRanges object with 3 ranges and 3 metadata columns:  
## seqnames ranges strand | exon\_id exon\_name exon\_rank  
## <Rle> <IRanges> <Rle> | <integer> <character> <integer>  
## [1] chr1 11874-12227 + | 1 <NA> 1  
## [2] chr1 12613-12721 + | 3 <NA> 2  
## [3] chr1 13221-14409 + | 5 <NA> 3  
## -------  
## seqinfo: 93 sequences (1 circular) from hg19 genome  
##   
## $uc010nxq.1  
## GRanges object with 3 ranges and 3 metadata columns:  
## seqnames ranges strand | exon\_id exon\_name exon\_rank  
## <Rle> <IRanges> <Rle> | <integer> <character> <integer>  
## [1] chr1 11874-12227 + | 1 <NA> 1  
## [2] chr1 12595-12721 + | 2 <NA> 2  
## [3] chr1 13403-14409 + | 6 <NA> 3  
## -------  
## seqinfo: 93 sequences (1 circular) from hg19 genome  
##   
## $uc010nxr.1  
## GRanges object with 3 ranges and 3 metadata columns:  
## seqnames ranges strand | exon\_id exon\_name exon\_rank  
## <Rle> <IRanges> <Rle> | <integer> <character> <integer>  
## [1] chr1 11874-12227 + | 1 <NA> 1  
## [2] chr1 12646-12697 + | 4 <NA> 2  
## [3] chr1 13221-14409 + | 5 <NA> 3  
## -------  
## seqinfo: 93 sequences (1 circular) from hg19 genome  
##   
## ...  
## <82957 more elements>

You should now find the ranges of exons grouped by transcripts (the list elements).

* SAQ1: Please tell how many transcripts in total are defined in the hg19 TxDb object.
* Your answer: 82960

## 4. Filter the Transcripts by the Chromosome Number and Transcript Lengths:

The computational cost for the RNA 2ndary structures is expensive for long transcripts. Therefore, we need to subset the transcripts GRangelist into the transcripts on chromosome 21 and with transcript width > 100 and <= 500.

P.S. Please use the same variable as before to store the filtered transcripts.

## === Hint code, fill the xs ============= ##  
#transcripts <- transcripts[seqnames(transcripts) == "xxxx"]  
#tx\_width <- sum(width(transcripts))  
#transcripts <- transcripts[xxxxxx]  
## ===== Enter your code below ============ ##  
  
transcripts <- transcripts[seqnames(transcripts) == "chr21"]  
tx\_width <- sum(width(transcripts))  
transcripts <- transcripts[tx\_width>100 & tx\_width<=500]

* SAQ2: Please tell how many transcripts in total satisfy the criteria above.
* Your answer: 69

## 5. Extract the Transcript Sequences from the Genome:

Apply the function extractTranscriptSeqs() to extract the DNA sequences of the mature RNA transcripts from genomes. Please type ?extractTranscriptSeqs() for the usage of the function. The sequence output should be stored into the variable named by tx\_seqs.

## === Hint code, fill the xs ============= ##  
#tx\_seqs <- extractTranscriptSeqs(xxx,xxxxx)  
## ===== Enter your code below ============ ##  
tx\_seqs <- extractTranscriptSeqs(Hsapiens,transcripts)

Check the outlook of tx\_seqs:

tx\_seqs

## A DNAStringSet instance of length 69  
## width seq names   
## [1] 180 CGCGACTGCGGCGGCGGTGGTGG...CCCGCCGGCCGCCTTTCTCGCG uc021wgv.1  
## [2] 477 GTCTGAATTTTTTTTGCTTATTA...AATATTCTTGGTCATTCAGCAG uc021whv.1  
## [3] 109 GTCTATGGCCATACCACCCTGAA...AGAAAAAAAGTTGATGTAAATA uc021whx.1  
## [4] 312 TAATAACATCTACCAAACAGTTT...TGTGAAGATGAAGGTAGAAATT uc021wid.1  
## [5] 483 ATGTCCTACAACTGCTGCTCTAG...CTTCTATCAATTCACCTGCTAA uc011acw.2  
## ... ... ...  
## [65] 273 ATCCTACGCTATGGAGTACAGAC...CACCCCACGCCCCAGTCGAAAG uc010gny.1  
## [66] 492 ATCCCTGACTCGGGGTCGCCTTT...AATTAAAAGAGATCGATATTAA uc002zax.1  
## [67] 103 GAGATGCTGAGCAGCAGAAGTGC...AGAATCAGAGGATGATCTGGGA uc002zfd.3  
## [68] 447 AGACCAGCCCTGTCCTCTGCGCC...GCCAGCAGGCTCTCTGAACTCC uc002zfs.1  
## [69] 339 TCAAGTCTTTCCGAGGGGCCAGT...GATTTTCACAGGTATCCAGCTT uc031rwc.1

tx\_seqs$uc002ytg.1

## 418-letter "DNAString" instance  
## seq: GGTGAAAAGGTGGAGGGGCTACAAGCTCAAGCCCTA...GATTTGCACCTGATATTCAGGGATCACTCTCAAAAT

as.character(tx\_seqs$uc002ytg.1)

## [1] "GGTGAAAAGGTGGAGGGGCTACAAGCTCAAGCCCTATATCCTTGGAGAGCCAAAAAAGACAACCACTTAAATTTTAACAAAAATGATGTCATCACCGTCCTGGAACAGCAAGACATGTGGTGGTTTGGAGAAGTTCAAGGTCAGAAGGGTTGGTTCCCCAAGTCTTACGTGAAACTCATTTCAGGGCCCATAAGGAAGTCTACAAGCATGGATTCTGGTTCTTCAGAGAGTCCTGCTAGTCTAAAGCGAGTAGCCTCTCCAGCAGCCAAGCCGGTCGTTTCGGGAGAAGGGCTCTGGAACTGCTGGGAAAACAGGGAGTTTAGGAAAAAAACCTGGTCCCTCCTGTAACGTGAAATAAAAAGAAAGAAATTTGCCCTGCCATGATTTGCACCTGATATTCAGGGATCACTCTCAAAAT"

* SAQ3: Please paste the DNA sequence for the transcript uc002ytg.1 below.
* Your answer: “GGTGAAAAGGTGGAGGGGCTACAAGCTCAAGCCCTATATCCTTGGAGAGCCAAAAAAGACAACCACTTAAATTTTAACAAAAATGATGTCATCACCGTCCTGGAACAGCAAGACATGTGGTGGTTTGGAGAAGTTCAAGGTCAGAAGGGTTGGTTCCCCAAGTCTTACGTGAAACTCATTTCAGGGCCCATAAGGAAGTCTACAAGCATGGATTCTGGTTCTTCAGAGAGTCCTGCTAGTCTAAAGCGAGTAGCCTCTCCAGCAGCCAAGCCGGTCGTTTCGGGAGAAGGGCTCTGGAACTGCTGGGAAAACAGGGAGTTTAGGAAAAAAACCTGGTCCCTCCTGTAACGTGAAATAAAAAGAAAGAAATTTGCCCTGCCATGATTTGCACCTGATATTCAGGGATCACTCTCAAAAT”

## 6. Save the Sequences in the Disk as fasta File:

Now, save the DNAStringSet object of variable tx\_seqs as a FASTA file on your disk, the FASTA file should be named by tx\_seqs.fasta.

This step is easy since the export of DNAStringSet is supported by the function writeXStringSet().

## === Hint code, fill the xs ============= ##  
#writeXStringSet(xxx,xxxx)  
## ===== Enter your code below ============ ##  
writeXStringSet(tx\_seqs,"tx\_seqs.fasta")

* SAQ4: Please report the first 4 lines of the tx\_seqs.fasta file. Please try to explain the components of the observed format.
* Your answer: uc021wgv.1 CGCGACTGCGGCGGCGGTGGTGGGGGGAGCCGCGGGGATCGCCGAGGGCCGGTCGGCCGCCCCGGGTGCCGCGCGGTGCC GCCGGCGGCGGTGAGGCCCCGCGCGTGTGTCCCGGCTGCGGTCGGCCGCGCTCGAGGGGTCCCCGTGGCGTCCCCTTCCC CGCCGGCCGCCTTTCTCGCG Explanation: The first line is the name of transcript and the corresponding sequence is followed by, composed of A,T,C,G.

## 7. Run RNAfold for the fasta File with Bash Command on the Linux System:

RNAfold is a command-line tool in the Vienna RNA package to predict RNA secondary structures with RNA sequences. Hence, for this question, you need to call the Linux bash command within R.

RNAfold can be downloaded at: <https://www.tbi.univie.ac.at/RNA/>, and you could install it on your own computer. But by this time, on our Linux server, RNAfold has already been installed.

Please run RNAfold with the previously saved FASTA file tx\_seq.fasta.

Set the temperature of folding into 70 degrees with argument --temp=70, and then save the standard output of RNAfold into a file named by RNAstructure.txt.

Please notice that in Linux bash, we can channel the output of a bash command into a text file with > mark, such as:

Command --argument=x INPUT.fasta > OUTPUT\_saved.txt

Also, you could call the system command line within R using the system() function.

## === Hint code, fill the xs ============= ##  
#system("xxx --xxxx xxxxx > xxxxxx")  
# RNAfold [OPTIONS] [<input0.fa>] [<input1.fa>]...  
## ===== Enter your code below ============ ##  
system("RNAfold --temp=70 tx\_seqs.fasta > RNAstructure.txt")

* SAQ5: Open the pdf file for the predicted RNA structure of the transcript uc002ytg.1, How many hairpin loops, interior loops, bulge loop, and multi-branch loop are presented in the RNA structure, respectively?
* Your answer: 6 hairpin loops, 7 interior loops, 3 bulge loop, 3 multi-branch loops in the RNA structure respectively
* SAQ6: Report the first 3 lines of the file RNAstructure.txt, please also interpret the format.
* Your answer:

The first 3 lines are: uc021wgv.1 CGCGACUGCGGCGGCGGUGGUGGGGGGAGCCGCGGGGAUCGCCGAGGGCCGGUCGGCCGCCCCGGGUGCCGCGCGGUGCCGCCGGCGGCGGUGAGGCCCCGCGCGUGUGUCCCGGCUGCGGUCGGCCGCGCUCGAGGGGUCCCCGUGGCGUCCCCUUCCCCGCCGGCCGCCUUUCUCGCG (((((..((((((((((.((.((((((.((((((((((((.((..(((((((((((((((.(((((.(((((((((.(((((((….))))..))).))))))).))..)))))..))))))))))).))))..)))))))))))))).)))))).))))))).)))))…..))))) (-79.28) The first line is the name of transcript and the second line is the sequence of that sequence, composed of A,G,C,U. The last line of the text output contains the predicted MFE structure as dot-bracket notation and its free energy -79.28 kcal/mol. A dot in the dot-bracket notation represents an unpaired position, while a base pair (i, j) is represented by a pair of matching parentheses at position i and j.

## 8. Read the output of RNAfold into GRanges:

The next step is to read the predicted RNA structures, which are stored using parentheses representation, into R. The basic idea for the code below is to match for the “(” or “.” in the output file of RNAfold. Then, for hybridized and non-hybridized regions, it will construct the string matching results into GRanges object on the transcript coordinate.

Please fill in the missing portion of the following code chunk to make it run correctly.

## Read the output file into R:  
RNAfold\_out <- readLines("RNAstructure.txt")  
Struc\_pred <- RNAfold\_out[seq(3, by = 3, length.out = length(tx\_seqs))]  
  
## Remove the energy scores attached at the end:  
Struc\_pred <- gsub(" .\*", "", Struc\_pred)  
Struc\_pred <- BStringSet(Struc\_pred)  
names(Struc\_pred) <- names(tx\_seqs)  
Struc\_pred

## A BStringSet instance of length 69  
## width seq names   
## [1] 180 (((((..((((((((((.((.((...)))))).))))).....))))) uc021wgv.1  
## [2] 477 ..............((((................)))))))..)))). uc021whv.1  
## [3] 109 ............((((.(((((.......................... uc021whx.1  
## [4] 312 ........((((((......(((...))))))......)))))).... uc021wid.1  
## [5] 483 ..........(((((((((((((......................... uc011acw.2  
## ... ... ...  
## [65] 273 .......((..(((.(((((......)...))).))............ uc010gny.1  
## [66] 492 ............((((((.............................. uc002zax.1  
## [67] 103 ......((((....((((.((((.......)))).............. uc002zfd.3  
## [68] 447 .............((((.(((((...(((....)))).)))))..... uc002zfs.1  
## [69] 339 .........((((((...((((....)..................... uc031rwc.1

## Construct the GRanges object for hybridized and nonhybridized regions:  
  
## === Hint code, fill the xs ============= ##  
#Hyb\_irl <- lapply(vmatchPattern("xxx",Struc\_pred), reduce)  
#nonHyb\_irl <- lapply(Hyb\_irl, xxxx) # Note: the `xxxx` is one of the inter-range method, please see BIO214\_refcard.rmd.  
## ===== Enter your code below ============ ##  
  
nonHyb\_irl <- lapply(vmatchPattern(".",Struc\_pred), reduce)  
Hyb\_irl <- lapply(nonHyb\_irl, gaps)  
  
##===== Your Code is finished till here =====##  
  
##Convert the IrangesList into GRanges  
  
irl2grl <- function(irl) GRangesList( mapply(function(x,y) GRanges(seqnames = y,  
 ranges = x,  
 strand = "\*"),irl,names(irl)) )  
  
Hyb\_gr <- unlist(irl2grl(Hyb\_irl))  
nonHyb\_gr <- unlist(irl2grl(nonHyb\_irl))  
  
Hyb\_gr ##The Granges for Hybridized regions on the transcript

## GRanges object with 1828 ranges and 0 metadata columns:  
## seqnames ranges strand  
## <Rle> <IRanges> <Rle>  
## uc021wgv.1 uc021wgv.1 8-17 \*  
## uc021wgv.1 uc021wgv.1 19-20 \*  
## uc021wgv.1 uc021wgv.1 22-27 \*  
## uc021wgv.1 uc021wgv.1 29-40 \*  
## uc021wgv.1 uc021wgv.1 42-43 \*  
## ... ... ... ...  
## uc031rwc.1 uc031rwc.1 277-279 \*  
## uc031rwc.1 uc031rwc.1 284-286 \*  
## uc031rwc.1 uc031rwc.1 297-299 \*  
## uc031rwc.1 uc031rwc.1 303-306 \*  
## uc031rwc.1 uc031rwc.1 313-318 \*  
## -------  
## seqinfo: 69 sequences from an unspecified genome; no seqlengths

nonHyb\_gr ##The Granges for non-Hybridized (Looped) regions on the transcript

## GRanges object with 1897 ranges and 0 metadata columns:  
## seqnames ranges strand  
## <Rle> <IRanges> <Rle>  
## uc021wgv.1 uc021wgv.1 6-7 \*  
## uc021wgv.1 uc021wgv.1 18 \*  
## uc021wgv.1 uc021wgv.1 21 \*  
## uc021wgv.1 uc021wgv.1 28 \*  
## uc021wgv.1 uc021wgv.1 41 \*  
## ... ... ... ...  
## uc031rwc.1 uc031rwc.1 280-283 \*  
## uc031rwc.1 uc031rwc.1 287-296 \*  
## uc031rwc.1 uc031rwc.1 300-302 \*  
## uc031rwc.1 uc031rwc.1 307-312 \*  
## uc031rwc.1 uc031rwc.1 319-339 \*  
## -------  
## seqinfo: 69 sequences from an unspecified genome; no seqlengths

* SAQ7: How many hybridized regions and loop regions are predicted by RNAfold on those transcripts?
* Your answer: 1828 hybridized regions and 1897 loop regions

## 9. Map from transcript coordinates to genome coordinates:

The next step is to map the GRanges from the transcript coordinates into Genome coordinates.

Whenever we need to do this kind of task in R, we generally rely on 2 powerful Bioconductor functions: mapFromTranscripts and mapToTranscripts. If our input range is on the transcript and the target range is on the genome, we should use mapFromTranscripts, conversely, we should use mapTotranscripts. Please check the help docomentation using ? + function name for the detailed explanations of their usage.

## === Hint code, fill the xs ============= ##  
#Hyb\_gr <- mapxxxTranscripts(xxx,xxxx)  
#nonHyb\_gr <- mapxxxTranscripts(xxx,xxxx)  
## ===== Enter your code below ============ ##  
Hyb\_gr <- mapFromTranscripts(Hyb\_gr,transcripts)  
nonHyb\_gr <- mapFromTranscripts(nonHyb\_gr,transcripts)  
##===== Your Code is finished till here =====##  
Hyb\_gr

## GRanges object with 1828 ranges and 2 metadata columns:  
## seqnames ranges strand | xHits transcriptsHits  
## <Rle> <IRanges> <Rle> | <integer> <integer>  
## uc021wgv.1 chr21 9825839-9825848 + | 1 1  
## uc021wgv.1 chr21 9825850-9825851 + | 2 1  
## uc021wgv.1 chr21 9825853-9825858 + | 3 1  
## uc021wgv.1 chr21 9825860-9825871 + | 4 1  
## uc021wgv.1 chr21 9825873-9825874 + | 5 1  
## ... ... ... ... . ... ...  
## uc031rwc.1 chr21 47705088-47705090 - | 1824 69  
## uc031rwc.1 chr21 47705081-47705083 - | 1825 69  
## uc031rwc.1 chr21 47705068-47705070 - | 1826 69  
## uc031rwc.1 chr21 47705061-47705064 - | 1827 69  
## uc031rwc.1 chr21 47705049-47705054 - | 1828 69  
## -------  
## seqinfo: 93 sequences from an unspecified genome; no seqlengths

nonHyb\_gr

## GRanges object with 1897 ranges and 2 metadata columns:  
## seqnames ranges strand | xHits transcriptsHits  
## <Rle> <IRanges> <Rle> | <integer> <integer>  
## uc021wgv.1 chr21 9825837-9825838 + | 1 1  
## uc021wgv.1 chr21 9825849 + | 2 1  
## uc021wgv.1 chr21 9825852 + | 3 1  
## uc021wgv.1 chr21 9825859 + | 4 1  
## uc021wgv.1 chr21 9825872 + | 5 1  
## ... ... ... ... . ... ...  
## uc031rwc.1 chr21 47705084-47705087 - | 1893 69  
## uc031rwc.1 chr21 47705071-47705080 - | 1894 69  
## uc031rwc.1 chr21 47705065-47705067 - | 1895 69  
## uc031rwc.1 chr21 47705055-47705060 - | 1896 69  
## uc031rwc.1 chr21 47705028-47705048 - | 1897 69  
## -------  
## seqinfo: 93 sequences from an unspecified genome; no seqlengths

* SAQ7: How the seqnames field of the GRanges changed before and after the transcript to genome mapping? please explain the reason behind this change.
* Your answer: the ‘seqnames’ field of the GRanges before the mapping is changed to ‘chromesome’ field. The reason behind this change is that the mapFromTranscripts function uses the seqname and the names of transcripts to determine the mapping pairs

## 10. Extract the GC content under each regions:

In this step, we will extract the sequence content, particularly the GC content, behind the hybridized regions and the looped regions. GC content is defined as the proportion of G and C within a given nucleotide sequence.

Then, construct 2 vectors named by Hyb\_GC and nonHyb\_GC storing the GC contents for the previously predicted hybridized and non-hybridized regions.

You may need the help of the following functions in this step: - Views(): for extracting sequences underneath GRanges on BSgenome object. - DNAStringSet(): for the convertion of the BSgenome views into a set of DNA strings so that we could calculate their sequence content. - letterFrequency():for the calculation of the proportion / frequency of nucleotides within XStringSet.

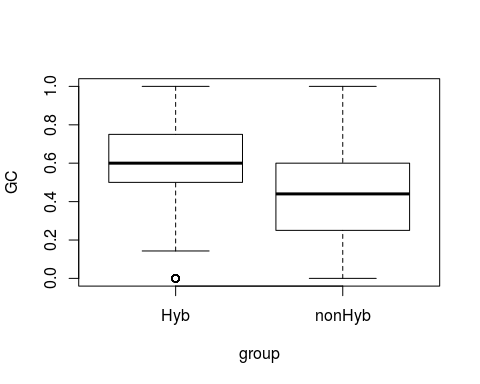
Please check the documentation for these functions with ? to get more clues.

## === Hint code, fill the xs, =========================== ##  
## === The xs might be nested with > 1 functions ========= ##  
#Hyb\_GC <- letterFrequency(xxxxxxx, "xx", as.prob=TRUE)  
#nonHyb\_GC <- letterFrequency(xxxxxxx, "xx", as.prob=TRUE)  
## ===== Enter your code below ========================== ##  
Hyb\_GC <- letterFrequency(DNAStringSet(Views(genome, Hyb\_gr)), "GC", as.prob=TRUE)  
nonHyb\_GC <- letterFrequency(DNAStringSet(Views(genome,nonHyb\_gr)), "GC", as.prob=TRUE)

## 11. Draw a box plot of the GC content stratified by regions:

Apply the R base function boxplot() to draw a box plot based on the constructed data.frame plot\_df.

plot\_df <- data.frame(GC = c(Hyb\_GC, nonHyb\_GC),  
 group = rep(c("Hyb", "nonHyb"),  
 c(length(Hyb\_GC), length(nonHyb\_GC))  
 ))  
## === Hint code, fill the xs === ##  
## boxplot(xx~xxx, data = plot\_df)   
## ====Enter your code below====== ##  
boxplot(GC~group, data = plot\_df)



* SAQ8: What are the median values of GC contents for the hybridized region and non-hybridized region? Please explain why one group is higher than the other from the molecular perspective.
* Your answer: The median values of GC contents for the hybridized region is 0.6. The median values of GC contents for the unhybridized region is 0.44 The reason behind this phenonmenon the frequency of stable secondary structure increases as GC content increases, and the stability of RNA structure has relation with the hybridization situation. The more loops RNA have, the less stable structure of RNA.

## 12. Test the difference in means using t-test and wilcox-test:

Conduct the t.test and wilcox.test to test for the difference of sample means between hybridized GC contents and looped GC contents.

## === Hint code, fill the xs === ##  
## t.test(xx, xxx)  
## wilcox.test(xx, xxx)  
## ====Enter your code below====== ##  
t.test(Hyb\_GC,nonHyb\_GC)

##   
## Welch Two Sample t-test  
##   
## data: Hyb\_GC and nonHyb\_GC  
## t = 18.214, df = 3381.7, p-value < 2.2e-16  
## alternative hypothesis: true difference in means is not equal to 0  
## 95 percent confidence interval:  
## 0.1414246 0.1755459  
## sample estimates:  
## mean of x mean of y   
## 0.5972862 0.4388010

wilcox.test(Hyb\_GC,nonHyb\_GC)

##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: Hyb\_GC and nonHyb\_GC  
## W = 2356799, p-value < 2.2e-16  
## alternative hypothesis: true location shift is not equal to 0

* SAQ9: Please interpret the result of the 2 statistical tests, what can you conclude from them? What is the difference between wilcox.test and t.test? If we have a small sample size for each group, i.e. < 30, which statistical test is more robust?
* Your answer:

1. Both tests reject the null hypothesis since p values are far smaller than 0.05.
2. The critical difference between these tests is that the test from Wilcoxon is a non-parametric test, while the t-test is a parametric test. Meanwhile, t-test requires the population need to be normally distributed, while the wilcoxon test doesn’t require that.
3. If I have a small sample size, I will prefert to choose wilcoxon test. Although t-test is generally used in small sample size, in this case, we do not know the normality of the data. Therefore, considering the robustness, I prefer to use wilcoxon test.

## Session info

sessionInfo()

## R version 3.6.3 (2020-02-29)  
## Platform: x86\_64-pc-linux-gnu (64-bit)  
## Running under: Ubuntu 16.04.6 LTS  
##   
## Matrix products: default  
## BLAS: /usr/lib/libblas/libblas.so.3.6.0  
## LAPACK: /usr/lib/lapack/liblapack.so.3.6.0  
##   
## locale:  
## [1] LC\_CTYPE=en\_US.UTF-8 LC\_NUMERIC=C   
## [3] LC\_TIME=en\_US.UTF-8 LC\_COLLATE=en\_US.UTF-8   
## [5] LC\_MONETARY=en\_US.UTF-8 LC\_MESSAGES=en\_US.UTF-8   
## [7] LC\_PAPER=en\_US.UTF-8 LC\_NAME=C   
## [9] LC\_ADDRESS=C LC\_TELEPHONE=C   
## [11] LC\_MEASUREMENT=en\_US.UTF-8 LC\_IDENTIFICATION=C   
##   
## attached base packages:  
## [1] stats4 parallel stats graphics grDevices utils datasets   
## [8] methods base   
##   
## other attached packages:  
## [1] TxDb.Hsapiens.UCSC.hg19.knownGene\_3.2.2  
## [2] GenomicFeatures\_1.38.1   
## [3] AnnotationDbi\_1.48.0   
## [4] Biobase\_2.46.0   
## [5] BSgenome.Hsapiens.UCSC.hg19\_1.4.0   
## [6] BSgenome\_1.54.0   
## [7] rtracklayer\_1.46.0   
## [8] Biostrings\_2.54.0   
## [9] XVector\_0.26.0   
## [10] GenomicRanges\_1.38.0   
## [11] GenomeInfoDb\_1.22.0   
## [12] IRanges\_2.20.2   
## [13] S4Vectors\_0.24.3   
## [14] BiocGenerics\_0.32.0   
##   
## loaded via a namespace (and not attached):  
## [1] SummarizedExperiment\_1.16.1 progress\_1.2.2   
## [3] tidyselect\_1.0.0 xfun\_0.12   
## [5] purrr\_0.3.4 lattice\_0.20-41   
## [7] vctrs\_0.2.2 htmltools\_0.4.0   
## [9] BiocFileCache\_1.10.2 yaml\_2.2.1   
## [11] blob\_1.2.1 XML\_3.99-0.3   
## [13] rlang\_0.4.4 pillar\_1.4.3   
## [15] glue\_1.4.0 DBI\_1.1.0   
## [17] rappdirs\_0.3.1 BiocParallel\_1.20.1   
## [19] bit64\_0.9-7 dbplyr\_1.4.2   
## [21] matrixStats\_0.55.0 GenomeInfoDbData\_1.2.2   
## [23] stringr\_1.4.0 zlibbioc\_1.32.0   
## [25] evaluate\_0.14 memoise\_1.1.0   
## [27] knitr\_1.28 biomaRt\_2.42.0   
## [29] curl\_4.3 Rcpp\_1.0.3   
## [31] openssl\_1.4.1 DelayedArray\_0.12.2   
## [33] bit\_1.1-15.1 Rsamtools\_2.2.1   
## [35] hms\_0.5.3 askpass\_1.1   
## [37] digest\_0.6.23 stringi\_1.4.5   
## [39] dplyr\_0.8.4 grid\_3.6.3   
## [41] tools\_3.6.3 bitops\_1.0-6   
## [43] magrittr\_1.5 tibble\_2.1.3   
## [45] RCurl\_1.98-1.1 RSQLite\_2.2.0   
## [47] crayon\_1.3.4 pkgconfig\_2.0.3   
## [49] Matrix\_1.2-18 prettyunits\_1.1.1   
## [51] assertthat\_0.2.1 rmarkdown\_2.1   
## [53] httr\_1.4.1 R6\_2.4.1   
## [55] GenomicAlignments\_1.22.1 compiler\_3.6.3