BScBI-CG

Practicals Report

Jan Izquierdo

Exercise 05

— 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 [Engel et al., 2013] 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...

mkdir -v $WDR/seqs/donor_ex05

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/donor_ex05/refgenome_alldonorsites_allchr.tbl \
      2> $WDR/seqs/donor_ex05/refgenome_alldonorsites_allchr.log;

gawk '{print $2}' $WDR/seqs/donor_ex05/refgenome_alldonorsites_allchr.log | \
  sort | uniq -c;
#     47 skipping
#    315 using
```

```
wc -l $WDR/seqs/donor_ex05/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/donor_ex05/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/donor_ex05/refgenome_alldonorsites_sequences.fasta
egrep -v '^>' $WDR/seqs/donor_ex05/refgenome_alldonorsites_sequences.fasta \
            > $WDR/seqs/donor_ex05/refgenome_alldonorsites_sequences.tbl

egrep -c '^>' $WDR/seqs/donor_ex05/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:

\begin{center} \texttt{ex_seqs/donor_50_sequences.setX.tbl} \texttt{ex_seqs/donor_50_sequences.setY.tbl} \texttt{ex_seqs/donor_50_sequences.setZ.tbl}
\texttt{ex_seqs/donor_100_sequences.setX.tbl} \texttt{ex_seqs/donor_100_sequences.setY.tbl} \texttt{ex_seqs/donor_100_s
\end{center}

```
# You can use the "shuf" command to randomize record lines,
# and "head" to recover top n-elements of the shuffled list.
#
shuf $WDR/seqs/donor_ex05/refgenome_alldonorsites_sequences.tbl | \
  head -50 - > $WDR/ex_seqs/donor_50_sequences.setX.tbl;

# you will need to repeat three times for the setX, setY, and setZ
# as well as for acceptors. Include your commands here:

#%

for num in 50 100; do {

    for rep in X Y Z; do {

      cat $WDR/seqs/donor_ex05/refgenome_alldonorsites_sequences.tbl| shuf| head -n$num > $WDR/ex_seqs/don

      }; done

  }; done

#-%#


#
```

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).

```
#%

# step 4 - For acceptor

mkdir -v $WDR/seqs/acceptor_ex05

gawk 'BEGIN{
        while (getline<ARGV[1]>0) { NOSS[$1"."$2":"$3]++; };
        ARGV[1]="";
      }
      $3 ~ /CDS\.(internal|terminal)/ { #acceptor sites
        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, $4 - 1;
        } else {         # reverse features
          print $1, $7, $5 + 1;
        };
      }' $WDR/seqs/translational_frameshifts.tbl \
         $WDR/seqs/refgenome_allcdsannotated_allchr.gff \
       > $WDR/seqs/acceptor_ex05/refgenome_allacceptorsites_allchr.tbl \
      2> $WDR/seqs/acceptor_ex05/refgenome_allacceptorsites_allchr.log;


# step 5 - Clip acceptor sequences
gawk '{ if ($2 == "+") print $1, $2, $3 - 24, $3 + 3; #change to given values
                  else print $1, $2, $3 - 3, $3 + 24; #change to given values
        }' $WDR/seqs/acceptor_ex05/refgenome_allacceptorsites_allchr.tbl | \
  while read chrom strnd acc_start acc_end;
    do {
      echo "# trimming $chrom ($strnd) $acc_start $acc_end" 1>&2;
      if [ "X$strnd" == "X+" ];
        then
            samtools faidx $WDR/seqs/refgenome_chromosomes.fasta \
                           $chrom.Scer:$acc_start-$acc_end ;
        else
            samtools faidx $WDR/seqs/refgenome_chromosomes.fasta \
                           $chrom.Scer:$acc_start-$acc_end | \
                revseq -sequence fasta::stdin -outseq stdout;
      fi;
    }; done > $WDR/seqs/acceptor_ex05/refgenome_allacceptorsites_sequences.fasta


egrep -v '^>' $WDR/seqs/acceptor_ex05/refgenome_allacceptorsites_sequences.fasta \
            > $WDR/seqs/acceptor_ex05/refgenome_allacceptorsites_sequences.tbl

#-%#
```

Filter sets of 50 and 100 random sequences

```
#%
```

```
for num in 50 100; do {

    for rep in X Y Z; do {

      cat $WDR/seqs/acceptor_ex05/refgenome_allacceptorsites_sequences.tbl| shuf| head -n$num > $WDR/ex_se

      }; done

  }; done

#-%#
```

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.

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

#%

cd ex_seqs

ln $WDR/data/* .

ln $WDR/seqs/donor_ex05/refgenome_alldonorsites_sequences.tbl ./donor_allsequences.setALL.tbl
ln $WDR/seqs/acceptor_ex05/refgenome_allacceptorsites_sequences.tbl ./acceptor_allsequences.setALL.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 [Wagih, 2017][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);

path<-Sys.getenv("WDR")

# loading the sequence set
```

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

```
donors.setALL <- readLines(paste0(path, "/seqs/donor_ex05/refgenome_alldonorsites_sequences.tbl")); #Chang

# 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)");

# 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=paste0(path, "/images/alldonors_logo+pictogram.png"), #Changed the path
       P, height=6, width=9, units="in", dpi=600)
```

Create and execute the file from the docs folder

```
#%
gedit $WDR/docs/freq_vis.R

Rscript docs/freq_vis.R

#-%#
```

Static

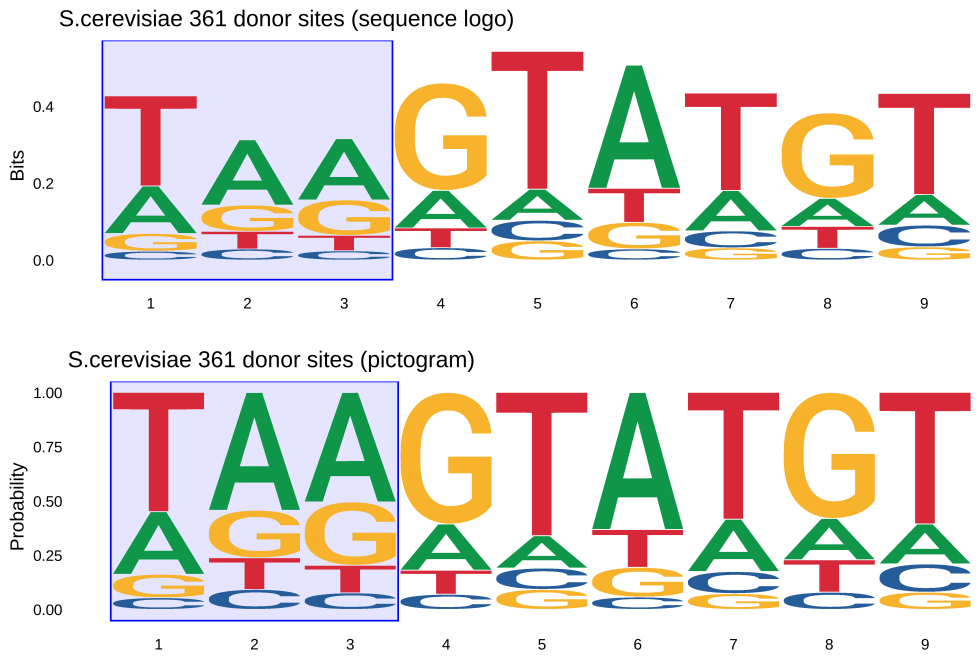

Figure 1: **Visualizing PWM generated from a set of *S. cerevisiae* donor sites.** Top panel shows a sequence logo, while the bottom corresponds to a pictogram, for a set of 361 donor site sequences from yeast CDSs annotations. Blue area on the left defines the three exonic final nucleotide positions.

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.

```
#%
touch $WDR/docs/sequence_logo.R

#define the bash function
run_Seq_rscript() {
    #Target directory
    dir="$WDR/ex_seqs"

    #rscript is 2nd input
    rsc=$1

    #Dir exists?
    if [[ ! -d "$dir" ]]; then
        echo "Directory $dir does not exist."
        return 1
  fi

    #Rscript exists?
    if [[ -f "$file" ]]; then
    echo "R file $rsc does not exist."
        return 1
    fi

    #Run files to Rscript
    for file in "$dir"/*; do
      if [[ -f "$file" ]]; then
        echo "Processing $file"
        Rscript "$rsc" "$file"
        echo "Done $file"
    fi
    done
```

```
}

rsc="$WDR/docs/sequence_logo.R"

run_Seq_rscript $rsc

#-%#
```

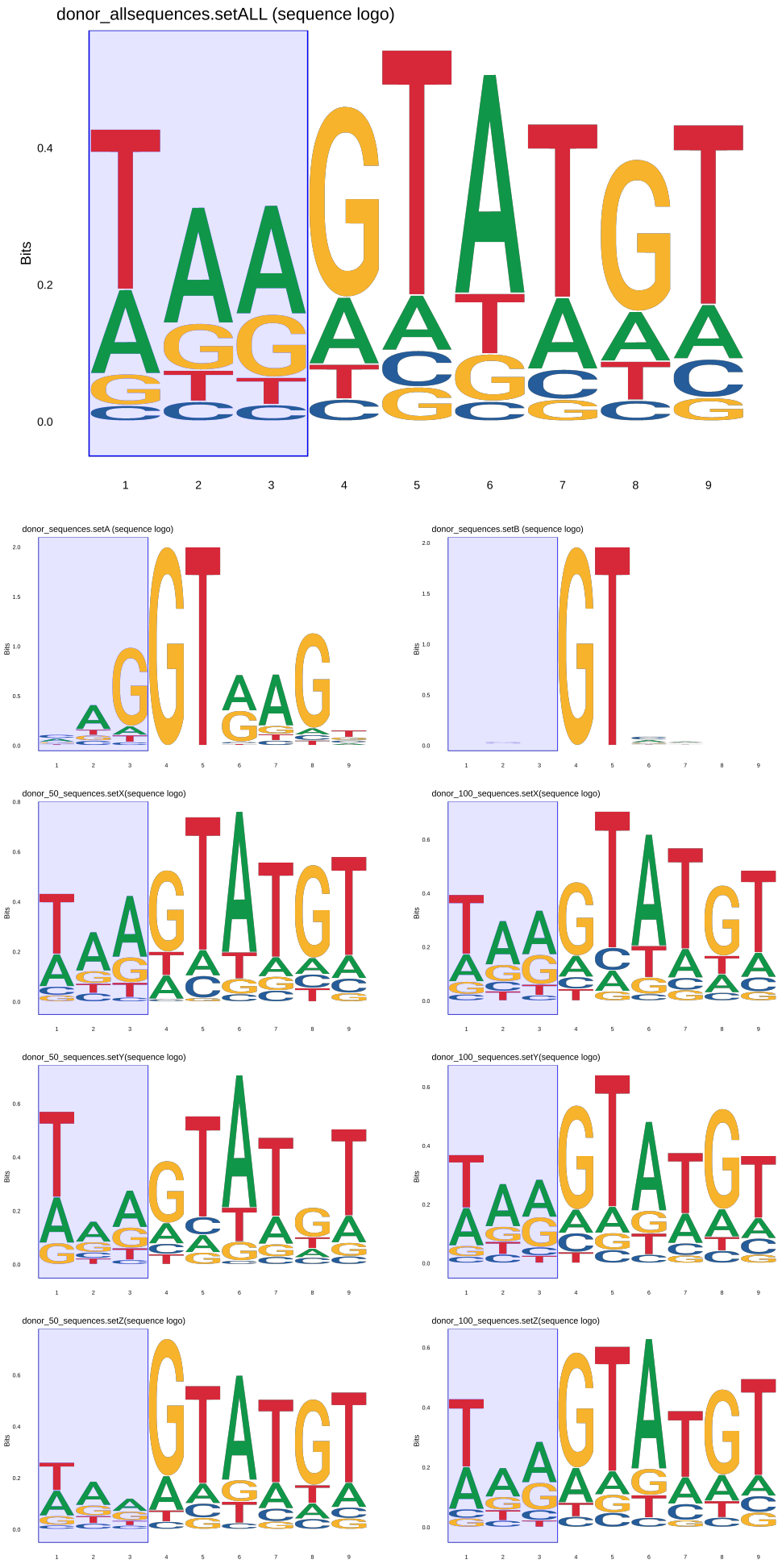

Figure 2: **Visualizing PWM generated from a sets of *S. cerevisiae* donor sites.** Show sequence logos of all donor sequences

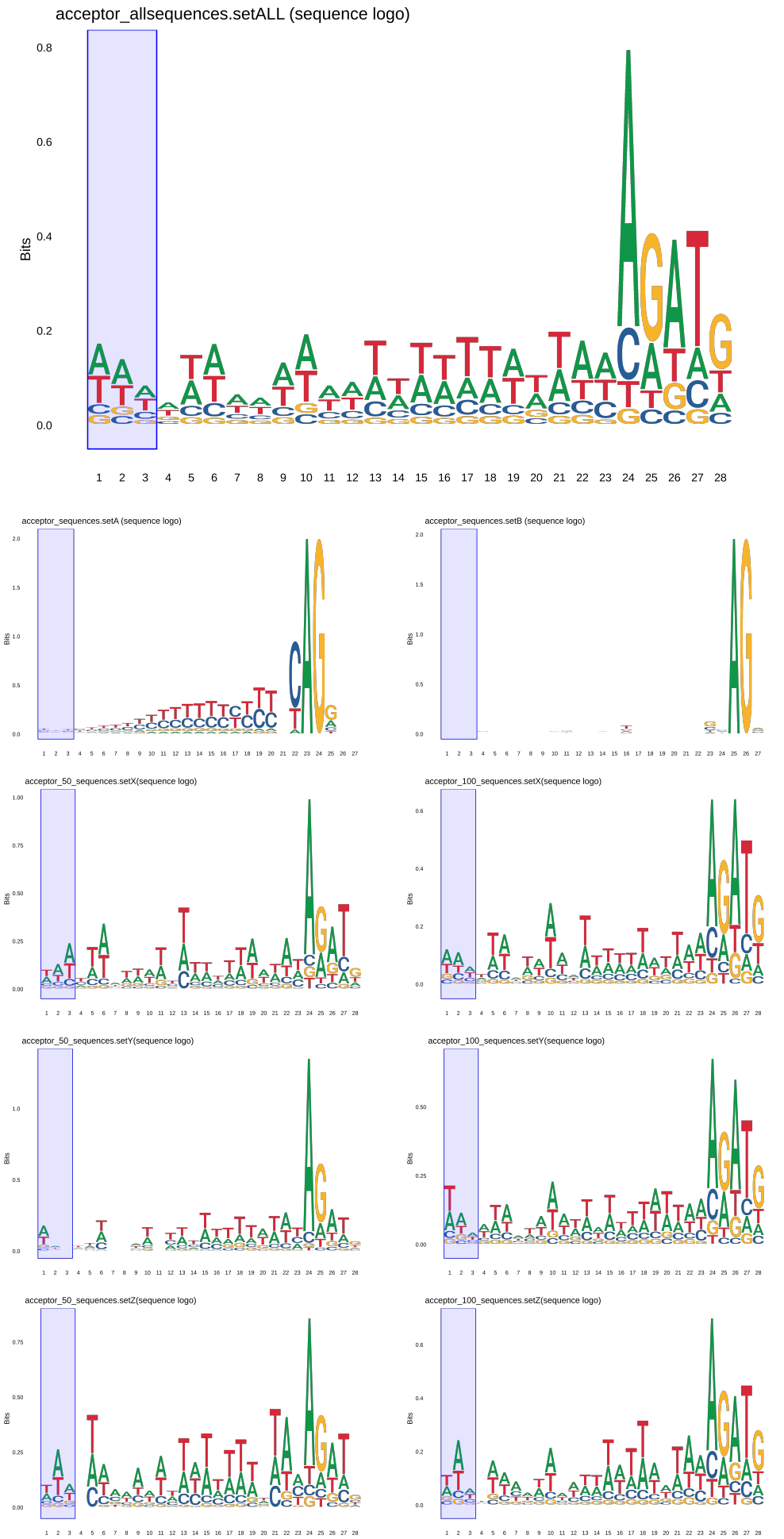

Figure 3: **Visualizing PWM generated from a sets of *S. cerevisiae* acceptor sites.** Show sequence logos of all acceptor sequences

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 20):

```
##
## 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 18) 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 14).

```
$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.

#%

dir="$WDR/ex_seqs"

for file in "$dir"/*; do #file with path and extension

  file_cut=$(echo $file|cut -f12 -d'/' | rev | cut -d'.' -f2,3 | rev) #only the name, without extension an

  echo $file

  #if acceptor first segment ends at 24
  if [[ "$file_cut" =~ acceptor ]]; then
    $BIN/makepwms.pl 24 $file > $WDR/data/$file_cut.pwm
  fi

  #if donor first segment ends at 4
  if [[ "$file_cut" =~ donor ]]; then
    $BIN/makepwms.pl 4 $file > $WDR/data/$file_cut.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
```

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.

All .pwm files in data to .predicted_splice_sites.tbl Parse and compare the files to assess TP, FP, FN and accuracy to fill the latex table

```
#%

TableFiller() {

  for type in donor acceptor; do

    echo "Type is: $type"
    echo ""

    #Annotated
    annotated_list=$(grep 'chrXV ' "$WDR/seqs/${type}_ex05/refgenome_all${type}sites_allchr.tbl" | awk '{p
```

```
    an=$(echo "$annotated_list"|wc -l)

    #Predicted
    dir_data="$WDR/data"

    for file in "$dir_data"/${type}*predicted_splice_sites.tbl; do #file with path and extension

      #Predicted count
      pred=$(cat $file |wc -l)

      #TP=lines in both
      TP=$(awk '{print $1}' "$file" | grep -Fxf <(echo "$annotated_list") |wc -l)

      #FP=Lines in predicted file that are not in annotated.
      FP=$(awk '{print $1}' "$file" | grep -Fvxf <(echo "$annotated_list") | wc -l)

      #FN=Lines in annotated and not predicted
      FN=$(echo "$annotated_list" | grep -Fvxf <(awk '{print $1}' "$file") | wc -l)

      #TN=Lines missing in all #Nt written

      SN=$(echo "scale=6; $TP / ($TP + $FN)"|bc)
      SP=$(echo "scale=6; $TP / ($TP + $FP)"|bc)

      file_cut=$(echo $file|cut -f12 -d'/' | rev | cut -d'.' -f3 | rev) #only the name, without extension

      echo "  For $file_cut:"
      echo "  "
      echo "    Predicted: $pred"
      echo "    Annotated: $an"
      echo "    TP: $TP"
      echo "    FP: $FP"
      echo "    FN: $FN"
      echo "    Sensitivity: $SN"
      echo "    Specificity: $SP"

      echo "  ---------------------------------------"

    done;

    echo ""
    echo "################################################"
    echo ""

  done

}

TableFiller > $WDR/data/Accuracy_table.txt

#-%#
```

Use this table file to fill the latex tables.

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.

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, "precission") 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	12	12	12	12	12	12	12	12	12
Predicted	217526	21642	56613	236951	222108	218400	173635	189587	227654
TP	8	3	3	8	7	7	7	7	7
TN									
FP	217518	21639	56610	236943	222101	218393	173628	189580	218393
FN	4	9	9	4	5	5	5	5	5
Sensitivity	.666	.250	.250	.666	.583	.583	.583	.583	.666
Specificity	.000036	.000138	.000052	.000033	.000031	.000032	.000040	.000036	.000035

Table 2: **Summary of accuracy assessment results for acceptor splice sites.**
Sensitivity (Sn, "recall") calculated as $Sn = \frac{TP}{TP+FN}$, specificity (Sp, "precission") 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	12	12	12	12	12	12	12	12	12
Predicted	344203	17475	57960	216192	246931	209367	297971	299059	316952
TP	4	3	0	3	2	3	4	4	5
TN									
FP	344199	17472	57960	216189	246929	209364	297967	299055	316947
FN	8	9	12	9	10	9	8	8	7
Sensitivity	.333	.250	0	.250	.166	.250	.333	.333	.416
Specificity	.000011	.000171	0	.000013	.000008	.000014	.000013	.000013	.000015

5 Discussion

This study used Position Weight Matrices (PWMs) to predict splice sites on chromosome XV in Saccharomyces cerevisiae. PWMs, which capture sequence composition, showed varied accuracy across different datasets. Overall, donor site predictions had higher sensitivity (up to 66.6% in sets like setALL and setX.50) compared to acceptor sites, which peaked at 41.6% in setZ.100. This suggests that donor sites may have more consistent patterns than acceptor sites, improving prediction accuracy. Specificity, however, was low across all splice site types, highlighting the difficulty in filtering out false positives. PWMs assume each sequence position is independent, which doesn't fully capture biological dependencies, limiting their effectiveness. Interestingly, more data didn't always improve accuracy; sometimes smaller sets performed better. Still, setALL, with more diverse sequences, maintained steady sensitivity for both splice types.

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: CG_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
```

```
# Score all possible signals using the PWM

$i = 0;
while ($i < scalar(@vseq) - $pos) {

  my $k = 0;
  my $p = 0;

  while ($k < $pos) {

    $p = $p + $pwm{ $vseq[$i + $k] }[$k];
    $k++;

  }; # while $k

  $p > 0 &&
      print STDOUT $i + $pwm{P}, " ", sprintf("%10.3f", $p), "\n";

  $i++;

}; # while $i
```

freq_vis.R

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

path<-Sys.getenv("WDR")

# loading the sequence set
donors.setALL <- readLines(paste0(path, "/seqs/donor_ex05/refgenome_alldonorsites_sequences.tbl")); #Changed the path

# 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)");

# 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); #Commented so it doesn't generate a pdf

# saving the picture
P <- gridExtra::arrangeGrob(p1, p2);
ggsave(file=paste0(path, "/images/alldonors_logo+pictogram.png"), #Changed the path
       P, height=6, width=9, units="in", dpi=600)
```

sequence_logo.R

```
library(ggplot2)
library(ggseqlogo)
library(gridExtra)

path<-Sys.getenv("WDR")

args = commandArgs(trailingOnly=TRUE)

print(length(args))

if (length(args)<1) {
  stop("MISSING FILE NAME.n", call.=FALSE)
}

seqFile<-args[1]
```

```
#name of the sequencs
seqName<-result <- sub("\\.tbl$", "", basename(seqFile))

# loading sequence sets
set_seq <- readLines(seqFile)

# 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( set_seq, method = 'bits' ) +
  ggtitle(paste0(seqName, " (sequence logo)"))

ggsave(file=paste0(path, "/images/", seqName,"_seqLogos.png"), #Changed the path
       p1, height=6, width=9, units="in", dpi=600)
```

6.2.3 Shell global vars and settings for this project

projectvars.sh

```
## ############################################################################################
##
##   projectvars.sh
##
##   A BASH initialization file for BScBI-CG practical exercise folders
##
## ############################################################################################
##
##                 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.
##
## ############################################################################################

#
# Base dir
export WDR=$PWD; # IMPORTANT: If you provide the absolute path, make sure
                 #            that your path DOES NOT contains white-spaces
                 #            otherwise, you will get weird execution errors.
                 #            If you cannot fix the dir names containing such white-space
                 #            chars, you MUST set this var using the current folder '.'
                 #            instead of '$PWD', i.e:    export WDR=.;
export BIN=$WDR/bin;
export DOC=$WDR/docs;

#
# Formating chars
export TAB=$'\t';
export RET=$'\n';
export LC_ALL="en_US.UTF-8";

#
# pandoc's vars
NM="Izquierdo_Jan";                  #-> IMPORTANT: SET YOUR SURNAME and NAME ON THIS VAR,
RB="README_BScBICG2425_exercise05"; #->            MUST FIX ON MARKDOWN README FILE
                                     #->            FROM TARBALL (AND INSIDE TOO)
RD="${RB}_${NM}";
PDOCFLGS='markdown+pipe_tables+header_attributes';
PDOCFLGS=$PDOCFLGS'+raw_tex+latex_macros+tex_math_dollars';
PDOCFLGS=$PDOCFLGS'+citations+yaml_metadata_block';
PDOCTPL=$DOC/BScBI_CompGenomics_template.tex;
export RD PDOCFLGS PDOCTPL;

#
function ltx2pdf () {
    RF=$1;
    /usr/bin/pdflatex $RF.tex;
    /usr/bin/bibtex $RF;
    /usr/bin/pdflatex $RF.tex;
    /usr/bin/pdflatex $RF.tex;
}
```

```
function runpandoc () {
  /usr/bin/pandoc -f $PDOCFLGS      \
         --template=$PDOCTPL        \
         -t latex --natbib          \
         --number-sections          \
         --highlight-style pygments \
         -o $RD.tex $RD.md;
  ltx2pdf $RD;
}

#
# add your bash defs/aliases/functions below...
export REFDIR="S288C_reference_genome_R64-5-1_20240529";
export REFGEN="S288C_reference_sequence_R64-5-1_20240529"
```

6.3 About this document

This document was be compiled into a PDF using `pandoc` (see `projectvars.sh` from previous subsection) and some `LaTeX` packages installed in this linux system. `synaptic`, `apt-get` or `aptitude` can be used to retrieve and install those tools from linux repositories. As the `raw_tex` extension has been provided to the `markdown_github` and `tex_math_dollars` formats, now this document supports inline LaTeX and inline formulas!

You can get further information from the following links about the Mark Down syntax, as well as from the manual pages (just type `man pandoc` and/or `man pandoc_markdown`).

7 References

S.R. Engel, F.S. Dietrich, D.G. Fisk, G. Binkley, R. Balakrishnan, M.C. Costanzo, S.S. Dwight, B.C. Hitz, K. Karra, R.S. Nash, S. Weng, E.D. Wong, P. Lloyd, MS. Skrzypek, S.R. Miyasato, M. Simison, and J.M. Cherry. The Reference Genome Sequence of *Saccharomyces cerevisiae*: Then and Now. *G3 (Bethesda)*, 4(3):389–98, 2013. doi: 10.1534/g3.113.008995. PMID:24374639.

O Wagih. `ggseqlogo`: a versatile `R` package for drawing sequence logos. *Bioinformatics*, 33(22):3645–47, 2017. doi: 10.1093/bioinformatics/btx469. PMID:29036507.