bioinformatics\_functions.R

dlaev

2023-06-18

######################################################  
# bioinformatics\_functions.R  
#  
# Description: This file contains functions for calculating GC content, counting nucleotides, and translating DNA sequences.  
# Author: Kevin Zhou Mastascusa  
# Date: 6.18.2023  
#  
# This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License  
# as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.  
#  
# This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;  
# without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  
# See the GNU General Public License for more details.  
#  
# You should have received a copy of the GNU General Public License along with this program.  
# If not, see <http://www.gnu.org/licenses/>.  
#  
######################################################  
  
  
  
# Function: calculate\_gc\_content  
calculate\_gc\_content <- function(sequence) {  
 sequence <- toupper(sequence)  
 gc\_count <- sum(strsplit(sequence, "")[[1]] %in% c("G", "C"))  
 total\_count <- nchar(sequence)  
 gc\_content <- gc\_count / total\_count \* 100  
 return(gc\_content)  
}  
  
# Function: count\_nucleotides  
count\_nucleotides <- function(sequence) {  
 sequence <- toupper(sequence)  
 nucleotide\_counts <- table(strsplit(sequence, "")[[1]])  
 return(nucleotide\_counts)  
}  
  
# Function: translate\_sequence  
# Function: translate\_sequence  
translate\_sequence <- function(sequence) {  
 sequence <- toupper(sequence)  
 genetic\_code <- c("TTT", "TTC", "TTA", "TTG", "CTT", "CTC", "CTA", "CTG",  
 "ATT", "ATC", "ATA", "ATG", "GTT", "GTC", "GTA", "GTG",  
 "TCT", "TCC", "TCA", "TCG", "CCT", "CCC", "CCA", "CCG",  
 "ACT", "ACC", "ACA", "ACG", "GCT", "GCC", "GCA", "GCG",  
 "TAT", "TAC", "CAT", "CAC", "CAA", "CAG", "AAT", "AAC",  
 "AAA", "AAG", "GAT", "GAC", "GAA", "GAG", "TGT", "TGC",  
 "TGG", "CGT", "CGC", "CGA", "CGG", "AGT", "AGC", "AGA",  
 "AGG", "GGT", "GGC", "GGA", "GGG",  
 "TAA", "TAG", "TGA") # Stop codons added  
 amino\_acids <- 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", "H", "H", "Q", "Q", "N", "N",  
 "K", "K", "D", "D", "E", "E", "C", "C",  
 "W", "R", "R", "R", "R", "S", "S", "R",  
 "R", "G", "G", "G", "G",  
 "\*", "\*", "\*") # Stop codons represented as "\*"  
 codons <- strsplit(sequence, "(?<=.{3})", perl = TRUE)[[1]]  
 translated\_sequence <- unname(sapply(codons, function(codon) { # Remove names from the vector  
 index <- match(codon, genetic\_code)  
 if (!is.na(index)) {  
 amino\_acids[index]  
 } else {  
 "X"  
 }  
 }))  
  
 # Find the position of the first stop codon  
 stop\_codon\_index <- match("\*", translated\_sequence)  
  
 # If a stop codon was found, cut the sequence at that point  
 if (!is.na(stop\_codon\_index)) {  
 translated\_sequence <- translated\_sequence[1:(stop\_codon\_index-1)]  
 }  
  
 return(translated\_sequence)  
}  
  
  
# Test: calculate\_gc\_content  
test\_calculate\_gc\_content <- function(sequence) {  
 gc\_content <- calculate\_gc\_content(sequence)  
 expected\_gc\_content <- sum(strsplit(sequence, "")[[1]] %in% c("G", "C")) / nchar(sequence) \* 100  
 tolerance <- 0.01 # Adjust the tolerance level as needed  
  
 diff <- abs(gc\_content - expected\_gc\_content)  
 print(paste("GC Content:", gc\_content))  
 print(paste("Expected GC Content:", expected\_gc\_content))  
 print(paste("Difference:", diff))  
 print(paste("Tolerance:", tolerance))  
  
 stopifnot(diff <= tolerance)  
}  
  
# Test: count\_nucleotides  
test\_count\_nucleotides <- function() {  
 sequence <- "ATGCCGTAATGGCCTAAG"  
 nucleotide\_counts <- count\_nucleotides(sequence)  
 expected\_counts <- c(A = 4, T = 5, G = 4, C = 5)  
 stopifnot(all(sort(nucleotide\_counts) == sort(expected\_counts)))  
}  
  
# Test: translate\_sequence  
# Test: translate\_sequence  
# Test: translate\_sequence  
# Test: translate\_sequence  
# Test: translate\_sequence  
# Test: translate\_sequence  
test\_translate\_sequence <- function() {  
 sequence <- "ATGCCGTAATGGCCTAAG"  
 translated\_sequence <- translate\_sequence(sequence)  
 print(translated\_sequence)  
 expected\_sequence <- c("M", "P")  
 stopifnot(identical(translated\_sequence, expected\_sequence))  
}  
  
  
  
# Run the test again  
test\_translate\_sequence()

## [1] "M" "P"

# Run all tests  
test\_calculate\_gc\_content("ATGCCGTAATGGCCTAAG")

## [1] "GC Content: 50"  
## [1] "Expected GC Content: 50"  
## [1] "Difference: 0"  
## [1] "Tolerance: 0.01"

test\_count\_nucleotides()  
test\_translate\_sequence()

## [1] "M" "P"

# TODO: Checklist  
# - [ ] Update the `translate\_sequence` function to handle stop codons correctly  
# - [ ] Update the `test\_translate\_sequence` function to use the expected sequence with stop codons  
# - [ ] Test the `translate\_sequence` function to ensure it returns the expected translated sequence  
# - [ ] Review and update the comments and documentation for the functions  
# - [ ] Handle edge cases and error cases in the functions (e.g., empty string, invalid characters, sequence length not multiple of 3)  
# - [ ] Perform additional tests and validation for all functions to ensure correct behavior in various scenarios  
# - [ ] Refactor the code or functions for better readability or performance if necessary  
# - [ ] Update or add any additional functions or features as needed  
  
# End of TODO checklist