BScBI-CG Practicals Report

Maria Cobo

Exercise 5

— November 14, 2024—

ESCI upf. School of International Studies

Computational Genomics 2024–25

BSc in Bioinformatics

UPF/ESCI – UPC – UB – UAB

Contents

List of Tables

List of Figures

1 Introduction

We are going to extract a set of known sequences that define specific signal features. Then we will build simple models in an attempt to predict location for those signals over an already annotated genomic sequence. This will allow us to evaluate the accuracy of our models, based on Position Weight Matrices (PWMs), by comparing those predictions against the known features.

1.1 Objectives

- To gather signal specific sequences from a set of coordinates and a genome sequence set.
- To compare different models obtained from a varying amount of evidences.
- To scan a sequence and predict a set of putative signals.
- To evaluate simple accuracy measures over the results obtained by different models.
- To introduce LaTeX BibTeX bibliographic files, as well as the SI units package.

1.2 Prerequisites

1.2.1 Installing required software

As for the previous practicals, we must ensure that at least `pandoc` and `pdflatex` commands are running smoothly over our report files. If you still need to install the base software, please refer to `exercise_00` and `exercise_01`, as well as the short tutorials from the Computational Genomics Virtual Campus at ESCI. Remind that we assume that you are using a Debian-based linux distribution, so we will show only the corresponding set of commands for that distribution.

```
##################
##
##  Hopefully, we will not need to install
##    any further tool for this exercise... ;^D
##
##  ... Just few R packages with the install.packages function maybe.
##
##  However, feel free to add here any installation command
##    that you may have required to complete this exercise.
```

1.2.2 Initializing the main report files

As in the previous exercises, remember to download first the exercise tarball from the Computational Genomics Virtual Campus at ESCI, unpack this file, modify the files accordingly to the user within the exercise folder, and set it as the current working directory for the rest of the exercise...

```
# You probably have already done this step.
tar -zxvf BScBI_CG2425_exercise_05.tgz
cd exercise_05

# Rename report file including your "NAME" and "SURNAME"
mv -v README_BScBICG2425_exercise05_SURNAME_NAME.md \
      README_BScBICG2425_exercise05_yourSurname_yourName.md

# Open exercise files using your text editor of choice
# (for instance vim, emacs, gedit, sublime, atom, ...);
# fix "NAME" and "SURNAME" placeholders on them
# and save those changes before continuing.
emacs  projectvars.sh \
       README_BScBICG2425_exercise05_yourSurname_yourName.md &

# Let's start with some initialization.
source projectvars.sh
echo $WDR

# Once you have run the commands that are already in the initial
```

```
# MarkDown document, you are probably ready to run this:
runpandoc
```

Let's start with the analyses, and may the shell be with you...

2 Datasets

We will continue working on the genomic datasets for the budding yeast *Saccharomyces cerevisiae*. We assume you still have some of the initial files downloaded from that genome repository. The *S. cerevisiae* (strain S288C) assembly R64 reference genome version [**?**] has 16 chromosome sequences, plus the mitochondrion genome, totaling 12 157 105 bp. However, we will be focussing on the already available annotation over chromosome XV (1 091 291 bp).

```
###
##### IMPORTANT #####  OPTION A  #####
###
#   You should have this data by now from previous exercises.
#     We have included here for completeness shake, just in case
#     you want to download again because you removed the data already.
#
mkdir $WDR/seqs;

URL="https://downloads.yeastgenome.org/sequence/S288C_reference/genome_releases";
wget $URL/S288C_reference_genome_Current_Release.tgz \
     -O $WDR/seqs/S288C_reference_genome_Current_Release.tgz;

# you may consider copying those vars to your projectvars.sh file
export REFDIR="S288C_reference_genome_R64-5-1_20240529"
export REFGEN="S288C_reference_sequence_R64-5-1_20240529"

pushd $WDR/seqs/;
tar -zxvf S288C_reference_genome_Current_Release.tgz;
popd;

gunzip -c $WDR/seqs/$REFDIR/$REFGEN.fsa.gz | \
  infoseq -only -length -noheading -sequence fasta::stdin \
    2> /dev/null | \
  gawk '{ s+=$1 } END{ print "# Yeast genome size (v.R64): "s }';
# Yeast genome size (v.R64): 12157105


###
##### IMPORTANT #####  OPTION B  #####
###
#   If you already downloaded the sequence for exercise_03,
#   then you can just link the sequence folder for the current exercise.
#
ln -vs ../exercise_03/seqs $WDR/;
```

2.1 Annotated donor splice sites

Among the files packed into the *S. cerevisiae* genome tarball, we have a `GFF` file containing the current gene set in chromosomal absolute sequence coordinates. We will filter out those genes having two or more exons, in order to extract the coordinates and sequence substrings that will define the complete set of annotated donor sites.

```
#
# step 0.1.- fixing sequence names
gunzip -c $WDR/seqs/$REFDIR/$REFGEN.fsa.gz | \
  gawk '/^>/ {
          id=substr($1,2);
          sq=$NF;
```

```
          sub(/^.*=/,  "", sq);
          sub(/\].*$/, "", sq);
          chr= (sq == "circular") ? "mt" : sq;
          $1=">chr"chr".Scer "id;
        }
        { print $0 }' - \
        > $WDR/seqs/refgenome_chromosomes.fasta;

egrep -c '^>' $WDR/seqs/refgenome_chromosomes.fasta;
#                   17 refgenome_chromosomes.fasta

#
# step 0.2.- how many annotated features do we have in the GFF file?
export REFGFF="saccharomyces_cerevisiae_R64-5-1_20240529.gff.gz"
zegrep -cv '^\#|^[ \t]*$' $WDR/seqs/$REFDIR/$REFGFF;
#   180337


#
# step 1.- filter out coding exons (CDS)
gunzip -c $WDR/seqs/$REFDIR/$REFGFF | \
  gawk '$3 ~ /CDS/ {
          gn=$9;
          sub(/^Parent=/,"",gn);
          sub(/;.*$/,"",gn);
          $9=gn"."$1;
          print $0;
        }' \
        > $WDR/seqs/refgenome_allcds_allchr.gff;

wc -l $WDR/seqs/refgenome_allcds_allchr.gff;
#          7072 refgenome_allcds_allchr.gff


#
# step 2.- calculating number of CDSs per gene
gawk '{ print $9; }
     '  $WDR/seqs/refgenome_allcds_allchr.gff | uniq -c | sort -nr \
      > $WDR/seqs/refgenome_allcds_allchr.genes.tbl;

wc -l $WDR/seqs/refgenome_allcds_allchr.genes.tbl;
#          6710 refgenome_allcds_allchr.genes.tbl

#
# step 2.1.- checking for "aberrant" splice sites
zegrep 'translational_frameshift' \
      $WDR/seqs/$REFDIR/$REFGFF | wc
#    47     423    6040
# we must be aware that there are 47 splice sites that were
# artificially added to the corresponding gene-structure to fix frame-shifts
#
# getting the gene names and coord for those artificial splice sites
gunzip -c $WDR/seqs/$REFDIR/$REFGFF | \
  gawk '$3 ~ /translational_frameshift/ {
          gn=$9;
          sub(/^Parent=/,"",gn);
          sub(/;.*$/,"",gn);
          print gn, $1, $4;
        }' - \
        > $WDR/seqs/translational_frameshifts.tbl;

#
# step 3.- annotate CDSs for latter analyses
```

```
gawk 'BEGIN{
        while (getline<ARGV[1]>0) {
          if ($1 > 1) GN[$2]=$1;
        };
        ARGV[1]="";
      }
      {
        ex=$3;
        CNT[$9]++;
        if ($9 in GN) {
          if (CNT[$9] == 1) {             # first CDSexon    (+)
            $3=($7 == "+") ? "CDS.first" : "CDS.terminal";
          } else {
            if (CNT[$9] == GN[$9]) { # terminal CDSexon (+)
              $3=($7 == "+") ? "CDS.terminal" : "CDS.first";
            } else {                        # internal CDSexon (+)
              $3="CDS.internal";
            };
          };
        } else {                            # single CDS exon
          $3="CDS.single";
        };
        print $0;
      }' $WDR/seqs/refgenome_allcds_allchr.genes.tbl \
         $WDR/seqs/refgenome_allcds_allchr.gff \
       > $WDR/seqs/refgenome_allcdsannotated_allchr.gff;

wc -l  $WDR/seqs/refgenome_allcdsannotated_allchr.gff;
#           7072 refgenome_allcdsannotated_allchr.gff


#
# step 4.- retrieve the donor chromosome coords for genes having at least
#          one intron (avoiding fake frameshifts), we also know that
#          all CDS.first and CDS.internal will end up with a donor site...
gawk 'BEGIN{
        while (getline<ARGV[1]>0) { NOSS[$1"."$2":"$3]++; };
        ARGV[1]="";
      }
      $3 ~ /CDS\.(first|internal)/ {
        if ($9":"$4 in NOSS || $9":"($4-1) in NOSS || $9":"($4+1) in NOSS ||
            $9":"$5 in NOSS || $9":"($5-1) in NOSS || $9":"($5+1) in NOSS) {
          print "# skipping "$0 | "cat 1>&2";
          next;
        };
        print "# using "$0 | "cat 1>&2";
        if ($7 == "+") { # forward features
          print $1, $7, $5 + 1;
        } else {         # reverse features
          print $1, $7, $4 - 1;
        };
      }' $WDR/seqs/translational_frameshifts.tbl \
         $WDR/seqs/refgenome_allcdsannotated_allchr.gff \
       > $WDR/seqs/refgenome_alldonorsites_allchr.tbl \
      2> $WDR/seqs/refgenome_alldonorsites_allchr.log;

gawk '{print $2}' $WDR/seqs/refgenome_alldonorsites_allchr.log | \
  sort | uniq -c;
#     47 skipping
#    315 using

wc -l $WDR/seqs/refgenome_alldonorsites_allchr.tbl
#           315 refgenome_alldonorsites_allchr.tbl
```

```
#
# step 5.- clip the donor sequences
gawk '{ if ($2 == "+") print $1, $2, $3 - 3, $3 + 5;
                  else print $1, $2, $3 - 5, $3 + 3;
        }' $WDR/seqs/refgenome_alldonorsites_allchr.tbl | \
  while read chrom strnd dnr_start dnr_end;
    do {
      echo "# trimming $chrom ($strnd) $dnr_start $dnr_end" 1>&2;
      if [ "X$strnd" == "X+" ];
        then
            samtools faidx $WDR/seqs/refgenome_chromosomes.fasta \
                           $chrom.Scer:$dnr_start-$dnr_end ;
        else
            samtools faidx $WDR/seqs/refgenome_chromosomes.fasta \
                           $chrom.Scer:$dnr_start-$dnr_end | \
                revseq -sequence fasta::stdin -outseq stdout;
      fi;
    }; done > $WDR/seqs/refgenome_alldonorsites_sequences.fasta
egrep -v '^>' $WDR/seqs/refgenome_alldonorsites_sequences.fasta \
           > $WDR/seqs/refgenome_alldonorsites_sequences.tbl

egrep -c '^>' $WDR/seqs/refgenome_alldonorsites_sequences.fasta
#                   315 refgenome_alldonorsites_sequences.fasta
```

The previous protocol produced a complete set of donor sites sequences in tabular format. Now we need to filter three sets of 50 sequences and another three sets of 100 sequences, chosen at random. Include the commands you have ran to produce those sequence sets. Name the tabular files as follows:

donor_50_sequences.setX.tbl, donor_50_sequences.setY.tbl, donor_50_sequences.setZ.tbl,
donor_100_sequences.setX.tbl, donor_100_sequences.setY.tbl, donor_100_sequences.setZ.tbl

```
# You can use the "shuf" command to randomize record lines,
# and "head" to recover top n-elements of the shuffled list.
# you will need to repeat three times for the setX, setY, and setZ
# as well as for acceptors. Include your commands here:
#
shuf $WDR/seqs/refgenome_alldonorsites_sequences.tbl | \
  head -50 - > $WDR/data/donor_50_sequences.setX.tbl;
shuf $WDR/seqs/refgenome_alldonorsites_sequences.tbl | \
  head -50 - > $WDR/data/donor_50_sequences.setY.tbl;
shuf $WDR/seqs/refgenome_alldonorsites_sequences.tbl | \
  head -50 - > $WDR/data/donor_50_sequences.setZ.tbl;
shuf $WDR/seqs/refgenome_alldonorsites_sequences.tbl | \
  head -100 - > $WDR/data/donor_100_sequences.setX.tbl;
shuf $WDR/seqs/refgenome_alldonorsites_sequences.tbl | \
  head -100 - > $WDR/data/donor_100_sequences.setY.tbl;
shuf $WDR/seqs/refgenome_alldonorsites_sequences.tbl | \
  head -100 - > $WDR/data/donor_100_sequences.setZ.tbl;

#
```

2.2 Annotated acceptor splice sites

You must extract the set of all acceptor sequences following the same procedure. Take into account that the acceptors define the start coordinates for internal and terminal exons and that you should cut 24 and 3 nucleotides from intron and exon respectively (we have used 3 and 6 from exon and intron respectively for the donor sites).

On the other hand, you should include in the forthcoming analyses the two datasets listed below, a pair of files for donor sites and two more for the acceptors.

```
gawk 'BEGIN{
        while (getline<ARGV[1]>0) { NOSS[$1"."$2":"$3]++; };
        ARGV[1]="";
      }
      $3 ~ /CDS\.(terminal|internal)/ {
        if ($9":"$4 in NOSS || $9":"($4-1) in NOSS || $9":"($4+1) in NOSS ||
            $9":"$5 in NOSS || $9":"($5-1) in NOSS || $9":"($5+1) in NOSS) {
          print "# skipping "$0 | "cat 1>&2";
          next;
        };
        print "# using "$0 | "cat 1>&2";
        if ($7 == "+") { # forward features
          print $1, $7, $5 + 1;
        } else {         # reverse features
          print $1, $7, $4 - 1;
        };
      }' $WDR/seqs/translational_frameshifts.tbl \
         $WDR/seqs/refgenome_allcdsannotated_allchr.gff \
       > $WDR/seqs/refgenome_allacceptorsites_allchr.tbl \
      2> $WDR/seqs/refgenome_allacceptorsitess_allchr.log;

gawk '{print $2}' $WDR/seqs/refgenome_allacceptorsitess_allchr.log | \
  sort | uniq -c;
#     47 skipping
#    315 using

wc -l $WDR/seqs/refgenome_allacceptorsites_allchr.tbl
#           315 refgenome_allacceptorsites_allchr.tbl

gawk '{ if ($2 == "+") print $1, $2, $3 - 24, $3 + 3
                  else print $1, $2, $3 - 3, $3 + 24;
        }' $WDR/seqs/refgenome_allacceptorsites_allchr.tbl | \
  while read chrom strnd dnr_start dnr_end;
    do {
      echo "# trimming $chrom ($strnd) $dnr_start $dnr_end" 1>&2;
      if [ "X$strnd" == "X+" ];
        then
            samtools faidx $WDR/seqs/refgenome_chromosomes.fasta \
                           $chrom.Scer:$dnr_start-$dnr_end ;
        else
            samtools faidx $WDR/seqs/refgenome_chromosomes.fasta \
                           $chrom.Scer:$dnr_start-$dnr_end | \
                revseq -sequence fasta::stdin -outseq stdout;
      fi;
    }; done > $WDR/seqs/refgenome_allacceptorsites_sequences.fasta
egrep -v '^>' $WDR/seqs/refgenome_allacceptorsites_sequences.fasta \
            > $WDR/seqs/refgenome_allacceptorsites_sequences.tbl

egrep -c '^>' $WDR/seqs/refgenome_allacceptorsites_sequences.fasta
#                   315 refgenome_allacceptorsites_sequences.fasta


shuf $WDR/seqs/refgenome_allacceptorsites_sequences.tbl | \
  head -50 - > $WDR/data/acceptor_50_sequences.setX.tbl;
shuf $WDR/seqs/refgenome_allacceptorsites_sequences.tbl | \
  head -50 - > $WDR/data/acceptor_50_sequences.setY.tbl;
shuf $WDR/seqs/refgenome_allacceptorsites_sequences.tbl | \
  head -50 - > $WDR/data/acceptor_50_sequences.setZ.tbl;
shuf $WDR/seqs/refgenome_allacceptorsites_sequences.tbl | \
  head -100 - > $WDR/data/acceptor_100_sequences.setX.tbl;
shuf $WDR/seqs/refgenome_allacceptorsites_sequences.tbl | \
  head -100 - > $WDR/data/acceptor_100_sequences.setY.tbl;
shuf $WDR/seqs/refgenome_allacceptorsites_sequences.tbl| \
```

```
  head -100 - > $WDR/data/acceptor_100_sequences.setZ.tbl;

ls -1 $WDR/data/*or_sequences.set?.tbl;
#
#  donor_sequences.setA.tbl
#  donor_sequences.setB.tbl
#
#  acceptor_sequences.setA.tbl
#  acceptor_sequences.setB.tbl
```

3 Analysis of the splice sites sequence sets

3.1 Frequencies visualization

We can visualize the frequencies for a rapid comparative exploration of the models we can compute for the different sequence sets, those we have obtained on the previous section, before using them to score putative splice sites on a query sequence. We can take advantage of the `ggseqlogo` `R` package [?][1].

```
# let's play with an specialized ggplot2 module: ggseqlogo
    install.packages("ggseqlogo");
# or get from github:
#   library(devtools);
#   devtools::install_github("omarwagih/ggseqlogo");
# or install in conda environment:
#   conda install -c conda-forge r-ggseqlogo
#
install.packages("gridExtra"); # unless you already have it installed

require(ggplot2);
require(ggseqlogo);
require(gridExtra);

# loading the sequence set
donors.setALL <- readLines("seqs/refgenome_alldonorsites_sequences.tbl");

# making a sequence logo
p1 <- ggplot() + theme_logo() +
        annotate('rect', xmin = 0.5, xmax = 3.5, ymin = -0.05, ymax = Inf,
             alpha = .1, col='#0000FF', fill='#0000FF55') +
        geom_logo( donors.setALL, method = 'bits' ) +
        ggtitle("S.cerevisiae 361 donor sites (sequence logo)") +
        guides(color = "none", fill = "none");

# drawing a pictogram
p2 <- ggplot() + theme_logo() +
        annotate('rect', xmin = 0.5, xmax = 3.5, ymin = -0.05, ymax = Inf,
             alpha = .1, col='#0000FF', fill='#0000FF55') +
        geom_logo( donors.setALL, method = 'prob' ) +
        ggtitle("S.cerevisiae 361 donor sites (pictogram)");

# to view on the side panel
gridExtra::grid.arrange(p1, p2);

# saving the picture
P <- gridExtra::arrangeGrob(p1, p2);
ggsave(file="images/alldonors_logo+pictogram.png",
       P, height=6, width=9, units="in", dpi=600)
```

[1] `ggseqlogo`: https://omarwagih.github.io/ggseqlogo/

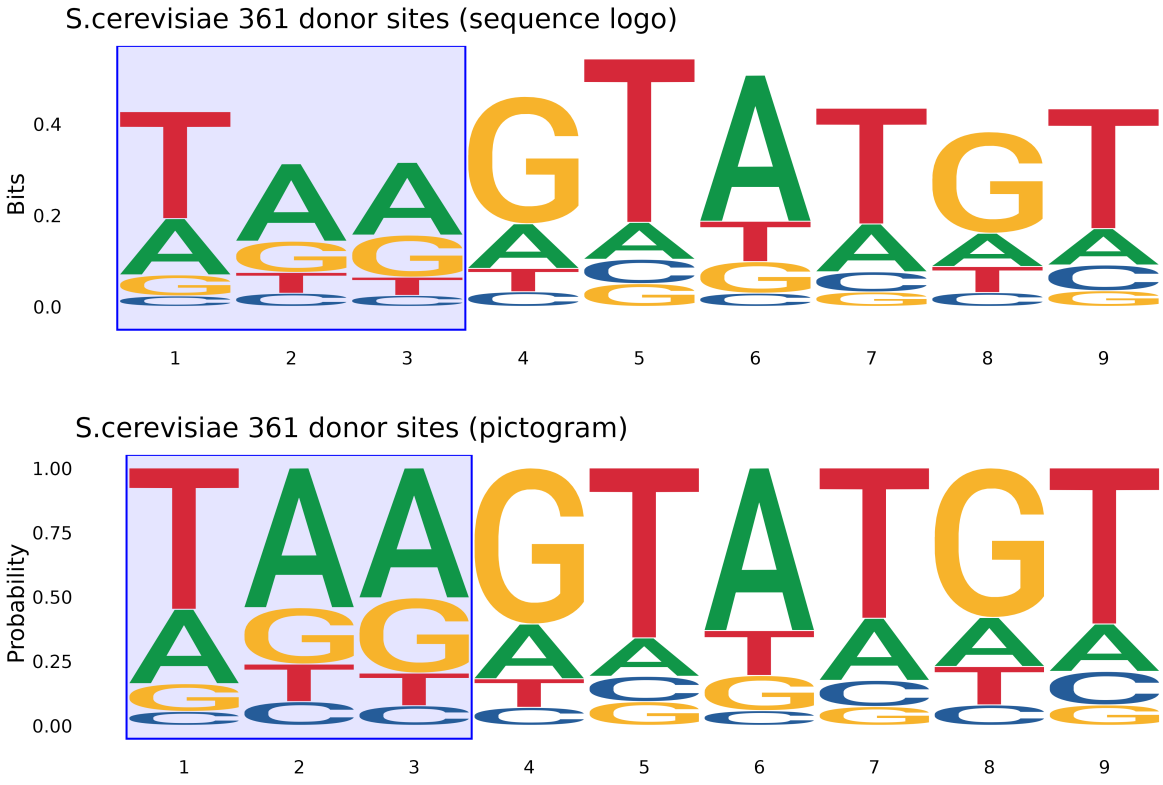

Figure 1: **Visualizing PWM generated from a set of *S. cerevisiae* donor sites.** Top panel shows a sequence logo, while the bottom corresponds to a pictogram, for a set of 361 donor site sequences from yeast CDSs annotations. Blue area on the left defines the three exonic final nucleotide positions.

IMPORTANT: Define a table inside a figure float that combines all the sequence logos, like the one shown at the top panel from Figure 1, for both donors and acceptors, on the nine (×2) sequence sets you have obtained from the previous section.

```
acceptors.setALL <- readLines("seqs/refgenome_allacceptorsites_sequences.tbl");
# making a sequence logo
p1 <- ggplot() + theme_logo() +
    annotate('rect', xmin = 24.5, xmax = 27.5, ymin = -0.05, ymax = Inf,
        alpha = .1, col='#0000FF', fill='#0000FF55') +
    geom_logo( acceptors.setALL, method = 'bits' ) +
    ggtitle("S.cerevisiae 361 acceptor sites (sequence logo)") +
    theme(plot.title = element_text(size = 40, face = "bold")) +
    theme(plot.title = element_text(size = 20, face = "bold"));
ggsave(file="images/allacceptors_logo.png",
    p1, height=6, width=9, units="in", dpi=600)

for (num in c("50", "100")){
    for (set in c("setX", "setY", "setZ")){
        for (site in c("donor", "acceptor")){
            file <- readLines(paste0("data/", site, "_", num, "_sequences.", set, ".tbl"))
            if (site == "acceptor"){ ggplot() + theme_logo() +
                annotate('rect', xmin = 24.5, xmax = 27.5, ymin = -0.05, ymax = Inf,
                    alpha = .1, col='#0000FF', fill='#0000FF55') +
                geom_logo( file, method = 'bits' ) +
                ggtitle(paste0("Logo: S.cerevisiae ", set, " that has ", num, " ", site, " sites")) +
                theme(plot.title = element_text(size = 20, face = "bold"));}
            else{ ggplot() + theme_logo() +
                annotate('rect', xmin = 0.5, xmax = 3.5, ymin = -0.05, ymax = Inf,
                    alpha = .1, col='#0000FF', fill='#0000FF55') +
                geom_logo( file, method = 'bits' ) +
```

```
                ggtitle(paste0("Logo: S.cerevisiae ", set, " that has ", num, " ", site, " sites")) +
                theme(plot.title = element_text(size = 20, face = "bold"));}
            ggsave(file= paste0("images/", num, "_", set, "_", site, "_logo.png"),
                height=6, width=9, units="in", dpi=600) } } }

for (splice_site in c("acceptor", "donor")){
    for (set in c("setA", "setB")){
        file <- readLines(paste0("data/", splice_site, "_sequences.", set, ".tbl"))
        if (splice_site == "acceptor"){ ggplot() + theme_logo() +
            annotate('rect', xmin = 24.5, xmax = 27.5, ymin = -0.05, ymax = Inf,
                alpha = .1, col='#0000FF', fill='#0000FF55') +
            geom_logo( file, method = 'bits' ) +
            ggtitle(paste0("Logo: S.cerevisiae ", set, " of ", splice_site, " sites")) +
            theme(plot.title = element_text(size = 20, face = "bold")); }
        else{ ggplot() + theme_logo() +
            annotate('rect', xmin = 0.5, xmax = 3.5, ymin = -0.05, ymax = Inf,
                alpha = .1, col='#0000FF', fill='#0000FF55') +
            geom_logo( file, method = 'bits' ) +
            ggtitle(paste0("Logo: S.cerevisiae ", set, " of ", splice_site, " sites")) +
            theme(plot.title = element_text(size = 20, face = "bold")); }
        ggsave(file= paste0("images/",set, "_", splice_site, "_logo.png"),
            height=6, width=9, units="in", dpi=600) } }
```

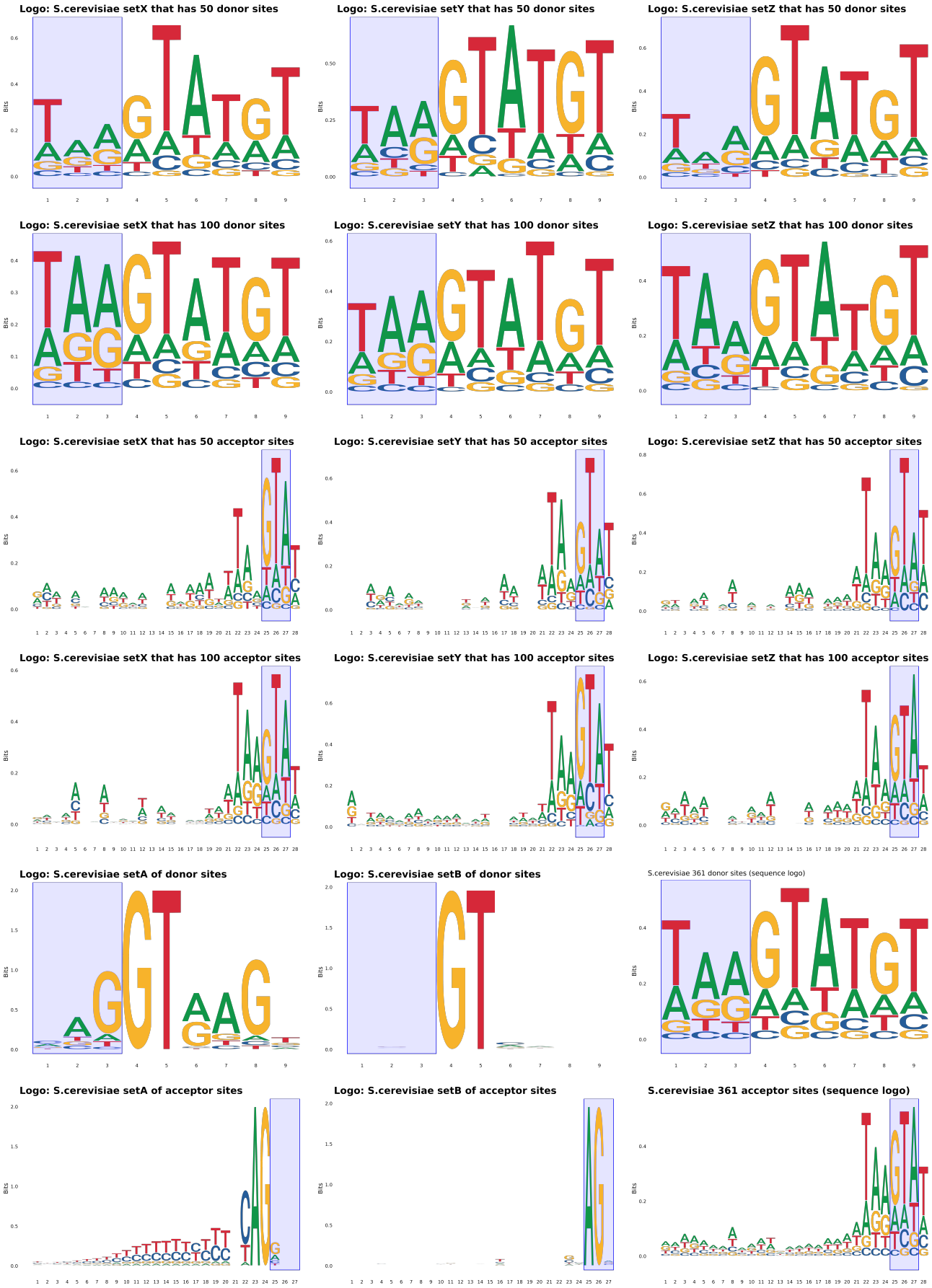

Figure 2: **`dnadiff` Visualization of the sequence logos for both donors and acceptors.** In these plots, the blue area defines the three exonic final nucleotide positions when ploting the sequence logo of the acceptor sites.

3.2 Compute PWMs

At this stage we should compute the Position Weight Matrices (PWMs) for the same set of **donor** sets. Here we have an example of the record structure for PWM models used by `pwmfinder.pl` script (see Appendix 6.2.2 on page 18):

```
##
## A simple Position Weight Matrix
## derived from a set of donor sites
##
P0       A        C       G        T
01   0.302    0.483  -0.305   -0.856
02   0.817   -0.667  -0.743   -0.474
03  -1.143   -0.782   1.123   -1.660
04   -9999    -9999   0.000    -9999
05   -9999    -9999   -9999    0.000
06   1.083   -2.097   0.135   -2.246
07   1.032   -1.093  -0.627   -1.111
08  -1.218   -1.479   1.257   -1.534
09  -0.411   -0.358  -0.136    0.492
XX 3
```

Let's consider a set of fixed-length DNA sequences like the ones we saved into the `donor_sequences.tbl` file. The corresponding weight matrix is computed for the probability distribution of the set of nucleotides found at each position in the aligned sequences set. Simply, we count ocurrences of each of the four nucleotides at a given position, in our case the alphabet is $\Sigma_{DNA} = \{A, C, G, T\}$. Those nucleotide frequencies are then normalized to fit the interval $[0, 1]$. The resulting matrix contains the same information but in terms of proportions. The real model has to be compared with a background model, in our example we assume a uniform distribution as a background model (random sequences model, thus $P(A) = P(C) = P(G) = P(T) = 0.25$). The weight of each nucleotide at each position is obtained by dividing scores for the annotated "real" set by the "random" set, in a log-likelyhood ratio. We use a very small number, like -999, to define the unprobable nucleotides found at a given position (a value close to infinity).

The `makepwms.pl` script (see Appendix 6.2.2 on page 16) takes a tabular file with a set of sequences and returns a matrix record suitable to be used later on by `pwmfinder.pl` (see page 12).

```
# donors
$BIN/makepwms.pl 4 $WDR/seqs/refgenome_alldonorsites_sequences.tbl \
                 > $WDR/data/donors_setALL.pwm

# Use a shell loop to run throught the remaining eight donor
#   and the nine acceptor site sequence sets.
#
## IMPORTANT: ##
#
#  The first argument of makepwms.pl just tells
#    the program where ends the first segment;
#    it should be 24 for the acceptor sequences
#    if you followed all the instructions of this protocol.


# Iterate over all files with extension .tbl
for i in data/*.tbl; do
  # Get the base filename without the directory path
  base_file=$(basename "$i")

  # Check if the filename starts with "donor"
  if [[ $base_file == donor* ]]; then
    # Use 4 as the first parameter for donor files
    $BIN/makepwms.pl 4 "$WDR/$i" > "$WDR/data/${base_file}.pwm"
  else
    # Use 24 as the first parameter for acceptor files
    $BIN/makepwms.pl 24 "$WDR/$i" > "$WDR/data/${base_file}.pwm"
  fi
done

# acceptors
$BIN/makepwms.pl 4 $WDR/seqs/refgenome_allacceptorsites_sequences.tbl \
```

```
                > $WDR/data/acceptors_setALL.pwm


# Iterate over all files with extension .tbl
for i in data/*.tbl; do
  # Get the base filename without the directory path
  base_file=$(basename "$i")

  # Check if the filename starts with "acceptor"
  if [[ $base_file == acceptors* ]]; then
    # Use 4 as the first parameter for donor files
    $BIN/makepwms.pl 4 "$WDR/$i" > "$WDR/data/${base_file}.pwm"
  else
    # Use 24 as the first parameter for acceptor files
    $BIN/makepwms.pl 24 "$WDR/$i" > "$WDR/data/${base_file}.pwm"
  fi
done
```

4 Splice Sites Prediction

Once we have the matrices properly formatted we will use a `Perl` script, included in Appendix 6.2.2. Let's analyze an already annotated sequence so we will be able to perform the accuracy assessment later on.

```
# number of already whole genome annotated donors by chromosome
gawk '{ print $1 }' $WDR/seqs/refgenome_alldonorsites_allchr.tbl | \
  sort | uniq -c
#        3 chrI
#       27 chrII
#        8 chrIII
#       34 chrIV
#       11 chrIX
#       31 chrmt
#       15 chrV
#        7 chrVI
#       22 chrVII
#       19 chrVIII
#       15 chrX
#       13 chrXI
#       26 chrXII
#       28 chrXIII
#       19 chrXIV
#       12 chrXV
#       25 chrXVI

# let's try chrXV, for instance
samtools faidx $WDR/seqs/refgenome_chromosomes.fasta \
           chrXV.Scer \
         > $WDR/seqs/refgenome_chrXV.fasta
```

4.1 Scanning sequences to annotate sites.

```
$BIN/pwmfinder.pl $WDR/data/donors_setALL.pwm \
                < $WDR/seqs/refgenome_chrXV.fasta \
                > $WDR/data/donors_setALL.predicted_splice_sites.tbl

# A : 0.209 1.119 1.014 -0.233 -0.745 1.337 -0.051 -0.392 -0.441
# C : -2.212 -1.441 -1.714 -1.840 -1.345 -2.129 -1.299 -1.655 -0.942
# G : -1.212 -0.191 0.224 1.278 -1.441 -0.873 -1.776 1.216 -1.655
# T : 1.135 -0.776 -0.977 -1.170 1.401 -0.518 1.224 -0.745 1.278
# Intron first nucleotide: 4
```

```
$BIN/pwmfinder.pl $WDR/data/acceptors_setALL.pwm \
                < $WDR/seqs/refgenome_chrXV.fasta \
                > $WDR/data/acceptors_setALL.predicted_splice_sites.tbl

# A : 0.401 0.442 0.596 0.643 0.559 0.469 0.469 0.701 0.429 0.456 0.482 0.387 0.144 0.415 0.286 0.401 0.49
# 0.387 0.571 0.678 -0.032 1.224 1.127 -0.299 -0.544 1.359 -0.233
# C : -0.714 0.005 -0.776 -0.627 -0.170 -0.368 -0.544 -0.441 -0.233 -0.684 -0.466 -0.655 -0.544 -0.368 -0.
# -0.392 -0.417 -0.873 -0.714 -0.840 -2.051 -1.776 -1.544 -2.299 -0.942 -2.129 -0.714
# G : 0.286 -0.776 -0.345 -0.014 -0.655 -0.417 -0.070 -0.807 -0.441 0.058 -0.599 -0.441 0.160 -0.392 -0.07
# -0.368 -0.599 -0.129 -0.466 -0.655 -1.599 -0.345 -0.109 1.263 -2.129 -1.255 -1.544
# T : -0.233 0.076 0.160 -0.322 0.005 0.127 -0.032 0.093 0.093 -0.051 0.286 0.401 0.127 0.177 0.271 0.177
# 0.301 0.240 0.271 1.293 -0.776 -0.873 -0.776 1.359 -0.345 1.135
# Intron first nucleotide: 4
```

You need to repeat the procedure for the nine sets of donors and acceptors (`ALL`, `setA`, `setB`, `setX.50`, `setY.50`, `setZ.50`, `setX.100`, `setY.100`, `setZ.100`). Just take into account that the current version of the `pwmfinder.pl` script only scans the forward strand of the analyzed genomic sequences.

```
for i in data/*.pwm;
do
$BIN/pwmfinder.pl $WDR/${i} \
                < $WDR/seqs/refgenome_chrXV.fasta \
                > $WDR/${i}.predicted_splice_sites.tbl
done
mkdir data/predicted
mv data/*predicted*.tbl data/predicted/
```

4.2 Accuracy assessment.

By now you should have sets of genomic coords that define the location of annotated and predicted signals. Fill the table 1 with the corresponding results.

```
for i in data/predicted/*;
do
echo $i
cat $i | wc -l
done

# For donors:
for i in data/predicted/donor*
do
echo $i
python3 $WDR/bin/comput_params.py $WDR/$i $WDR/seqs/refgenome_alldonorsites_allchr.tbl
done

# For acceptors:
for i in data/predicted/acceptor*
do
echo $i
python3 $WDR/bin/comput_params.py $WDR/$i $WDR/seqs/refgenome_allacceptorsites_allchr.tbl
done
```

Generate a similar table for the acceptor sites; describe which set yields better predictions when looking at those results for the two types of signals, donors or acceptors. Can we guess the reason, if any, behind such a difference?

Table 1: **Summary of accuracy assessment results for donor splice sites.**
Sensitivity (Sn, "recall") calculated as $Sn = \frac{TP}{TP+FN}$, specificity (Sp, "precision") as $Sp = \frac{TP}{TP+FP}$.
Results shown were computed over *S. cerevisiae* chromosome XV annotations and predictions.

Measure	setALL	setA	setB	setX.50	setY.50	setZ.50	setX.100	setY.100	setZ.100
Annotated	22	22	22	22	22	22	22	22	22
Predicted									
TP	2	0	0	4	3	3	4	4	4
TN									
FP	217524	21642	56613	219919	161015	179820	215479	201749	212736
FN	11	13	13	9	10	10	9	9	9
Sensitivity	0.1538	0.0	0.0	0.3077	0.2308	0.2308	0.3077	0.3077	0.3077
Specificity	9.19e-6	0.0	0.0	1.818e-5	1.863e-5	1.668e-5	1.856e-5	1.983e-5	1.880e-5

Table 2: **Summary of accuracy assessment results for acceptor splice sites.**
Sensitivity (Sn, "recall") calculated as $Sn = \frac{TP}{TP+FN}$, specificity (Sp, "precision") as $Sp = \frac{TP}{TP+FP}$.
Results shown were computed over *S. cerevisiae* chromosome XV annotations and predictions.

Measure	setALL	setA	setB	setX.50	setY.50	setZ.50	setX.100	setY.100	setZ.100
Annotated	22	22	22	22	22	22	22	22	22
Predicted									
TP	8	0	0	3	3	1	3	2	3
TN									
FP	350681	17475	57960	271715	274709	243948	332106	224143	318639
FN	5	13	13	10	10	12	10	11	10
Sensitivity	0.6154	0.0	0.0	0.2308	0.2308	0.0769	0.2308	0.1538	0.2308
Specificity	2.281e-5	0.0	0.0	1.104e-5	1.092e-5	4.099e-6	9.033e-6	8.923e-6	9.415e-6

5 Discussion

IMPORTANT Discuss your results here (around 300 words). And remember to include in the Appendices section (see page 15), any extra script you wrote from this exercise `bin` folder using the `loadfile` macro. We can take advantage of the LaTeX referencing capabilities, as described in the first exercise template.

In this practical we kept working with the genomic datasets for the building yeast `Saccharomyces cerevisiae` and we focused on the available annotation over chromosome XV. First, we filtered those genes having two or more exons, extracting their coordinates and sequencing substrings that define the complete set of annotated donor sites. Then, we randomly filtered three sets of 50 and 100 sequences and, repeated these for the acceptor splice sites.

Then, the analysis proceeded on visualizing the sequence logos of the splice sites (donors and acceptors) for the different sequence sets. In these plots, the blue area defines the three exonic final nucleotide positions. So, these show that there is a higher degree of confidence of finding 'TAA' before finding the donor splice site. We did not expect this as the step before filtered out the terminal exons so there is no need for stop codons (TAA, TGA or TAG). Moreover, the plots for the donor splice site show that there is a high degree of confidence of finding 'ATG' after the acceptor splice site. Again, this is unexpected as we removed the star exons. Also, setA and setB do not follow this pattern.

Next, we proceeded to compute the Position Weight Matrices (PWMs) for each of the sets (ALL, setA, setB, setX.50, setY.50, setZ.50, setX.100, setY.100, setZ.100) and used them to predict the donor and acceptor splice sites. Finally, we scanned the sequences to annotate sites and got sets of genomic coordinates that define the location of annotated and predicted signals. From these, we got a summary table where we can see that setA and setB are not accurate at all, (expect fot the acceptors of setB) as they have values of zero for both sensitivity and specificity. Additionally, setALL has really high sensitivity and small specificity, with a lot of FP. And, for the six sets, we can see that for the donor and acceptor splice sites there is a small sensitivity and even smaller specificity and a high number og FP.

In conclusion, the PWMs identified splice sites and a variability in performance for the different sets. They had high False Positives, and small sensitivities and specificities. So, improving the models could be better for predictions and accuracy.

6 Appendices

6.1 Software

We have used the following versions:

```
uname -a
# Linux aleph 5.15.0-48-generic #54-Ubuntu SMP
# Fri Aug 26 13:26:29 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux

R --version
# R version 4.3.1 (2023-06-16) -- "Beagle Scouts"
# Copyright (C) 2023 The R Foundation for Statistical Computing
# Platform: x86_64-conda-linux-gnu (64-bit)

infoseq -version
# EMBOSS:6.6.0.0

wget --version
# GNU Wget 1.21.2 built on linux-gnu.

pandoc --version
# pandoc 3.1.3
# Features: +server +lua
# Scripting engine: Lua 5.4

mamba --version
# mamba 1.4.2
# conda 23.3.1

gunzip --version
# gunzip (gzip) 1.13

perl -v
# This is perl 5, version 32, subversion 1 (v5.32.1)
# built for x86_64-linux-gnu-thread-multi


packageVersion("ggplot2");
# [1] '3.4.4'
packageVersion("ggseqlogo");
# [1] '0.1'
packageVersion("gridExtra");
# [1] '2.3'
```

6.2 Supplementary files

6.2.1 conda environment dependencies for the exercise

environment.yml

```
#
## ############################################################################################
##
##   environment.yml
##
##   Defining conda/mamba software dependencies to run BScBI-CG practical exercises.
##
## ############################################################################################
##
##                 CopyLeft 2024 (CC:BY-NC-SA) --- Josep F Abril
##
##   This file should be considered under the Creative Commons BY-NC-SA License
##   (Attribution-Noncommercial-ShareAlike). The material is provided "AS IS",
##   mainly for teaching purposes, and is distributed in the hope that it will
##   be useful, but WITHOUT ANY WARRANTY; without even the implied warranty
```

```
##   of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
##
## ##############################################################################################
#
# To install software for the exercise use the following command:
#
#    conda env create --file environment.yml
#
# then run the command below to activate the conda environment:
#
#    conda activate BScBI-CG2425_exercises
#
name: BScBI-CG2425_exercises
channels:
  - bioconda
  - conda-forge
  - defaults
dependencies:
  - htop
  - vim
  - emacs
  - gawk
  - perl
  - python
  - biopython
  - wget
  - curl
  - gzip
  - texlive-core
  - pandoc
  - pandocfilters
  - emboss
  - jellyfish
  - sra-tools
  - seqtk
  - fastqc
  - trimmomatic
  - samtools
  - bamtools
  - picard
  - bwa
  - bowtie2
  - soapdenovo2
  - igv
  - blast
  - gnuplot
  - graphicsmagick
  - mummer
  - hmmer
  - bbmap
  - metaeuk
  - busco=5.5.0
  - perl-text-soundex
  - repeatmasker
  - trf
  # R-packages
  - r-ggplot2
  - r-reshape2
  - r-gridextra
  - r-ggseqlogo
```

6.2.2 Project specific scripts

makepwms.pl

```
#!/usr/bin/perl
#
## ##############################################################################################
##
##   makepwms.pl - computing a PWM model from a set of signal sequences
##
## ##############################################################################################
##
##                 CopyLeft 2024 (CC:BY-NC-SA) --- Josep F Abril
##
##   This file should be considered under the Creative Commons BY-NC-SA License
##   (Attribution-Noncommercial-ShareAlike). The material is provided "AS IS",
```

```
##   mainly for teaching purposes, and is distributed in the hope that it will
##   be useful, but WITHOUT ANY WARRANTY; without even the implied warranty
##   of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
##
## ##############################################################################
#

use strict;
use warnings;

if (scalar(@ARGV) < 2) {

  print STDERR "ERROR: makepwms.pl requires two arguments an integer and a sequence file in tabular format\n";
  exit(1);

};

my $col = $ARGV[0];
my $file = $ARGV[1];

if (!open(SEQSFILE,"< $file")) {

  print STDERR "ERROR: makepwms.pl cannot open $file file\n";
  exit(1);

};

my @pwm    = ();         # storing PWM counter in a matrix
my @nucleotides = qw( A C G T N );
my %nucs   = qw( A 0 C 1 G 2 T 3 N 4 SUM 5 );
my @zeroes = (0) x (scalar(@nucleotides) + 1);

# loading absolute frequencies matrix
my $c = 0;
while (<SEQSFILE>) {

  next if /^\#/o;
  next if /^\s*$/o;
  chomp;
  my $line = uc($_);

  for (my $i = 0; $i < length($line); $i++) {
      my $nuc = substr($line, $i, 1);
      exists($pwm[$i]) || ($pwm[$i] = [ @zeroes ]);
      my $j = exists($nucs{$nuc}) ? $nucs{$nuc} : $nucs{"N"};
      $pwm[$i][$j]++;
  };

  $c++;

}; # while

print STDERR "# Processed $c sequences...\n";
close(SEQSFILE);

my $I = scalar(@pwm);
my $J = scalar(@zeroes);

# relative frequencies matrix
for (my $i = 0; $i < $I; $i++) {
    for (my $j = 0; $j < $J; $j++) {
        $pwm[$i][$J] += $pwm[$i][$j];
        print STDERR "$i $j $pwm[$i][$j]\n";
    };
};

# computing loglikelyhood matrix
for (my $i = 0; $i < $I; $i++) {
    for (my $j = 0; $j < $J; $j++) {
        $pwm[$i][$j] = &loglikelyhood($pwm[$i][$j]/$pwm[$i][$J]);
        print STDERR "$i $j $pwm[$i][$j]\n";
    };
};

# writing the matrix
print STDOUT <<"EOF";
##
## A simple Position Weight Matrix
## derived from a set of sequences
```

```
## from $file
##
EOF

print STDOUT "PO",
             join("", map { sprintf("%10s", $_) } @nucleotides), "\n";

$J = scalar(@nucleotides) - 1;
my $k = 0;
for (my $i = 0; $i < $I; $i++) {
    my @tmp = @{ $pwm[$i] };
    print STDERR sprintf("%02d", $k+1)," @tmp\n";
    print STDOUT sprintf("%02d", ++$k),
                 join("", map { sprintf("%10.3f", $_) } @tmp[(0..$J)]), "\n";
};

print STDOUT "XX $col\n";

exit(0);

sub loglikelyhood() {
    my ($score,$lklhd);
    $score = shift;
    $lklhd = $score == 0 ? -9999 : log($score / 0.25) / log(2);
    return $lklhd;
} # loglikelyhood
```

pwmfinder.pl

```
#!/usr/bin/perl
#
## ##############################################################################################
##
##   pwmfinder.pl - finding possible locations for a signal modeled with a PWM model
##
## ##############################################################################################
##
##                 CopyLeft 2024 (CC:BY-NC-SA) --- Josep F Abril
##
##   This file should be considered under the Creative Commons BY-NC-SA License
##   (Attribution-Noncommercial-ShareAlike). The material is provided "AS IS",
##   mainly for teaching purposes, and is distributed in the hope that it will
##   be useful, but WITHOUT ANY WARRANTY; without even the implied warranty
##   of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
##
## ##############################################################################################
#

use strict;
use warnings;

if (scalar(@ARGV) < 1) {

  print STDERR "ERROR: pwmfinder.pl requires a PWM file\n";
  exit(1);

};

if (!open(MATFILE,"< $ARGV[0]")) {

  print STDERR "ERROR: pwmfinder.pl cannot open $ARGV[0] file\n";
  exit(1);

};

my %pwm = ();         # storing PWM in a hash
my @nucleotides = (); # array to store nucleotides chars
my $reading_pwm = 0;  # Flag: are we reading scores for PWM or not
my $pos = 0;          # Temporary variable to store the current position
                      # on the PWM being read.

while (<MATFILE>) {

  next if /^\#/o;
  next if /^\s*$/o;

  my $line = $_;
```

```
  ($reading_pwm == 0
   && $line =~ m/\AP0\s+(\w+)\s+(\w+)\s+(\w+)\s+(\w+)/
   ) && do {

      $nucleotides[0] = $1;
      $nucleotides[1] = $2;
      $nucleotides[2] = $3;
      $nucleotides[3] = $4;

      $reading_pwm = 1;

      next;

  }; # if $reading_pwm == 0

  ($reading_pwm == 1) && do {

      $line =~ m/\A\d+\s+([\-\d\.]+)\s+([\-\d\.]+)\s+([\-\d\.]+)\s+([\-\d\.]+)/
        && do {

          $pwm{$nucleotides[0]}[$pos] = $1;
          $pwm{$nucleotides[1]}[$pos] = $2;
          $pwm{$nucleotides[2]}[$pos] = $3;
          $pwm{$nucleotides[3]}[$pos] = $4;

          $pos++;

          next;

      };

      $line =~ m/\AXX\s+(\d+)/
        && do {

          $pwm{P}      = $1;
          $reading_pwm = 0;

      };

  }; # if $reading_pwm == 1

}; # while

my $i = 0;
my $len = scalar(@nucleotides);

while ($i < $len) {

  print STDERR "$nucleotides[$i] : @{ $pwm{ $nucleotides[$i] } }\n";

  $i++;

}; # while

print STDERR "Intron first nucleotide: $pwm{P}\n";

# Reading a DNA sequence in fasta format from file

my $iden = <STDIN>;  # Reading first line of fasta file
                     # which conatins the sequence identifier ">XXXX"...
my @seql = <STDIN>;  # Reading the sequence from the rest of lines...
                     # The above two lines will only work
                     # when fasta file has only one sequence...
my $seq  = q{};      # A string to concatenate all the sequence lines...

$i = 0;
while ($i < scalar(@seql)) {

  chomp($seql[$i]);
  $seq = $seq . $seql[$i];
  $i = $i + 1;

};

$seq = uc($seq); # Convert all nucleotides to UpperCase

my @vseq = split //o, $seq;
         # Storing sequence string into a nucleotide vector
```