Project 5. Genomic Data Science for Epigenetics

Table of Contents

## The Outline for Project 5.

Project 5 aims to discover a set of genomic features that could explain and predict patterns of epigenetic status on the genome. Firstly, we will try to apply basic Bioconductor tools to extract the genomic land markers from the gene annotations. Then, we will analyze a published epigenetic modification data using the extracted annotations. Explicitly, we will investigate its consensus motifs, exon lengths, as well as its meta distributions on genes. In the end, we will try to use machine learning to develop a prediction tool for the epigenetic modification, and see if we can integrate the genome topologies discovered from the EDA to enhance its prediction performance.

This project tries to reproduce the key findings in the following papers, please check them if you need more background information:

*1.Dominissini D, Moshitch-Moshkovitz S, Schwartz S, Salmon-Divon M, Ungar L, Osenberg S, Cesarkas K, Jacob-Hirsch J, Amariglio N, Kupiec M: Topology of the human and mouse m6A RNA methylomes revealed by m6A-seq. Nature 2012, 485(7397):201-206.*

*2.Meyer KD, Yogesh S, Paul Z, Olivier E, Mason CE, Jaffrey SR: Comprehensive analysis of mRNA methylation reveals enrichment in 3’ UTRs and near stop codons. Cell 2012, 149(7):1635–1646.*

*3.Chen K, Wei Z, Zhang Q, Wu X, Rong R, Lu Z, Su J, de Magalhães JP, Rigden DJ, Meng J: WHISTLE: a high-accuracy map of the human N6-methyladenosine (m6A) epitranscriptome predicted using a machine learning approach. Nucleic Acids Research 2019.*

## 1. Extract Transcriptomic Landmarkers

First, our goal is to extract some important genomic features for the later analysis. Retrieve the 4 transcriptomic landmarks shown below from the TxDb package of hg19.

* Transcription Start Sites (TSS).
* Transcription Ending Sites (TES).
* Stop Codon.
* Start Codon.

To achieve this, we need to rely on the combination of a set of “intra-range methods”. The extracted landmarks should be GRanges object with width = 1. Please store the results into the variables of TSS, TES, Start\_codon, and Stop\_codon, respectively.

Hint: transcripts() can extract the GRanges of full transcript (with introns) from TxDb object, and cdsBy() can retrieve a GRangesList that contains exons of CDS for each transcript / genes. Please try to understand these data structures, and be careful that the cdsBy() returns multiple exons of a CDS, not the full CDS with introns.

## === Hint code, fill the "\_\_\_" ============= ##  
# library(TxDb.\_\_\_.UCSC.\_\_\_.knownGene)  
# txdb\_hg19 <- TxDb.\_\_\_.UCSC.\_\_\_.knownGene  
# TSS <- \_\_\_(transcripts(txdb\_hg19), width = \_\_\_, fix = \_\_\_)  
# TES <- \_\_\_(transcripts(txdb\_hg19), width = \_\_\_, fix = \_\_\_)  
# Start\_codon <- \_\_\_(unlist(\_\_\_(cdsBy(txdb\_hg19, by = \_\_\_))), width = \_\_\_, fix = \_\_\_)  
# Stop\_codon <- \_\_\_(unlist(\_\_\_(cdsBy(txdb\_hg19, by = \_\_\_))),width = \_\_\_, fix = \_\_\_)  
# #Some extra hints, the missing functions above are all intra-range methods  
## ===== Enter your code below =============== ##  
library(TxDb.Hsapiens.UCSC.hg19.knownGene)  
txdb\_hg19 <- TxDb.Hsapiens.UCSC.hg19.knownGene  
TSS <- resize(transcripts(txdb\_hg19), width = 1, fix = "start")  
TES <- resize(transcripts(txdb\_hg19), width = 1, fix= "end" )  
Start\_codon <- resize(unlist(range(cdsBy(txdb\_hg19, by = "tx"))), width = 3, fix = "start")  
Stop\_codon <- resize(unlist(range(cdsBy(txdb\_hg19, by = "tx"))),width = 3, fix = "end")

unique(TSS)

## GRanges object with 51385 ranges and 2 metadata columns:  
## seqnames ranges strand | tx\_id tx\_name  
## <Rle> <IRanges> <Rle> | <integer> <character>  
## [1] chr1 11874 + | 1 uc001aaa.3  
## [2] chr1 69091 + | 4 uc001aal.1  
## [3] chr1 321084 + | 5 uc001aaq.2  
## [4] chr1 321146 + | 6 uc001aar.2  
## [5] chr1 322037 + | 7 uc009vjk.2  
## ... ... ... ... . ... ...  
## [51381] chrUn\_gl000237 2686 - | 82956 uc011mgu.1  
## [51382] chrUn\_gl000241 36875 - | 82957 uc011mgv.2  
## [51383] chrUn\_gl000243 11501 + | 82958 uc011mgw.1  
## [51384] chrUn\_gl000243 13608 + | 82959 uc022brq.1  
## [51385] chrUn\_gl000247 5816 - | 82960 uc022brr.1  
## -------  
## seqinfo: 93 sequences (1 circular) from hg19 genome

unique(TES)

## GRanges object with 46124 ranges and 2 metadata columns:  
## seqnames ranges strand | tx\_id tx\_name  
## <Rle> <IRanges> <Rle> | <integer> <character>  
## [1] chr1 14409 + | 1 uc001aaa.3  
## [2] chr1 70008 + | 4 uc001aal.1  
## [3] chr1 321115 + | 5 uc001aaq.2  
## [4] chr1 321207 + | 6 uc001aar.2  
## [5] chr1 326938 + | 7 uc009vjk.2  
## ... ... ... ... . ... ...  
## [46120] chrUn\_gl000237 1 - | 82956 uc011mgu.1  
## [46121] chrUn\_gl000241 20433 - | 82957 uc011mgv.2  
## [46122] chrUn\_gl000243 11530 + | 82958 uc011mgw.1  
## [46123] chrUn\_gl000243 13637 + | 82959 uc022brq.1  
## [46124] chrUn\_gl000247 5787 - | 82960 uc022brr.1  
## -------  
## seqinfo: 93 sequences (1 circular) from hg19 genome

* SAQ1: How many unique coordinates for TSS, TES, Start codons, and Stop codons exist on the genome assembly hg19? Is there more alternative usage for the start codons or stop codons? How about the alternative usage for the transcription starts and ends?
* Answer:

TSS 82960 TES 82960 start codon 63691 stop codon 63691

The start codon is the first codon of a messenger RNA (mRNA) transcript translated by a ribosome. The start codon always codes for methionine in eukaryotes and Archaea and a modified Met (fMet) in bacteria, mitochondria and plastids. The most common start codon is AUG (i.e., ATG in the corresponding DNA sequence)

The stop codon is a nucleotide triplet within messenger RNA that signals a termination of translation into proteins

The transcription start site is the location where the transcription of mRNA starts at the 5’-end of a gene sequence while the transcription end site is the location where transcription ends at the 3’-end of a gene sequence. They are all in the non-coding regions and function as control or influence the transcript process

### Examine the Sequence Around the Start and Stop Codons

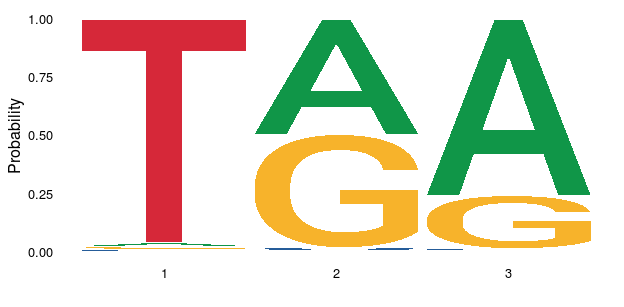
In this step, we will extract the sequences of the 3 bases windows centered by the Start and Stop condons, then, we will plot the seqeunce logo around the stop and start codons.

A sequence logo is a graphical representation of the conservation on amino acid or nucleic acid fragments. In R, the sequence logo can be plotted using function ggseqlogo() in the package ggseqlogo. The input of ggseqlogo() can be a string vector of nucleotide sequences that have the same string lengths. Please set the method = “prob”, so that it will display the proportion of bases at each nucleotide position.

## === Hint code, fill the "\_\_\_" ============= ##  
#library(ggseqlogo)  
#library(BSgenome.\_\_\_.UCSC.\_\_\_)  
#ggseqlogo(as.vector( \_\_\_( \_\_\_(Hsapiens,Start\_codon) ) ), method = "prob")  
#ggseqlogo(as.vector( \_\_\_( \_\_\_(Hsapiens,Stop\_codon) ) ), method = "prob")  
## ===== Enter your code below =============== ##  
library(ggseqlogo)  
library(BSgenome.Hsapiens.UCSC.hg19)  
ggseqlogo(as.vector( DNAStringSet( Views(Hsapiens,Start\_codon) ) ), method = "prob")



ggseqlogo(as.vector( DNAStringSet( Views(Hsapiens,Stop\_codon) ) ), method = "prob")



Afterward, check if the generated sequence logos being mostly consistent with the codon sequences in the common knowledge.

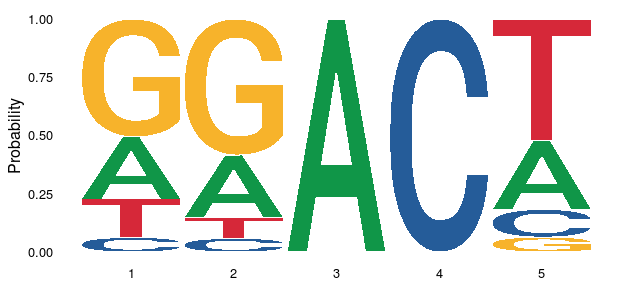
## 2. Identify Sequence Motif of an Epigenetic Modification

Subsequently, we will begin to analyze an epigenetic dataset downloaded from GEO. There is a BED file named GSE63753.bed under the project directory. The BED file was downloaded from the supplementary data of the GEO dataset [GSE63753](https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63753).

The GEO data is published by a study that detect the m6A modifications using miCLIP. miCLIP is a sequencing technology that can detect nucleotide modifications on the transcriptome at a single-base resolution. The file contains one of the processed miCLIP sample in the study, which reveals a single-based methylome of m6A in the HEK293 cell line.

Use the function rtraklayer::import() to load the BED file GSE63753.bed into R, and save the GRanges in a variable named m6A. Subsequently, extract the sequences under the 5bp window centered by the m6A modification. Using the same method of the previous step, plot the frequency of the nucleotides for the pentamer sequences.

## === Hint code, fill the "\_\_\_" ============= ##  
# m6A <- rtracklayer::import(\_\_\_)  
# sequence <- \_\_\_  
# ggseqlogo( sequence, \_\_\_ )  
# ===== Enter your code below =============== ##  
m6A <- rtracklayer::import("/home/yuxuan/BIO214/Project5/GSE63753.bed")  
sequence <- as.vector( DNAStringSet( Views(Hsapiens,m6A+2) ))  
ggseqlogo( sequence, method = "prob" )



* SAQ2: Check the IUPAC nucleotide ambiguity codes defined in the variable IUPAC\_CODE\_MAP. Then, describe a consensus motif using 5 letters, such that the motif you named can capture >70% of the sequences in the sequence logo plot. The answer cannot be too vague (such as NNNNN), the scope of your motif should be as narrow as possible in terms of the percentage among all possible base combinations of a pentamer. Please briefly explain your reasoning.
* Answer: DRACH As can be seen from the sequence logo picture, among the 5 base sequnce, the third and the forth bases are determined (3-A,4-C). Regarding to the other three bases, their sequence all contain four bases possibilities. Owing to ensure the scope is as narrow as possible, I therefore exclude the least possible base condition and retained the base D,R,H to 1st, 2rd,5th respectively. The proportion of the m6A matching in consensus sequnce “DRACH” is 0.773

You can evaluate the proportion of the m6A matching with:

IUPAC\_CODE\_MAP

## A C G T M R W S Y K V   
## "A" "C" "G" "T" "AC" "AG" "AT" "CG" "CT" "GT" "ACG"   
## H D B N   
## "ACT" "AGT" "CGT" "ACGT"

mean(vcountPattern("DRACH", DNAStringSet( Views( Hsapiens, m6A + 2) ), fixed = F)) #Change some bases in GGACA into ambiguity code so that it can include more motifs

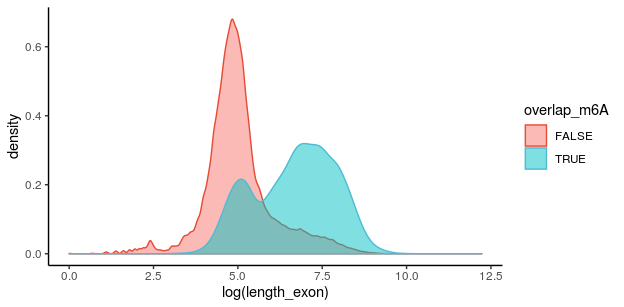
## [1] 0.7728607

## 3. Plot the Length of Exons

Next, we will explore the correlation between exon lengths and m6A modification. One of the EDA approaches is to plot the exon length distribution for exons containing or not containing the m6A sites.

Fill in the code below so that we will create 2 vectors of the same length. The first vector named length\_exon contains length of each exon on hg19, while the second vector named overlap\_m6A is a dummy variable (logic), and it will be true if the exon contains m6A. Then, run the code afterwards to plot the densities of exon lengths stratified by the overlapping status with the modification.

## === Hint code, fill the "\_\_\_" ============= ##  
#ex\_hg19 <- \_\_\_(\_\_\_) #extract exons of hg19 using exons()  
#length\_exon <- \_\_\_(\_\_\_) #extract the lengths for each of the exons  
#overlap\_m6A <- \_\_\_%\_\_\_%\_\_\_ #retrieve a dummy variable, true if exon overlapps with m6A  
# ===== Enter your code below =============== ##  
ex\_hg19 <- exons(txdb\_hg19) #extract exons of hg19 using exons()  
length\_exon <- width(ex\_hg19) #extract the lengths for each of the exons  
overlap\_m6A <- ex\_hg19%over%m6A #retrieve a dummy variable, true if exon overlapps with m6A  
# ===== Your code is finished ================ ##  
library(ggplot2)  
library(ggsci)  
ggplot(data.frame(length\_exon,overlap\_m6A)) +  
 geom\_density(aes(x = log(length\_exon),  
 colour = overlap\_m6A,  
 fill = overlap\_m6A), alpha = 0.5) +   
 theme\_classic() + scale\_color\_npg()



* SAQ3: Please interpret the density plot. What is the difference between the exons containing the modification and the exons without the modification? Are we more likely to observe m6A on long exons? If there is a boundary of exon length to classify the exons containing or not containing the m6A, what boundary would you choose?
* Answer: Generally, the exons containing the modification have longer length, about e^7 bp in length. On the contrary, the exons without the modification are about e^5 bp. Based on the figure, the average area (peak) of long exons tend to locate on the right side of the axis (blue plot), in other word, the m6a tend to be find in long exons. There is a boundary for exon length to classify the exons containing or not containing the m6A, I would choose e^6 bp, since there is an intersection about two lines.

## 4. Plot the Topology (Distribution) of Markers on Genes

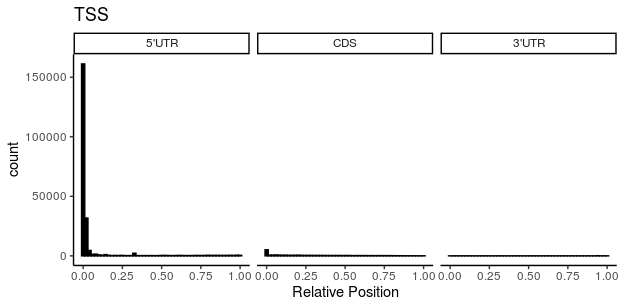
Next, we will draw a meta-gene plot. Specifically, the figure summarizes the spatial distribution of genomic features on transcript coordinates relative to start and the stop codons. You could refer to Figure 5D in Meyer KD.et.al Nature (2012) as an example.

The information we need to draw this graph is not complex: the only required values are the relative positions of each site on the 5’UTR, CDS, and 3’UTR. After calculating the relative positions, all we have to do is to draw 3 histograms of the relative position of each region, and the final distribution is just a combination of these 3 histograms.

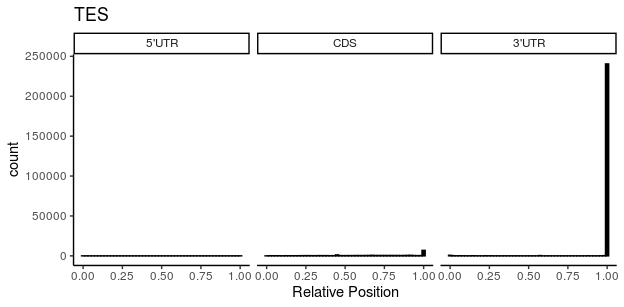
In the following code chunk, you need to create a function called relative\_pos\_on\_region(). The function will return the relative position of its input GRanges (parameter x) on the GRangesList (parameter region). The relative position on a region is defined as the distance toward the 5’ Start of the region divided by the full length of the region.

For example, if a site is located on the 200bp downstream of the start of a 5’UTR, and the length of the 5’UTR is 1000, then its relative position on 5’UTR is 200/1000 = 0.2. If some entries of x are not in the region, the function should omit those entries and only return the relative positions for entries mapped to the region.

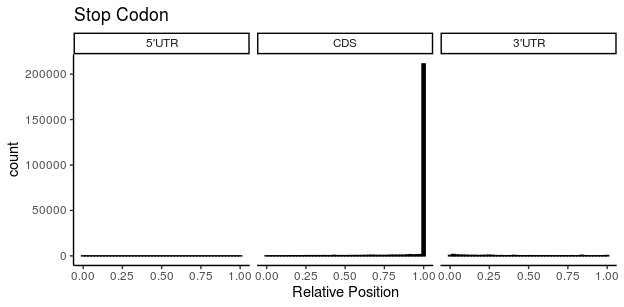
## === Hint code, fill the "\_\_\_" ============= ##  
#relative\_pos\_on\_region <- function(x, region){  
#region\_map <- map\_\_\_Transcripts(\_\_\_, \_\_\_)  
#region\_width <- sum(\_\_\_(\_\_\_))[region\_map$\_\_\_]  
#start\_on\_region <- \_\_\_(region\_map)  
#return(\_\_\_/)  
#}  
# ===== Enter your code below =============== ##  
library (GenomicAlignments)  
relative\_pos\_on\_region <- function(x, region){  
region\_map <- mapToTranscripts(x, region)  
region\_width <- sum(width(region))[region\_map$transcriptsHits]  
start\_on\_region <- start(region\_map)  
return(start\_on\_region/region\_width)  
}  
# ===== Your code is finished ================ ##  
plot\_tx <- function(marker,utr5\_gr,cds\_gr,utr3\_gr,marker\_name){  
  
 utr5\_pos <- relative\_pos\_on\_region(marker,utr5\_gr)  
 cds\_pos <- relative\_pos\_on\_region(marker,cds\_gr)  
 utr3\_pos <- relative\_pos\_on\_region(marker,utr3\_gr)  
  
 pldf <- data.frame(relative\_pos = c(utr5\_pos, cds\_pos, utr3\_pos),  
 tx\_region = rep(c("5'UTR","CDS","3'UTR"),  
 c(length(utr5\_pos),length(cds\_pos),length(utr3\_pos)))  
 )  
  
 pldf$tx\_region = factor(pldf$tx\_region,levels = c("5'UTR","CDS","3'UTR"))  
  
 ggplot(pldf) +  
 geom\_histogram(aes(x=relative\_pos),bins = 50,  
 colour = "black", fill = "black") +  
 facet\_wrap(~tx\_region) +  
 theme\_classic() +  
 labs(title = marker\_name, x = "Relative Position")  
} #What this function do is just organize the data and plot the histogram.  
  
#Extract the regions for 5'UTR, CDS, and 3'UTR  
UTR5 <- fiveUTRsByTranscript(txdb\_hg19)   
CDS <- cdsBy(txdb\_hg19, by = "tx")  
UTR3 <- threeUTRsByTranscript(txdb\_hg19)  
  
#Then we will generate a serious of plot for the topology of previously extracted tx landmarkers  
plot\_tx(TSS,UTR5,CDS,UTR3,"TSS")



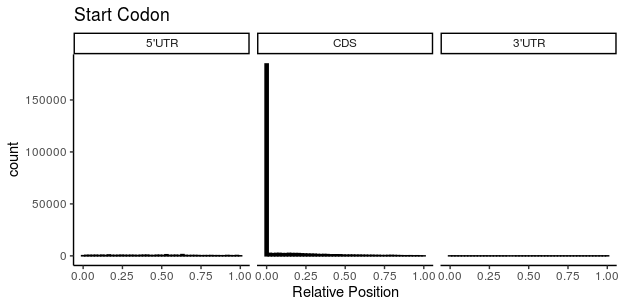
plot\_tx(TES,UTR5,CDS,UTR3,"TES")



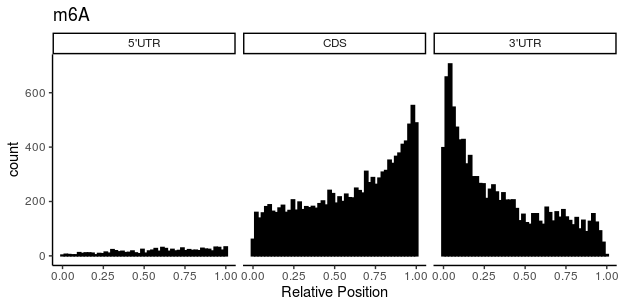
plot\_tx(Stop\_codon,UTR5,CDS,UTR3,"Stop Codon")



plot\_tx(Start\_codon,UTR5,CDS,UTR3,"Start Codon")



plot\_tx(m6A,UTR5,CDS,UTR3,"m6A")



* SAQ4: Please interpret the computed topology distributions for m6A. Which transcript regions are more likely to contain the modification sites? Are the relative positions on the regions important for m6A? Around which transcript landmarks (e.x. TSS, start codon) is the modification most enriched? Please explain your reasoning.
* Answer: CDS regions are most likely to contain the modification site The relative positions on the regions are important for m6A since the modifications sites would change with the relative position as shown in the figure The modification are enriched around the beginning of 5’UTR of TSS, the beginning of CDS of start codon, the end of 3’UTR of TES, the end of CDS of Stop codon. Because as shwon in the figure, the modifications enriched regions where the transcript landmarks work.

## 5. Engineering Genomic Features for Site Prediction of Epigentic Markers

In project 3, we introduced the unsupervised learning methods for the prediction of tissue labels from gene expression data. Now, we want to use the machine learning model again to predict the modification status on consensus motifs. In this case, we want to use a supervised learning methods, which means that the target of the prediction is provided to the model during the model training process.

Our training data is stored in the mcols() of m6A\_ml, which contains 1000 training instances for both the positive and negative data. The metadata is a DataFrame that has columns for target and sequence-based features. The sequence-based features are 30 variables extracted only from the nucleotide sequences surrounding the modification sites, and their calculation is NOT dependent on the gene annotations (from only BSgenome but not from TxDb). What you need to do is to create the following annotation based genomic features as the additional metadata columns in m6A\_ml:

* UTR5: a logical variable indicating the site overlapping with 5’UTR.
* CDS: a logical variable indicating the site overlapping with CDS.
* UTR3: a logical variable indicating the site overlapping with 3’UTR.
* long\_exon: a logical variable for sites overlapping with long exons (exon length > XXX, XXX is a boundary number choosen by you).
* Stop\_codon: a logical variable for sites overlapping with the XXX bp centered by a stop codon. (XXX is also choosen by you)
* UTR3\_pos: a real number value for the relative position of the site on 3’UTR (0 if not on 3’UTR).
* CDS\_pos: a real number value for the relative position of the site on CDS (0 if not on CDS).

Please fill the following code chunk to add the above-mentioned genomic features into the prediction models.

After adding the features, we will run a high-level package to automatically build 4 prediction models and report their performance using 5 folds cross-validation. The 4 models are a combination of 2 ML algorithms (SVM and RandomForest) and 2 feature types (sequence feature along and sequence + genomic features).

You will get full score for this part if the genomic features you engineered can lead to an AUROC of more than 0.70 in any one of the ML algorithms.

m6A\_ml <- readRDS("m6A\_ml.rds") #m6A\_ml is a GRanges, its metadata columns are features used in the prediction model  
  
## === Hint code, fill the "\_\_\_" ============= ##  
#m6A\_ml$UTR5 <- m6A\_ml %\_\_\_% \_\_\_ #add a dummy feature for 5'UTR  
#m6A\_ml$CDS <- m6A\_ml \_\_\_ \_\_\_ #add a dummy feature for 5'CDS  
#m6A\_ml$UTR3 <- m6A\_ml \_\_\_ \_\_\_ #add a dummy feature for 3'UTR  
#m6A\_ml$long\_exon <- m6A\_ml \_\_\_ \_\_\_ #add a dummy feature for long exon (> a boundary number)  
#m6A\_ml$Stop\_codon <- m6A\_ml \_\_\_ \_\_\_ #add a dummy feature for stop codon flanked by a choosen number  
#m6A\_ml$UTR3\_pos <- WhistleR:::relative\_pos\_map(m6A\_ml, UTR3, 0, F) #a feature for relative position on UTR3  
#m6A\_ml$CDS\_pos <- WhistleR:::relative\_pos\_map(m6A\_ml, CDS, 0, F) #a feature for relative position on CDS  
# ===== Enter your code below =============== ##  
m6A\_ml$UTR5 <- m6A\_ml %over% UTR5 #add a dummy feature for 5'UTR  
m6A\_ml$CDS <- m6A\_ml %over% CDS #add a dummy feature for 5'CDS  
m6A\_ml$UTR3 <- m6A\_ml %over% UTR3 #add a dummy feature for 3'UTR  
m6A\_ml$long\_exon <- m6A\_ml %over% ex\_hg19 #add a dummy feature for long exon (> a boundary number)  
m6A\_ml$Stop\_codon <- m6A\_ml %over% Stop\_codon#add a dummy feature for stop codon flanked by a choosen number  
m6A\_ml$UTR3\_pos <- WhistleR:::relative\_pos\_map(m6A\_ml, UTR3, 0, F) #a feature for relative position on UTR3  
m6A\_ml$CDS\_pos <- WhistleR:::relative\_pos\_map(m6A\_ml, CDS, 0, F) #a feature for relative position on CDS  
# ===== Your code is finished ================ ##  
library(perflite)  
library(knitr)  
set.seed(102)  
  
perf\_results <- performance\_class(  
 y = list(  
 target\_1 = as.factor(m6A\_ml$m6A),  
 target\_2 = as.factor(m6A\_ml$m6A)  
 ), #list of response vectors  
 X = list(  
 sequence\_feature = data.frame( mcols(m6A\_ml)[,2:31] ),  
 add\_genomic\_feature = data.frame( mcols(m6A\_ml)[,-1] )   
 ), #list of feature matrixes  
 k = 5, #number of folds in cross validation  
 p = 1, #number of parallel computation  
 cv\_f = c(svm = svm\_class,  
 randomForest = randomForest\_class) #list of classifier functions.  
)

## [1] "Start to test on learning method: svm"  
## [1] "Using data pairs: sequence\_feature and target\_1."  
## [1] "Fold 1 training..."  
## [1] "Fold 2 training..."  
## [1] "Fold 3 training..."  
## [1] "Fold 4 training..."  
## [1] "Fold 5 training..."  
## [1] "Using data pairs: add\_genomic\_feature and target\_2."  
## [1] "Fold 1 training..."  
## [1] "Fold 2 training..."  
## [1] "Fold 3 training..."  
## [1] "Fold 4 training..."  
## [1] "Fold 5 training..."  
## [1] "Start to test on learning method: randomForest"  
## [1] "Using data pairs: sequence\_feature and target\_1."  
## [1] "Fold 1 training..."  
## [1] "Fold 2 training..."  
## [1] "Fold 3 training..."  
## [1] "Fold 4 training..."  
## [1] "Fold 5 training..."  
## [1] "Using data pairs: add\_genomic\_feature and target\_2."  
## [1] "Fold 1 training..."  
## [1] "Fold 2 training..."  
## [1] "Fold 3 training..."  
## [1] "Fold 4 training..."  
## [1] "Fold 5 training..."

kable(perf\_results[[1]],   
 caption = names(perf\_results)[1])

svm

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | AUROC | ACC | ERR | SENS | SPEC | MCC |
| sequence\_feature | 0.6053 | 0.587 | 0.413 | 0.603 | 0.571 | 0.1741 |
| add\_genomic\_feature | 0.7080 | 0.661 | 0.339 | 0.627 | 0.695 | 0.3227 |

kable(perf\_results[[2]],  
 caption = names(perf\_results)[2])

randomForest

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | AUROC | ACC | ERR | SENS | SPEC | MCC |
| sequence\_feature | 0.6077 | 0.5770 | 0.4230 | 0.565 | 0.589 | 0.1540 |
| add\_genomic\_feature | 0.7057 | 0.6465 | 0.3535 | 0.617 | 0.676 | 0.2935 |

* SAQ5: For the prediction made by SVM, after adding genomic features (annotation-based), by how much is the AUROC improved compared with only using the sequence features? Can you explain such an improvement? If an annotation-based feature is important at improving the prediction performance, can we infer that the annotation is biologically/scientifically significant for the predicted epigenetic marker? Please explain your reasoning.
* Answer: After adding genomic features, the AUROC improved from 0.61-0.71. This improvement is resulted from the adding genomic features since it covers more influcing factors and analyzes the information in more perspective, rather than restricting to the sequence.

The annotation-based feature is important at improving the prediction performance so we can infer the annotation is biologically significant for the predicted epigenetic marker since many factors influence the identification, which suggests more features to be included.Plus, as can be seen in this experiment, the performance improved.

## Session Info

sessionInfo()

## R version 3.6.3 (2020-02-29)  
## Platform: x86\_64-pc-linux-gnu (64-bit)  
## Running under: Ubuntu 16.04.6 LTS  
##   
## Matrix products: default  
## BLAS: /usr/lib/libblas/libblas.so.3.6.0  
## LAPACK: /usr/lib/lapack/liblapack.so.3.6.0  
##   
## locale:  
## [1] LC\_CTYPE=en\_US.UTF-8 LC\_NUMERIC=C   
## [3] LC\_TIME=en\_US.UTF-8 LC\_COLLATE=en\_US.UTF-8   
## [5] LC\_MONETARY=en\_US.UTF-8 LC\_MESSAGES=en\_US.UTF-8   
## [7] LC\_PAPER=en\_US.UTF-8 LC\_NAME=C   
## [9] LC\_ADDRESS=C LC\_TELEPHONE=C   
## [11] LC\_MEASUREMENT=en\_US.UTF-8 LC\_IDENTIFICATION=C   
##   
## attached base packages:  
## [1] stats4 parallel stats graphics grDevices utils datasets   
## [8] methods base   
##   
## other attached packages:  
## [1] knitr\_1.28   
## [2] perflite\_1.0.0   
## [3] GenomicAlignments\_1.22.1   
## [4] Rsamtools\_2.2.1   
## [5] SummarizedExperiment\_1.16.1   
## [6] DelayedArray\_0.12.2   
## [7] BiocParallel\_1.20.1   
## [8] matrixStats\_0.55.0   
## [9] ggsci\_2.9   
## [10] ggplot2\_3.2.1   
## [11] BSgenome.Hsapiens.UCSC.hg19\_1.4.0   
## [12] BSgenome\_1.54.0   
## [13] rtracklayer\_1.46.0   
## [14] Biostrings\_2.54.0   
## [15] XVector\_0.26.0   
## [16] ggseqlogo\_0.1   
## [17] TxDb.Hsapiens.UCSC.hg19.knownGene\_3.2.2  
## [18] GenomicFeatures\_1.38.1   
## [19] AnnotationDbi\_1.48.0   
## [20] Biobase\_2.46.0   
## [21] GenomicRanges\_1.38.0   
## [22] GenomeInfoDb\_1.22.0   
## [23] IRanges\_2.20.2   
## [24] S4Vectors\_0.24.3   
## [25] BiocGenerics\_0.32.0   
##   
## loaded via a namespace (and not attached):  
## [1] colorspace\_1.4-1 class\_7.3-16   
## [3] farver\_2.0.3 WhistleR\_0.99.2   
## [5] bit64\_0.9-7 lubridate\_1.7.8   
## [7] prodlim\_2019.11.13 interactiveDisplayBase\_1.24.0  
## [9] codetools\_0.2-16 splines\_3.6.3   
## [11] pROC\_1.16.1 caret\_6.0-85   
## [13] dbplyr\_1.4.2 shiny\_1.4.0   
## [15] BiocManager\_1.30.10 compiler\_3.6.3   
## [17] httr\_1.4.1 assertthat\_0.2.1   
## [19] Matrix\_1.2-18 fastmap\_1.0.1   
## [21] lazyeval\_0.2.2 later\_1.0.0   
## [23] htmltools\_0.4.0 prettyunits\_1.1.1   
## [25] tools\_3.6.3 gtable\_0.3.0   
## [27] glue\_1.4.0 GenomeInfoDbData\_1.2.2   
## [29] reshape2\_1.4.3 dplyr\_0.8.4   
## [31] rappdirs\_0.3.1 Rcpp\_1.0.3   
## [33] vctrs\_0.2.2 nlme\_3.1-144   
## [35] iterators\_1.0.12 timeDate\_3043.102   
## [37] xfun\_0.12 gower\_0.2.1   
## [39] stringr\_1.4.0 mime\_0.9   
## [41] lifecycle\_0.2.0 XML\_3.99-0.3   
## [43] AnnotationHub\_2.18.0 zlibbioc\_1.32.0   
## [45] MASS\_7.3-51.5 scales\_1.1.0   
## [47] ipred\_0.9-9 hms\_0.5.3   
## [49] promises\_1.1.0 RColorBrewer\_1.1-2   
## [51] yaml\_2.2.1 curl\_4.3   
## [53] memoise\_1.1.0 biomaRt\_2.42.0   
## [55] rpart\_4.1-15 stringi\_1.4.5   
## [57] RSQLite\_2.2.0 highr\_0.8   
## [59] BiocVersion\_3.10.1 randomForest\_4.6-14   
## [61] foreach\_1.5.0 e1071\_1.7-3   
## [63] lava\_1.6.7 rlang\_0.4.4   
## [65] pkgconfig\_2.0.3 bitops\_1.0-6   
## [67] evaluate\_0.14 lattice\_0.20-41   
## [69] ROCR\_1.0-11 purrr\_0.3.4   
## [71] recipes\_0.1.9 labeling\_0.3   
## [73] bit\_1.1-15.1 tidyselect\_1.0.0   
## [75] plyr\_1.8.5 magrittr\_1.5   
## [77] R6\_2.4.1 generics\_0.0.2   
## [79] DBI\_1.1.0 pillar\_1.4.3   
## [81] withr\_2.2.0 survival\_3.1-8   
## [83] RCurl\_1.98-1.2 nnet\_7.3-13   
## [85] tibble\_2.1.3 crayon\_1.3.4   
## [87] BiocFileCache\_1.10.2 rmarkdown\_2.1   
## [89] progress\_1.2.2 grid\_3.6.3   
## [91] data.table\_1.12.8 blob\_1.2.1   
## [93] ModelMetrics\_1.2.2.1 GenomicScores\_1.10.0   
## [95] digest\_0.6.25 xtable\_1.8-4   
## [97] httpuv\_1.5.2 openssl\_1.4.1   
## [99] munsell\_0.5.0 askpass\_1.1