Project report for R (with Basic dna analysis)

qin.he

August 2019

1 Introduction

As the establishment of Human Genome Project(HGP), to determine the entire human genome DNA sequence is one of the ultimate goal for the genome researchers. Since the first detailed linkage map was published, different researches were emphasized on the genome mapping both functionally and dynamically. With the human genome sequencing developing rapidly, different analysis tools are developed helping researchers to conduct computation with regards to the bio-informatics aspects studying of genes, cells and molecules. As R is one commonly used open source language with tools open for both researching and developing, this report is to give a brief introduction from the basic usage of R to the sequence analysis. Due to the recent work on cellular and genomic analysis, some basic introduction related to prominent algorithms will also be covered, from the statistical analysis view.

2 Basic Usage of R

As R is designed for statistical analysis, the language is mainly introduced from objective-oriented programmingly. In addition to some basic statistics(descriptively and linearly), for instances, the summary() giving the most commonly used statistics including mean, median, standard deviation, quantile and range, other advanced functions which can be applied to S3 or S4 classes is also of out interest because of their usage in regression, classification and other statistical and machine learning related task completion. To start with, some basic operation and distribution will be reviewed: A little bit different from the MATLAB, the defaulted remainder from division is expressed not as mod and there is the division especially designed for integer, The notation of the numbers are also mainly similar to MATLAB but with a little difference with the check of infinity/not a number and null object: x¡- 1e-10
is.infinite(x)
¿ [1] FALSE
is.finite(x)
¿ [1] TRUE

Not a number (undefined result, use is.nan to test)
x¡- 0/0
x
¿ [1] NaN
is.nan(x)
¿ [1] TRUE
Null object (use is.null to test)
x¡- NULL
is.null(x)
¿ [1] TRUE
Not available/missing value (use is.na to test)

x¡- NA
is.na(x)
¿ [1] TRUE
is.nan(x)
¿ [1] FALSE
One useful command for NA replacement: x[is.na(x)]¡- 0
Different from The basic learning materials can be found on our summer course website. In addition to the replacement of NA I mentioned t last, as the strictness of unity of variables in table format, there is always necessities to fill NA to substitute blank spaces.(Encountered in this project frequently). One way is to use fill properties in some functions embedded in extended libraries. For instancesin "read.csv" and "cbindfill" and etc. Writing the function by yourself is also convenient. Note that, like MATLAB is advantaged in matrix operation, R is also not good at operation large complexity problem with for loops. Alternatively, sapply and lapply functions are recommended. Here, a short script from this project in filling NA for DNA GC location table is exhibited as one example: LengthA¡- max(countA)
LengthC¡- max(countC)
LengthT¡- max(countT)
LengthG¡- max(countG)
fillLine¡- function(L, sequence)
if (length(sequence)¡ L) sequence¡- c(sequence, rep(NA,
L-length(sequence)))
return(sequence)

posATb¡- t(sapply(seq(1,length(posA)), function(i)
fillLine(LengthA, as.vector(posA[i][[1]]))))
posCTb¡- t(sapply(seq(1,length(posC)), function(i)
fillLine(LengthC, as.vector(posC[i][[1]]))))
posTTb¡- t(sapply(seq(1,length(posT)), function(i)
fillLine(LengthT, as.vector(posT[i][[1]]))))
posGTb¡- t(sapply(seq(1,length(posG)), function(i)
fillLine(LengthG, as.vector(posG[i][[1]]))))

After having a general understand of the data, using summary, it is usually required to preprocess the data especially for biological and health data. TO get rid of the noise and achieve stationary/ unbiased/ specially distributed data, different transformation and normaliztion are usually conducted combining baseline correction. Here, we introduced some useful transformation and normaliation methods combining this project(Structure introduction of the project data will be included in next chapter):

I. Overview the raw data and preprocess

boxplot(rawData, main = "rawData Gene Expressions")
GSMstat["TotalRaw"]¡-
as.vector(summary(as.numeric(rawData)))

normalization:

1. Z-transform is conducted here, it is commonly used for normalization of lightly skewed data which is required to be processed later on norm distributed:

---

z = x-mean/std

---

normData¡- rawData
normData¡- sapply(seq(1,10), function(i) normData[,i]¡-
(rawData[,i] - mean(rawData[,i]))/sd(rawData[,i])
) GSMstat["TotalNorm"]¡-
as.vector(summary(as.numeric(normData)))
boxplot(normData, main = "normData Gene Expressions")
kurtosis.norm.test(normData)

According to the norml test, it is normally distributed with p value being 2.6e-16 less than 0.05 alpha statistics.

2.Log-Normalization(large-scale skewed data) and MAnormalization:

0.Logtransform(against the too large scale) and because of the information is stored
in 2 bits, we thus choose base as 2: x = log2(x)
1.Calculated M(difference between origin and lagged terms) and A(average of these terms) as following:
M = x(n+1)- x(n) can be set as lagged more than 1 if necessary
A = 0.5*(x(n+1)- x(n)) baasically the mean of the sequences of interest
Microarray data is often normalized within arrays to control for systematic biases in dye coupling and
hybridization efficiencies, as well as other technical biases in the DNA probes and
the print tip used to spot the array.By minimizing these systematic variations, true biological differences can be found.
(In the dye coupling, M is, therefore, the binary logarithm of the intensity ratio or difference
between log intensities and A is the average log intensity for a dot in the plot.)

2. Check the MA plot, if there is bias(from 0) go to 3 else stop normalization
3. LOESS noormalization(local weighted regression): Regression and predict the
new center of M Iterated with A: Mc = loess(M,A) and stepwisely updte the M = M- Mc

---

logData¡- log2(rawData)
boxplot(logData)
GSMstat["TotalLog"]¡- summary(as.numeric(logData))
kurtosis.norm.test(logData)

Kurtosis test for normality

data: logData
T = 5.2636, p-value ¡ 1.2e-16

M¡- logData
diff¡- logData
MAData¡- logData
A¡- logData
avg¡- logData
look into differences between samples(supposed to be small,close to zero)
logData¡- cbind(logData,logData[,1])
M¡- sapply(seq(1,10), function(i) M[,i]¡- (logData[,i+1]-logData[,i]))
A¡- sapply(seq(1,10), function(i) A[,i]¡- 0.5*(logData[,i+1]+logData[,i]))
MA Plot of random unrepeated sampled 100 genes(from all length(diff))
set.seed(123)
Ind¡- sample(seq(1,length(M)/10), 100)
smoothScatter(avg[Ind,], diff[Ind,], col = 1, main="MA plot for 100 random genes", xlab="A", ylab="M")
abline(h=c(-1,1), col="red")

according to the result, there is no bias of M thus does not need to do the LOESS normalization
LOESS normalization
Mc¡- A[,1]
MAData¡- sapply(seq(1,10), function(i) l¡- loess(M[Ind,i] A[Ind,i])
Mc¡- predict(l, A[,i])
Mc[is.na(Mc)]¡- 0
print(Mc)
return(MAData[,i]¡- (M[,i] - Mc)))
boxplot(MAData, main = "MA-Normed Gene Expressions")
logData¡- log2(rawData)
GSMstat["TotalMA"]¡- summary(as.numeric(MAData))
write.csv(GSMstat, file = "GSMstat.csv", col.names = T)

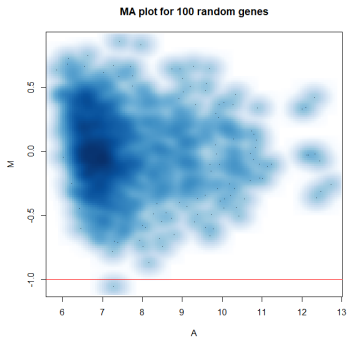


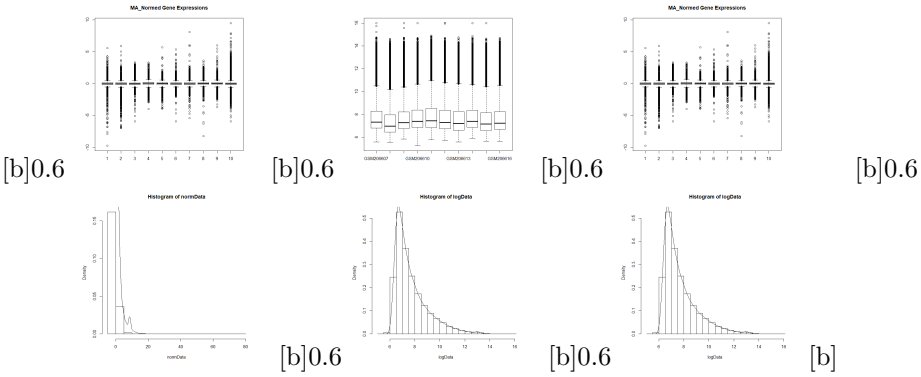
Figure 1: The Universe

kurtosis.norm.test(MAData)
According to the norml test, it is normally distributed with p value being 2.2e-16 less than 0.05 alpha statistics.

Notice that: MA plots are used to visualize
intensity-dependent ratio of raw microarray data
(microarrays typically show a bias here, with higher A resulting in higher —M—, i.e. the brighter the spot the more likely an observed difference between sample and control).
The MA plot uses M as the y-axis and A as the x-axis and gives a quick overview of the distribution of the data.
In many microarray gene expression experiments, an underlying assumption is that most of the genes would not see any change in their expression;
therefore, the majority of the points on the y-axis (M) would be located at 0, since Log(1) is 0.

3.Scale-normalization(if data is extremely skewed) 0.Logtransform(against the too large scale) and because of the information is stored
in 2 bits, we thus choose base as 2: x = log2(x)
1.Calculated MAD(meadian of the absolute value of standard devition)
2. scale the data with initiation: scaleData = logData
3.MAD scale normalization(stcastically) which is to normalized the scaleData by first subtracting median and then scaled by 1/MAD:
MAD¡- sapply(seq(1,10), function(i) MAD[i]¡- median(abs(sd(logData[i]))))
scaleData¡- sapply(seq(1,10), function(i) scaleData[,i]¡- (scaleData[,i] - median(logData[,i]))/MAD[i])

[h!] As listed in the figure the upper three from (a)-(c) gives the boxplots of rawData after z-transformation, Log transformation and MA normalization separately. It is obvious that the mean of the z-transformed and MAnormed datasets are close to zero for all the 10 samples while is above 0 for log data showing the bias still exhists for log data. The figure scale also fits the box



[b]0.6 [b]0.6 [b]0.6 [b]0.6 [b]0.6 [b]

Figure 2: (a)GE after Z-Normed(b)GE after Log-transform(c)GE after MANormalize(d)Histogram of GE after Z-Norm(e)Histogram of GE after Log-transform(f) Histogram of GE after MANormaliz

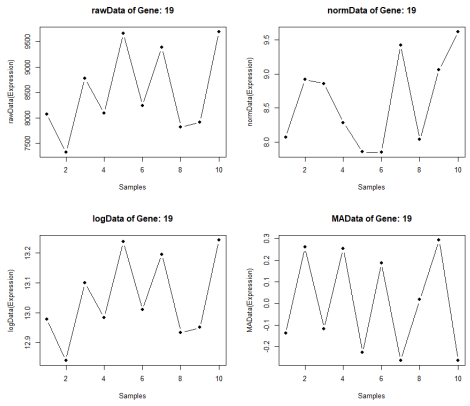
plot's certain parameters: the IQR(Q3 -Q1) is also close to zero for z-normed and MA-normed data while is about 1 for log data.In addition, even normally distributed, there still many gene expressions distribute out of whisker ([Q1-1.5*IQR, Q3+ 1.5*IQR]). Because of the transformation scale, the log data and MA which on basis of log scale both have smaller range with the normdata being several times larger. The symmetry also only exhits in the MA-Normed data.On contrary, the histograms of them also verify the bias of the dataset after log-trasformation and z-transformation. The skewness again exhists significantly in data after log and z transformation. The data within 3 sigma interval also is in accordance with the whisker of the boxplot figures.

3 Basic DNA Analysis

After the preprocess, the expression distribute without bias and skewness. First, we choose one specific gene to check it's expression in 10 samples again further. The rawData, logData and MAData all shows similar trend within 10 samples. For checking stationariness, the MA Data shows the difference between two adjacent samples less than 0.5 and the largest expression exhists at 5th and 10th samples and the smallest appears at the second sample. Next, the exploration goes to the protein sequence statistics. The fasta data ustilied is the dna sequences, 411553 length in all included in the list. To do the sequence analysis of our interest, usually, one single specific sequence is chosen. Here, we try to check one plant used in helping sleeplessness. As protein is the most important component in any mechanisms, including binding, docking and etc. Here, we starts with checking its structure and analysis on amino acid sequences. Through its annotation: ——————————————————————

attr(,"Annot")

¿¿ [1] "¿n19 — AB015469 — snRNA — Arabidopsis thaliana (thale cress) —



U2 — NONCODE v2.0 — NULL — NULL — -1.2887100 — -0.2345336”

---

the probe information, dna name, dna type and etc. the information of the mocular test is not completed, the only mocular test result recorded is SNR635 and SNR532 without F pixels and B pixels and further test results.

The basic sequence structure information can be read through command: count and the first parameter is the count for nucleotides number: ————————

---

count(nc,1)

a	c	g	t
47	45	40	67

---

and the count for dinucleotides number:

---

count(nc,2)

aa	ac	ag	at	ca	cc	cg	ct	ga	gc	gg	gt	ta	tc	tg	tt
9	10	10	17	11	8	7	19	8	12	11	9	18	15	12	22

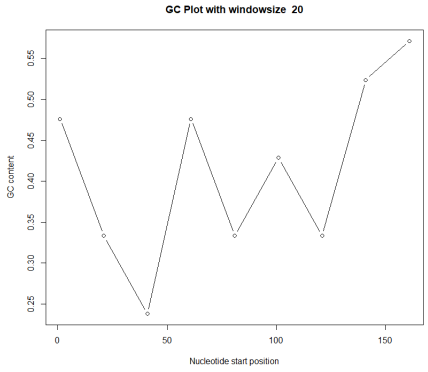
For dna sequence analysis, the GC ratio is of interest at the first place due to its markedly variable. These variations in GC ratio within the genomes of more complex organisms result in the determination of isochores, some islet regions. GC-rich isochores contributes in mapping these specific regions of he genome which is related to the amino acid seuqnce analysis as well. The GC ratio again can be read through the library function GC(), for instances, we can get the total GC content ratio being: 0.4271357 which is less than half. However, according to the low SNR for both 635 and 532, we consider apply slice window to the whole sequence:

```
slidingwindowGCplot ¡-
function(windowsize,inputseq)
GCwindow ¡- seq(1, length(inputseq) - windowsize, by = windowsize)
Chunks ¡- length(GCwindow)
chunks¡-seq(1,Chunks)
for(i in seq(1,Chunks))
chunk¡- inputseq[GCwindow[i]:(GCwindow[i] + windowsize)]
chunkGC¡- GC(chunk)
print(chunkGC)
chunks[i]¡- chunkGC

plot(GCwindow,chunks,type="b",xlab="Nucleotide
start position",ylab="GC content",main=paste("GC Plot with windowsize ",
windowsize))

slidingwindowGCplot(20,nc[1,])
```

As amino acids are the building blocks of proteins and are also used by the cell



for 19th ncrna.png

Figure 4: GCplot for 19th ncrna

as neurotransmitters, sources of energy and a variety of other processes, it is also useful to analyze the amino acids components for some mocular diagnosis of disease in pathophysiology. First, we can check the composition for amino acid using: aaComp(nc)

	Number	Molepercentage
Tiny	1	100
Small	1	100
Aliphatic	1	100
Aromatic	0	0
NonPolar	1	100
Polar	0	0
Charged	0	0
Basic	0	0
Acidic	0	0

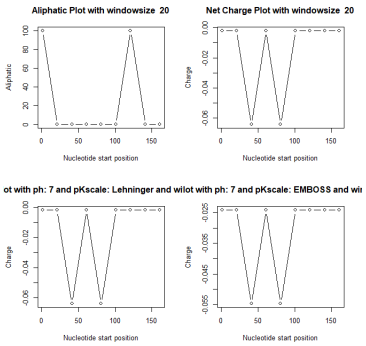
In the first fragment of the total protein sequence(length: 200), for instance, there is one tiny, one small, one aliphatic and one non polar atom with 100 percent individually. If we are interested in the index of them specifically, there are also functions can be called. For instance, the aIndex() for reading the index of aliphatic component. Notice that, the aliphatic components, as flammable, work mainly in enabling hydrocarbon functioning as fuel, we thus usually test it with regard to metabolizing of energy in tissues or organs,which is of our interest considering the effect on sleep quality and deepness. Another parameter of interest can be charge wich shows relation to either environment and ionizability of the certain protein. Generally, when pH is less than the pKscale, the protonate form of amino acid chan predominates, leaving the acidic side chins with a charge approaching 0 and the basic side chains with a charge approaching to +1. Conversely, when the pH is greater than pKscale, the deprotonated form predominates, leacing the charge approaching -1 and basic side chains a charge approaching 0. Uniquely, the charge on indicidual amino acid side chains can ary when near a group of non-polar or highly charged side chains. Here, we check the pH as 7 and pKscale as "Legninger" and "EMBOSS" as example:

```
slidingwindowaaCompplot ¡- function(windowsize,inputseq)
window ¡- seq(1, length(inputseq) - windowsize, by = windowsize)
Chunks ¡- length(window)
chunkaliphs¡-seq(1,Chunks)
chunkQs¡-seq(1,Chunks)
chunkQ1s¡-seq(1,Chunks)
chunkQ2s¡-seq(1,Chunks)
for(i in seq(1,Chunks))
chunk¡- inputseq[window[i]:(window[i] + windowsize)]
chunkaliph¡- aIndex(chunk)
chunkQ¡- charge(chunk)
chunkQ1¡- charge(chunk, pH = 7, pKscale = "Lehninger")
chunkQ2¡- charge(chunk, pH=7, pKscale="EMBOSS")
chunkaliphs[i]¡- chunkaliph
chunkQs[i]¡- chunkQ
chunkQ1s[i]¡- chunkQ1
chunkQ2s[i]¡- chunkQ2
```

write.csv(t(c(chunkaliphs,chunkQs, chunkQ1s, chunkQ2s)),file = "aaCompTb.csv",
col.names = TRUE)
print(data.frame("Aliphatic" = chunkaliphs,"Net Charge" = chunkQs, "Charge
with Lehninger" = chunkQ1s,"Charge with EMBOSS" = chunkQ2s))
layout(matrix(c(1,2,3,4), 2,2, byrow = TRUE))
plot(window,chunkaliphs,type="b",xlab="Nucleotide start position",ylab="Aliphatic",main=paste("Aliphatic
Plot with windowsize ", windowsize))
plot(window,chunkQs,type="b",xlab="Nucleotide start position",ylab="Charge",main=paste("Net
Charge Plot with windowsize ", windowsize))
plot(window,chunkQ1s,type="b",xlab="Nucleotide start position",ylab="Charge",main=paste("ChargePlot
with ph: 7 and pKscale: Lehninger and windowsize ", windowsize))
plot(window,chunkQ2s,type="b",xlab="Nucleotide start position",ylab="Charge",main=paste("ChargePlot
with ph: 7 and pKscale: EMBOSS and windowsize ", windowsize))

slidingwindowaaCompplot(20, nc)

On the other hand, the hydrophobicity which is commonly used to predict the



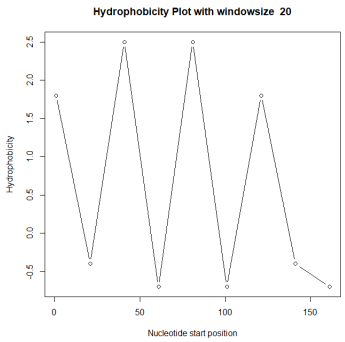
for 19th ncrna with sliding window.png

Figure 5: aaComp for 19th ncrna

transmembrane phapha-helices of membrane proteins is also one property can be studied. Similarly, just change the computation function into hydrophobicity(), we can have the result(code is not included here again.)

4 Summary

To summarize, R is a useful tool in statistics analysis and the preprocess, including normalization and baseline correction can be objectively applied. ks-test for distribution is specifically useful in quality control(QC), Similarly, other methods can be chosen alternatively such as t-test(Welch Two sample t-test), then follows some general test. As the usage in bio-informatic aspects research, we introduce the dna and protein test specifically as one example. Some implemen-



for 19th ncrna.png

Figure 6: Hydrophobicity for 19th ncrna

tation combining basic calculation, data acquisition, transforming(cleaning) and processing are covered.

5 Appendix A: Data Acquistion and description

url ="$http://noncode.org/datadownload/ncrna_NONCODE[v3.0].fasta.tar.gz$"
download.file(url, destfile = "411553.fasta.tar.gz", model = "w", headers = NULL)
untar("D:/Introduction to R/411553.fasta.tar.gz", exdir ="D:/Introduction to R/411553")
$ncrna <- -read.fasta(file = "411553/ncrna_NONCODE[v3.0].fasta")$

6 Appendix B: Extended Libraries

```
if (!requireNamespace("BiocManager", quietly = TRUE))
        install.packages("BiocManager")                    library(genefilter)
        BiocManager::install("Biobase")                    library(Biobase)
        BiocManager::install("genefilter")                 library(limma)
        BiocManager::install("affy")                       library(affy)
        BiocManager::install("oligo")                      library(oligo)
        BiocManager::install("GEOquery")                   library(GEOquery)
        install.packages("normtest")                       library(normtest)
        install.packages("reshape2")                       library(reshape2)
        install.packages("rnaseqWrapper")                  library(gplots)
        install.packages("seqinr")                         library(seqinr)
        install.packages("ape")                            library(ape)
        BiocManager::install(c("DESeq","topGO"))           library(Biostrings)
        install.packages("Peptides")                       library(Peptides)
```

[2] [1] [3]

References

[1] *Biostar*.

[2] Chen Lu. *Introduction to R*.

[3] QinHe. *GithubCode*.