Restriction_site_fitter

AFP

4 de octubre de 2018

Intro

Inspired on the second epigenetics TD exercise 3.
Introduce in each sequence a restriction site in order to clone it in the proper Open Reading Frame (ORF) and right orientation.

Writing complement DNA and reverse direction functions

Complement_dna basically changes the original strand to the complementary bases
“A”=“T”, “T”=“A”,“G”=“C”,“C”=“G”

```
##complement of the DNA sequence
complement_DNA<-function(DNA_seq){
m=strsplit(DNA_seq,"")
m=m[[1]]
c_DNA_seq=unname(sapply(m, switch, "A"="T", "T"="A","G"="C","C"="G"))
c_DNA_seq=paste(c_DNA_seq, collapse="")
c_DNA_seq
}
```

Reverse_DNA will flip the sequence from 5’->3’ to 3’->5’

```
reverse_DNA<- function(DNA_seq){
  bp_split=strsplit(DNA_seq,"")
  rev_site_split=rev(bp_split[[1]])
  rev_site<- paste(rev_site_split, collapse="")
  rev_site
}
```

gene sequence

First step, input the DNA sequence to be studied in a character vector called “seq1”

```
seq1="CCACGCGTCCGCCCACGCGTCCGGGAGGCACTAGGGATGGTCCGCAGGATTGGACTGATACAGAGGCCGCCACGGAGCCCGCCGGAGCCA
CCGTTCCTGCTGCTGCCGCCGCTGCCCGAATCGGAACCGTCGGGCCGCAGCCGCCGGCAATGCCGCGAAGGAAGAGGAATGCAGGCAGTAGTTCAG
ATGGAACCGAAGATTCCGATTTTCTACAGA"

#a test to see the complementary sequence of seq1
complement_DNA(seq1)
```

```
## [1] "GGTGCGCAGGCGGGTGCGCAGGCCCTCCGTGATCCCTACCAGGCGTCCTAACCTGACTATGTCTCCGGCGGTGCCTCGGGCGGCCTCG
GTGGCAAGGACGACGACGGCGGCGACGGGCTTAGCCTTGGCAGCCCGGCGTCGGCGGCCGTTACGGCGCTTCCTTCTCCTTACGTCCGTCATCAAG
TCTACCTTGGCTTCTAAGGCTAAAAGATGTCT"
```

restriction site enzimes

Second, the input of all the restriction sites and their enzimes names.
This is wrote in a way that an unlimited number of enzimes can be inputted.
They are stored in a list called site_mut

```
site_mut=list(HindIII = "AAGCTT",
EcoRI = "GAATTC",
Asp718I = "GGATACC",
KpnI = "GGTACC",
BamHI ="GGATCC" ,
EcoRV = "GATATC",
NotI = "CGGCCG" ,
XhoI = "CTCGAG" ,
XbaI ="TCTAGA")

#an example of all the reversed restriction site enzimes
lapply(site_mut,reverse_DNA)
```

```
## $HindIII
## [1] "TTCGAA"
##
## $EcoRI
## [1] "CTTAAG"
##
## $Asp718I
## [1] "CCATAGG"
##
## $KpnI
## [1] "CCATGG"
##
## $BamHI
## [1] "CCTAGG"
##
## $EcoRV
## [1] "CTATAG"
##
## $NotI
## [1] "GCCGGC"
##
## $XhoI
## [1] "GAGCTC"
##
## $XbaI
## [1] "AGATCT"
```

respect for the ORF

What has to be tested are the parts that do not concern the ORF and also to get rid of the stop codon.
For that, the original DNA string “seq1” will be separated in 3 parts, and only will use the parts that are outside the ORF for the analysis (including the TAG stop codon).
The DNA seq before the start codon will be called “begining_seq”.

```
#DNA seq before start
start= "ATG"
start_pos=regexpr("ATG", seq1)[1]
start_coding=substr(seq1, start_pos, nchar(seq1))
begining_seq=substr(seq1, 1, start_pos)
#example of the begining seq
begining_seq
```

```
## [1] "CCACGCGTCCGCCCACGCGTCCGGGAGGCACTAGGGA"
```

The DNA seq after the stop codon will be called “final_seq”

```
#DNA seq after stop
stop_pos=regexpr("TAG", start_coding)[1]
final_seq=substr(start_coding, stop_pos, nchar(start_coding))
#example of the final sequence
final_seq
```

```
## [1] "TAGTTCAGATGGAACCGAAGATTCCGATTTTCTACAGA"
```

check for exact matches on the DNA strand

This code will search 4 cases: + enzime sequence over the DNA strand at the begining and final seq
+ reversed enzime sequence over the DNA strand at the begining and final seq (checkpoint)
+ enzime sequence over the complementary DNA strand at the begining and final seq
+ reversed enzime sequence over the complementary DNA strand at the begining and final seq (checkpoint)

If there is a -1 result it means it did not find a match.

```
#check for exact matches
#1st case: exact match
lapply(site_mut, regexpr,begining_seq)
lapply(site_mut, regexpr,final_seq)
#2nd case: exact reverse match
lapply(lapply(site_mut,reverse_DNA), regexpr,begining_seq)
lapply(lapply(site_mut,reverse_DNA), regexpr,final_seq)
#3rd case: complementary chain exact match
lapply(site_mut, regexpr,complement_DNA(begining_seq))
lapply(site_mut, regexpr,complement_DNA(final_seq))
#4th case: complementary chain  reverse match
lapply(lapply(site_mut,reverse_DNA), regexpr,complement_DNA(begining_seq))
lapply(lapply(site_mut,reverse_DNA), regexpr,complement_DNA(final_seq))
```

Conclusion: no exact matches. The output was not plotted because it takes too much space.

levenshtain distance search FUNCTION

In information theory, linguistics and computer science, the Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. It is named after the Soviet mathematician Vladimir Levenshtein, who considered this distance in 1965. (wikipedia) The implemented package in R is called stringdist. It has a function called amatch that searches for matches with a given distance.

levenshtain distance search FUNCTION

I wrote a function that does the following:
Given the inputs:
1. List of enzimes
2. Any DNA sequence (in the example the sequences outside tbe ORF)
3. Levenshtain distance (could be 1,2,3…) 4. Start or final analysis Does the following steps:
1. creates a sequence from 1 to the length of the restriction site sequences. Combinations of the DNA string from begining to end minus length of the enzime sequence -1 2. Conditionally, stores only the positive results of the levenshtein matchs with the following information:
+ exact matched positions of the DNA string starting from left to right of the original sequence
+ the sequence that was matched on the DNA string
+ the enzime sequence that was matched (to see the distance)
+ the type of match: either original or reversed enzime, either on the original DNA or its complement

For start seq: it also checks that from the final position, it does not disrupt the triplet order (it searches for multiples of 3) For the end seq:it only analyses that the enzime contains the position 1 in order to cut the STOP sequence.

Function code

The function name is “find_DNA_match”

```
#Loading the library to the working environment
library(stringdist)
```

```
## Warning: package 'stringdist' was built under R version 3.5.1
```

```
#Levenshtain distance search FUNCTION
find_DNA_match<- function (enzime, DNA_seq, distance, type){

if (type=="start"){
  combs=list()
  m=nchar(DNA_seq)-nchar(enzime)
  #this is for generalizing, the length of the enzime seq -1
  enzime_spaces<- nchar(enzime)-1
for (i in 1:m){
  #first for amatch, create 1:6 combinations from begining to end- the length
  #of the enzime -1
  sequence<-substr(DNA_seq,i, i+enzime_spaces)
  #same direction enzime, DNA strand
  if (!is.na(amatch(enzime, sequence, method='osa',maxDist=distance))){
    chk_mult=i-nchar(DNA_seq)
    if(chk_mult%%3==0){
  combs[[paste0("E_S", i)]]<-list("position"= seq(i,i+enzime_spaces), "sequence"=sequence, "enzi
me"=enzime, "type"= "same direction enzime, DNA strand")
    }
  }

  }

  #reverse direction enzime, DNA strand
  if (!is.na(amatch(reverse_DNA(enzime), sequence, method='osa',maxDist=distance))){
    chk_mult=i-nchar(DNA_seq)
    if(chk_mult%%3==0){
    combs[[paste0("RE_S", i)]]<-list("position"= seq(i,i+enzime_spaces), "sequence"=sequence, "e
nzime"=reverse_DNA(enzime), "type"= "reverse direction enzime, DNA strand")
    }
    }

  #same direction enzime, complementary DNA strand
  if (!is.na(amatch(enzime, complement_DNA(sequence), method='osa',maxDist=distance))){
    chk_mult=i-nchar(DNA_seq)
    if(chk_mult%%3==0){
    combs[[paste0("E_CS", i)]]<-list("position"= seq(i,i+enzime_spaces), "sequence"= complement_
DNA(sequence), "enzime"=enzime, "type"= "same direction enzime, complement DNA strand")
    }
    }
  # reverse direction enzime, complement DNA strand
  if (!is.na(amatch(reverse_DNA(enzime), complement_DNA(sequence), method='osa',maxDist=distanc
e))){
    chk_mult=i-nchar(DNA_seq)
    if(chk_mult%%3==0){

    combs[[paste0("RE_CS", i)]]<-list("position"= seq(i,i+enzime_spaces), "sequence"= complement
_DNA(sequence), "enzime"=reverse_DNA(enzime), "type"= "reverse direction enzime, complement DNA
 strand")
    }

}
```

```
}



if (type=="final"){
#dna search function for the post-stop codon
#it HAS to cut at the 1 position
  combs=list()
  m=nchar(DNA_seq)-nchar(enzime)
  #this is for generalizing, the length of the enzime seq -1
  enzime_spaces<- nchar(enzime)-1
  for (i in 1){
    #first for amatch, create 1:6 combinations from begining to end-5
    sequence<-substr(DNA_seq,i, i+enzime_spaces)
    #same direction enzime, DNA strand
    if (!is.na(amatch(enzime, sequence, method='osa',maxDist=distance))){
      #chk_mult=i-nchar(DNA_seq)
     # if(chk_mult%%3==0){
        combs[[paste0("E_S", i)]]<-list("position"= seq(i,i+enzime_spaces), "sequence"=sequence,
 "enzime"=enzime, "type"= "same direction enzime, DNA strand")
      #}
    }

  }

  #reverse direction enzime, DNA strand
  if (!is.na(amatch(reverse_DNA(enzime), sequence, method='osa',maxDist=distance))){
   # chk_mult=i-nchar(DNA_seq)
    #if(chk_mult%%3==0){
      combs[[paste0("RE_S", i)]]<-list("position"= seq(i,i+enzime_spaces), "sequence"=sequence,
"enzime"=reverse_DNA(enzime), "type"= "reverse direction enzime, DNA strand")
    #}
  }

  #same direction enzime, complementary DNA strand
  if (!is.na(amatch(enzime, complement_DNA(sequence), method='osa',maxDist=distance))){
    #chk_mult=i-nchar(DNA_seq)
    #if(chk_mult%%3==0){
      combs[[paste0("E_CS", i)]]<-list("position"= seq(i,i+enzime_spaces), "sequence"= complemen
t_DNA(sequence), "enzime"=enzime, "type"= "same direction enzime, complement DNA strand")
    #}
  }
  # reverse direction enzime, complement DNA strand
  if (!is.na(amatch(reverse_DNA(enzime), complement_DNA(sequence), method='osa',maxDist=distanc
e))){
    #chk_mult=i-nchar(DNA_seq)
    #if(chk_mult%%3==0){

      combs[[paste0("RE_CS", i)]]<-list("position"= seq(i,i+enzime_spaces), "sequence"= compleme
nt_DNA(sequence), "enzime"=reverse_DNA(enzime), "type"= "reverse direction enzime, complement DN
A strand")
    #}

  }
```

```
}
  combs

}
```

Results

The input is the list of enzimes “site_mut”, the begining sequence and for a distance of 1.
To interpret the results, because in the begining (START) sequence the wanted codon is at the final part, the best fit should be the enzimes that are closest to its final part.
For the final sequence (the one after the stop codon) the desired position should be the one closest to the first base, so there is no mean in known the length of the string.
In this example, the begining sequence has 37 bases. Thus, the best fit would be those that are closest to the number 37.
On the final sequence there is only site 1 containing results so there should be no problem.

```
#Now, search in the begining and final sequences: begining_seq,final_seq
#find_DNA_match (enzime, DNA_seq, distance)
#for the begining seq, find the closest to the end
nchar(begining_seq)
```

```
## [1] 37
```

```
str(lapply(site_mut, find_DNA_match, begining_seq, 1, "start"))
```

```
## List of 9
##  $ HindIII: list()
##  $ EcoRI  : list()
##  $ Asp718I: list()
##  $ KpnI   : list()
##  $ BamHI  : list()
##  $ EcoRV  : list()
##  $ NotI   : list()
##  $ XhoI   : list()
##  $ XbaI   : list()
```

```
str(lapply(site_mut, find_DNA_match, begining_seq, 2, "start"))
```

```
## List of 9
##  $ HindIII: list()
##  $ EcoRI  : list()
##  $ Asp718I: list()
##  $ KpnI   :List of 2
##   ..$ RE_S31:List of 4
##   .. ..$ position: int [1:6] 31 32 33 34 35 36
##   .. ..$ sequence: chr "CTAGGG"
##   .. ..$ enzime  : chr "CCATGG"
##   .. ..$ type    : chr "reverse direction enzime, DNA strand"
##   ..$ E_CS31:List of 4
##   .. ..$ position: int [1:6] 31 32 33 34 35 36
##   .. ..$ sequence: chr "GATCCC"
##   .. ..$ enzime  : chr "GGTACC"
##   .. ..$ type    : chr "same direction enzime, complement DNA strand"
##  $ BamHI  :List of 2
##   ..$ RE_S31:List of 4
##   .. ..$ position: int [1:6] 31 32 33 34 35 36
##   .. ..$ sequence: chr "CTAGGG"
##   .. ..$ enzime  : chr "CCTAGG"
##   .. ..$ type    : chr "reverse direction enzime, DNA strand"
##   ..$ E_CS31:List of 4
##   .. ..$ position: int [1:6] 31 32 33 34 35 36
##   .. ..$ sequence: chr "GATCCC"
##   .. ..$ enzime  : chr "GGATCC"
##   .. ..$ type    : chr "same direction enzime, complement DNA strand"
##  $ EcoRV  :List of 2
##   ..$ RE_S31:List of 4
##   .. ..$ position: int [1:6] 31 32 33 34 35 36
##   .. ..$ sequence: chr "CTAGGG"
##   .. ..$ enzime  : chr "CTATAG"
##   .. ..$ type    : chr "reverse direction enzime, DNA strand"
##   ..$ E_CS31:List of 4
##   .. ..$ position: int [1:6] 31 32 33 34 35 36
##   .. ..$ sequence: chr "GATCCC"
##   .. ..$ enzime  : chr "GATATC"
##   .. ..$ type    : chr "same direction enzime, complement DNA strand"
##  $ NotI   :List of 2
##   ..$ E_S10:List of 4
##   .. ..$ position: int [1:6] 10 11 12 13 14 15
##   .. ..$ sequence: chr "CGCCCA"
##   .. ..$ enzime  : chr "CGGCCG"
##   .. ..$ type    : chr "same direction enzime, DNA strand"
##   ..$ E_S22:List of 4
##   .. ..$ position: int [1:6] 22 23 24 25 26 27
##   .. ..$ sequence: chr "CGGGAG"
##   .. ..$ enzime  : chr "CGGCCG"
##   .. ..$ type    : chr "same direction enzime, DNA strand"
##  $ XhoI   :List of 3
##   ..$ E_S22  :List of 4
##   .. ..$ position: int [1:6] 22 23 24 25 26 27
##   .. ..$ sequence: chr "CGGGAG"
##   .. ..$ enzime  : chr "CTCGAG"
```

```
##   .. ..$ type    : chr "same direction enzime, DNA strand"
##   ..$ E_S31  :List of 4
##   .. ..$ position: int [1:6] 31 32 33 34 35 36
##   .. ..$ sequence: chr "CTAGGG"
##   .. ..$ enzime  : chr "CTCGAG"
##   .. ..$ type    : chr "same direction enzime, DNA strand"
##   ..$ RE_CS31:List of 4
##   .. ..$ position: int [1:6] 31 32 33 34 35 36
##   .. ..$ sequence: chr "GATCCC"
##   .. ..$ enzime  : chr "GAGCTC"
##   .. ..$ type    : chr "reverse direction enzime, complement DNA strand"
##  $ XbaI   : list()
```

```
#str(lapply(site_mut, find_DNA_match, begining_seq, 3, "start"))

#for the final seq, fin the closest to the begining
#find the closest to 1

str(lapply(site_mut, find_DNA_match, final_seq, 1, "final"))
```

```
## List of 9
##  $ HindIII: list()
##  $ EcoRI  : list()
##  $ Asp718I: list()
##  $ KpnI   : list()
##  $ BamHI  : list()
##  $ EcoRV  : list()
##  $ NotI   : list()
##  $ XhoI   : list()
##  $ XbaI   : list()
```

```
str(lapply(site_mut, find_DNA_match, final_seq, 2, "final"))
```

```
## List of 9
##  $ HindIII: list()
##  $ EcoRI  :List of 2
##   ..$ E_S1  :List of 4
##   .. ..$ position: int [1:6] 1 2 3 4 5 6
##   .. ..$ sequence: chr "TAGTTC"
##   .. ..$ enzime  : chr "GAATTC"
##   .. ..$ type    : chr "same direction enzime, DNA strand"
##   ..$ RE_CS1:List of 4
##   .. ..$ position: int [1:6] 1 2 3 4 5 6
##   .. ..$ sequence: chr "ATCAAG"
##   .. ..$ enzime  : chr "CTTAAG"
##   .. ..$ type    : chr "reverse direction enzime, complement DNA strand"
##  $ Asp718I: list()
##  $ KpnI   : list()
##  $ BamHI  : list()
##  $ EcoRV  : list()
##  $ NotI   : list()
##  $ XhoI   :List of 2
##   ..$ RE_S1:List of 4
##   .. ..$ position: int [1:6] 1 2 3 4 5 6
##   .. ..$ sequence: chr "TAGTTC"
##   .. ..$ enzime  : chr "GAGCTC"
##   .. ..$ type    : chr "reverse direction enzime, DNA strand"
##   ..$ E_CS1:List of 4
##   .. ..$ position: int [1:6] 1 2 3 4 5 6
##   .. ..$ sequence: chr "ATCAAG"
##   .. ..$ enzime  : chr "CTCGAG"
##   .. ..$ type    : chr "same direction enzime, complement DNA strand"
##  $ XbaI   : list()
```

```
#str(lapply(site_mut, find_DNA_match, final_seq, 3, "final"))
```

the exercise

“Introduce in each sequence a restriction site in order to clone the gene in the proper Open Reading Frame (ORF) and right orientation.

For the end with 2 substitutions EcoRI->same direction enzime, DNA strand (is after the end marks) XhoI ->reverse direction enzime, DNA strand

For the start, position 36 with 2 substitutions: KpnI-> “reverse direction enzime, DNA strand” BamHI->“reverse direction enzime, DNA strand” EcoRV-> reverse direction enzime, DNA strand XhoI ->same direction enzime, DNA strand

If XhoI were the end restriction site, it would cut also at the begining because it fits correctly on the two sites.

only 1 posibility: If EcoRI was the end restriction site, BamHI is the only that fits and they go oposite directions on the DNA strand. BamHi is the only that is before EcoRI, the others are after meaning it would paste the DNA backwards, so the START signal would be at the end.

Conclusion With 2 substitutions, the best restriction site enzimes would be EcoRI & BamHI.

ADITIONAL INFORMATION: I also analized for 3 substitutions, and EcoRI fits with 3 substitutions at the begining, so there is still a chance that it could be pasted backwards. However, it is the best fit.