031920.1

Ange, Chufan, Emma

3/19/2020

ECEV320 final project: primer design Ange Uwimana, Chufan Cai, Emma Wilkinson

1.Why primer design is important:

2.How the code works:

Input: 1) a DNA sequence (the single strand DNA from a double strand DNA) arranged from 5' to 3' in a .txt file, not space, tab or comma separated and all upper case.

2) the exact position where the primers are required, enter as "b" in the code. Output: Two data frames: 1) all appropriate primers around position "b", from 5' to 3'; 2) all appropriate complement primers around position "b", from 5' to 3'; Example: The user would like to amplify by PCR between 1680bp to 1900bp within the human TET2 gene. First, assign "b" as "1680" and run the code, the code gives out all primers and complement primers that can be used as forward primers; Second, assign "b" as "1900" and run the code, the code gives out all primers and complement primers that can be used as reverse primers.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
plasmid_sequence <- read.delim("./human TET2 sequene single strand.txt", header = FALSE, stringsAsFacto

#assign value to b: b is the position in a gene where we want primers;
#here is an example: we want primers around 1680bp, so b=1680, and you can use any numbers

b <- 1680

#-----------------------------------------------------------------------------------------------
#clean up raw data and convert to data frame format (from 5' to 3')

plasmid_sequence <- strsplit(plasmid_sequence[1,1],"")
plasmid_sequence <- data.frame(plasmid_sequence[1], row.names = NULL,
                               check.rows = FALSE, check.names = TRUE,
                               fix.empty.names = TRUE, stringsAsFactors=FALSE)

#based on the plasmid sequence, generate all primers around b position, and named "primer_precursors".
```

```
#a = primer length -1



a <- 23
primer_precursor_23 <- NULL
for (i in 0:a){
  primer_precursor_23 <- rbind(primer_precursor_23, plasmid_sequence[(b-i):(b+a-i),], .id = NULL)}

primer_precursor_23 <- data.frame(primer_precursor_23, row.names = NULL,
                                  check.rows = FALSE, check.names = TRUE,
                                  fix.empty.names = TRUE, stringsAsFactors=FALSE)
```

3.Dissection of each part of the code:

Part I: Import DNA sequence, extract all possible primers and complement primers (precursors) with length range from 18 to 24 bp. (Chufan)

Part II: Keep precursors that have 40-60% GC content, and have a melting temperature (Tm) of 50-60°C. (Emma)

i.The guanine-cytosine (GC) content of a primer is the percentage of guanine and cytosine present in a DNA sequence. The GC
content is calculated by: (Count(G+C)/(Count(A+C+T+G). The GC content is presented as a decimal, corresponding to a %. G-C base pairs, unlike A-T base pairs, have a unique triple hydrogen bond, which are harder to break, and can contribute to the overall stability of the primer. Additionally, primers with high GC content have higher melting temperatures. Therefore, it is common practice to design PCR primers to have a GC content within a range of 0.4-0.6 (or 40%-60%). This intermediate range ensures that the primers have sufficient stability without contributing to a high melting temperature.

Our code first generates the GC content of all of the primer and complementary primer precursors. The GC content is calculated using the "GC" function found in the "seqinr" package. Shown is the workflow for the 24bp primer_precursors generated in Part I.

The input used to calculate the GC and Tm are the primer/complementary primer precursors generated in Part I mentioned above.In this example, we use the 24bp primer precursor as the input. Using the package "seqinr", we will then fun the function "GC" over each row of the dataframe containing the 24bp primer precursors. The apply function is used in this example to run the GC function (FUN=GC) over each row (MARGIN =1). The output is then a numeric vector corresponding to the GC content of these primers.

```
options(repos="https://cran.rstudio.com" )
#install seqinr package
install.packages("seqinr")
```

```
##
## The downloaded binary packages are in
##  /var/folders/h6/k12mj6kx7qb8g5spjwn0zmtm0000gn/T//RtmptO3VRk/downloaded_packages
```

```
  library(seqinr)
```

```
##
## Attaching package: 'seqinr'
```

```
## The following object is masked from 'package:dplyr':
##
##     count
```

```
#calculate the GC content found in the primer_precursors_23 dataframe (which corresponds to the 24bp pr
pp_24_GC <-apply(as.matrix(primer_precursor_23), MARGIN = 1, FUN =GC)
```

Next, we add the numeric vector to the end of the primer_precursor data frame, labeling it as "GC Content"

```
#add numeric vector to primer_precursor_23 dataframe, naming it "GC Content"
primer_precursor_23["GC Content"] <- pp_24_GC
```

After calculating the GC content of every primer (found by row), the primers that are not within the range of 0.4-0.6 (40%-60%) are filtered out, and a new list of the primers that meet this threshold is created. If the GC content is between >=0.4 or <= 0.6, the code will create a new file called pp_(bplength)bp_pass_GC_threshold. In this case, it is pp_24bp_pass_GC_threshold. This is done using the filter function in the dplyr library.

```
#run dplyr function, filter the primer_precursors that have a GC content betwen 0.4-0.6
library(dplyr)
pp_24bp_pass_GC_threshold <- filter(primer_precursor_23, primer_precursor_23$`GC Content` >= 0.4, prime
```

This workflow is then continued to include the complementary_primer_precursors from 18-24bp.

ii.The melting temperature (Tm) of a primer is also a major factor in PCR efficiency. By definition, the melting temperature of a primer is the temperaturewherein 50% of the double-stranded DNA input is dissociated into single-stranded DNA. There are many ways to calculate the Tm, with factors such as GCcontent and length of primer effecting overall Tm. A common method is using the Wallace Method, which is calculated by the following equation: Tm = 4(G + C) + 2(A + T)=°C. However, this method is known to only be accurate for primers between 14-20bp and is biased towards shorter primers, as it takes into account total primer length in the calculation. A more accurate melting temperature can be calculated using the Nearest Neighbor thermodynamics.This method is considered to be more accurate than the Wallace Method and takes into consideration not only nucleotide content, but sequence as well. A version of this formula is seen here:

Tm = $((\Delta H)/(A + \Delta S + R\ln(C/4)))$-273.15 + 16.6log[Na+]

In this equation ΔH is the enthalpy change (in kcal mol-1) during which the double-stranded DNA dissociates into sinlge-stranded DNA in the initial melt step. A is equal to -0.0108 kcal mol-1, which accounts for the helix initiation during this melt phase. ΔS involves the entropy change that occurs during the initial melt phase. R is the gas constant, measured at 0.00199 kcal mol-1. C is the oligonucleotide concentration that is usually standardized between 25-50nM, depending on the application. -273.15 is a conversion factor used to convert between Kelvin and C. Additionally, [NA+] is the sodium ion concentration, standardized around 50mM. The oligonucleotide concentration and [NA+] can be changed to fit the PCR reaction. Other factors, such as ions within the PCR reaction, can shift the Tm as well.

Our code takes the primers that meet the GC threshold and computes the melting temperature for these primers using the Tm_NN (nearest neighbor) function found in the "Tm Calculator" package. Once the Tm_NN is calculated for each primer, by row, the primers that do not have a Tm between 50°-60°C are filtered out, similar to the GC-content workflow.The "Biostrings" package is also necessary for calcualting the Tm, as the Tm Calculator package requires strings, rather than characters, which is converted by the c2s function found within this package.

It is important to note that Tm_NN contains default primer concentrations to 25nM and [NA+] to 50mM. The Tm_NN function can also accommodate applications that require other concentrations of primers, or different input concentrations of other ions associated in PCR reactions, such as Mg 2+ , which is defaulted to 0 mM. It is up to the user to change the default settings.

To calculate the Tm, we first create an empty vector (pp_24bp_pass_GC_threshold_Tm). Once this empty vector is created, the colon within the for loop creates a sequence of numbers from 1 to 15, and loops over i = 1, i= 2,etc. The for loop runs the function Tm_NN over each row of the pp_24bp_pass_GC_threshold dataframe and creates a numeric vector containing the calcualted Tm values for that dataframe.

```
#install TmCalculator package
install.packages("TmCalculator")
```

```
##
```

```
## The downloaded binary packages are in
##  /var/folders/h6/k12mj6kx7qb8g5spjwn0zmtm0000gn/T//RtmptO3VRk/downloaded_packages
```

```
library(TmCalculator)
```

```
##
## Attaching package: 'TmCalculator'
```

```
## The following objects are masked from 'package:seqinr':
##
##     c2s, GC, s2c
```

```
#installBiostrings pacakge
install.packages("Biostrings")
```

```
## Warning: package 'Biostrings' is not available (for R version 3.6.1)
```

```
library(Biostrings)
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which, which.max, which.min
```

```
## Loading required package: S4Vectors
```

```
## Warning: package 'S4Vectors' was built under R version 3.6.2
```

```
## Loading required package: stats4
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, rename
```

```
## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

## Warning: package 'IRanges' was built under R version 3.6.2

##
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## Loading required package: XVector

##
## Attaching package: 'Biostrings'

## The following object is masked from 'package:TmCalculator':
##
##     complement

## The following object is masked from 'package:seqinr':
##
##     translate

## The following object is masked from 'package:base':
##
##     strsplit
```

```
#24bp
pp_24bp_pass_GC_threshold_Tm <- c()

for (i in 1:nrow(pp_24bp_pass_GC_threshold)){
  pp_24bp_pass_GC_threshold_Tm <- c(pp_24bp_pass_GC_threshold_Tm, Tm_NN(as.matrix(c2s(pp_24bp_pass_GC_t
}
```

As previously mentioned, high GC content can contribute to a high melting temperature, as a higher melting temperature is needed to break the triple hydrogen bond present between the G-C base pair. Therefore, it is commonly recommended that the primer Tm is between 50°-60°C. Melting temperatures above this range can increase the probability of secondary annealing.

Our next step is to filter the melting temperatures that are within 50°-60°C. To do this, we add the vector containing the Tm, pp_24bp_pass_GC_threshold_Tm, and add it to the pp_24bp_pass_GC_threshold dataframe using the cbind function. We then rename that column "Tm". In this case, this was the 26th row (#bp +2), but this will change for bp of other sizes.

To filter, we use the dplyr function, similar to what was used to filter the GC content. This function is used to filter the melting temperatures of the primers that are between 50°-60°C. The primers that have a melting temperature within this range are filtered into a dataframe labeled pp_24bp_pass_GC_and_Tm_threshold.

```
#add Tm generated to pp_#bp_pass_GC_threshold
pp_24bp_pass_GC_threshold <-cbind(pp_24bp_pass_GC_threshold, pp_24bp_pass_GC_threshold_Tm)

#rename new column generated "Tm"
colnames(pp_24bp_pass_GC_threshold)[26] <- "Tm"

#filter out the Tm that are not within 50-60 degrees
library(dplyr)
```

```
#use the dplyr filter function to filter melting temperatures between 50-60 degrees.The primers that me
#placed into a new dataframe called pp_#bp_pass_GC_and_Tm_threshold

#24bp
pp_24bp_pass_GC_and_Tm_threshold <- filter(pp_24bp_pass_GC_threshold, pp_24bp_pass_GC_threshold$Tm >=50
```

This process is also repeated for all complementary primer precursors. The output of the expanded code should include primers precursors and complementary primer precursors that have a GC content betwwen 0.4-0.6 and a melting temperature 50°-60°C. Once both forward and reverse primers are designed (see Part I) it is recommended that forward and reverse primers be within 5° of each other to ensure that the PCR reaction occurs efficiently on both sides of the gene of interest.

Part III: Get rid of precursors that don't start or end with ½ G/C. (Ange)

4.Strengths of our code: All primers generated are from 5' to 3', so no further reverse complement is needed.

5.Shortages of our code:

Whether the primers are self or double complemented hasn't been tested. Primer pairs should have a Tm within 5°C of each other. Make sure there are no highly repetitive sequences. You can look at primer pairs and filter out those that have >4 repeats. Primer pairs should not have complementary regions. Primer pairs should have a Tm within 5°C of each other.(Need to explain why-I note why we cant have the primers have complementary regions below)

Our code does not cover another essential aspect of PCR primer design, which involves primers that do not self-anneal, forming primer-primer interactions, alternatively called primer dimers. This step should be done after designing both forward and reverse primers. Primer dimers can form due to complementary bases in the sequences, thus leading to off-target amplification and inhibiting the amplification of the DNA target sequence. While we note that primers should not have complementary regions, our code does not contain an algorithm that correctly predicts primer dimers. Many currently available programs caclulate and incorporate the Gibbs free energy (ΔG) of association in determining primer-dimer formation (a ΔG of around (-5) to (-6) is standard, but is largely application based). Therefore, to overcome this shortcoming, we recommend either using software, such as the OligoAnalyzer Tool by IDT, or the PrimerRoc algorithm developed by Johnson et al.(2019) to determine the presence of primer dimers once both forward and reverse strands are designed.