Workshop bioinfo 2021 part1

Frédéric Silvestre

2/9/2021

```
knitr::opts_chunk$set(echo = TRUE, include=TRUE, cache=TRUE, results="hide", warning=FALSE, error=TRUE,
```

Change the general options in each chunk: eval=T (run the chunk); results=markup (show the results) Option/Command I to create a chunk Shift/option (to [Option (to { | shift/option L ~ option n

#set working directory

```
getwd()
setwd("~/methylome")
ls()
rm(list = ls())
tempdir() #save all the data from R
#to open /var folder > command + shift + G (write /var)
```

#install R v4 and R studio v 1.4

install Bioconductor 3.12

Bioconductor: https://www.bioconductor.org/ search for available packages: https://www.bioconductor.org /packages/release/BiocViews.html#____Software

```
# Check the version of BiocManager on your computer
BiocManager::version()

if(!requireNamespace("BiocManager", quietly=TRUE))
        install.packages("BiocManager")
BiocManager::install(version="3.12")

library(BiocManager)

#To check the number of packages available in Bioconductor
avail <- BiocManager::available()
length(avail)
avail
#update your packages
BiocManager::install()
```

#To obtain help

```
?mean
help.start()
sessionInfo()
help(package = "Biobase")
vignette(package="Biobase")
vignette("ExpressionSetIntroduction")
browseVignettes(package="Biobase")
```

```
BiocManager::install("Biostrings")
library(Biostrings)

#pacman package to install and load packages automatically
if(!require(pacman)){install.packages("pacman");require(pacman)}

if(!require(c("GenomicRanges", "GenomeInfoDb", "IRanges", "Biostrings", "BSgenome", "rtracklayer", "Anno
BiocManager::install(c("GenomicRanges", "GenomeInfoDb", "IRanges", "Biostrings", "BSgenome", "rtracklay

p_load(GenomicRanges, GenomeInfoDb, IRanges, Biostrings, BSgenome, rtracklayer, remotes, AnnotationHub,


if(!require("remotes")){
    install.packages("remotes")
    library(remotes)
}
```

#To save and load R objects

```
saveRDS(tiles, file="tiles1.RDS")
b <- readRDS("tiles1.RDS")
```

#Working with sequences

#create a DNAstring object

```
DNA <- DNAString("TCAG") #create a Biostring object
DNA
class(DNA)

a <- as.character(DNA) #give the sequence
class(a)

length(DNA) #count the number of nucleotides
b <- DNA[2:3]
class(b)
b

DNA2 <- DNAString("AAGCTAT")
DNA2
```

#create a set of DNAstring objects

```
DNA3 <- DNAStringSet(c("TCA", "AAATCG"))
DNA3
class(DNA3)
length(DNA3) #count the number of objects Biostring
width(DNA3) #count the number of nucleotides

set1 <- DNA3[1]
set1
width(set1)

unique(DNA3)
```

```r
duplicated(DNA3)
```

#combine 2 DNAString objects

```r
seqs1 <- DNAStringSet(list(seq1=DNA, seq2=DNA2)) #combine and give a name
seqs1
class(seqs1)
seqs1$seq1
```

#seq in FASTA format of the gene CDR2 of Kmar (downloaded from https://www.ensembl.org/Kryptolebias_marmoratus/Gene/Sequence?db=core;g=ENSKMAG00000010508;r=LWHD01000005.1:3041959-3048914;t=ENSKMAT00000014204)

```r
seqs2 <- readDNAStringSet(filepath="kmarCDR2.fa", format="fasta")

?readDNAStringSet

seqs2
names(seqs2)
class(seqs2)
length(seqs2)
width(seqs2)

seqs2

vmatchPattern("AGGAGGT", seqs2) #retrieve the motif in the sequence

subseq1 <- subseq(seqs2, 3,10)
subseq1
subseq1 <- subseq(seqs2, start=8, end=3029)
subseq1

print(as.character(seqs2))

letterFrequency(seqs2, "T")
letterFrequency(seqs2,"GC")
letterFrequency(seqs2,"GC")/width(seqs2) #GC contain

dinucleotideFrequency(seqs2)

?dinucleotideFrequency

translate(seqs2)

methods(class="DNAString")

subseq2 <- subseq(seqs2, 20,100)
subseq2
class(subseq2)
```

#combine 2 subsequences in a single DNAStringSet object of length 2

```r
singleseq <- c(unlist(subseq1), unlist(subseq2))
singleseq
seqlist <- list(subseq1, subseq2)#combine the 2 subsequences in 2 DNAStringSet object of length 1
```

```
seqlist

widths <- sapply(seqlist, function(x) sum(width(x)))
widths
seqviews <- successiveViews(singleseq, widths)
seqviews
class(seqviews)

seqstring1 <- as(seqviews, "DNAStringSet")
seqstring1
```

```
singleseq2 <- c(unlist(seqs2), unlist(subseq1))#combine 2 sequences from the downloaded sequence
singleseq2
seqlist2 <- list(seqs2, subseq1)
seqlist2
widths2 <- sapply(seqlist2, function(x) sum(width(x)))
widths2
seqviews2 <- successiveViews(singleseq2, widths2)
seqstring2 <- as(seqviews2, "DNAStringSet")
seqstring2
```

#Work on set of genes: GRanges

```
myseq <- DNAString("ACCATTGATTAT")
myset <- DNAStringSet(list(chrI=myseq, chrII=DNAString("ACGTACGT")))
myset
range1 <- GRanges("chrI", IRanges(start = 3, end = 5), strand = "+")
range1
class(range1)

?GRanges

seqnames(range1)
getSeq(myset, range1)

mcols(range1)$wobble <- 10 #create metadata columns
range1
seqinfo(range1) #???
range2 <- GRanges(c("chrZ","chrY"), IRanges(start=c(5,10), end=c(35,45)), strand="+")
range2
range3 <- c(range1, range2)
range3
range3[1:2,]
```

#download gtf file from your species: http://www.ensembl.org/Kryptolebias_marmoratus/Info/Index #to find all the files to download here: https://www.ensembl.org/info/data/ftp/index.html

```
features <- import("kmargtf.gtf")
features
mcols(features) <- mcols(features)[,c("type","gene_name","gene_id")]
features
class(features)
feat <- subset(features, gene_name=="CDR2")
feat
```

#to find references for different species using AnnotationHub

```
?AnnotationHub
ah <- AnnotationHub()
length(ah)
colnames(mcols(ah))
table(ah$species)
records <- query(ah,"Kryptolebias marmoratus")
records
table(records$genome)
table(records$sourcetype)
mcols(records)[,c("title", "rdataclass", "sourcetype")]
ah[["AH64640"]]
```

#ne pas voir query(ah, c(“OrgDb”, “Kryptolebias marmoratus”)) sc_orgdb <- ah[[“AH86088”]] columns(sc_orgdb) head(keys(sc_orgdb, “GENENAME”)) select(sc_orgdb, “GO:0000002”, c(“GENENAME”, “SYMBOL”), keytype=“GO”) keys(sc_orgdb, “SYMBOL”) select(sc_orgdb, “1-acylglycerol-3-phosphate O-acyltransferase 3”, c(“SYMBOL”, “ACCNUM”, “ALIAS”, “ENTREZID”, “ONTOLOGY”, “REFSEQ”), keytype=“GENENAME”)

working with expressionSet

```
read.table("expressiondata.txt", header=TRUE, sep="\t", row.names=1)
exprs <- as.matrix(read.table("expressiondata.txt", header=TRUE, sep="\t", row.names=1))
exprs
class(exprs)
dim(exprs)
colnames(exprs)
head(exprs[,1:3])
head(exprs[1:3,])
head(exprs[1:3])

minimalSet <- ExpressionSet(assayData=exprs)
minimalSet
pData <- read.table("pData.txt", row.names=1, header=TRUE, sep="\t")
pData
summary(pData)

all(rownames(pData)==colnames(exprs))

names(pData)
class(pData)
sapply(pData, class)
pData[c(2, 4), c("Sex")]
pData[pData$Age>20,]

phenoData <- new("AnnotatedDataFrame", data=pData)
phenoData

exampleSet <- ExpressionSet(assayData=exprs, phenoData=phenoData)
exampleSet
exampleSet$Sex[1:3]
featureNames(exampleSet)
sampleNames(exampleSet)
varLabels(exampleSet)
```

```r
mat <- exprs(exampleSet)
mat
males <- exampleSet[,exampleSet$Sex=="M"]
males
exprs(males)
```

```r
#SummarizedExperiment for RNAseq

data(airway, package="airway")
se <- airway
se
assays(se)
assays(se)$counts
rowRanges(se)
colData(se)
se[,se$dex=="trt"]
metadata(se)
```

```r
sessionInfo()
```

```r
#Correction for multiple comparisons
if(!require("qvalue")){
    install.packages("qvalue")
    library(qvalue)
}

?qvalue # proportion of false positive when the test is positive ; compare to Bonferoni and FDR
data(hedenfalk)
qvalues <- qvalue(hedenfalk$p, fdr.level=0.05, method="bootstrap")$q
qvalues
bonferroni <- p.adjust(hedenfalk$p, method="bonferroni")
fdr <- p.adjust(hedenfalk$p, method="fdr")
plot(hedenfalk$p,qvalues,pch=19,ylim=c(0,1),
     xlab="raw P-values",ylab="adjusted P-values")
points(hedenfalk$p,bonferroni,pch=19,col="red")
points(hedenfalk$p,fdr,pch=19,col="blue")
legend("bottomright",legend=c("q-value","FDR (BH)","Bonferroni"),
       fill=c("black","blue","red"))
```