ActividadColaborativa

2022-04-11

#Integrantes: #Fernanda Elizabeth Romo Alarcon - A01639261 #César Arnaldo Cabrera Chávez - A01642244 #Javier Hernández Garza A01635390 #Inés Alejandro Garcia Mosqueda - A00834571 #José Antonio Juárez Pacheco - A00572186

1. Escribe una función que genere una secuencia aleatoria de DNA de tamaño “n”.

#rDNA -> Cadena aleatoria generada

```
randomDNA=function(){
  n = 35
  #n=readline(prompt="n numero de datos: ")
  #n=as.integer(n)
  v=c("A","T","G","C","N","-")
  DNA=sample(v,n,replace=TRUE)

return (DNA)
}

rDNA=randomDNA()
print(rDNA)
```

```
##  [1] "G" "N" "N" "T" "-" "T" "C" "T" "T" "-" "N" "N" "N" "-" "N" "G" "-" "G" "T"
## [20] "C" "-" "A" "A" "C" "T" "T" "N" "-" "A" "G" "N" "T" "C" "G" "T"
```

2. Codifica una función que calcula el tamaño de una secuencia de DNA.

```
size=function(DNA){
  return(length(DNA))
}
size_DNA=size(rDNA)
print(paste("Size",size_DNA))
```

```
## [1] "Size 35"
```

3. Crea una función que recibe una secuencia de DNA e imprime el porcentaje de cada base en la secuencia.

```
porcentajes=function(DNA,sizeDNA){
  v=table(DNA)
  i=1
  n=sizeDNA
  A=0
  C=0
  G=0
  Tc=0
  Guion=0
  if ("A" %in% DNA){
    A=(v[i]/n)*100
    i=i+1
  }
  if ("C" %in% DNA){
    C=(v[i]/n)*100
    i=i+1
  }
  if ("G" %in% DNA){
    G=(v[i]/n)*100
    i=i+1
  }
  if ("T" %in% DNA){
    Tc=(v[i]/n)*100
    i=i+1
  }
  if ("N" %in% DNA){
    N=(v[i]/n)*100
    i=i+1
  }
  if ("-" %in% DNA){
    Guion=(v[i]/n)*100
    i=i+1
  }
  return (c(A,C,G,Tc,N,Guion))
}
p=porcentajes(rDNA,size_DNA)
print(rDNA)
```

```
##  [1] "G" "N" "N" "T" "-" "T" "C" "T" "T" "-" "N" "N" "N" "-" "N" "G" "-" "G" "T"
## [20] "C" "-" "A" "A" "C" "T" "T" "N" "-" "A" "G" "N" "T" "C" "G" "T"
```

```
print(paste("A",p[1],"%","/","C",p[2],"%","/","G",p[3],"%","/","T",p[4],"%","/","N",p[5],"%","/"
,"Omitido",p[6],"%"))
```

```
## [1] "A 17.1428571428571 % / C 8.57142857142857 % / G 11.4285714285714 % / T 14.2857142857143
% / N 22.8571428571429 % / Omitido 25.7142857142857 %"
```

4. Programa una función que transcribe DNA a RNA, usa la siguiente tabla:

```
complemento=function(DNA){
  DNAc=c()
  for (i in 1:length(DNA)){
    if(DNA[i]=="A"){
      DNAc[i]="T"
    }else if (DNA[i]=="T"){
      DNAc[i]="A"
    }else if(DNA[i]=="C"){
      DNAc[i]="G"
    }else if(DNA[i]=="G"){
      DNAc[i]="C"
    }else if(DNA[i]=="-"){
      DNAc[i]="-"
    }else if(DNA[i]=="N"){
      DNAc[i]="N"
    }
  }
  return (transcribir(DNAc))
}

transcribir=function(DNA){
  RNA=c()
  for (i in 1:length(DNA)){
    if(DNA[i]=="A"){
      RNA[i]="U"
    }else if (DNA[i]=="T"){
      RNA[i]="A"
    }else if(DNA[i]=="C"){
      RNA[i]="G"
    }else if(DNA[i]=="G"){
      RNA[i]="C"
    }else if(DNA[i]=="-"){
      RNA[i]="-"
    }else if(DNA[i]=="N"){
      RNA[i]="N"
    }
  }
  return (RNA)
}

transcribirComplemento=function(DNA){
  RNAc = complemento(DNA)
  RNAc = transcribirComplemento(DNA)
}

RNAc=transcribir(rDNA)
print(RNAc)
```

```
##  [1] "C" "N" "N" "A" "-" "A" "G" "A" "A" "-" "N" "N" "N" "-" "N" "C" "-" "C" "A"
## [20] "G" "-" "U" "U" "G" "A" "A" "N" "-" "U" "C" "N" "A" "G" "C" "A"
```

```
print(rDNA)
```

```
##  [1] "G" "N" "N" "T" "-" "T" "C" "T" "T" "-" "N" "N" "N" "-" "N" "G" "-" "G" "T"
## [20] "C" "-" "A" "A" "C" "T" "T" "N" "-" "A" "G" "N" "T" "C" "G" "T"
```

5. Crea una función que traduce una secuencia de RNA a una secuencia de proteínas.

```
traduce = function(i){

    if(i == 'UUU' || i == 'UUC'){
      amino = 'Phe'
    } else if(i == 'UUA' || i == 'UUG' || i == 'CUU' || i == 'CUC' || i == 'CUA' || i == 'CUG' )
{
      amino = 'Leu'
    } else if(i == 'AUU' || i == 'AUC' || i == 'AUA'){
      amino = 'Ile'
    } else if(i == 'AUG'){
      amino = 'Met'
    } else if(i == 'GUU' || i == 'GUC' || i == 'GUA' || i == 'GUG'){
      amino = 'Val'
    } else if(i == 'UCU' || i == 'UCC' || i == 'UCA' || i == 'UCG' || i == 'AGU' || i == 'AGC'){
      amino = 'Ser'
    } else if(i == 'CCU' || i == 'CCC' || i == 'CCA' || i == 'CCG'){
      amino = 'Pro'
    } else if(i == 'ACU' || i == 'ACC' || i == 'ACA' || i == 'ACG'){
      amino = 'Thr'
    } else if(i == 'GCU' || i == 'GCC' || i == 'GCA' || i == 'GCG'){
      amino = 'Ala'
    } else if(i == 'UAU' || i == 'UAC'){
      amino = 'Tyr'
    } else if(i == 'CAU' || i == 'CAC'){
      amino = 'His'
    } else if(i == 'CAA' || i == 'CAG'){
      amino = 'Gln'
    } else if(i == 'AAU' || i == 'AAC'){
      amino = 'Asn'
    } else if(i == 'AAA' || i == 'AAG'){
      amino = 'Lys'
    } else if(i == 'GAU' || i == 'GAC'){
      amino = 'Asp'
    } else if(i == 'GAA' || i == 'GAG'){
      amino = 'Glu'
    } else if(i == 'UGU' || i == 'UGC'){
      amino = 'Cys'
    } else if(i == 'UGG'){
      amino = 'Trp'
    } else if(i == 'CGU' || i == 'CGC' || i == 'CGA' || i == 'CGG' || i == 'AGA' || i == 'AGG'){
      amino = 'Arg'
    } else if(i == 'GCU' || i == 'GGC' || i == 'GGA' || i == 'GGG'){
      amino = 'Gly'
    } else if(i == 'UAA' || i == 'UAG' || i == 'UGA'){
      amino = 'STOP'
    } else {
      amino = "N"
    }
    return (amino)

  }
```

```
traduccion_proteina <- function(DNA){

codones = c()
i = 1
j = 1



while (i < length(DNA)){
  codones[j] = sprintf('%s%s%s', DNA[i], DNA[i+1], DNA[i+2])
  j = j + 1
  i = i + 3
}

lista_tripletes <- c()
for(i in codones){
  tri = traduce(i)
  lista_tripletes <- append(lista_tripletes, tri, after = length(lista_tripletes))
}
print(lista_tripletes)
}
print(transcribir(rDNA))
```

```
##  [1] "C" "N" "N" "A" "-" "A" "G" "A" "A" "-" "N" "N" "N" "-" "N" "C" "-" "C" "A"
## [20] "G" "-" "U" "U" "G" "A" "A" "N" "-" "U" "C" "N" "A" "G" "C" "A"
```

```
traduccion_proteina(transcribir(rDNA))
```

```
##  [1] "N"   "N"   "Glu" "N"   "N"   "N"   "N"   "Leu" "N"   "N"   "N"   "N"
```

6. Para representar una molécula de ADN de doble hebra basta con escribir la secuencia de una de sus hebras. Consideremos, por ejemplo, la secuencia TGCGATAC. Como no se indica lo contrario, se asume que la secuencia está escrita en sentido 5'→3' y, por lo tanto, se trata de la hebra directa (forward strand): Hebra directa: 5'-TGCGATAC-3' Análogamente, si decido escribir esta misma secuencia empezando por el extremo 3' se obtiene la hebra inversa (reverse strand): Hebra inversa: 3'-CATAGCGT-5' Escribe una función que recibe una hebra directa y regresa la hebra inversa.

```
generar_lista <- function(){
  n <- readline(prompt = "Enter size of DNA: ")
  nucleotido_list <- c()
  for (i in 1:n){
    nucleotido <- readline(prompt = "Enter nucleotido: ")
    nucleotido_list <- append(nucleotido_list, nucleotido, after = length(nucleotido_list))
  }
  return(nucleotido_list)
}

lista_inversa <- function(lista_i){
  nucleotido_list <- c()
  for (i in 0:length(lista_i)){
    nucleotido_list = append(nucleotido_list, lista_i[length(lista_i)-i], after = length(nucleot
ido_list))
  }
  return(nucleotido_list)
}

#lista_nucleotidos = generar_lista()
lista_nucleotidos = rDNA
lista_nucleotidos
```

```
##  [1] "G" "N" "N" "T" "-" "T" "C" "T" "T" "-" "N" "N" "N" "-" "N" "G" "-" "G" "T"
## [20] "C" "-" "A" "A" "C" "T" "T" "N" "-" "A" "G" "N" "T" "C" "G" "T"
```

```
lista_inversa(lista_nucleotidos)
```

```
##  [1] "T" "G" "C" "T" "N" "G" "A" "-" "N" "T" "T" "C" "A" "A" "-" "C" "T" "G" "-"
## [20] "G" "N" "-" "N" "N" "N" "-" "T" "T" "C" "T" "-" "T" "N" "N" "G"
```

7. Normalmente se representa la molécula escribiendo primero la hebra directa y debajo la hebra complementaria (complementary strand). La hebra complementaria se escribe en sentido 3’→5’ para que las bases de ambas hebras queden emparejadas:

a. Hebra directa: 5’-TGCGATAC-3’
b. Hebra complementaria: 3’-ACGCTATG-5’. Escribe una función qué recibe una hebra directa y obtiene la hebra complementaria.

```
generar_lista <- function(){
  d <- readline(prompt = "Ingresa el tamanio de la hebra: ")
  nucleotido_list <- c()
  for (i in 1:d){
    nucleotido <- readline(prompt = "Ingresa el nucleotido: ")
    nucleotido_list <- append(nucleotido_list, nucleotido, after = length(nucleotido_list))
  }
  return(nucleotido_list)
}

lista_complementaria <- function(lista_i){
  cnucleotido_list <- c()
  for (i in 1:length(lista_i)){
    if(lista_i[i]=="A"){
      cnucleotido_list[i] <- "T"
    } else if (lista_i[i]=="T"){
      cnucleotido_list[i] <- "A"
    } else if(lista_i[i]=="C"){
      cnucleotido_list[i] <- "G"
    } else if(lista_i[i]=="G"){
      cnucleotido_list[i] <- "C"
      #cnucleotido_list[i] <- append(cnucleotido_list, "C", after = length(cnucleotido_list))
    }else if(lista_i[i]=="-"){
      cnucleotido_list[i] <- "-"
    }else if(lista_i[i]=="N"){
      cnucleotido_list[i] <- "N"
    }
  }
  return(cnucleotido_list)
}
#lista_nucleotidos = generar_lista()
lista_nucleotidos = rDNA
print(lista_nucleotidos)
```

```
##  [1] "G" "N" "N" "T" "-" "T" "C" "T" "T" "-" "N" "N" "N" "-" "N" "G" "-" "G" "T"
## [20] "C" "-" "A" "A" "C" "T" "T" "N" "-" "A" "G" "N" "T" "C" "G" "T"
```

```
lista_complementaria(lista_nucleotidos)
```

```
##  [1] "C" "N" "N" "A" "-" "A" "G" "A" "A" "-" "N" "N" "N" "-" "N" "C" "-" "C" "A"
## [20] "G" "-" "T" "T" "G" "A" "A" "N" "-" "T" "C" "N" "A" "G" "C" "A"
```

8. Si escribimos la secuencia de la hebra complementaria en sentido inverso (5'→3') se obtiene la complementaria inversa (reverse-complement):

a. Hebra complementaria: 3'-ACGCTATG-5'
b. Hebra complementaria inversa: 5'-GTATCGCA-3' Escribe la función en R para obtener la hebra complementaria inversa, desde una hebra complementaria.

```
print(complemento(rDNA))
```

```
##  [1] "G" "N" "N" "U" "-" "U" "C" "U" "U" "-" "N" "N" "N" "-" "N" "G" "-" "G" "U"
## [20] "C" "-" "A" "A" "C" "U" "U" "N" "-" "A" "G" "N" "U" "C" "G" "U"
```

```
print(lista_inversa(complemento(rDNA)))
```

```
##  [1] "U" "G" "C" "U" "N" "G" "A" "-" "N" "U" "U" "C" "A" "A" "-" "C" "U" "G" "-"
## [20] "G" "N" "-" "N" "N" "N" "-" "U" "U" "C" "U" "-" "U" "N" "N" "G"
```

9. Prueba cada una de las funciones y recuerda que las secuencias pueden contener caracteres especiales donde aparte de ATGC, en ADN, y AUGC, en ARN, pudimos ver algunos guiones (omitidos) y N (desconocido) nucleótido.

Caso de prueba: * Generar cadena codificante * Encontrar su tamaño * Porcentaje de cada nucleotido * Doble helice formada (cadena codificante y cadena molde) * Transcripcion de ADN a ARN y cadena molde

```
ADN = randomDNA()
tam_ADN = size(ADN)
porcentajes(ADN,tam_ADN)
```

```
##        -        A        C        G        N        T
## 11.42857 25.71429 22.85714 11.42857 11.42857 17.14286
```

```
print("Helices formadas:")
```

```
## [1] "Helices formadas:"
```

```
print(ADN)
```

```
##  [1] "A" "T" "G" "T" "G" "A" "N" "A" "-" "C" "C" "C" "-" "T" "G" "A" "A" "C" "T"
## [20] "G" "A" "N" "A" "N" "T" "-" "A" "A" "-" "T" "C" "N" "C" "C" "C"
```

```
print(lista_complementaria(ADN))
```

```
##  [1] "T" "A" "C" "A" "C" "T" "N" "T" "-" "G" "G" "G" "-" "A" "C" "T" "T" "G" "A"
## [20] "C" "T" "N" "T" "N" "A" "-" "T" "T" "-" "A" "G" "N" "G" "G" "G"
```

```
print("Transcripcion de ADN a ARN:")
```

```
## [1] "Transcripcion de ADN a ARN:"
```

```
ARN=transcribir(ADN)
print(ARN)
```

```
##  [1] "U" "A" "C" "A" "C" "U" "N" "U" "-" "G" "G" "G" "-" "A" "C" "U" "U" "G" "A"
## [20] "C" "U" "N" "U" "N" "A" "-" "U" "U" "-" "A" "G" "N" "G" "G" "G"
```

```
print(lista_complementaria(ADN))
```

```
##  [1] "T" "A" "C" "A" "C" "T" "N" "T" "-" "G" "G" "G" "-" "A" "C" "T" "T" "G" "A"
## [20] "C" "T" "N" "T" "N" "A" "-" "T" "T" "-" "A" "G" "N" "G" "G" "G"
```

```
print("Doble helice inversa --->directa y complementaria inversas<--- :")
```

```
## [1] "Doble helice inversa --->directa y complementaria inversas<--- :"
```

```
print(lista_inversa(ADN))
```

```
##  [1] "C" "C" "C" "N" "C" "T" "-" "A" "A" "-" "T" "N" "A" "N" "A" "G" "T" "C" "A"
## [20] "A" "G" "T" "-" "C" "C" "C" "-" "A" "N" "A" "G" "T" "G" "T" "A"
```

```
print(lista_inversa(lista_complementaria(ADN)))
```

```
##  [1] "G" "G" "G" "N" "G" "A" "-" "T" "T" "-" "A" "N" "T" "N" "T" "C" "A" "G" "T"
## [20] "T" "C" "A" "-" "G" "G" "G" "-" "T" "N" "T" "C" "A" "C" "A" "T"
```

```
print("Traduccion de ARN a Proteina")
```

```
## [1] "Traduccion de ARN a Proteina"
```

```
print(transcribir(ADN))
```

```
##  [1] "U" "A" "C" "A" "C" "U" "N" "U" "-" "G" "G" "G" "-" "A" "C" "U" "U" "G" "A"
## [20] "C" "U" "N" "U" "N" "A" "-" "U" "U" "-" "A" "G" "N" "G" "G" "G"
```

```
traduccion_proteina(transcribir(ADN))
```

```
##  [1] "Tyr" "Thr" "N"   "Gly" "N"   "Leu" "Thr" "N"   "N"   "N"   "N"   "N"
```