Project 4

Santiago Castro Dau, June Monge, Rachita Kumar, Sarah Lötscher

3/24/2022

Problem 8: Profile HMMs: Estimating match emission probabilities

What are the estimated match emission probabilities of the profile HMM in Figure 1?

The model

```
## The Example Data

bat <- c("A","G","-","-","-","C")
rat <- c("A","-","A","G","-","C")
cat <- c("A","G","-","-","-","C")
gnat <- c("-","G","A","A","A","C")
goat <- c("A","G","-","-","A","C")
M <- rbind(bat,rat,cat,gnat,goat)

# Alphabet
A=c("A","C","G","T")
```

Find out which positions are in Match state and which are in Insert state

```
# Get the positions which are Insertstate and Match state
match = which(colSums(M!="-")>(dim(M)[1]/2)) #Vector with Match positions
insertion = c(1:dim(M)[2])[!c(1:dim(M)[2]) %in% match] #Vector with insert positions
```

```
#Matrix of all pos????????????????
E=matrix(,0,dim(M)[2])
for (n in A){
  e<-apply(M==n,2,sum)
  E <- rbind(E,e)
}
rownames(E)=A
```

Calculate the Emission probabilities of a match

```
# Emission probabilities of match
E_match_prob=t(t(E[,match]+1)/colSums(E[,match]+1))
E_match_prob
```

```
##    [,1]  [,2]      [,3]
## A 0.625 0.125 0.1111111
## C 0.125 0.125 0.6666667
## G 0.125 0.625 0.1111111
## T 0.125 0.125 0.1111111
```

Problem 9: Estimating insert emission probabilities

What are the estimated insert emission probabilities of the profile HMM in Figure 1?

```r
consecutive_in=split(insertion, cumsum(c(1, diff(insertion) != 1)))
E_cons=matrix(,length(A),0)
for (insert in consecutive_in){
  s = rowSums(E[,insert])
  E_cons = cbind(E_cons,s)
}
rownames(E_cons)=A
colnames(E_cons)=c(1:ncol(E_cons))

# Emission probabilities of insertion
E_insertion_prob=t(t(E_cons+1)/colSums(E_cons+1))
E_insertion_prob
```

```
##     1
## A 0.6
## C 0.1
## G 0.2
## T 0.1
```

Problem 10: Estimating transition probabilities

What are the estimated transmission probabilities in the profile HMM in Figure 1?

```r
#Transition probabilities
counts.only=FALSE

#Get state Matrix
SM=matrix(FALSE,dim(M)[1],dim(M)[2])
SM[,match][M[,match]!='-']='M'
SM[,match][M[,match]=='-']='D'
SM[,insertion][M[,insertion]!='-']='I'

#Initate Transition Matrix
transitions <- c("MM","MD","MI","IM","ID","II","DM","DD","DI")
Tr=matrix(0,length(transitions),dim(M)[2]+1)
rownames(Tr)=transitions


#Create the Transition counts for every possible szenario
for (i in 1:dim(SM)[1]){
  prev <- 'M'
  prevStateNum <- 0
  for(j in 1:(dim(SM)[2])) {
    newState<- SM[i,j]
    if (newState==FALSE){
      prevStateNum <- j
      next
    }
    transition <- paste(prev, newState, sep="")
    Tr[transition, prevStateNum +1] <- Tr[transition, prevStateNum +1] +1
    prevStateNum <- j
    prev <- newState
  }
  transition<- paste(prev, 'M', sep="")
  Tr[transition, prevStateNum +1] <- Tr[transition, prevStateNum +1] +1
}

#Add the Insertion state columns to the corresponding match state
for (insert in rev(consecutive_in)){
  s = rowSums(Tr[,c(insert[1]-1,insert)])
  Tr[,insert[1]-1]=s
  Tr=Tr[,-insert] #Delete the insertion columns
}


#Add pseudo-count and Get the Transition probabilities
if (!counts.only) {
    Tr <- Tr+1
    for(i in 1:ncol(Tr)) {
      Tr[1:3,i] <- Tr[1:3,i] / sum(Tr[1:3,i])
      Tr[4:6,i] <- Tr[4:6,i] / sum(Tr[4:6,i])
      Tr[7:9,i] <- Tr[7:9,i] / sum(Tr[7:9,i])
    }
}

colnames(Tr)=c(0:(ncol(Tr)-1))
Tr
```

```
##            0         1         2         3
## MM 0.6250000 0.4444444 0.6000000 0.7500000
## MD 0.2500000 0.2222222 0.2000000 0.1250000
## MI 0.1250000 0.3333333 0.2000000 0.1250000
## IM 0.3333333 0.1666667 0.6666667 0.3333333
## ID 0.3333333 0.1666667 0.1666667 0.3333333
## II 0.3333333 0.6666667 0.1666667 0.3333333
## DM 0.3333333 0.4000000 0.3333333 0.3333333
## DD 0.3333333 0.2000000 0.3333333 0.3333333
## DI 0.3333333 0.4000000 0.3333333 0.3333333
```

Problem 11: Protein family membership classification

(1) Import functions

```
#Get Functions
source("./code/profileHMM.R")
```

(2) Read the two alignments

```r
#Read in Data
GTP_binding_proteins = parseAlignment("./data/GTP_binding_proteins.txt")
ATPases = parseAlignment("./data/ATPases.txt")
```

(3) Parametrise two profile HMMs

Parametrise two profile HMMs for each protein family

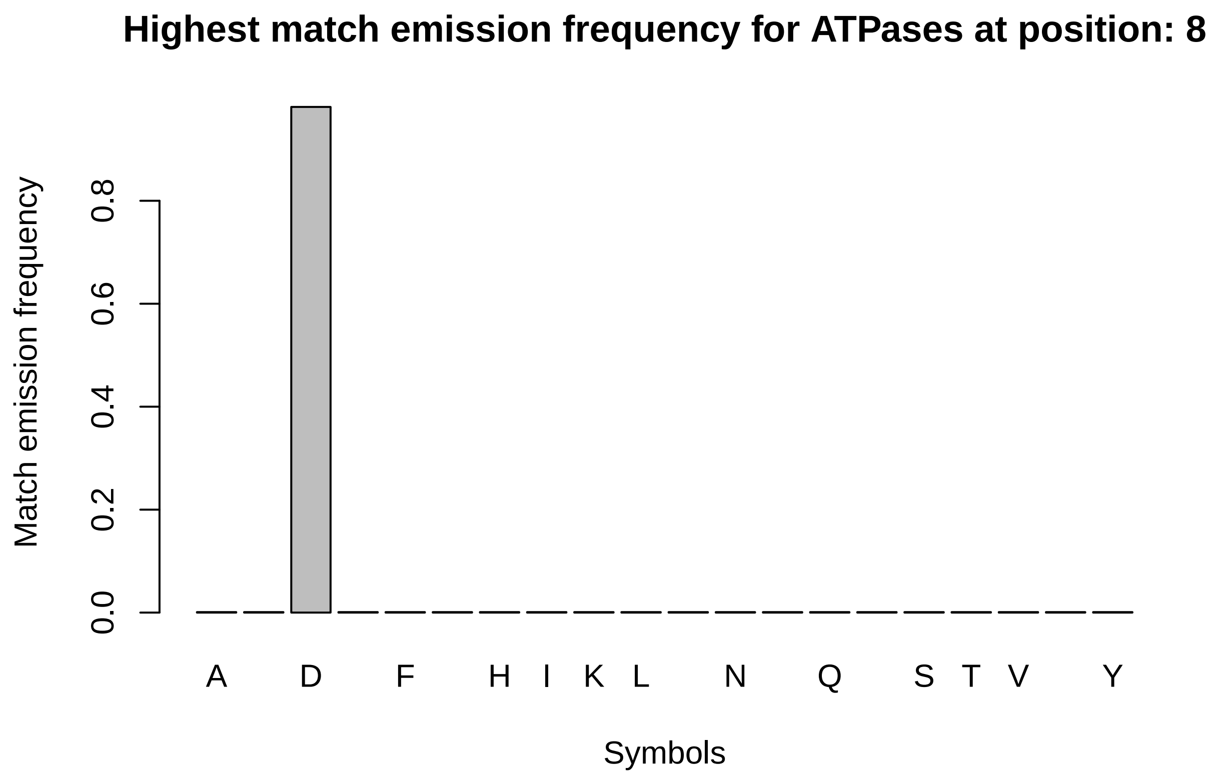
```r
#Get the profile HMMS
profileHMM_GTP = learnHMM(GTP_binding_proteins)
profileHMM_ATPases = learnHMM(ATPases)
```

(4) Highest match and highest insert emission frequencies for ATPases

```r
#Highest match emission frequencies
match_frequency_positions_ATPases =
  which(profileHMM_ATPases$mE == max(profileHMM_ATPases$mE,na.rm=TRUE),
        arr.ind = TRUE)[2]
print(match_frequency_positions_ATPases)
```

```
## [1] 8
```

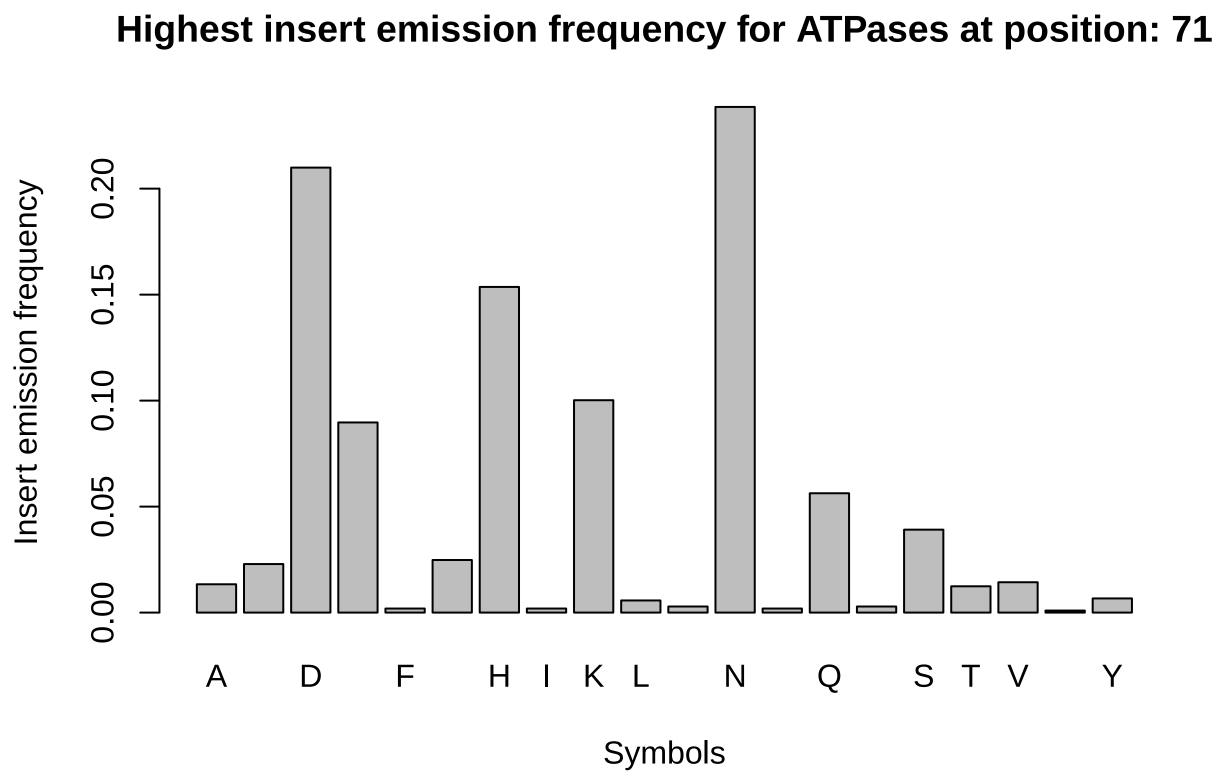
```r
barplot(profileHMM_ATPases$mE[,match_frequency_positions_ATPases],
        names.arg=rownames(profileHMM_ATPases),
        main = paste0("Highest match emission frequency for ATPases at position: ",
                      match_frequency_positions_ATPases),
        xlab = "Symbols",
        ylab = "Match emission frequency")
```



```
#Highest insert emission frequencies
insert_frequency_positions_ATPases =
  which(profileHMM_ATPases$iE == max(profileHMM_ATPases$iE,na.rm=TRUE),
        arr.ind = TRUE)[2]
print(insert_frequency_positions_ATPases)
```

```
## [1] 71
```

```
barplot(profileHMM_ATPases$mE[,insert_frequency_positions_ATPases],
        names.arg=rownames(profileHMM_ATPases),
        main = paste0("Highest insert emission frequency for ATPases at position: ",
                      insert_frequency_positions_ATPases),
        xlab = "Symbols",
        ylab = "Insert emission frequency")
```

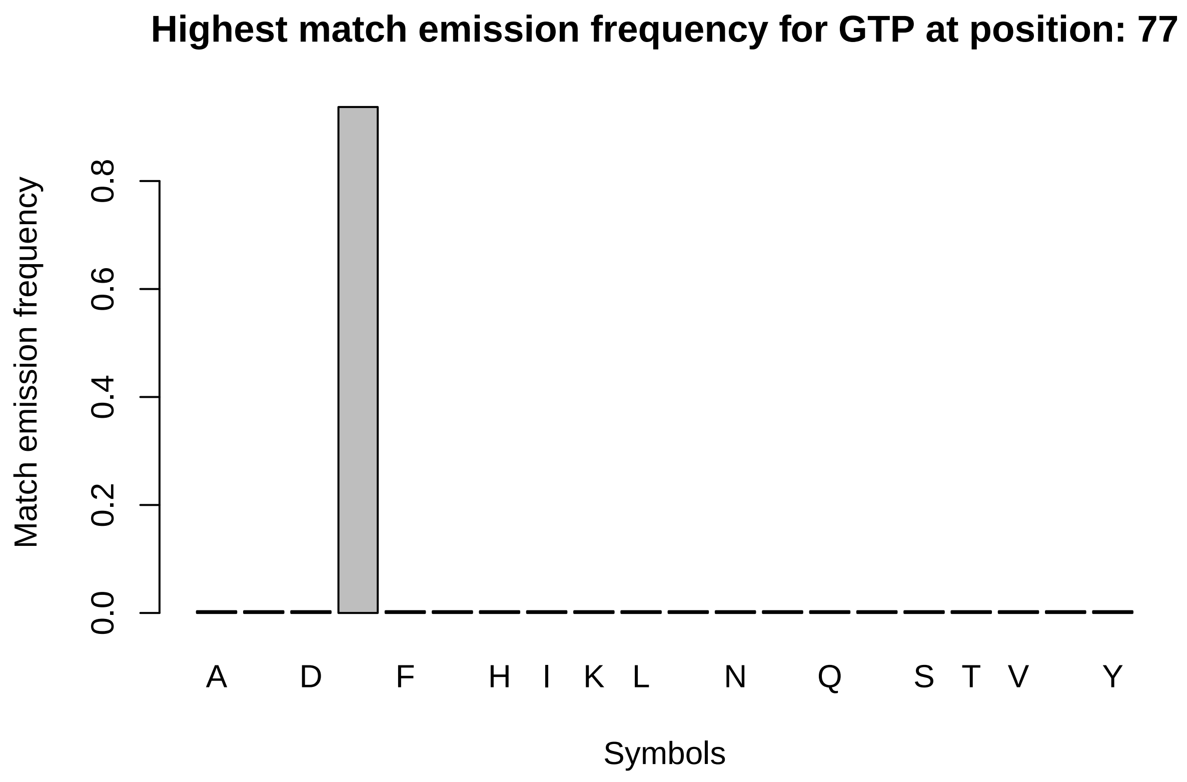


(4) Highest match and highest insert emission frequencies for GTP

```
#Highest match emission frequencies
match_frequency_positions_GTP =
  which(profileHMM_GTP$mE == max(profileHMM_GTP$mE,na.rm=TRUE),
                                      arr.ind = TRUE)[2]
print(match_frequency_positions_GTP)
```

```
## [1] 77
```

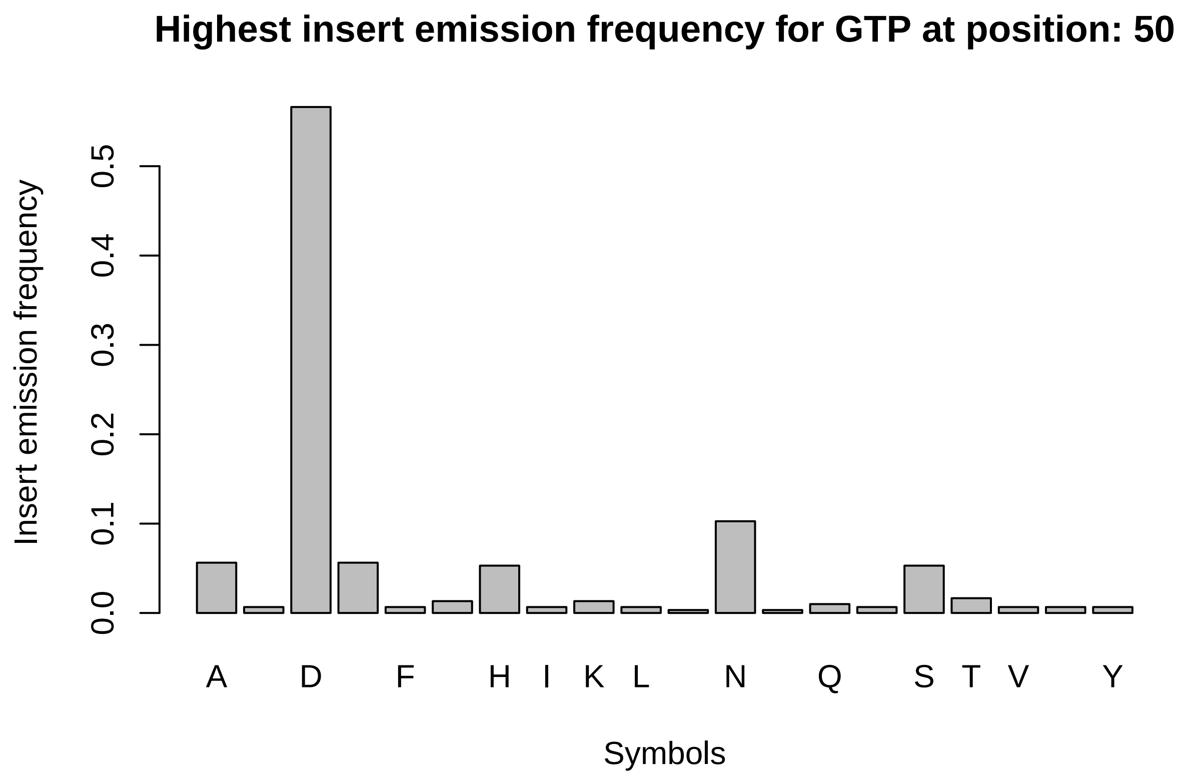
```
barplot(profileHMM_GTP$mE[,match_frequency_positions_GTP],
        names.arg=rownames(profileHMM_GTP),
        main = paste0("Highest match emission frequency for GTP at position: ",
                      match_frequency_positions_GTP),
        xlab = "Symbols",
        ylab = "Match emission frequency")
```



```r
#Highest insert emission frequencies
insert_frequency_positions_GTP =
  which(profileHMM_GTP$iE == max(profileHMM_GTP$iE,na.rm=TRUE),
                                       arr.ind = TRUE)[2]
print(insert_frequency_positions_GTP)
```

```
## [1] 50
```

```r
barplot(profileHMM_GTP$mE[,insert_frequency_positions_GTP],
        names.arg=rownames(profileHMM_GTP),
        main = paste0("Highest insert emission frequency for GTP at position: ",
               insert_frequency_positions_GTP),
        xlab = "Symbols",
        ylab = "Insert emission frequency")
```



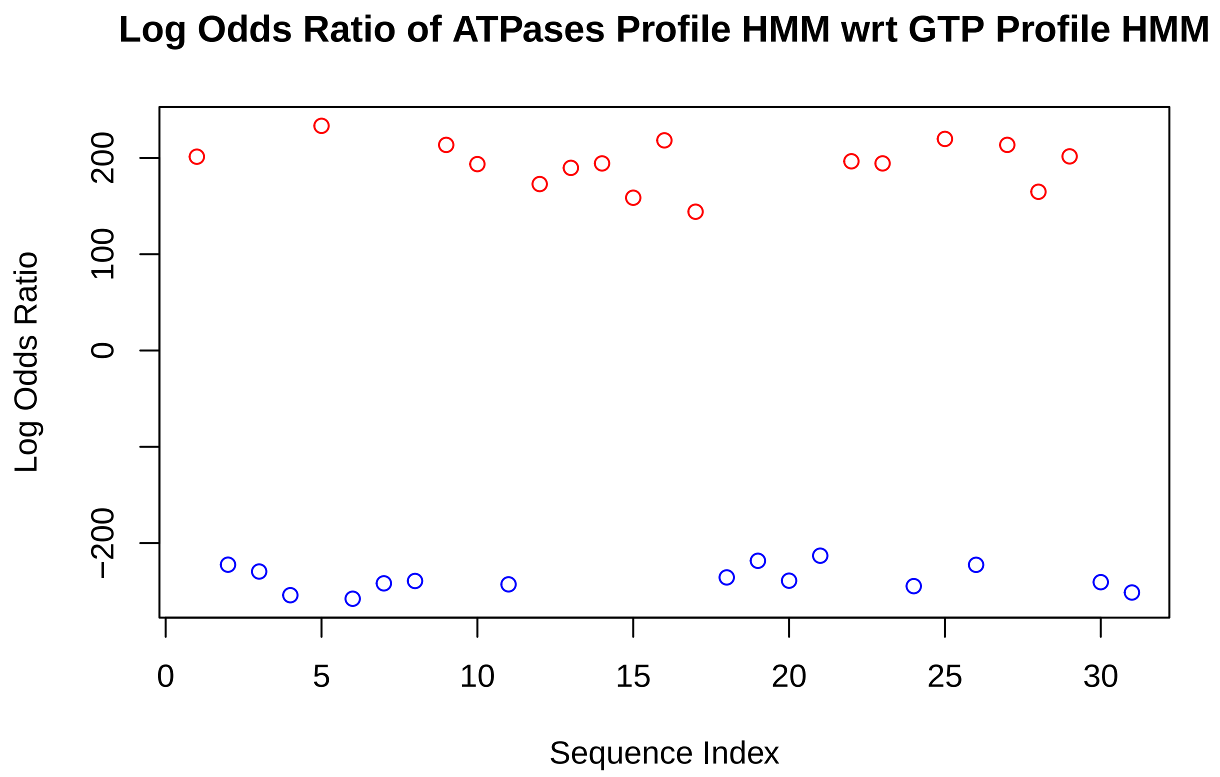
(5) Load the protein sequences into a list

```r
#Load the protein sequence
unclassifiedProteins <- parseProteins(proteinsFile =
                                        "./data/Unclassified_proteins.txt" )
```

(6) Obtain the log odds ratio and plot the results

```r
P_ATPases <- lapply(X = unclassifiedProteins, forward, HMM=profileHMM_ATPases)
P_GTP <- lapply(X = unclassifiedProteins, forward, HMM=profileHMM_GTP)

#Log-Odds Ratio of ATPases Profile HMM wrt GTP Profile HMM
q <- unlist(P_ATPases)-unlist(P_GTP)
plot(q,col=ifelse(q>=0,"red","blue"),
     main="Log Odds Ratio of ATPases Profile HMM wrt GTP Profile HMM",
     ylab = "Log Odds Ratio", xlab = "Sequence Index")
```



```
unclassifiedATPases = as.numeric(which(q >=0,arr.ind = TRUE))
no_unclassifiedATPases = length(unclassifiedATPases)
print(paste0("The number of predicted ATPases is ",no_unclassifiedATPases))
```

```
## [1] "The number of predicted ATPases is 16"
```

```
print("The unclassified proteins that are ATPases have indices")
```

```
## [1] "The unclassified proteins that are ATPases have indices"
```

```
print(unclassifiedATPases)
```

```
##  [1]  1  5  9 10 12 13 14 15 16 17 22 23 25 27 28 29
```

```
unclassifiedGTP = as.numeric(which(q < 0,arr.ind = TRUE))
no_unclassifiedGTP = length(unclassifiedGTP)
print(paste0("The number of predicted GTPs is ",no_unclassifiedGTP))
```

```
## [1] "The number of predicted GTPs is 15"
```

```
print("The unclassified proteins that are GTPs have indices")
```

```
## [1] "The unclassified proteins that are GTPs have indices"
```

```r
print(unclassifiedGTP)
```

```
##  [1]  2  3  4  6  7  8 11 18 19 20 21 24 26 30 31
```

```r
print("Yes, for each protein there is a clear answer since
      the log odds ratio is strictly above or below zero.")
```

```
## [1] "Yes, for each protein there is a clear answer since \n      the log odds ratio is strictly abov
```