R_biolmol_A01254805

Daniel Barreras, Yair Beltrán, Victor Symonds

2023-04-20

PARTE 1

Crea una función que genere una secuencia aleatoria de nucleótidos de ADN (A, T, G y C) de tamaño “n”. Ejecútala solicitando una secuencia aleatoria de ADN de 30 nucleótidos y muestra el resultado impreso en consola.

```
crear_secuencia_adn <- function(n = 30) {

  nucleotidos <- c("A", "T", "G", "C")
  secuencia <- sample(nucleotidos, n, replace=TRUE)


  return(paste(secuencia, collapse=""))
}



secuencia_aleatoria <- crear_secuencia_adn(30);
cat(secuencia_aleatoria)

## AGATGTTGGAACTTGCGCCGCATTTGCATT
```

Crea una función que calcule el tamaño de una secuencia de ADN. Utilízala para calcular el tamaño de la secuencia que generaste en el punto 1 y muestra el resultado impreso en consola.

```
calcular_tamaño_secuencia <- function(secuencia) {
  return(nchar(secuencia))
}

tamaño_secuencia <- calcular_tamaño_secuencia(secuencia_aleatoria)

cat("El tamaño de la secuencia de ADN es:", tamaño_secuencia)

## El tamaño de la secuencia de ADN es: 30
```

Crea una función que recibe una secuencia de DNA e imprime el porcentaje de cada base (A, C, G y T) en la secuencia. Ejecútala sobre la secuencia que generaste en el punto 1 y muestra el resultado impreso en consola.

```
calcular_porcentaje_bases <- function(secuencia){

  n <- nchar(secuencia)
  a <- 0
```

```
  c <- 0
  g <- 0
  t <- 0

  for (i in 1:n) {
    base <- substr(secuencia, i, i)
    if (base == "A") {
      a <- a + 1
    } else if (base == "C") {
      c <- c + 1
    } else if (base == "G") {
      g <- g + 1
    } else if (base == "T") {
      t <- t + 1
    }
  }

  por_a <- a/n * 100
  por_c <- c/n * 100
  por_g <- g/n * 100
  por_t <- t/n * 100

  cat("Porcentaje de bases A:", round(por_a, digits = 2), "%\n")
  cat("Porcentaje de bases C:", round(por_c, digits = 2), "%\n")
  cat("Porcentaje de bases G:", round(por_g, digits = 2), "%\n")
  cat("Porcentaje de bases T:", round(por_t, digits = 2), "%\n")
}

calcular_porcentaje_bases(secuencia_aleatoria)
```

```
## Porcentaje de bases A: 20 %
## Porcentaje de bases C: 20 %
## Porcentaje de bases G: 26.67 %
## Porcentaje de bases T: 33.33 %
```

Crea una función que recibe una hebra directa y regresa la hebra inversa. Ejecútala sobre la secuencia que generaste en el punto 1 y muestra el resultado impreso en consola.

```
calcular_hebra_inversa <- function(secuencia){
  inversa <- ""
  for (i in nchar(secuencia):1) {
    inversa <- paste0(inversa, substr(secuencia, i, i))
  }
  return(inversa)
}

hebra_inversa <- calcular_hebra_inversa(secuencia_aleatoria)
cat("Secuencia aleatoria: ", secuencia_aleatoria, "\n")
```

```
## Secuencia aleatoria:  AGATGTTGGAACTTGCGCCGCATTTGCATT

cat("Hebra inversa: ", hebra_inversa)

## Hebra inversa:  TTACGTTTACGCCGCGTTCAAGGTTGTAGA
```

Crea una función qué recibe una hebra directa y obtiene la hebra complementaria. Ejecútala sobre la secuencia que generaste en el punto 1 y muestra el resultado impreso en consola

```
calcular_hebra_complementaria <- function(secuencia){
  complementaria <- ""
  for (i in 1:nchar(secuencia)) {
    base <- substr(secuencia, i, i)
    if (base == "A") {
      complementaria <- paste0(complementaria, "T")
    } else if (base == "T") {
      complementaria <- paste0(complementaria, "A")
    } else if (base == "G") {
      complementaria <- paste0(complementaria, "C")
    } else if (base == "C") {
      complementaria <- paste0(complementaria, "G")
    }
  }
  return(complementaria)
}

hebra_complementaria <- calcular_hebra_complementaria(secuencia_aleatoria)

cat("Secuencia aleatoria: ", secuencia_aleatoria, "\n")

## Secuencia aleatoria:  AGATGTTGGAACTTGCGCCGCATTTGCATT

cat("Hebra complementaria: ", hebra_complementaria)

## Hebra complementaria:  TCTACAACCTTGAACGCGGCGTAAACGTAA
```

PARTE 2

Crea una función que transcribe ADN a ARN. Es decir, que recibe una secuencia molde de nucleótidos de ADN y devuelve la secuencia del transcrito correspondiente que se transcribiría a partir de dicha secuencia. Ejecútala sobre la secuencia que generaste en el punto 1 y muestra el resultado impreso en consola.

```
transcribir_adn_arn <- function(secuencia) {
  arn <- ""
  for(i in 1:nchar(secuencia)){
    base <- substr(secuencia, i, i)
    if(base == "T"){
      arn <- paste0(arn, "U")
    }else{
```

```
      arn <- paste0(arn, base)
    }
  }
  return(arn)
}

arn_transcrito <- transcribir_adn_arn(secuencia_aleatoria)
cat("Secuencia aleatoria: ", secuencia_aleatoria, "\n")

## Secuencia aleatoria:  AGATGTTGGAACTTGCGCCGCATTTGCATT

cat("Secuencia transcrita a ARN: ", arn_transcrito)

## Secuencia transcrita a ARN:  AGAUGUUGGAACUUGCGCCGCAUUUGCAUU
```

Crea una función que te sirva para encontrar codones de inicio y de terminación en una secuencia dada de ARN. A la secuencia presente entre un codón de inicio y uno de terminación en un gen se le conoce como marco de lectura. Ejecútala sobre la secuencia que generaste en el punto anterior (6) y muestra el resultado impreso en consola.

```
encontrar_codones <- function(secuencia){
  codon_inicio = "AUG"
  codon_final = c("UAA", "UAG", "UGA")

  secuencia <- paste0(codon_inicio, secuencia, "UAAUAGUGA")

  codones <- c();


  for(i in  seq(1, nchar(secuencia), by=3)){

    j <- (i+2)/3;

    codones[j] <- paste0(substr(secuencia, i, i), substr(secuencia, i+1,
i+1), substr(secuencia, i+2, i+2));
  }
  cat("Codones de la secuencia: ", codones, "\n")

  for(i in 1:length(codones)){
    if(codones[i] == codon_inicio){
      cat("Marco de lectura: START, ")
    }else if(codones[i] %in% codon_final){
      cat("STOP")
      break
    } else {
      cat(codones[i], ", ")
    }
  }
```

```
  return(codones)
}

codones <- encontrar_codones(arn_transcrito)

## Codones de la secuencia:  AUG AGA UGU UGG AAC UUG CGC CGC AUU UGC AUU UAA
UAG UGA
## Marco de lectura: START, AGA , UGU , UGG , AAC , UUG , CGC , CGC , AUU ,
UGC , AUU , STOP
```

Crea una función que traduce una secuencia de ARN a una secuencia de aminoácidos (proteína), siguiendo para ello las reglas del código genético y la representación en una letra de los aminoácidos.

```
traducir_arn <- function(codones){
  tabla_aminoacidos <- data.frame(
    codon = c("UUU", "UUC", "UUA", "UUG", "CUU", "CUC", "CUA", "CUG", "AUU",
"AUC", "AUA", "AUG",
              "GUU", "GUC", "GUA", "GUG", "UCU", "UCC", "UCA", "UCG", "CCU",
"CCC", "CCA", "CCG",
              "ACU", "ACC", "ACA", "ACG", "GCU", "GCC", "GCA", "GCG", "UAU",
"UAC", "UAA", "UAG",
              "CAU", "CAC", "CAA", "CAG", "AAU", "AAC", "AAA", "AAG", "GAU",
"GAC", "GAA", "GAG",
              "UGU", "UGC", "UGA", "UGG", "CGU", "CGC", "CGA", "CGG", "AGU",
"AGC", "AGA", "AGG",
              "GGU", "GGC", "GGA", "GGG"),
    amino_acido = c("F", "F", "L", "L", "L", "L", "L", "L", "I", "I", "I",
"M",
                    "V", "V", "V", "V", "S", "S", "S", "S", "P", "P", "P",
"P",
                    "T", "T", "T", "T", "A", "A", "A", "A", "Y", "Y", "STOP",
"STOP",
                    "H", "H", "Q", "Q", "N", "N", "K", "K", "D", "D", "E",
"E",
                    "C", "C", "STOP", "W", "R", "R", "R", "R", "S", "S", "R",
"R",
                    "G", "G", "G", "G")
  )

  cat("START ")
  for(i in seq_along(codones)){

    codon_encontrado <- codones[i] == tabla_aminoacidos$codon
    posicion <- which(codon_encontrado == TRUE)

    if(tabla_aminoacidos$amino_acido[posicion] == "STOP"){
      break
    }
    cat(tabla_aminoacidos$amino_acido[posicion], " ")
```

```
  }
  cat("STOP")
}

traducir_arn(codones)

## START M  R  C  W  N  L  R  R  I  C  I  STOP
```