RNA Splicing Classifier

Miguel Ángel Juárez Garzón

Sequence Extraction

```
dfall<-read.csv("splice.txt")
dfall[]<-lapply(dfall,as.character)
mall = length(dfall[,1]) # Number of instances
mall
```

```
## [1] 3190
```

Sequence pre-processing

```
toRemove <- c() # Remove uncertain base pairs in the data frame
j<-1
for (i in 1:mall){
enes <- lapply(strsplit(dfall[i,3], ""), function(x) which(x == "N"))
des <- lapply(strsplit(dfall[i,3], ""), function(x) which(x == "D"))
eses <- lapply(strsplit(dfall[i,3], ""), function(x) which(x == "S"))
erres <- lapply(strsplit(dfall[i,3], ""), function(x) which(x == "R"))
if (length(enes[[1]])>0 || length(des[[1]])>0
|| length(eses[[1]])>0 || length(erres[[1]])>0){
toRemove[j]<-i
j<-j+1
}
}
dfall<-dfall[-toRemove,]
mall <- length(dfall[,1]) # Number of instances after filtering
mall
```

```
## [1] 3175
```

Exercise 1: Generation of data sets

```
#Random sample
sample_splice <- sample(mall,0.7 * mall,replace = FALSE)
# Creating training and test dataset
splice_training <- dfall[sample_splice,]
splice_test <- dfall[-sample_splice,]
```

Training Set

```
splice_training
```

	Cl…	Instance	Sequence
	<chr>	<chr>	<chr>
2054	N	HUMF13A05-NEG-121	GGGGTAATTTTTTATGGAGAGGTCAATGACATCAAGACC
2974	N	HUMSTATHG2-NEG-1	AAGCTTTACTTTCTTCTTTACCTTCATACCCCTATTCTCTT
1797	N	HUMBLAST1-NEG-661	GTCTGCCTCAGTCCACCCTGTACCCTGGCCCGGTCCTT
1279	IE	HUMMHCW1B-ACCEPTOR-2658	AGGGCCCCTCACCTTCCCCTCCTTTCCCAGAGCCGTCT
625	EI	HUMPSAP-DONOR-170	GACCCAAGCAGCTGGAGGCTCTGTGTGTGGGTGAGTTT
1478	IE	HUMTNFAB-ACCEPTOR-5270	TCAGCTTTTTCTTTTCTCTCTCCTCTTCAGGATCATCTTC
2670	N	HUMMXB-NEG-1741	ACCAAACTGTTCAGAGCACGATTGAAGACATAAAAGTGA
791	IE	HUMA1GLY2-ACCEPTOR-3502	CCCCAGTCAGTCTCCTTGCTCCCCCTGCAGCTGACAAG
1394	IE	HUMPSAP-ACCEPTOR-1501	CCTACACATCCATGTCTCTTTTCTCTGCAGGCCCCATGG
2602	N	HUMMHCW1B-NEG-241	AATATTACCTGAGGTAAGGTAAGGCAAAGAGTGGGAGGC

1-10 of 2,222 rows Previous **1** 2 3 4 5 6 … 223 Next

```
# Table for training data
train_data_info <- table(splice_training$Class)
train_data_info
```

```
##
##   EI   IE    N
##  549  547 1126
```

Test Set

```
# Table for test data
test_data_info <- table(splice_test$Class)
test_data_info
```

```
##
##  EI  IE   N
## 213 218 522
```

Verosimility Estimation

```
prob_calc <- function(clase, bn, tot_class){
# Read the training set. Count the frequency of base A in each
# position for all sequences classified as IE
# m is the number of sequences in the training set
# The dataframe df refers to the training set
  n <- 60
  prob <- rep(0,n)
  m <- nrow(splice_training)

  for (i in 1:m){
    if (splice_training[i,1]==clase){
    aes <- lapply(strsplit(splice_training[i,3], ""), function(x) which(x == bn))
    prob[aes[[1]]] <- prob[aes[[1]]]+1
    }
  }
  prob <- prob/tot_class
}

prob_IE_A <- prob_calc ("IE", "A", train_data_info[2])
prob_IE_A
```

```
##  [1] 0.208409506 0.206581353 0.219378428 0.204753199 0.223034735 0.226691042
##  [7] 0.232175503 0.257769653 0.213893967 0.188299817 0.193784278 0.171846435
## [13] 0.148080439 0.151736746 0.131627057 0.124314442 0.100548446 0.082266910
## [19] 0.076782450 0.074954296 0.087751371 0.106032907 0.060329068 0.080438757
## [25] 0.054844607 0.073126143 0.234003656 0.021937843 0.996343693 0.001828154
## [31] 0.283363803 0.234003656 0.234003656 0.246800731 0.239488117 0.208409506
## [37] 0.255941499 0.228519196 0.224862888 0.244972578 0.250457038 0.228519196
## [43] 0.244972578 0.219378428 0.299817185 0.235831810 0.235831810 0.212065814
## [49] 0.265082267 0.248628885 0.250457038 0.285191956 0.244972578 0.212065814
## [55] 0.310786106 0.243144424 0.268738574 0.268738574 0.261425960 0.201096892
```

Exercise 2: Classifier Training

```
#Verosimility for all the train dta set class and base combination
prob_IE_C <- prob_calc ("IE", "C", train_data_info[2])
prob_IE_G <- prob_calc ("IE", "G", train_data_info[2])
prob_IE_T <- prob_calc ("IE", "T", train_data_info[2])
prob_EI_A <- prob_calc ("EI", "A", train_data_info[1])
prob_EI_C <- prob_calc ("EI", "C", train_data_info[1])
prob_EI_G <- prob_calc ("EI", "G", train_data_info[1])
prob_EI_T <- prob_calc ("EI", "T", train_data_info[1])
prob_N_A <- prob_calc ("N", "A", train_data_info[3])
prob_N_C <- prob_calc ("N", "C", train_data_info[3])
prob_N_G <- prob_calc ("N", "G", train_data_info[3])
prob_N_T <- prob_calc ("N", "T", train_data_info[3])
```

```
# Class definition
splice_training.classes <- c("EI","IE","N")
splice_training.classes
```

```
## [1] "EI" "IE" "N"
```

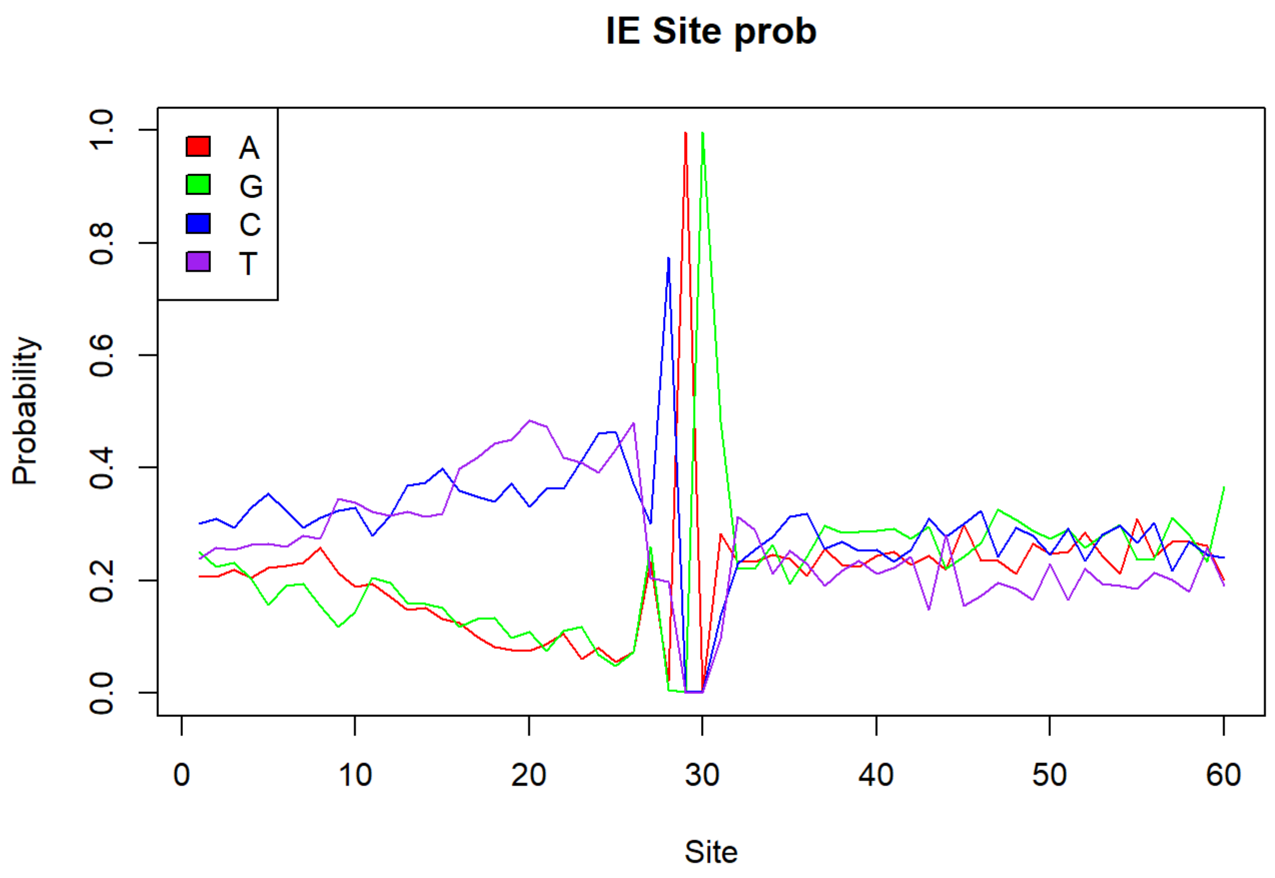
```
nclass <- length(splice_training.classes)
# Prior probabilities initialization
p.Cs <- c(0,0,0)

for (c in 1:nclass) {
  p.Cs[c] <- nrow(splice_training[splice_training$Class==splice_training.classes
[c],]) / nrow(splice_training)
}

p.Cs
```

```
## [1] 0.2470747 0.2461746 0.5067507
```

Exercise 3: Graphical Representation of Probabilities

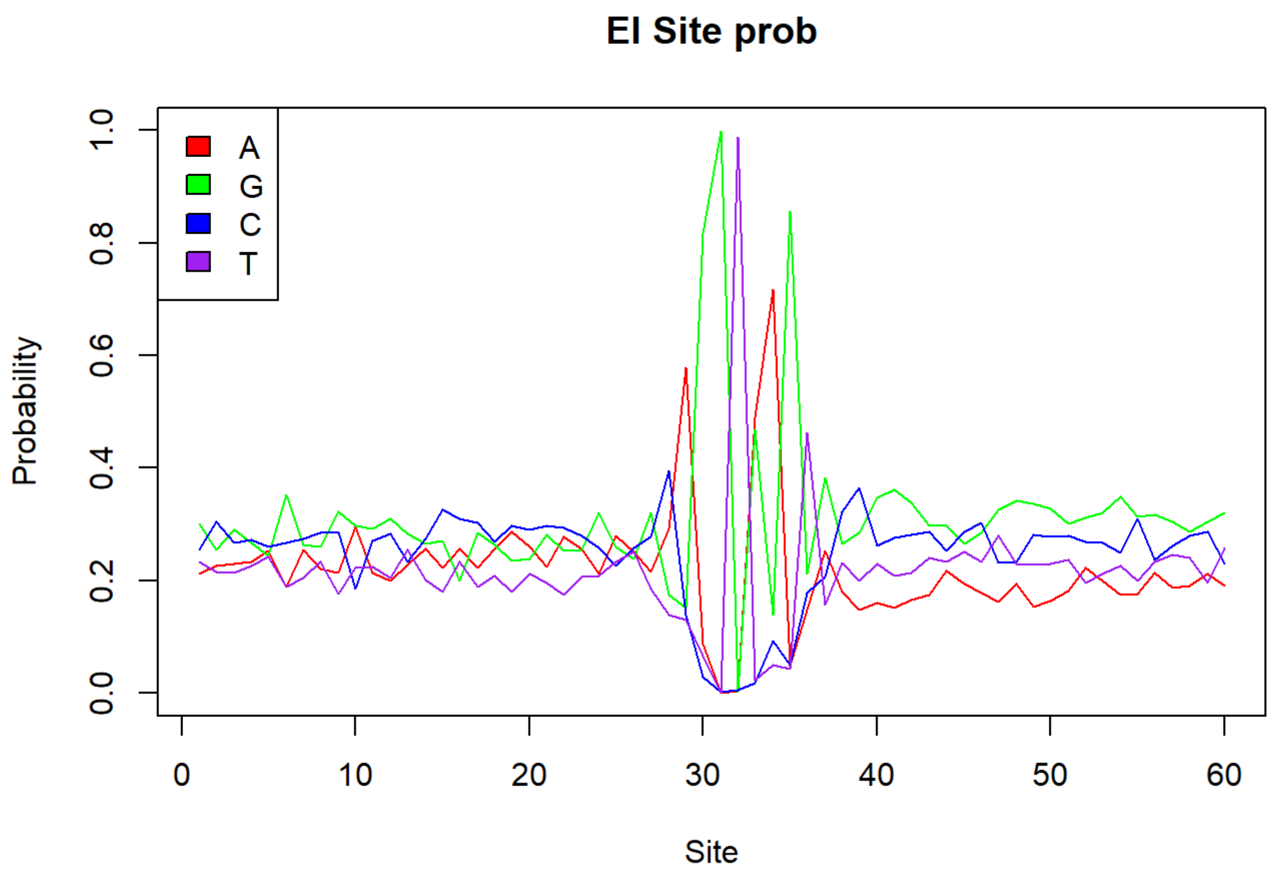
Plot IE

```
plot(1:60, prob_IE_A, ylim = c(0,1), col="red",type = "l" , ylab="Probability",xlab
="Site",
     main="IE Site prob")
lines(1:60, prob_IE_G, col="Green")
lines(1:60, prob_IE_C, col="Blue")
lines(1:60, prob_IE_T, col="Purple")
legend("topleft", c("A","G", "C", "T"),fill=c("red", "Green", "Blue", "Purple"))
```



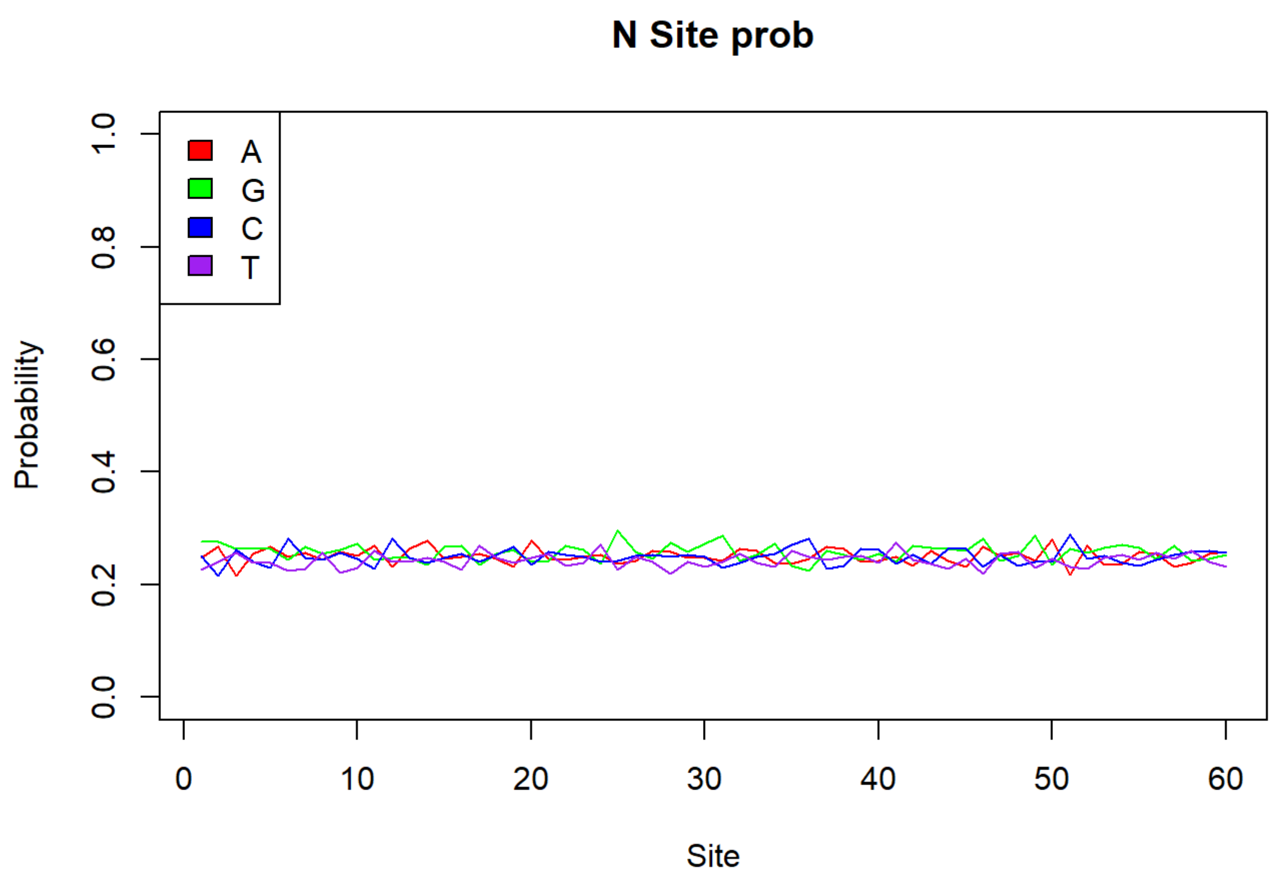
Plot EI

```
plot(1:60, prob_EI_A, ylim = c(0,1), col="red",type = "l" , ylab="Probability",xlab
="Site",
     main="EI Site prob")
lines(1:60, prob_EI_G, col="Green")
lines(1:60, prob_EI_C, col="Blue")
lines(1:60, prob_EI_T, col="Purple")
legend("topleft", c("A","G", "C", "T"),fill=c("red", "Green", "Blue", "Purple"))
```



Plot N

```
plot(1:60, prob_N_A, ylim = c(0,1) , col="red",type = "l" , ylab="Probability",xlab
="Site",
     main="N Site prob")
lines(1:60, prob_N_G, col="Green")
lines(1:60, prob_N_C, col="Blue")
lines(1:60, prob_N_T, col="Purple")
legend("topleft", c("A","G", "C", "T"),fill=c("red", "Green", "Blue", "Purple"))
```



Looking at the graphs, we can clearly see that there are highly conserved sequences in the intron-exon and exon-intron boundary sequences, while those sequences without boundaries are very heterogeneous throughout their sequence.

Exercise 4: Classification phase

```r
#Assigned class vector initialization
assigned_class <- c()
for (i in 1:sum(test_data_info)){
  EI_probs <- c()
  IE_probs <- c()
  N_probs <- c()
  seq <- strsplit(splice_test[i,3], "")
  #Probability for position on each class
  for (j in 1:60){
    if (seq[[1]][j] == "A"){
      EI_probs[j]<- prob_EI_A[[j]]
      IE_probs[j] <- prob_IE_A[[j]]
      N_probs[j]<- prob_N_A[[j]]
    } else if (seq[[1]][j] == "G"){
      EI_probs[j] <- prob_EI_G[[j]]
      IE_probs[j] <- prob_IE_G[[j]]
      N_probs[j] <- prob_N_G[[j]]
    } else if (seq[[1]][j] == "C"){
      EI_probs[j] <- prob_EI_C[[j]]
      IE_probs[j] <- prob_IE_C[[j]]
      N_probs[j]<- prob_N_C[[j]]
    } else if (seq[[1]][j] == "T"){
      EI_probs[j] <- prob_EI_T[[j]]
      IE_probs[j] <- prob_IE_T[[j]]
      N_probs[j] <- prob_N_T[[j]]
    }
  }

  logpc <- c()

  logpc[1] <- log(p.Cs[1])+sum(log(EI_probs))
  logpc[2] <- log(p.Cs[2])+sum(log(IE_probs))
  logpc[3] <- log(p.Cs[3])+sum(log(N_probs))

  #Class assignation
  assigned_class[i] <- splice_training.classes[which.max(logpc)]

}
```

Exercise 5: Confusion Matrix and Error Rate

```r
# Confusion Matrix
splice_test_labels <- dfall[-sample_splice,]$Class
confusion <- table(splice_test_labels, assigned_class, dnn=list('actual','predicte
d'))
confusion
```

```
##       predicted
## actual  EI  IE   N
##     EI 195   8  10
##     IE   5 206   7
##     N    9   5 508
```

```
#Error Rate Calculation
error_rate <- sum(sum(confusion-diag(diag(confusion))))/sum(colSums(confusion))
error_rate
```

```
## [1] 0.04616999
```

The error rate we obtain is low, around 0.06 - 0.04, depending on the simulation. This means that the classifier makes an error approximately 5% of the time, indicating that it makes few mistakes and can be considered a good classifier.