Comparative RNASeq Analysis

Thomas Nicholson

14/09/2020

Overview of sRNA

RNAs play a critical role in a wide range of biological functions such as:

- Transcription/Translation
  - rRNA, tRNA, 6sRNA etc.
- Immune response
  - CRISPR-cas
- Gene regulation
  - Riboswitches, sRNAs binding to mRNA etc.
- Virulence

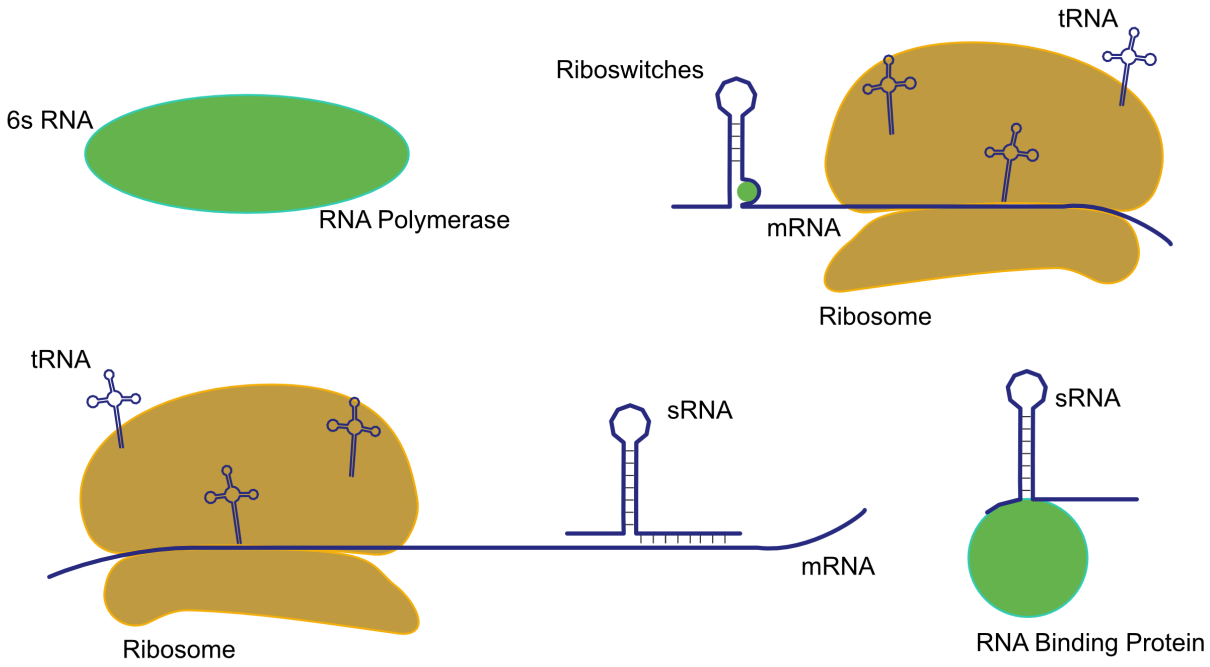


Figure 1: Figure 1. Examples of ncRNAs in bacteria

Overview of Methods

- Take RNASeq data from multiple genomes
  - 21 strain

  - 11 genera
  - 6 families
- Predict sRNAs based on expressed regions in RNASeq data
- Use multiple RNASeq datasets for each genome
- Consider a number of different approaches for evaluating the predicted regions
  - Conservation of transcription
  - Conservation of sequence (nhmmer search across genomes from the analysed clade).
  - GC content
  - Covariation observed in sequence alignments (using R-scape)
  - Secondary structure (minimum free energy from RNAAlifold and the Z score of the MFE from alifoldz)
  - Presence of ncRNA motifs (using the rmfam dataset)
- Two control groups will be used
  - Previously annotated sRNAs will be used as a positive control
  - random intergenic sequences of the same lengths as the predicted sRNAs will be used as a negative control

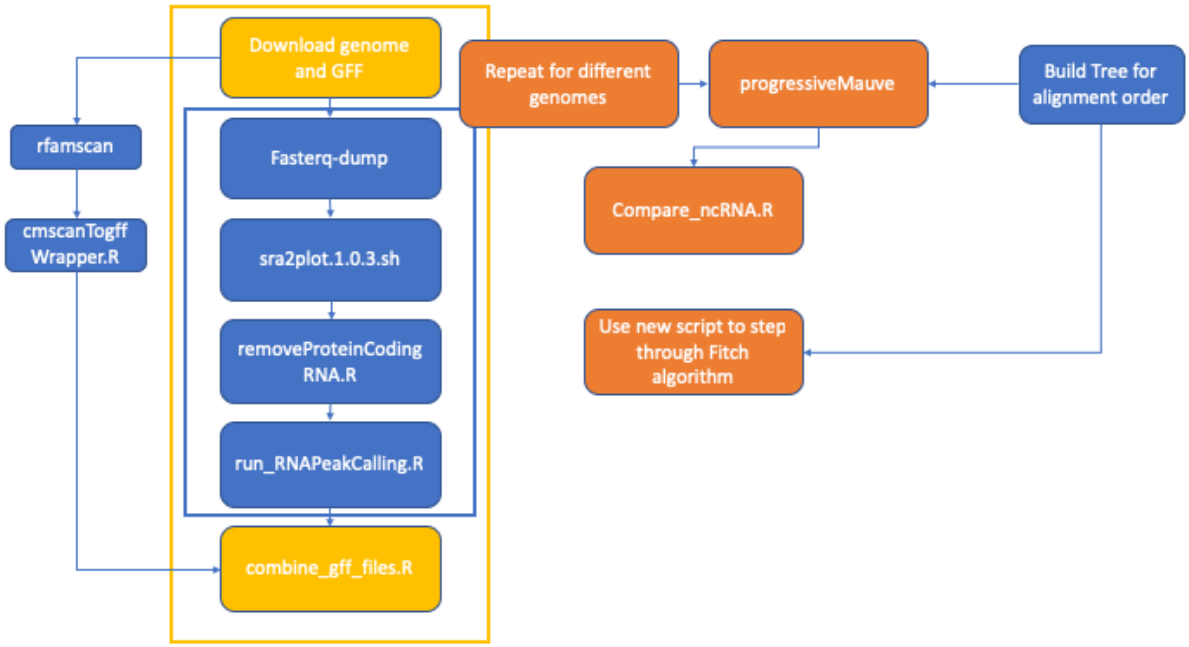


Figure 2: Figure 4. Workflow of methods

Summary of strains used

```
load("~/bin/r_git/R/r_files/accession_info.Rda")
accession_info <- accession_info %>%
  mutate(strain_short = substr(Strain, start = 1, stop = 30)) %>%
  select(Accession, RNASeq.file.counts, strain_short)
accession_info[1:20,]
```

```
##          Accession RNASeq.file.counts                  strain_short
## 1  GCA_000017745.1                  2       Escherichia coli E24377A
## 2  GCA_000017765.1                  8            Escherichia coli HS
## 3  GCA_000017985.1                  6 Escherichia coli B str. REL606
```

```
## 4  GCA_900186905.1               10              Escherichia coli
## 5  GCA_002843685.1               10 Escherichia coli str. K-12 sub
## 6  GCA_001559675.1                8          Escherichia coli K-12
## 7  GCA_000497505.1                4          Shigella dysenteriae
## 8  GCA_000283715.1                8               Shigella sonnei
## 9  GCA_000007405.1                8            Shigella flexneri
## 10 GCA_002504125.1                5 Salmonella enterica subsp. ent
## 11 GCA_000006945.2               10 Salmonella enterica subsp. ent
## 12 GCA_900184385.1               10 Salmonella enterica subsp. ent
## 13 GCA_000210855.2               10 Salmonella enterica subsp. ent
## 14 GCA_002813995.1               10 Salmonella enterica subsp. ent
## 15 GCA_002848605.1               10         Klebsiella pneumoniae
## 16 GCA_000220485.1                5 Klebsiella pneumoniae KCTC 224
## 17 GCA_001874505.1                2        Enterobacter hormaechei
## 18 GCA_002303275.1               10           Enterobacter cloacae
## 19 GCA_000747565.1                7              Serratia sp. SCBI
## 20 GCA_000438825.1                8        Serratia plymuthica S13
```

Download data and map reads

Scripts involved for each Accession

- *callPeaksforGenome.sh -g <GCA Accession>*
  - Only accessions with >4 RNASeq files are analysed
  - *fetch_genomes_from_GCA.sh -r <GCA Accession> -g*
    - The genome and gff files are downloaded from ncbi using the GCA acession
    - *-g* flag is for downloading GFF file
  - The RNASeq data is downloaded using *fasterq-dump* with a given accession
    - these are selected from a file (shown below) containing a list of RNASeq experiment IDs for each strain.
    - filtered for paired ends, Illumina HiSeq
  - *sra2plot.1.0.3.sh -s <SRA Accession> -r <GCA Accession> -d -n <Number of CPUs>*
    - Maps the reads
    - *-d* turns off the downloading function of the script as this is being done separately
  - *removeProteinCodingRNA.R -f <SRA Accession> -g <GCA Accession>*
  - *run_rnaPeakCalling.R -f <SRA Accession> -g <GCA Accession>*
  - *rfamscan <GCA Accession>*
    - Searches the given genome for rFam models and reformats output into GFF format
    - *cmscanToGffWrapper.R -f <GCA Accession>.tblout -g <GCA Accession>*
  - *combine_gff_files.R* -f *./gff_files/ -o <GCA Accession>*

```
load("~/bin/r_git/R/r_files/sra_rnaseq_files.Rda")
sra_rnaseq_files <- sra_rnaseq_files %>% select(GENOME_ACCESSION, ACCESSION, SPECIES)
sra_rnaseq_files[1:10,]
```

```
##    GENOME_ACCESSION ACCESSION                                      SPECIES
## 1   GCA_000009285.2 DRR008642                        Ralstonia eutropha H16
## 2   GCA_000009285.2 DRR008643                        Ralstonia eutropha H16
## 3   GCA_002310435.1 DRR012600 Staphylococcus aureus subsp. aureus str. Newman
## 4   GCA_002310435.1 DRR012601 Staphylococcus aureus subsp. aureus str. Newman
## 5   GCA_002310435.1 DRR012602 Staphylococcus aureus subsp. aureus str. Newman
## 6   GCA_002310435.1 DRR012603 Staphylococcus aureus subsp. aureus str. Newman
## 7   GCA_002310435.1 DRR012604 Staphylococcus aureus subsp. aureus str. Newman
```

```
## 8   GCA_002310435.1 DRR012605 Staphylococcus aureus subsp. aureus str. Newman
## 9   GCA_000021465.1 DRR013233                          Helicobacter pylori P12
## 10  GCA_000021465.1 DRR013234                          Helicobacter pylori P12
```

Call peaks on individual RNASeq experiments

- A plot file is produced. This contains a number for each nucleotide that indicates read depth.
- The read depth gets set to 0 for all coding regions of the file
  - This is done as identifying ncRNAs inside coding regions is a much more challenging problem than simply peak calling
- For the remaining positions, the read depth is normalised and any region where the read depth is above a threshold for >50 nt is called a peak.
  - Threshold is set to the equivalent of ~15 nt read depth before normalisation

callPeaksforGenome.sh

```
#!/bin/bash

##--------------------------------------------------------------------##
##---------------------------- Setup Variables ----------------------##
##--------------------------------------------------------------------##

FILE_PATH=`dirname $0`
number_of_sra="10"
output_path="./"
CPUS='6'
output_log=/dev/stdout
display_available_files="F"

##--------------------------------------------------------------------##
##------------------------ User Input Options ------------------------##
##--------------------------------------------------------------------##

while getopts "g:n:o:c:qth" arg; do
  case $arg in
    g)
      gca=$OPTARG
      ;;
    n)
      number_of_sra=$OPTARG
      ;;
    o)
      output_path=$OPTARG
      ;;
    c)
      CPUS=$OPTARG
      ;;
    q)
      output_log=$gca.log
      ;;
    t)
    display_available_files="T"
```

```
    ;;
    h)
echo '# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -'

      ;;

    esac
done

##------------------------------------------------------------------##
##------------------------- Tests For Inputs ---------------------##
##------------------------------------------------------------------##
if [[ -z $gca ]]; then
echo 'Error: GCA needed. Specify with -g <gca>'
echo ' '
echo 'Use -h for more help.'
echo ' '
exit
fi

counts=`grep $gca ~/phd/RNASeq/SRA_bacteria_RNAseq.txt | grep "PAIRED" | grep "Illumina HiSeq" | wc -l`
if (( $counts == 0 )); then
echo "No valid RNAseq datasets for $gca"

exit
fi

if [[ $display_available_files == "T" ]]; then
grep $gca ~/phd/RNASeq/SRA_bacteria_RNAseq.txt | grep "PAIRED" | grep "Illumina HiSeq"
exit
fi

##------------------------------------------------------------------##
##---------------------- Set up folders/files ---------------------##
##------------------------------------------------------------------##

cd $output_path
mkdir -p "$gca.data"
cd "$gca.data"
mkdir gff_files
echo "Output to $output_log"

if (( $counts > $number_of_sra )); then

grep $gca ~/phd/RNASeq/SRA_bacteria_RNAseq.txt | grep "PAIRED" | cut -f1 | head -n $number_of_sra > tmp

else

grep $gca ~/phd/RNASeq/SRA_bacteria_RNAseq.txt | grep "PAIRED" | cut -f1 > tmp1

fi


##------------------------------------------------------------------##
```

```
##---------------------- Download Genome and GFF ------------------##
##-----------------------------------------------------------------##

    if [[ -f "${gca}.fna" ]]; then
    echo "$gca.fna already downloaded."
    else
    echo "Downloading $gca Genome and GFF files"
    fetch_genomes_from_GCA.sh -r $gca -g >> $output_log
    fi

if [ $? -eq 0 ]; then
    echo " "
else
     echo "Error: Downloading $gca Genome and GFF files failed. See fetch_genomes_from_GCA.sh"
     exit $?
fi


##-----------------------------------------------------------------##
##-------------- Download and Process RNASeq Files ----------------##
##-----------------------------------------------------------------##
file_lines=`cat tmp1`

for line in $file_lines ;
do

    if [[ -f "${line}_sra_calls.gff" ]]; then

    echo "$line already downloaded."

    else

    echo "Downloading $line"
    fasterq-dump --split-3 -p $line >> $output_log
    echo "Mapping reads"
    sra2plot.1.0.3.sh -s $line -r $gca -d -n $CPUS  >> $output_log

    plot_lenegth=`wc -l $line.plot  | cut -d ' ' -f2`
    rm *.sam
        if [ $plot_lenegth -gt 0 ]; then
        rm ${line}*fwd.plot
        rm ${line}*.rev.plot
        rm fastq/${line}*.fastq
        rm trimmed/${line}*.fastq
        fi
    rm /Users/thomasnicholson/ncbi/public/sra/*.cache
    echo "Removing CDS"
    removeProteinCodingRNA.R -f $line -g $gca >> $output_log
    echo "Calling Peaks"
    run_rnaPeakCalling.R -f $line  -g $gca >> $output_log

    fi
    cp ${line}_sra_calls.gff ./gff_files/
```

```
done

##--------------------------------------------------------------------##
##---------------------- Search for rFam models ----------------------##
##--------------------------------------------------------------------##

rfamscan() { counts=$( bc -l <<< "scale=2;$(esl-seqstat $1.fna | grep ^"Total" | tr -s ' ' | cut -d ' '

if [[ -f "${gca}_ncRNA.gff" ]]; then
    echo "${gca}_ncRNA.gff exists"
else
    echo "Running cmscan using rfam models"
    rfamscan $gca  >> $output_log
fi

cp $gca.gff ./gff_files/
cp ${gca}_ncRNA.gff ./gff_files


##--------------------------------------------------------------------##
##----------------------- Combine GFF Files ----------------------##
##--------------------------------------------------------------------##

if [[ ! -f "${gca}_new_calls.txt" ]]; then
combine_gff_files.R -f ./gff_files/ -o $gca
fi

echo "Finished."
rm tmp1
```

fetch_genomes_from_GCA.sh

```
#!/bin/bash

##--------------------------------------------------------------------##
##---------------------------- Help Message ----------------------##
##--------------------------------------------------------------------##

usage(){
    echo "fetch_genomes_from_GCA.sh is a script for downloading a genome (and GFF file) from a GCA acces
Usage:
 fetch_genomes_from_GCA.sh [opts] [input]

Options:
    -h  Display this help

Input
    -r  Reference genome accession (required)
    -o  Output name
    -e Fasta file extension
    -g include the GFF file
```

```
"
}

##--------------------------------------------------------------------##
##----------------------- User Input Options --------------------------##
##--------------------------------------------------------------------##

while getopts "r:o:e:gh" arg; do
case $arg in
    r)
    GENOME=${OPTARG};;
    o)
    OUTPUT=${OPTARG};;
    e)
    EXTENSION=${OPTARG};;
    g)
      GFF='y'
      ;;
    h)
        usage
        exit
      ;;
    \?)
    echo "Unknown option: -${OPTARG}" >&2; exit 1;;
    esac
done

##--------------------------------------------------------------------##
##-------------------------- Tests For Inputs ------------------------##
##--------------------------------------------------------------------##

if [ -z ${GENOME} ]; then
    echo "Error: No input specified." >&2
    usage
    exit 1
fi

if [ -z ${OUTPUT} ]; then

OUTPUT=${GENOME}

fi

if [ -z ${EXTENSION} ]; then

EXTENSION="fna"

fi

##--------------------------------------------------------------------##
##----------------------- Get IDs for download -----------------------##
##--------------------------------------------------------------------##
```

```
AssemblyName=$(esearch -db assembly -query ${GENOME} | efetch -format docsum | xtract -pattern Document
refseqID=$(esearch -db assembly -query ${GENOME} | efetch -format docsum | xtract -pattern DocumentSumma

refseq1=$(echo $refseqID | head -c 7 | tail -c 3)
refseq2=$(echo $refseqID | head -c 10 | tail -c 3)
refseq3=$(echo $refseqID | head -c 13 | tail -c 3)



##------------------------------------------------------------------##
##---------------------- Download fasta file ----------------------##
##------------------------------------------------------------------##

if [ ! -f $OUTPUT.$EXTENSION ];then

fastaLink="ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/$refseq1/$refseq2/$refseq3/$refseqID._$AssemblyNa

downloadLink=$(echo $fastaLink | sed 's/\._/_/g')

curl $downloadLink > $OUTPUT.$EXTENSION.gz
sleep 1
gunzip $OUTPUT.$EXTENSION.gz

if [ $? -eq 0 ]; then
    echo " "
else
    exit $?
fi


echo "$OUTPUT.$EXTENSION downloaded using $downloadLink"

else

fastaLink="ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/$refseq1/$refseq2/$refseq3/$refseqID._$AssemblyNa

downloadLink=$(echo $fastaLink | sed 's/\._/_/g')

echo "$OUTPUT.$EXTENSION already downloaded. To download again use $downloadLink"


fi

##------------------------------------------------------------------##
##---------------------- Download GFF file ----------------------##
##------------------------------------------------------------------##

if [[ $GFF = 'y' ]]; then

if [ ! -f $OUTPUT.gff ];then


gffLink="ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/$refseq1/$refseq2/$refseq3/$refseqID._$AssemblyName
```

```
downloadLink=$(echo $gffLink | sed 's/\._/_/g')

curl $downloadLink > $OUTPUT.gff.gz
sleep 1
gunzip $OUTPUT.gff.gz

if [ $? -eq 0 ]; then
    echo " "
else
     exit $?
fi

echo "$OUTPUT.gff downloaded using $downloadLink"

else

gffLink="ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/$refseq1/$refseq2/$refseq3/$refseqID._$AssemblyName/

downloadLink=$(echo $gffLink | sed 's/\._/_/g')

echo "$OUTPUT.gff already downloaded. To download again use $downloadLink"


fi

fi
```

sra2plot.1.0.3.sh

```
#!/bin/sh
#Downloads fastq files from SRA, trims, maps and generates plotfiles for visualisation in artemis

#Dependencies:
#curl
#sratoolkit
#samtools 1.6 (older versions may not work for generating plotfiles)
#bowtie2
#trimmomatic 0.36

usage(){
    echo "sra2plot.sh is a wrapper script for downloading, mapping and visualising RNA-seq data from the
Usage sra2plot [opts] [input]

    Options:
        -h  Display this help

        Input
            -r  Reference genome accession (required)
        -s  SRA run accession or name of split fastq files (Required. Format: FILE_1.fastq FILE_2.fastq
            -n  Number of cores

        Turn off defaults
```

```
        -d  Turn off download. Default: download genome and SRA from NCBI if not found in working direc
            (Genome accession must be in Genbank nucleotide format: https://www.ncbi.nlm.nih.gov/Sequin/
        -t  Turn off trimming
        -m  Turn off mapping
        -p  Don't make plotfiles
        -x  Don't cleanup files"
}
TPATH="/Users/thomasnicholson/bin/Trimmomatic_binary-0.36"
OUTDIR=""
SRA=""
GENOME=""
TRIM=true
MAP=true
PLOT=true
CLEAN=true
DOWNLOAD=true
THREADS=1

while getopts :s:r:n:thdmpx opt; do
    case "${opt}" in
    h) usage;exit;;
    t) TRIM=false;;
    s) SRA=${OPTARG};;
    r) GENOME=${OPTARG};;
    d) DOWNLOAD=false;;
    m) MAP=false;;
    p) PLOT=false;;
    x) CLEAN=false;;
    n) THREADS=${OPTARG};;
    \?) echo "Unknown option: -${OPTARG}" >&2; exit 1;;
    :) echo "Missing option argument for -${OPTARG}" >&2; exit 1;;
    *) echo "Unimplemented option: -${OPTARG}" >&2; exit;;
    esac
done
shift $((${OPTIND}-1))

if [ -z ${GENOME} ] || [ -z ${SRA} ]; then
    echo "Error: No input specified." >&2
    usage
    exit 1
fi

if [ -z ${TPATH} ]; then
    echo "Error: Path to trimmomatic install folder is not set.\n" >&2
    exit 1
fi

if $DOWNLOAD;then
    if [ ! -f ${GENOME}.fna ];then
    fetch_genomes_from_GCA.sh -r ${GENOME} -g
    fi
    if [ ! -f ${SRA}_*.fastq ];then
    fastq-dump --split-3 ${SRA}
```

```
    fi
fi

if $TRIM;then
    if [ ! -d trimmed ];then
    mkdir trimmed
    fi
    java -jar ${TPATH}/trimmomatic-0.36.jar PE -threads `echo $((2*${THREADS}))` ${SRA}_1.fastq ${SRA}_1
fi

if $MAP; then
    #Build index of genome if necessary
    if [ ! -d index ]; then
    mkdir index
    fi
    bowtie2-build ${GENOME}.fna ${GENOME} &&

    mv *.bt2* index/

    bowtie2 -p `echo "$((${THREADS}))"` -x index/${GENOME} -1 trimmed/${SRA}_1_paired.fastq -2 trimmed/$
fi

if $PLOT;then
    samtools view -bS -@ ${THREADS} ${SRA}.sam > ${SRA}.bam
    samtools sort -@ ${THREADS} ${SRA}.bam > ${SRA}.sorted.bam
    # Forward strand.
    #alignments of the second in pair if they map to the forward strand
    samtools view -b -f 128 -F 16 -@ ${THREADS} ${SRA}.sorted.bam > ${SRA}.fwd1.bam
    samtools index ${SRA}.fwd1.bam
    #alignments of the first in pair if they map to the reverse strand
    samtools view -b -f 80 -@ ${THREADS} ${SRA}.sorted.bam > ${SRA}.fwd2.bam
    samtools index ${SRA}.fwd2.bam
    #combine alignments that originate on the forward strand
    samtools merge -f ${SRA}.fwd.bam ${SRA}.fwd1.bam ${SRA}.fwd2.bam
    samtools index ${SRA}.fwd.bam

    # Reverse strand
    #alignments of the second in pair if they map to the reverse strand
    samtools view -b -f 144 -@ ${THREADS} ${SRA}.sorted.bam > ${SRA}.rev1.bam
    samtools index ${SRA}.rev1.bam
    #alignments of the first in pair if they map to the forward strand
    samtools view -b -f 64 -F 16 -@ ${THREADS} ${SRA}.sorted.bam > ${SRA}.rev2.bam
    samtools index ${SRA}.rev2.bam
    #combine alignments that originate on the reverse strand.
    samtools merge -f ${SRA}.rev.bam ${SRA}.rev1.bam ${SRA}.rev2.bam
    samtools index ${SRA}.rev.bam

    #Generate plotfiles
    samtools mpileup -aa ${SRA}.fwd.bam > ${SRA}.fwd.mpileup
    samtools mpileup -aa ${SRA}.rev.bam > ${SRA}.rev.mpileup
    cat ${SRA}.fwd.mpileup | cut -f4 > ${SRA}.fwd.plot
    cat ${SRA}.rev.mpileup | cut -f4 > ${SRA}.rev.plot
    paste ${SRA}.rev.plot ${SRA}.fwd.plot > ${SRA}.plot
```

```
fi

if $CLEAN; then
    rm *.bam *.mpileup *.bai
    if [ ! -d fastq ]; then
        mkdir fastq
    fi
    mv ${SRA}_*.fastq fastq/
fi



#To-do
#add install checks
#add opts for directory outputs
#write readme
#make logs/verbose?
```

removeProteinCodingRNA.R

```
#!/usr/bin/env Rscript
suppressMessages(library('getopt'))



spec = matrix(c(
  'sra', 'f', 1, "character",
  'help' , 'h', 0, "logical",
  'stranded' , 's', 0, "logical",
  'gff' , 'g', 1, "character",
  'file_path', 'p', 2, "character",
  'range', 'r', 2, "integer",
  'out_name', 'o', 2, "character"
), byrow=TRUE, ncol=4)


opt = getopt(spec)

if ( !is.null(opt$help) ) {
  cat("removeProteinCoding.R version 1.0\n")
  cat(" \n")
  cat("Use removeProteinCoding.R <options> -f <sra plot file> -g <gff file>\n")
  cat(" \n")
  cat("Options:\n")
  cat("  -f <sra plot file> The file that contains the plot data. Do not inclue the .plot file extensio
  cat("  -g <gff file> The file that contains the gff data. Do not inclue the gff file extension\n")
  cat("  -s <stranded data> The data is stranded\n")
  cat("  -p <file path> The location of the other files and the output file\n")
  cat("  -r <protein coding range> The number of nucleotides either side of a CDS region that should al
  cat("  -o <output file name> The name of the output file. Do not inclue the gff file extension. The d
  q(status=1)
}
```

```
if ( is.null(opt$sra) ) {
  cat("Error: -f <sra plot file> is required.\n")
  q(status=1)
}
if ( is.null(opt$gff) ) {
  cat("Error: -g <gff file> is required.\n")
q(status=1)
}
suppressMessages(library(tidyverse))
suppressMessages(library(tjnFunctions))

if ( is.null(opt$file_path ) ) { opt$file_path = "." }
if ( is.null(opt$range ) ) { opt$range = 50 }
if ( is.null(opt$out_name ) ) { opt$out_name = opt$sra }
if(is.null(opt$stranded)){
  stranded <- F
}else{
  stranded <- T
}
sraName <- opt$sra
gffName <- opt$gff
filePath <- opt$file_path


plotDat <- read.table(paste(filePath, "/", sraName, ".plot", sep = ""))
gffDat <- read.table(paste(filePath, "/", gffName, ".gff", sep = ""), sep = "\t", fill = T, comment.cha

colnames(gffDat) <- c("sequence", "source", "feature", "start", "end", "score", "strand", "phase", "Atr

plotDat <- removeCDSregions(plotDat = plotDat, gffDat = gffDat, stranded = stranded, time.it = T)



cat(paste("Writing the plot output to ", filePath, "/", opt$out_name, "_ncRNA.plot\n", sep = ""))
write.table(plotDat%>%select(V1,V2), file = paste(filePath, "/", opt$out_name, "_ncRNA.plot", sep = "")
```

run_rnaPeakCalling.R

```
#!/usr/bin/env Rscript
suppressMessages(library('getopt'))

spec = matrix(c(
  'sra', 'f', 1, "character",
  'help' , 'h', 0, "logical",
  'stranded' , 's', 0, "logical",
  'quiet' , 'q', 0, "logical",
  'gff' , 'g', 1, "character",
  'file_path', 'p', 2, "character",
  'range', 'r', 2, "integer",
  'out_name', 'o', 2, "character"
), byrow=TRUE, ncol=4)
```

```
opt = getopt(spec)

if ( !is.null(opt$help) ) {
  cat("run_rnaPeakCalling.R version 1.0\n")
  cat(" \n")
  cat("Use run_rnaPeakCalling.R <options> -f <sra plot file> -g <gff file>\n")
  cat(" \n")
  cat("Options:\n")
  cat("  -f <sra plot file> The file that contains the plot data. Do not inclue the .plot file extensio
  cat("  -g <gff file> The file that contains the gff data. Do not inclue the gff file extension\n")
  cat("  -s <stranded data> The data is stranded\n")
  cat("  -q <quiet> Do not print any updates\n")
  cat("  -p <file path> The location of the other files and the output file\n")
  cat("  -r <protein coding range> The number of nucleotides either side of a CDS region that should al
  cat("  -o <output file name> The name of the output file. Do not inclue the gff file extension. The d
  q(status=1)
}

if ( is.null(opt$sra) ) {
  cat("Error: -f <sra plot file> is required.\n")
  q(status=1)
}
if ( is.null(opt$gff) ) {
  cat("Error: -g <gff file> is required.\n")
  q(status=1)
}



suppressMessages(library(tidyverse))
suppressMessages(library(tjnFunctions))
###--- column 1 is reverse and column 2 is forward ---###

if ( is.null(opt$file_path ) ) { opt$file_path = "." }
if ( is.null(opt$range ) ) { opt$range = 50 }
if ( is.null(opt$out_name ) ) { opt$out_name = opt$sra }
if(is.null(opt$stranded)){
  stranded <- F
}else{
  stranded <- T
}

if(is.null(opt$quiet)){
  quiet <- F
}else{
  quiet <- T
}

sraName <- opt$sra
gffName <- opt$gff
filePath <- opt$file_path
```

```
ptm <- proc.time()


gffDat  <- tryCatch({
  suppressWarnings(gffDat <- read.table(paste(filePath, "/", gffName, ".gff", sep = ""), sep = "\t", fi
  gffDat
}, error =  function(e) {
  cat(paste("Error: ", opt$file_path, "/", opt$gff, ".gff not found.\n", sep = ""))
  q(status=1)
})


plotDat  <- tryCatch({
  suppressWarnings(plotDat <- read.table(paste(filePath, "/", sraName, ".plot", sep = "")))
  plotDat
}, error =  function(e) {
  cat(paste("Error: ", opt$file_path, "/", opt$sra, ".plot not found.\n", sep = ""))
  q(status=1)
})

total <- (sum(plotDat$V1) + sum(plotDat$V2))/1000000

colnames(gffDat) <- c("sequence", "source", "feature", "start", "end", "score", "strand", "phase", "Atr

plotDatncRNA  <- tryCatch({
  ##Change this path or put the header file in the working directory
  suppressWarnings(plotDatncRNA <- read.table(paste(filePath, "/", sraName, "_ncRNA.plot", sep = "")))

  plotDatncRNA
}, error =  function(e) {
  plotDat <- read.table(paste(filePath, "/", sraName, ".plot", sep = ""))
  if(quiet == F){
    cat("Running removeCDSregions\n")

  }
  plotDatncRNA <- removeCDSregions(plotDat = plotDat, gffDat = gffDat, stranded = stranded, time.it = T

  plotDatncRNA
} )

plotDatncRNA$V1 <- plotDatncRNA$V1/total
plotDatncRNA$V2 <- plotDatncRNA$V2/total



cat("Running rnPeakCalling\n")

cat("Calling forward\n")
callsDatFwd <- rnaPeakCalling(dat = plotDatncRNA, col.num = 2, small_peaks = F, plot_threshold = 15/tota

cat("Calling reverse\n")
callsDatRev <- rnaPeakCalling(dat = plotDatncRNA, col.num = 1, small_peaks = F, plot_threshold = 15/tota
```

```
callsDatFwd <- callsDatFwd%>%mutate(strand = "+")
callsDatRev <- callsDatRev%>%mutate(strand = "-")

runningTime <- proc.time() - ptm
  printRunningTime(runningTime = runningTime)

  if("feature.length" %in% colnames(callsDatFwd) == F){
    print(colnames(callsDatFwd))
    print(head(callsDatFwd))
    cat("Warning: feature.length column not found in callsDatFwd.\n")
    quitStatus <- T
    callsDatFwdTmp <- callsDatFwd%>%filter(start != 0)%>%
      mutate(feature.length = stop - start)%>%
      mutate(feature.score = feature.length*mean.score)%>%
      filter(feature.score > 3)
  }else{
  callsDatFwdTmp <- callsDatFwd%>%filter(start != 0)%>%
    mutate(feature.score = feature.length*mean.score)%>%
    filter(feature.score > 3)
  }
  if("feature.length" %in% colnames(callsDatRev) == F){
    print(colnames(callsDatRev))
    print(head(callsDatRev))
    cat("Warning: feature.length column not found in callsDatRevTmp.\n")
    quitStatus <- T
    callsDatRevTmp <- callsDatRev%>%filter(start != 0)%>%
      mutate(feature.length = stop - start)%>%
      mutate(feature.score = feature.length*mean.score)%>%
      filter(feature.score > 3)
  }else{
  callsDatRevTmp <- callsDatRev%>%filter(start != 0)%>%
    mutate(feature.score = feature.length*mean.score)%>%
    filter(feature.score > 3)
  }

  # if(quitStatus == T){
  #   q(status=1)
  # }

gffMain <- readLines(paste(filePath, "/", gffName, ".gff", sep = ""))
gffMain <- data.frame(text = gffMain)
genomeInfo <- as.character(gffMain[8,1])
genomeBuild <- as.character(gffMain[4,1])
genomeSpecies <- as.character(gffMain[9,1])
accession <- strsplit(genomeInfo, " ")[[1]][2]



gffFwd <- callsDatFwdTmp%>%mutate(strand = "+",
                                                    source = "sraAlignedncRNAExpression",
                                                    seqname = accession,
                                                    median.val = round(mean.score*100),
                                                    feature = "ncRNA",
```

```
                                                frame = ".",
                                                attribute = paste("ID=rna_fwd_", row_number(), sep = "
  select(seqname, source, feature, start, stop, median.val, strand, frame, attribute)

gffRev <- callsDatRevTmp%>%mutate(strand = "-",
                                                source = "sraAlignedncRNAExpression",
                                                seqname = accession,
                                                median.val = round(mean.score*100),
                                                feature = "ncRNA",
                                                frame = ".",
                                                attribute = paste("ID=rna_rev_", row_number(), sep = "
  select(seqname, source, feature, start, stop, median.val, strand, frame, attribute)%>%
  arrange(as.numeric(start))



gff <- gffFwd%>%bind_rows(gffRev)%>%arrange(as.numeric(start))
gff <- gff%>%filter(start != 0)




fileConn<-file(paste(filePath, "/", opt$out_name, "_sra_calls.gff", sep = ""))
writeLines(c("##gff-version 3",
             "#!gff-spec-version 1.21",
             "#!processor R script (local) with manual add of top section",
             genomeBuild,
             paste("#!genome-build-accession NCBI_Assembly:", opt$gff, sep = ""),
             paste("#!annotation-date ", Sys.Date(), sep = ""),
             "#!annotation-source sraPlotSummary.R (local version)",
             genomeInfo,
             genomeSpecies), fileConn)
close(fileConn)

cat(paste("Writing the gff output to ", filePath, "/", opt$out_name, "_sra_calls.gff\n", sep = ""))
write.table(x = gff, file = paste(filePath, "/", opt$out_name, "_sra_calls.gff", sep = ""), row.names =
```

rfamscan

```
rfamscan() { counts=$( bc -l <<< "scale=2;$(esl-seqstat $1.fna | grep ^"Total" | tr -s ' ' | cut -d ' '
```

cmscanToGFFWrapper.R

```
#!/usr/bin/env Rscript
library('getopt')


spec = matrix(c(
  'cmscanOutput', 'f', 1, "character",
  'gcf', 'g', 1, "character",
  'help' , 'h', 0, "logical",
```

```
  'file_path', 'p', 2, "character",
  'out_name', 'o', 2, "character"
), byrow=TRUE, ncol=4)


opt = getopt(spec)
#
# opt$cmscanOutput <- "GCA_000017745.1.tblout"
# opt$gcf <- "GCA_000017745.1"
# opt$file_path <- "~/phd/RNASeq/escherichia/"
# opt$output <- "escherichia_test"

if ( !is.null(opt$help) ) {
  cat("cmscanToGffWrapper.R version 1.0\n\n")
  cat("Use cmscanToGffWrapper.R <options> -f <cmscan ouptut file> -g <gff file>\n\n")
  cat("Options:\n")
  cat("  -f <cmscan ouptut file> The file that contains the cmscan output\n")
  cat("  -g <gff file> The file that contains the gff data. Do not inclue the gff file extension\n")
  cat("  -f <file path> The location of the other files and the output file\n")
  cat("  -o <output file name> The name of the output file. Do not inclue the gff file extension. The d
   q(status=1)
}

if ( is.null(opt$cmscanOutput) ) {
  cat("Error: -f <cmscan ouptut file> is required.\n")
  q(status=1)
}
if ( is.null(opt$gcf) ) {
  cat("Error: -g <gff file> is required.\n")
  q(status=1)
}

library(tidyverse)
library(tjnFunctions)

if ( is.null(opt$file_path ) ) { opt$file_path = "." }
if ( is.null(opt$output ) ) { opt$output = opt$gcf }

rfamRes <- read.table(paste(opt$file_path, opt$cmscanOutput, sep = "/"), header = F, comment.char = "#"


gff <- cmscanToGff(rfamRes = rfamRes)


gffMain <- readLines(paste(opt$file_path, "/", opt$gcf, ".gff", sep = ""))
gffMain <- data.frame(text = gffMain)
genomeInfo <- as.character(gffMain[8,1])
genomeBuild <- as.character(gffMain[4,1])
genomeSpecies <- as.character(gffMain[9,1])
accession <- strsplit(genomeInfo, " ")[[1]][2]

fileConn<-file(paste(opt$file_path, "/",opt$output, "_ncRNA.gff", sep = ""))
writeLines(c("##gff-version 3",
```

```
            "#!gff-spec-version 1.21",
            "#!processor R script (local)",
            genomeBuild,
            paste("#!genome-build-accession NCBI_Assembly:", opt$gcf, sep = ""),
            paste("#!annotation-date ", Sys.Date(), sep = ""),
            "#!annotation-source cmscan (rFam) (local version)",
            genomeInfo,
            genomeSpecies), fileConn)
close(fileConn)


write.table(x = gff, file = paste(opt$file_path, "/",opt$output, "_ncRNA.gff", sep = ""), row.names = F
```

combine_gff_files.R

```
#!/usr/bin/env Rscript
suppressMessages(library('getopt'))


# getopts ------------------------------------------------------------------


spec = matrix(c(
  'sra', 'f', 1, "character",
  'gff', 'g', 1, 'character',
  'help' , 'h', 0, "logical",
  'stranded' , 's', 0, "logical",
  'quiet' , 'q', 0, "logical",
  'file_path', 'p', 2, "character",
  'out_name', 'o', 2, "character",
  'random_data', 'r', 1, "character"
), byrow=TRUE, ncol=4)


opt = getopt(spec)

if ( !is.null(opt$help) ) {
  cat("combine_gff_files.R version 1.0\n")
  cat(" \n")
  cat("Use combine_gff_files.R <options> -f <files>\n")
  cat(" \n")
  cat("Options:\n")
  cat("  -f <files> The gff files\n")
  cat("  -s <stranded data> The data is stranded\n")
  cat("  -r <random data> The file to remove CDS regions from\n")
  cat("  -q <quiet> Do not print any updates\n")
  cat("  -p <file path> The location of the other files and the output file\n")
  cat("  -o <output file name> The name of the output file. Do not inclue the gff file extension. The d
  q(status=1)
}

if ( is.null(opt$sra) ) {
```

```
  cat("Error: -f <files> is required.\n")
  q(status=1)
}

if ( is.null(opt$out_name) ) {
  cat("Error: -o <output file name> is required.\n")
  q(status=1)
}


# packages ------------------------------------------------------------------


suppressMessages(library(tidyverse))
suppressMessages(library(comparativeSRA))

# defining variables --------------------------------------------------------


if ( is.null(opt$file_path ) ) { opt$file_path = "." }
if ( is.null(opt$out_name ) ) { opt$out_name = opt$sra }
if(is.null(opt$stranded)){
  stranded <- F
}else{
  stranded <- T
}

if(is.null(opt$quiet)){
  quiet <- F
}else{
  quiet <- T
}


#####

file_path <- opt$file_path
files <- list.files(paste(file_path, opt$sra, sep = "/"), pattern = ".gff$")
# import data ---------------------------------------------------------------


#print(files)
dat <- data.frame(sequence = as.character("0"), source = as.character("0"), feature = as.character("0")
                  start = as.integer("0"), end = as.integer("0"), score = as.character("0"),
                  strand = as.character("0"), phase = as.character("0"), Atrribute = as.character("0"),
i <- 2
for(i in 1:length(files)){
  tmp  <- tryCatch({
    suppressWarnings(tmp <- read.table(paste(file_path, opt$sra, files[i], sep = "/"), comment.char = "
  }, error =  function(e) {
    cat(paste("Error: ", "row ", i, ", ", file_path, "/", opt$sra, "/", files[i], " cannot be opened.\n
    cat(paste(e, "\n"))
  })
```

```
  if(class(tmp) == "NULL"){
    next
  }

  if(ncol(tmp) != 9){
    cat(paste("Error: ", "row ", i, ", ", file_path, "/", opt$sra, "/", files[i], " contains ", ncol(tm
    next
  }

  colnames(tmp) <- c("sequence", "source", "feature", "start", "end", "score", "strand", "phase", "Atrr

  tmp <- tmp%>%mutate(file_name = files[i])%>%mutate(score = as.character(score))

  if(files[i] == opt$random_data){
    tmp <- tmp%>%
  filter(feature != "CDS", feature != "gene", feature != "pseudogene", feature != "exon", feature != "r
  }else{

  dat <- dat%>%bind_rows(tmp)
}
}
if(!is.null(opt$random_data)){
   ncRNAgff <- dat%>%
     filter(feature != "gene", feature != "pseudogene", feature != "exon", feature != "region")
}else{
ncRNAgff <- dat%>%
  filter(feature != "CDS", feature != "gene", feature != "pseudogene", feature != "exon", feature != "r
}

# main section  ---------------------------------------------------------------------


ncRNAgff <- ncRNAgff%>%arrange(start) %>% filter((end - start) > 0)# %>% arrange(strand)


mergedDat <- data.frame(sequence = as.character("0"), feature = as.character("0"),
                        start = as.integer("0"), end = as.integer("0"),
                        strand = as.character("0"), file_names = as.character("start_row"),
                        row_numbers = as.character("0"), prop_overlap = as.numeric(0), new_feature = F,
                        number_of_rnaseq_files = as.integer("0"),
                        score = as.character("0"),
                        stringsAsFactors = F)

##loop through the combined gff files and combine features that overlap
i <- 3
current_feature <- F #is there a current feature being written?
new_feature <- T

for(i in 1:(nrow(ncRNAgff))){
  ##check if the feature is already known
  if(ncRNAgff$source[i] != "sraAlignedncRNAExpression"){
    new_feature <- F
  }
```

```
##if there is no current feature then set a new start value
if(current_feature == F){
start_val <- ncRNAgff$start[i]
start_i <- i
end_val <- ncRNAgff$end[i]
}



##set the new end value
if(ncRNAgff$end[i] > end_val){
end_val <- ncRNAgff$end[i]
}

if(i == nrow(ncRNAgff)){

  ##check if the subsequent feature was contained within the first feature
  if(ncRNAgff$end[start_i] < end_val){
    prop_val <- (ncRNAgff$end[start_i] - ncRNAgff$start[i])/(end_val - start_val)
  }else{
    prop_val <- 1
  }

  tmp <- data.frame(sequence = ncRNAgff$sequence[i],
                    feature = ncRNAgff$feature[i],
                    start = start_val, end = end_val,
                    strand = ncRNAgff$strand[i],
                    file_names = paste(ncRNAgff$file_name[start_i:i], collapse = ","),
                    row_numbers = paste(c(start_i:i), collapse = ","),
                    prop_overlap = prop_val,
                    new_feature = new_feature,
                    number_of_rnaseq_files = length(start_i:i),
                    score = as.character(ncRNAgff$score[i]),
                    stringsAsFactors = F)
  mergedDat <- mergedDat%>%bind_rows(tmp)
  current_feature <- F
  new_feature <- T
}else{


##check if the cuurent end value overlaps with the next starting value and update the end value if it
if(end_val > ncRNAgff$start[i + 1]){
  end_val <- ncRNAgff$end[i + 1]
  current_feature <- T
}else{

  ##check if the subsequent feature was contained within the first feature
  if(ncRNAgff$end[start_i] < end_val){
  prop_val <- (ncRNAgff$end[start_i] - ncRNAgff$start[i])/(end_val - start_val)
  }else{
    prop_val <- 1
  }
```

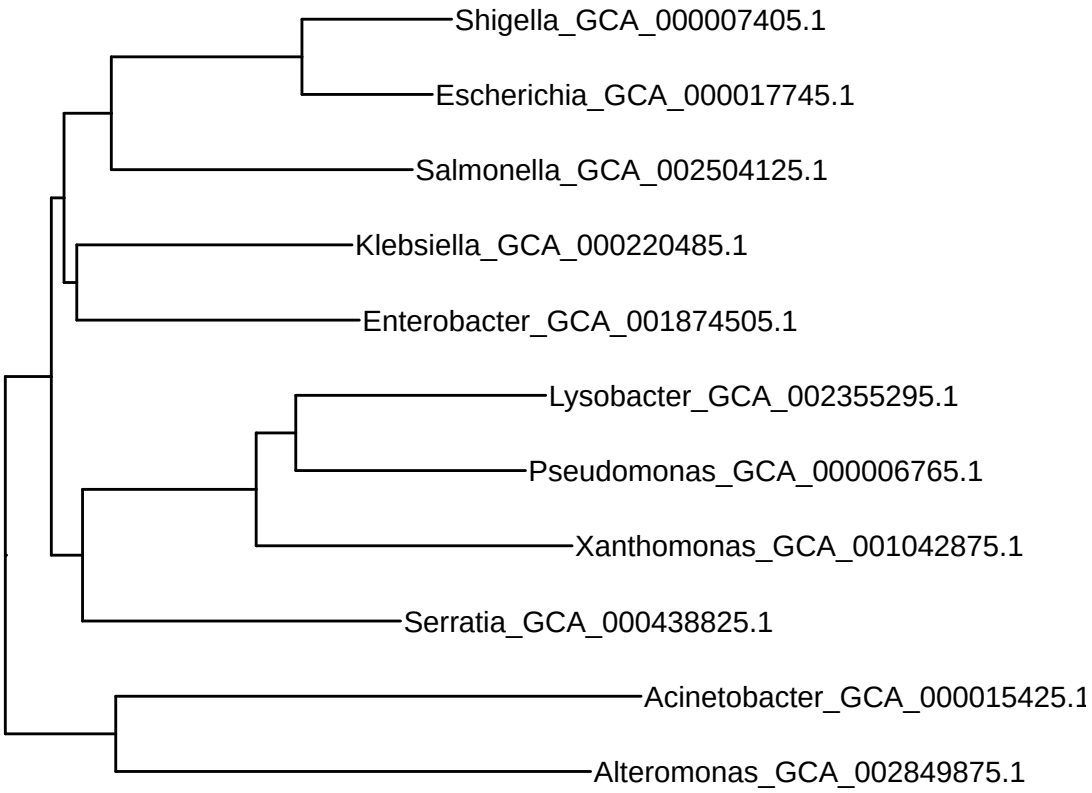
```
    tmp <- data.frame(sequence = ncRNAgff$sequence[i],
                      feature = ncRNAgff$feature[i],
                      start = start_val, end = end_val,
                      strand = ncRNAgff$strand[i],
                      file_names = paste(ncRNAgff$file_name[start_i:i], collapse = ","),
                      row_numbers = paste(c(start_i:i), collapse = ","),
                      prop_overlap = prop_val,
                      new_feature = new_feature,
                      number_of_rnaseq_files = length(start_i:i),
                      score = as.character(ncRNAgff$score[i]),
                      stringsAsFactors = F)
    mergedDat <- mergedDat%>%bind_rows(tmp)
    current_feature <- F
    new_feature <- T
  }
  }
}
```

```
mergedDat <- mergedDat%>%filter(number_of_rnaseq_files > 0, file_names != "start_row")

# if(!is.null(opt$random_data)){
#   mergedDat <- mergedDat %>% filter(file_names != opt$gff)
# }

mergedDat <- mergedDat %>% mutate(id =  paste(opt$out_name, row_number(), sep = "_"))

cat(paste("Writing the output to ", file_path, "/", opt$out_name, "_new_calls.txt\n", sep = ""))
write.table(x = mergedDat, file = paste(file_path, "/", opt$out_name, "_new_calls.txt", sep = ""), row.
```

Genome Alignments and Combining Files

- All genomes within a genus were aligned.
- One genome from each genus was used as an alignment against the other genomes

```
  tree <- read.tree("~/phd/RNASeq/alignments/all_alignments/genera_11.guide_tree")
  p <-ggtree(tree) +
  geom_tiplab() +
  xlim(0,0.8)
  p
```



Summary of Output Data

- From 21 strains and 11 genera, there were 292 RNASeq files.
  - *Escherichia* and *Shigella* are separated in the pyhlogenetic tree for this data
- This resulted in 53,485 expressed regions being predicted.
- There were 8335 known ncRNAs included in the analysis.

Combining GFF file

At this stage each individual RNASeq file has a corresponding gff file of SRA calls. There is also the original GFF file containing ncRNAs (along with CDS). Predictions of ncRNAs are made using rfam models and the output is made into a GFF file. There are 2 GFF files containing *known* ncRNAs and a number of GFF files containing predicted SRAs.

- feature files (.gff) files were all combined into a single *ACCESSION*_new_calls.txt file.
  - *combine_gff_files.r (done in gff_files folder)*
- After combining all the individual calls for each genome there were a total of 8906 putative sRNAs.
- For each sRNA that was predicted, a random intergenic region was selected.
  - *get_random_srna_sequences.py -a GCA_002208745.1*
  - the file containing new calls for a given genome was used.
  - this was done by randomly selecting a start site and taking the sequence from that location (for the same length as the orignial predicted sRNA).
  - coding regions were removed

- There were 15,072 random regions chosen

get_random_srna_sequences.py

```
for file in *.txt; do accession=`basename $file _new_calls.txt`; echo $accession; get_random_srna_sequer
```

```
#!/usr/bin/python

'''
file paths are hard coded
'''


import sys
from Bio import SeqIO
import getopt
import os
from BCBio import GFF
from Bio.Seq import Seq
from Bio.Alphabet import generic_dna
import random
import comparativeSRNA as srna


help = '''

'''

def usage():
    print help

def rungetopts():
    try:
        opts, args = getopt.getopt(sys.argv[1:], "a:sqh", ["accession", "shuffle", "quiet", "help"])
    except getopt.GetoptError as err:
        # print help information and exit:
        print(err) # will print something like "option -a not recognized"
        usage()
        sys.exit(2)
    accession = ""
    shuffled = False
    for o, a in opts:
            if o in ("-h", "--help"):
                usage()
                sys.exit()
            elif o in ("-a", "--accession"):
                accession = a
            elif o in ("-s", "--shuffle"):
                shuffled = True
            else:
                assert False, "unhandled option"
    if accession == "":
        print "-a <accession> missing. For more help use -h"
```

```
        sys.exit(2)
    return(accession, shuffled)


def main():

    accession, shuffled = rungetopts()
    print "Reading files"
    try:
        inFile = open("/Users/thomasnicholson/phd/RNASeq/new_calls/%s_new_calls.txt" % accession, 'r')

        fileLength = file_len("/Users/thomasnicholson/phd/RNASeq/new_calls/%s_new_calls.txt" % accessio
    except IOError:
        print "/Users/thomasnicholson/phd/RNASeq/new_calls/%s_new_calls.txt not found" % accession
        sys.exit(2)
    try:
        fastaFile = list(SeqIO.parse("/Users/thomasnicholson/phd/RNASeq/sequences/%s.fna" % accession,
    except IOError:
        print "/Users/thomasnicholson/phd/RNASeq/sequences/%s.fna not found" % accession
        sys.exit(2)

    print "Combining contigs"
    my_seq = srna.concatenateSequence(fastaFile)

    print "Getting intergenic sequence"
    random_seq = srna.intergenicSequence(accession, my_seq, shuffled)

    print "Getting intergenic positions"
    positions = srna.intergenicPositions(accession)

    print "Selecting random sRNAs"
    srna.selectRandomLocation(inFile, positions,fileLength, random_seq, accession)




if __name__ == "__main__":
    main()
```

For each genome there is now a single file containing all the SRA calls and whether they were previously found/predicted.

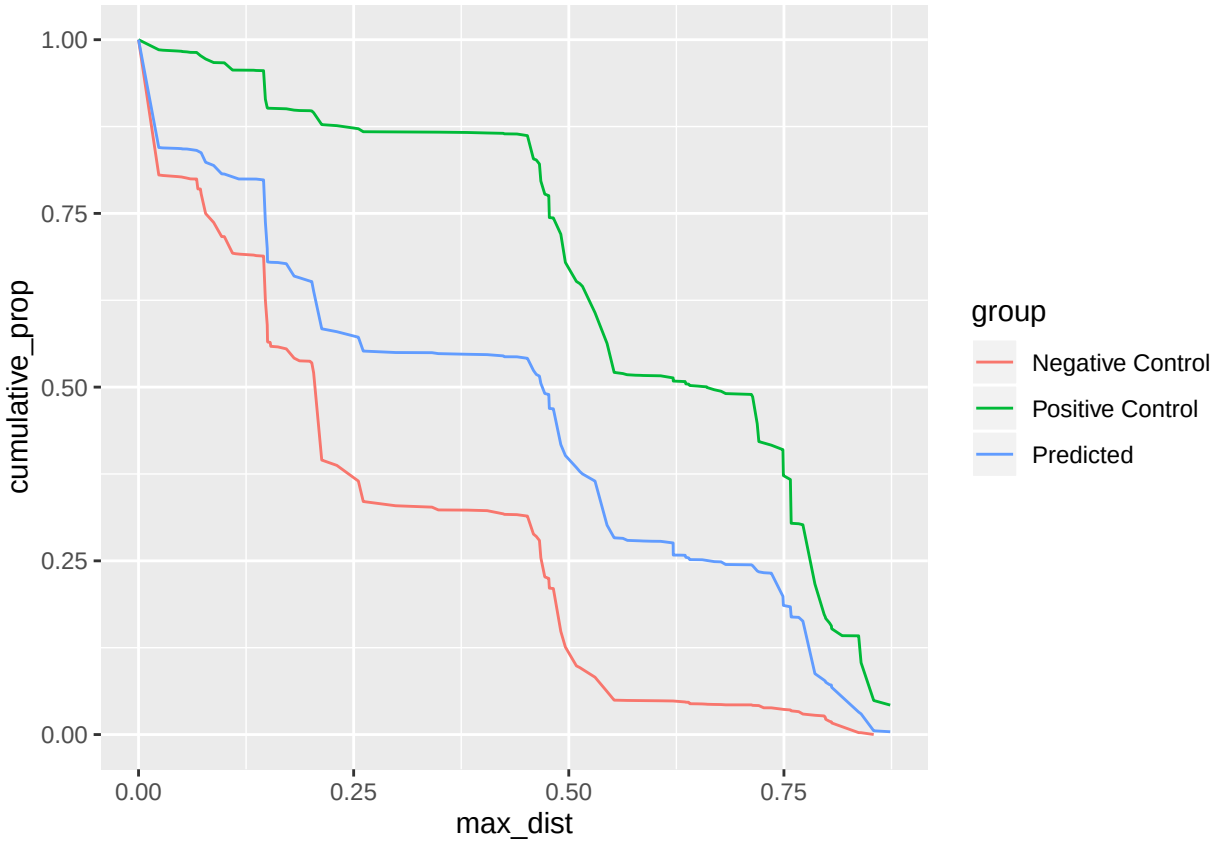
At this point genome alignments are done using MAUVE.

progressiveMauve –output=NAME.xmfa –output-guide-tree=NAME.tree –backbone-output=NAME.backbone GENOME_1 GENOME_2

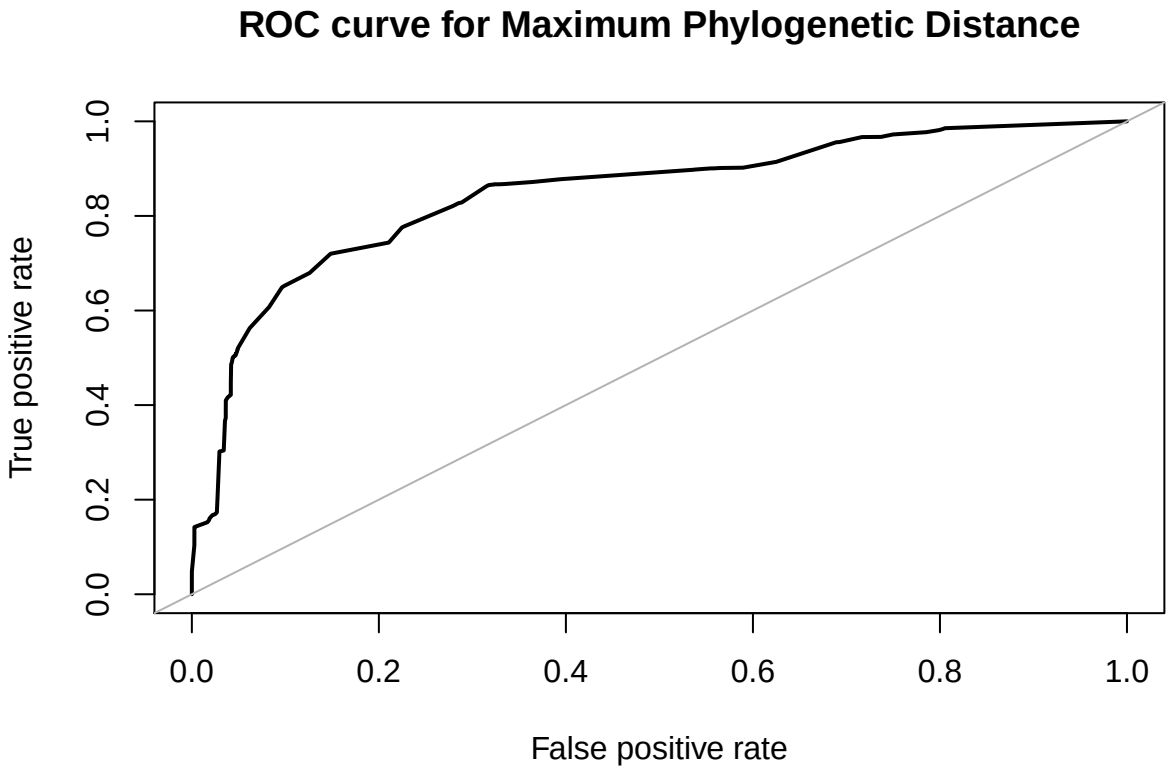
Results

Evolutionary Distance

```
load( file = "~/bin/r_git/R/maxDistsPC.Rda")
load(file = "~/bin/r_git/R/maxDistsPred.Rda")
load(file = "~/bin/r_git/R/maxDistsNC.Rda")
load(file = "~/bin/r_git/R/distsCumulativeCount.Rda")
dists <- distsPositive %>% bind_rows(distsPredicted, distsNegative)

  ggplot()+
  geom_line(data = distsCumulativeCount, aes(x = max_dist, y = cumulative_prop, group = group, colour =
```

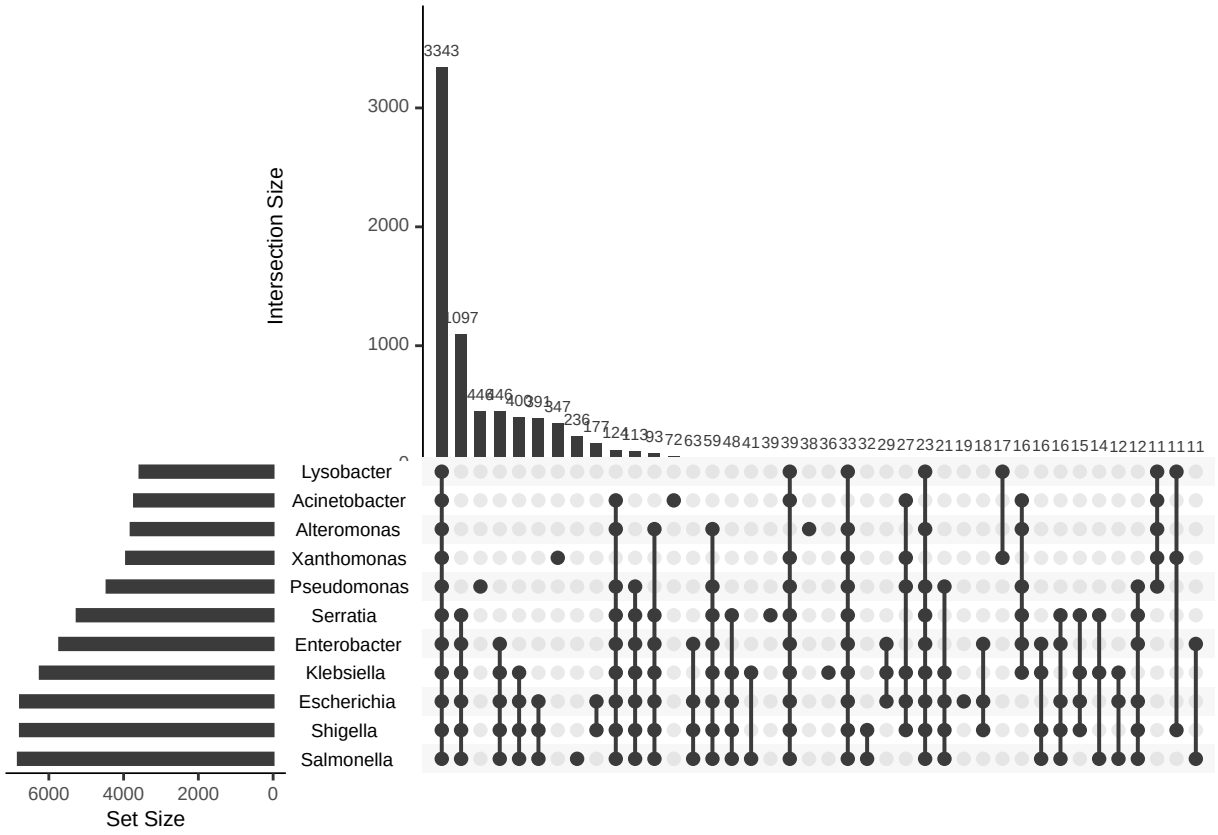



```
rocData <- dists %>% filter(group != "Predicted") %>% mutate(response = ifelse(group == "Positive Contr
roc.curve(response = rocData$response, predicted = rocData$max_dist,
          main="ROC curve for Maximum Phylogenetic Distance")
```




```
## Area under the curve (AUC): 0.847
```

```
load("~/bin/r_git/R/nhmmerGeneraUpsetR.Rda") #nhmmerGeneraUpsetR
UpSetR::upset(nhmmerGeneraUpsetR, sets = colnames(nhmmerGeneraUpsetR)[2:ncol(nhmmerGeneraUpsetR)], mb.ra
```




```
generaTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/genera_11_accession_only.guide_tree")
##check all data is there
nodes <- data.frame(generaTree$edge)
nodes$distances <- generaTree$edge.length
labels <- data.frame(names = generaTree$tip.label, X2 = c(1:length(generaTree$tip.label)))
treeDat <- nodes %>% full_join(labels)



pseudomonasTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/pseudomonas.guide_tree")
eschTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/escherichia.guide_tree")
shigTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/Shigella.tree")
salmTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/salmonella.guide_tree")
klebTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/Klebsiella.tree")
enterTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/Enterobacter.tree")
serrTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/serratia.guide_tree")
acinTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/acinetobacter.guide_tree")
xanthTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/xanthomonas.guide_tree")
alterTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/Altermonas.guide_tree")
# lysoTree <- read.tree("~/phd/RNASeq/alignments/all_alignments/escherichia.guide_tree")

generaMat <- cophenetic.phylo(x = generaTree)
pseudomonasMat <- cophenetic.phylo(x = pseudomonasTree)
eschMat <- cophenetic.phylo(x = eschTree)
shighMat <- cophenetic.phylo(x = shigTree)
salmMat <- cophenetic.phylo(x = salmTree)
klebMat <- cophenetic.phylo(x = klebTree)
```

```
enterMat <- cophenetic.phylo(x = enterTree)
serrMat <- cophenetic.phylo(x = serrTree)
acinMat <- cophenetic.phylo(x = acinTree)
xanthMat <- cophenetic.phylo(x = xanthTree)
alterMat <- cophenetic.phylo(x = alterTree)
lysoMat <- mat <- matrix(ncol = 1, nrow = 1)
rownames(lysoMat) <- "GCA_002355295.1"
colnames(lysoMat) <- "GCA_002355295.1"
lysoMat[1,1] <- 0

accession_info <- read.csv("~/phd/RNASeq/accession_info_all.csv", as.is = T)
#load("~/bin/r_git/R/r_files/accession_info.Rda")


mat <- matrix(ncol = nrow(accession_info), nrow = nrow(accession_info))

rownames(mat) <- accession_info$Accession
colnames(mat) <- accession_info$Accession


getPhyloDist <- function(mat, accession_info, dat, generaLookup) {
  for(i in 1:nrow(dat)){
  acc1 <- rownames(dat)[i]
  genus1 <- accession_info$Species[accession_info$Accession == acc1]
  accRef1 <- accession_info$Accession[accession_info$Species == genus1 & accession_info$Reference.Genome
  rowID <- match(acc1, rownames(mat))
  for(j in 1:ncol(mat)){
    acc2 <- colnames(mat)[j]
    genus2 <- accession_info$Species[accession_info$Accession == acc2]
    accRef2 <- accession_info$Accession[accession_info$Species == genus2 & accession_info$Reference.Gen
    colID <- j
    if(genus1 == genus2){
      lookupI <- match(acc1, rownames(dat))
      lookupJ <- match(acc2, colnames(dat))
      mat[rowID, colID] <- dat[lookupI, lookupJ]
      }else{
      lookupI <- match(accRef1, rownames(generaLookup))
      lookupJ <- match(accRef2, colnames(generaLookup))
      mat[rowID, colID] <- generaLookup[lookupI, lookupJ]
    }
  }
  }


  return(mat)
}


mat <- getPhyloDist(mat = mat, accession_info = accession_info, dat = eschMat, generaLookup = generaMat)
mat <- getPhyloDist(mat = mat, accession_info = accession_info, dat = shighMat, generaLookup = generaMa
mat <- getPhyloDist(mat = mat, accession_info = accession_info, dat = salmMat, generaLookup = generaMat)
mat <- getPhyloDist(mat = mat, accession_info = accession_info, dat = klebMat, generaLookup = generaMat)
mat <- getPhyloDist(mat = mat, accession_info = accession_info, dat = enterMat, generaLookup = generaMa
```

```
mat <- getPhyloDist(mat = mat, accession_info = accession_info, dat = serrMat, generaLookup = generaMat)
mat <- getPhyloDist(mat = mat, accession_info = accession_info, dat = acinMat, generaLookup = generaMat)
mat <- getPhyloDist(mat = mat, accession_info = accession_info, dat = xanthMat, generaLookup = generaMat
mat <- getPhyloDist(mat = mat, accession_info = accession_info, dat = alterMat, generaLookup = generaMat
mat <- getPhyloDist(mat = mat, accession_info = accession_info, dat = pseudomonasMat, generaLookup = gen
mat <- getPhyloDist(mat = mat, accession_info = accession_info, dat = lysoMat, generaLookup = generaMat)

phyloDistMat <- mat
save(phyloDistMat, file = "~/bin/r_git/R/phyloDistMatrix.Rda")

nhmmerDataframeSetup <- function(dat, contigLookup = "") {

  dat <- dat[,c(1:16)]
  colnames(dat) <-  c("target.name", "accession", "query.name", "accession.2", "hmmfrom", "hmmto", "ali
  dat <- dat %>% filter(accession == "-")
  dat <- dat %>%
    separate(col = target.name, into = c("t1", "t2", "t3"), sep = "_", remove = F, extra = "merge") %>%
    mutate(target.genome = paste(t1, t2, sep = "_")) %>%
    select(-t1, -t2, -t3)%>%
    separate(col = query.name, into = c("t1", "t2", "t3"), sep = "_", remove = F, extra = "merge") %>%
    mutate(query.genome = paste(t1, t2, sep = "_")) %>%
    select(-t1, -t2, -t3)
  dat <- dat %>% left_join(contigLookup, by = "target.genome")
  dat <- dat %>% mutate(target.genome = ifelse(!is.na(target.genome.accession), target.genome.accession
  return(dat)
  }
genomeCombinations <- function(dat, phyloDistMat){
  dat <- dat %>% mutate(match.id = paste(target.genome, query.genome, sep = ", "))
  datUnique <- dat %>% select(target.genome, query.genome, match.id) %>% unique() %>% mutate(distance =
  for (i in 1:nrow(datUnique)) {
    acc1 <- datUnique[i,1]
    acc2 <- datUnique[i,2]
    rowID <- match(acc1, table = rownames(phyloDistMat))
    colID <- match(acc2, table = colnames(phyloDistMat))
    datUnique$distance[i] <- phyloDistMat[rowID ,colID]

  }
  datUnique <- datUnique %>% select(match.id, distance)
 dat <- dat %>% left_join(datUnique, by = "match.id")
 return(dat)
}



datPositive <- read.table("~/phd/RNASeq/srna_seqs/version_1/positive_control.tbl", comment.char = "#", 
datPredicted <- read.table("~/phd/RNASeq/srna_seqs/version_1/predicted_2.tbl", comment.char = "#", fill
datNegative <- read.table("~/phd/RNASeq/srna_seqs/version_1/negative_control.tbl", comment.char = "#", 
contigLookup <- read.table("~/phd/RNASeq/sequences/contig_ids_accession.lookup", sep = "\t", comment.cha
colnames(contigLookup) <- c("target.genome", "target.genome.accession")
load(file = "~/bin/r_git/R/phyloDistMatrix.Rda")


# write.table(x = datNegative, file = "~/phd/RNASeq/srna_seqs/version_1/negative_control_no_shuffle_COR
```

```
datPositive <- nhmmerDataframeSetup(dat = datPositive, contigLookup = contigLookup)
datPredicted <- nhmmerDataframeSetup(datPredicted, contigLookup = contigLookup)
datNegative <- nhmmerDataframeSetup(datNegative, contigLookup = contigLookup)

datPositive <- genomeCombinations(dat = datPositive, phyloDistMat = phyloDistMat)
datPredicted <- genomeCombinations(dat = datPredicted, phyloDistMat = phyloDistMat)
datNegative <- genomeCombinations(dat = datNegative, phyloDistMat = phyloDistMat)
datNegative2 <- datNegative %>% filter(E.value < 1e-5)
datPredicted2 <- datPredicted %>% filter(E.value < 1e-5)
datPositive2 <- datPositive %>% filter(E.value < 1e-5)

max_val <- max(c(max(datPositive2$distance, na.rm = T), max(datNegative2$distance, na.rm = T)))
min_val <- min(c(min(datPositive2$distance, na.rm = T), min(datNegative2$distance, na.rm = T)))


distsPositive <- datPositive2 %>% filter(!is.na(distance)) %>% group_by(query.name) %>% summarise(max_di
distsPredicted <- datPredicted2 %>% filter(!is.na(distance)) %>% group_by(query.name) %>% summarise(max_
distsNegative <- datNegative2 %>% filter(!is.na(distance)) %>% group_by(query.name) %>% summarise(max_di

distsPositive <- distsPositive %>% mutate(group = "Positive Control")
distsPredicted <- distsPredicted %>% mutate(group = "Predicted")
distsNegative <- distsNegative %>% mutate(group = "Negative Control")

save(distsPositive, file = "maxDistsPC.Rda")
save(distsPredicted, file = "maxDistsPred.Rda")
save(distsNegative, file = "maxDistsNC.Rda")


cumulativeCounts <- function(dists, smooth = T){

  groups <- unique(dists$group)
  for(i in groups){
    dat <- dists %>% filter(group == i)
    dat <- dat %>% mutate(count = 1) %>%
    arrange(-max_dist) %>% group_by(group) %>%
    mutate(cumulativeCount = cumsum(count)) %>% ungroup() %>%
    group_by(group, max_dist) %>% summarise(cumulative_prop = max(cumulativeCount)/ nrow(dat))

    if(smooth){
      dat <- as.data.frame(spline(x = dat$max_dist,y =  dat$cumulative_prop))
    }
    dat <- dat %>% ungroup() %>% mutate(group = i)
    if(exists('combinedDat')){
      combinedDat <- combinedDat %>% bind_rows(dat)
    }else{
      combinedDat <- dat
    }
  }
  return(combinedDat)

}
```

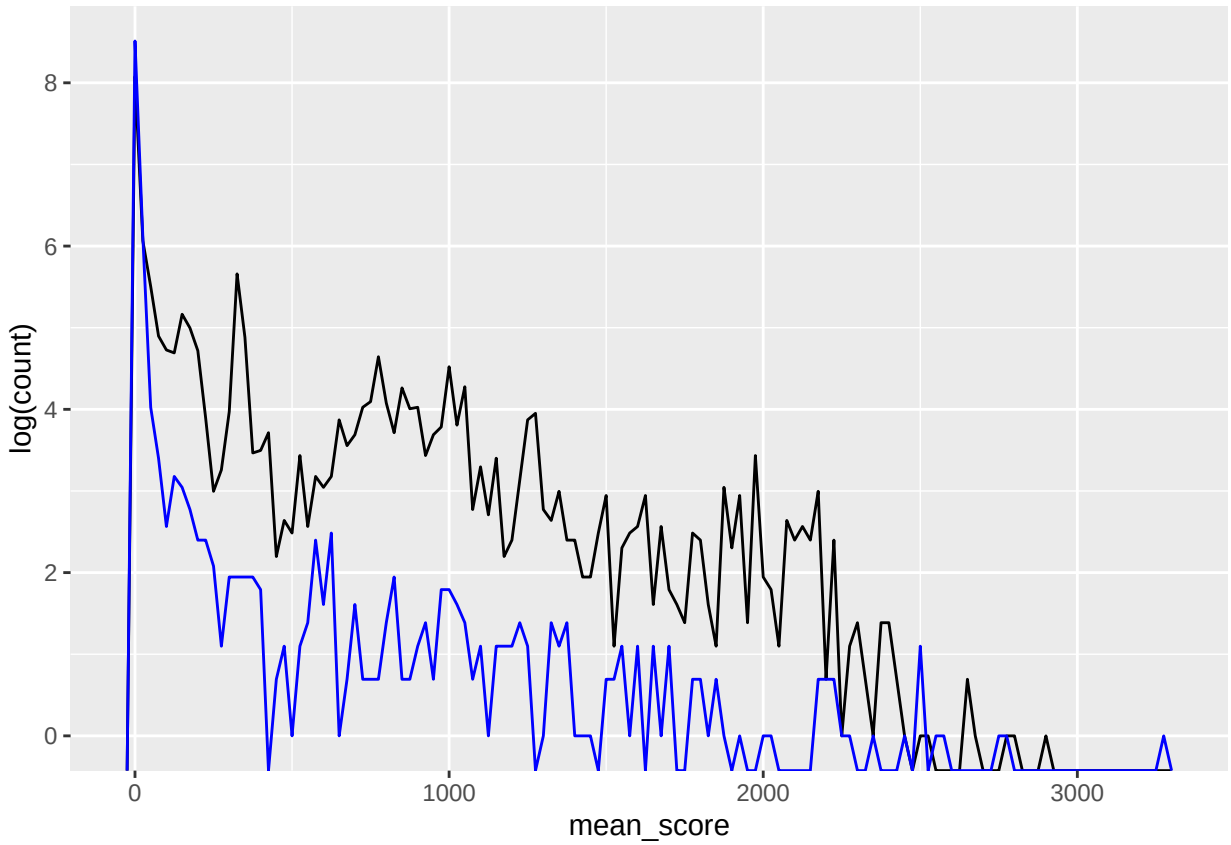
```
dists <- distsPositive %>% bind_rows(distsPredicted, distsNegative)


distsCumulativeCount <- cumulativeCounts(dists = dists, smooth = F)

save(distsCumulativeCount, file = "distsCumulativeCount.Rda")
```

```
nhmmerDataframeSetup <- function(dat, contigLookup = "") {

  dat <- dat[,c(1:16)]
  colnames(dat) <-  c("target.name", "accession", "query.name", "accession.2", "hmmfrom", "hmmto", "ali
  dat <- dat %>% filter(accession == "-")
  dat <- dat %>%
    separate(col = target.name, into = c("t1", "t2", "t3"), sep = "_", remove = F, extra = "merge") %>%
    mutate(target.genome = paste(t1, t2, sep = "_")) %>%
    select(-t1, -t2, -t3)%>%
    separate(col = query.name, into = c("t1", "t2", "t3"), sep = "_", remove = F, extra = "merge") %>%
    mutate(query.genome = paste(t1, t2, sep = "_")) %>%
    select(-t1, -t2, -t3)
  dat <- dat %>% left_join(contigLookup, by = "target.genome")
  dat <- dat %>% mutate(target.genome = ifelse(!is.na(target.genome.accession), target.genome.accession
  return(dat)
  }
genomeCombinations <- function(dat, phyloDistMat){
  dat <- dat %>% mutate(match.id = paste(target.genome, query.genome, sep = ", "))
  datUnique <- dat %>% select(target.genome, query.genome, match.id) %>% unique() %>% mutate(distance =
  for (i in 1:nrow(datUnique)) {
    acc1 <- datUnique[i,1]
    acc2 <- datUnique[i,2]
    rowID <- match(acc1, table = rownames(phyloDistMat))
    colID <- match(acc2, table = colnames(phyloDistMat))
    datUnique$distance[i] <- phyloDistMat[rowID ,colID]

  }
  datUnique <- datUnique %>% select(match.id, distance)
 dat <- dat %>% left_join(datUnique, by = "match.id")
 return(dat)
}



datPositive <- read.table("~/phd/RNASeq/srna_seqs/version_1/positive_control.tbl", comment.char = "#", 
contigLookup <- read.table("~/phd/RNASeq/sequences/contig_ids_accession.lookup", sep = "\t", comment.cha
colnames(contigLookup) <- c("target.genome", "target.genome.accession")
load(file = "~/bin/r_git/R/phyloDistMatrix.Rda")


# write.table(x = datNegative, file = "~/phd/RNASeq/srna_seqs/version_1/negative_control_no_shuffle_COR

datPositive <- nhmmerDataframeSetup(dat = datPositive, contigLookup = contigLookup)
datPositive <- genomeCombinations(dat = datPositive, phyloDistMat = phyloDistMat)
datPositive2 <- datPositive %>% filter(E.value < 1e-5)
```

```
load("~/bin/r_git/R/r_files/accession_info.Rda")

accession_info <- accession_info %>% select(Accession, Species) %>% dplyr::rename(target.genome = Acces

newRows <- data.frame(target.genome = c("GCA_000007385.1", "GCA_002355295.1", "GCA_000196795.1", "GCA_0

accession_info <- accession_info %>% bind_rows(newRows)

datPositive2 <- datPositive2 %>% left_join(accession_info, by = "target.genome")

accession_info <- accession_info %>% dplyr::rename(query.genome = target.genome, query.species = target
datPositive2 <- datPositive2 %>% left_join(accession_info, by = "query.genome")




mat <- matrix(nrow = length(unique(datPositive2$query.name)), ncol = length(unique(datPositive2$target.s
upsetDat <- as.data.frame(mat)
upsetDat[,1] <- as.character(unique(datPositive2$query.name))
colnames(upsetDat) <- c("name", as.character(unique(datPositive2$target.species)))


i <- 1
for (i in 1:nrow(upsetDat)) {
  id <- upsetDat$name[i]
  targetSpecies <- unique(datPositive2$target.species[datPositive2$query.name == id])
  colNums <- match(x = targetSpecies, table = colnames(upsetDat))
  upsetDat[i,colNums] <- 1
}

upsetDat[is.na(upsetDat)] <-  0

nhmmerGeneraUpsetR <- upsetDat
save(nhmmerGeneraUpsetR, file = "nhmmerGeneraUpsetR.Rda")
UpSetR::upset(upsetDat, sets = colnames(upsetDat)[2:ncol(upsetDat)], mb.ratio = c(0.55, 0.45), order.by
```

Covariation

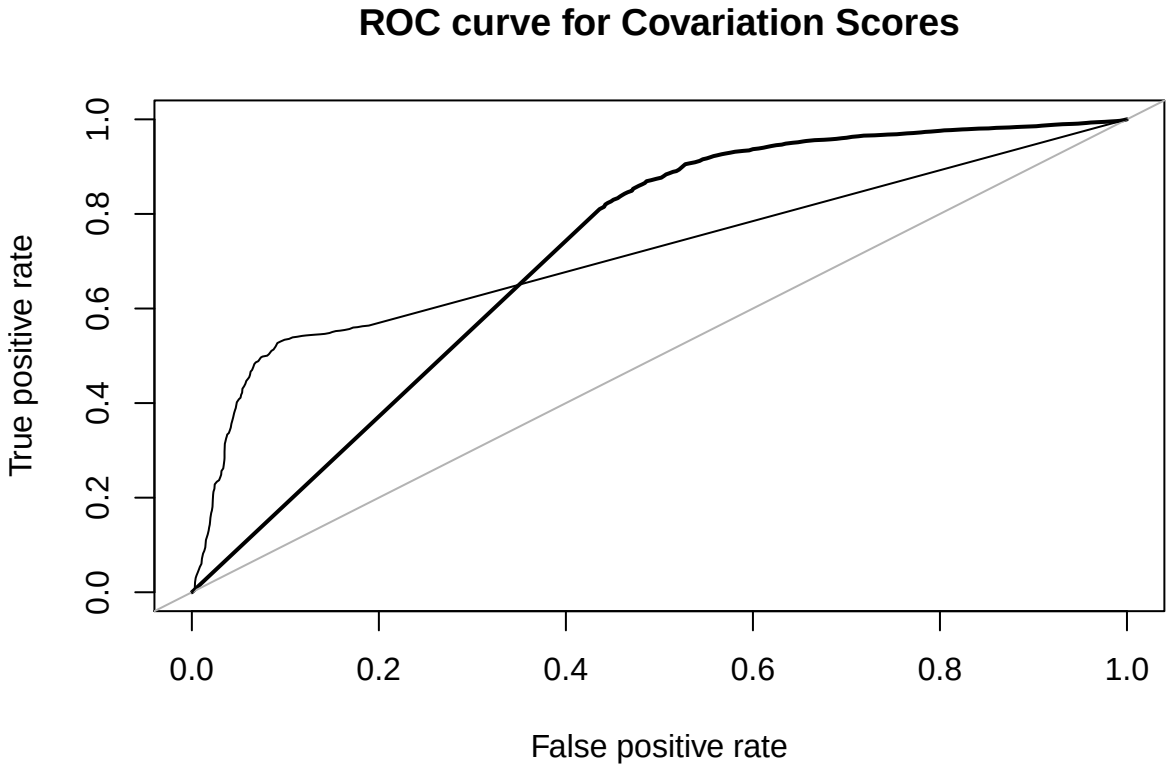
```
load("~/bin/r_git/R/pcCovariation.Rda")
load("~/bin/r_git/R/ncCovariation.Rda")
load("~/bin/r_git/R/predCovariation.Rda")

ggplot() +
  geom_freqpoly(data = pcCov, aes(x = mean_score, y = log(..count..)), binwidth = 25) +
  geom_freqpoly(data = ncCov, aes(x = mean_score, y = log(..count..)), binwidth = 25, colour = "blue")
```




```
pcCov <- pcCov %>% mutate(response = 1)
ncCov <- ncCov %>% mutate(response = 0)
rocData <- pcCov %>% bind_rows(ncCov)
roc.curve(response = rocData$response, predicted = rocData$min_eval,
          main="ROC curve for Covariation Scores")
```

```
## Area under the curve (AUC): 0.711
```

```
roc.curve(response = rocData$response, predicted = rocData$mean_score,
          main="ROC curve for Covariation Scores", add.roc = T)
```



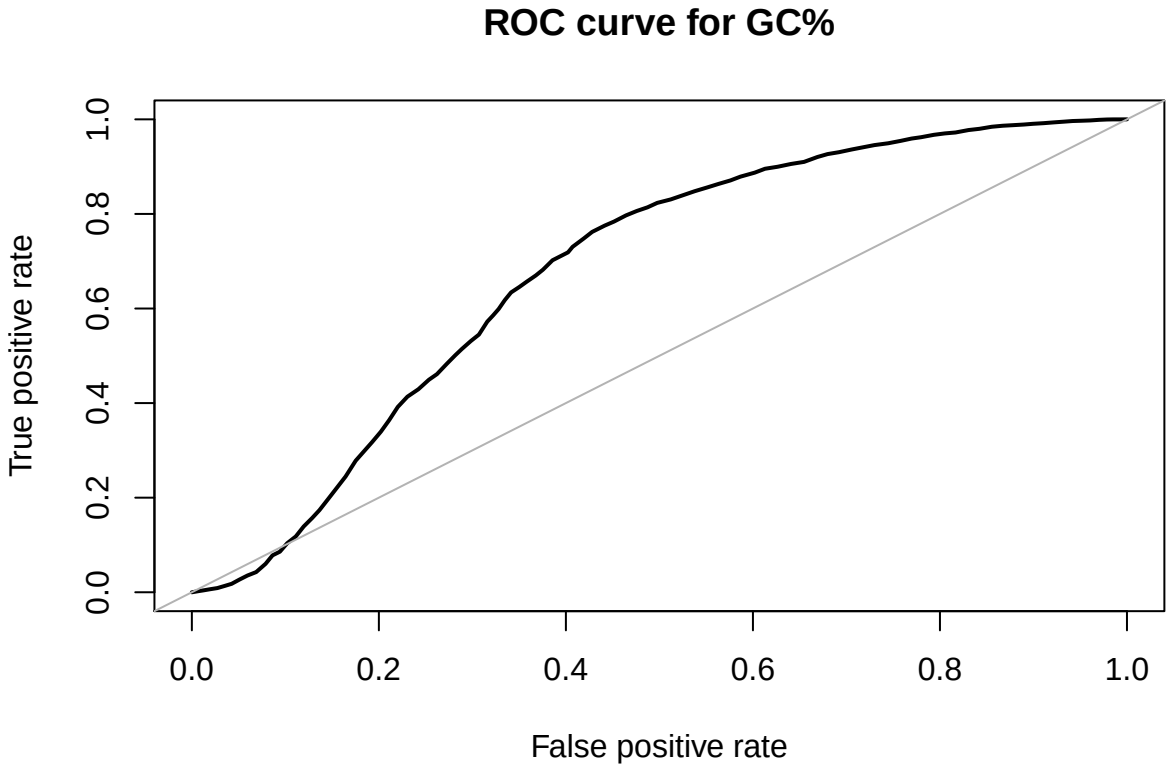

```
## Area under the curve (AUC): 0.718
```

```
pcCov <- read.table("~/phd/RNASeq/srna_seqs/version_1/positive_control/positive_control.rscape.cov", sep
ncCov <- read.table("~/phd/RNASeq/srna_seqs/version_1/negative_control/negative_control.rscape.cov", sep

predCov <- read.table("~/phd/RNASeq/srna_seqs/version_1/predicted/predicted.rscape.cov", sep = "\t", co

#colnames(pcCov) <- c("V1", "left_pos", "right_pos", "score", "e.value", "substitutions", "power")
#colnames(ncCov) <- c("V1", "left_pos", "right_pos", "score", "e.value", "substitutions", "power")

pcCov <- pcCov %>% mutate(ID = ifelse(V1 == "no significant pairs", left_pos, ID))


pcCov$score[pcCov$V1 == "no significant pairs"] <- 0
pcCov$e.value[pcCov$V1 == "no significant pairs"] <- 10
pcCov$power[pcCov$V1 == "no significant pairs"] <- 0
pcCov$substitutions[pcCov$V1 == "no significant pairs"] <- 0

pcCov$left_pos[pcCov$V1 == "no significant pairs"] <- "-"
pcCov$right_pos[pcCov$V1 == "no significant pairs"] <- "-"
pcCov$V1[pcCov$V1 == "no significant pairs"] <- "-"

ncCov <- ncCov %>% mutate(ID = ifelse(V1 == "no significant pairs", left_pos, ID))

ncCov$score[ncCov$V1 == "no significant pairs"] <- 0
ncCov$e.value[ncCov$V1 == "no significant pairs"] <- 10
ncCov$power[ncCov$V1 == "no significant pairs"] <- 0
ncCov$substitutions[ncCov$V1 == "no significant pairs"] <- 0

ncCov$left_pos[ncCov$V1 == "no significant pairs"] <- "-"
```

```
ncCov$right_pos[ncCov$V1 == "no significant pairs"] <- "-"
ncCov$V1[ncCov$V1 == "no significant pairs"] <- "-"

predCov <- predCov %>% mutate(ID = ifelse(V1 == "no significant pairs", left_pos, ID))


predCov$score[predCov$V1 == "no significant pairs"] <- 0
predCov$e.value[predCov$V1 == "no significant pairs"] <- 10
predCov$power[predCov$V1 == "no significant pairs"] <- 0
predCov$substitutions[predCov$V1 == "no significant pairs"] <- 0

predCov$left_pos[predCov$V1 == "no significant pairs"] <- "-"
predCov$right_pos[predCov$V1 == "no significant pairs"] <- "-"
predCov$V1[predCov$V1 == "no significant pairs"] <- "-"







pcCovMean <- pcCov %>% group_by(ID) %>% summarise(mean_score = mean(score))
pcCovMax <- pcCov %>% group_by(ID) %>% summarise(min_eval = min(e.value))
pcCov <- pcCovMean %>% full_join(pcCovMax, by = "ID")

ncCovMean <- ncCov %>% group_by(ID) %>% summarise(mean_score = mean(score))
ncCovMax <- ncCov %>% group_by(ID) %>% summarise(min_eval = min(e.value))
ncCov <- ncCovMean %>% full_join(ncCovMax, by = "ID")

predCovMean <- predCov %>% group_by(ID) %>% summarise(mean_score = mean(score))
predCovMax <- predCov %>% group_by(ID) %>% summarise(min_eval = min(e.value))
predCov <- predCovMean %>% full_join(predCovMax, by = "ID")

save(pcCov, file = "pcCovariation.Rda")
save(ncCov, file = "ncCovariation.Rda")
save(predCov, file = "predCovariation.Rda")
```

GC Content

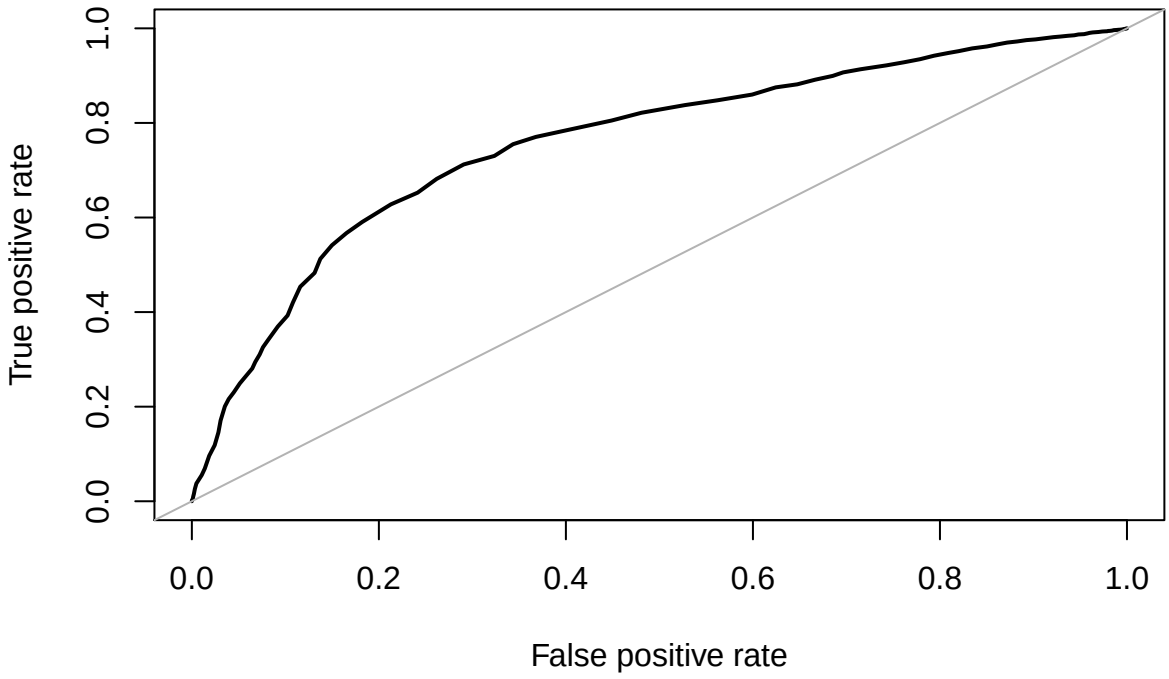
```
pcGC <- read.table("~/phd/RNASeq/srna_seqs/version_1/positive_control.gc", sep = "\t", comment.char = "#
ncGC <- read.table("~/phd/RNASeq/srna_seqs/version_1/negative_control_no_shuffle.gc", sep = "\t", comme
predGC <- read.table("~/phd/RNASeq/srna_seqs/version_1/predicted.gc", sep = "\t", comment.char = "#", as

pcGC <- pcGC %>% mutate(response = 1)
ncGC <- ncGC %>% mutate(response = 0)

rocData <- pcGC %>% bind_rows(ncGC)

roc.curve(response = rocData$response, predicted = rocData$V2,
          main="ROC curve for GC%")
```




```
## Area under the curve (AUC): 0.680
```

Secondary Structure

```
load("~/bin/r_git/R/pcAlifold.Rda")
load("~/bin/r_git/R/ncAlifold.Rda")
pcAlifold <- pcAlifold %>% mutate(response = 1)
ncAlifold <- ncAlifold %>% mutate(response = 0)

rocData <- pcAlifold %>% bind_rows(ncAlifold)
rocData$z_mean[is.na(rocData$z_mean)] <- 10
rocData$z_max[is.na(rocData$z_max)] <- 10

roc.curve(response = rocData$response, predicted = rocData$z_mean,
          main="ROC curve for MFE")
```



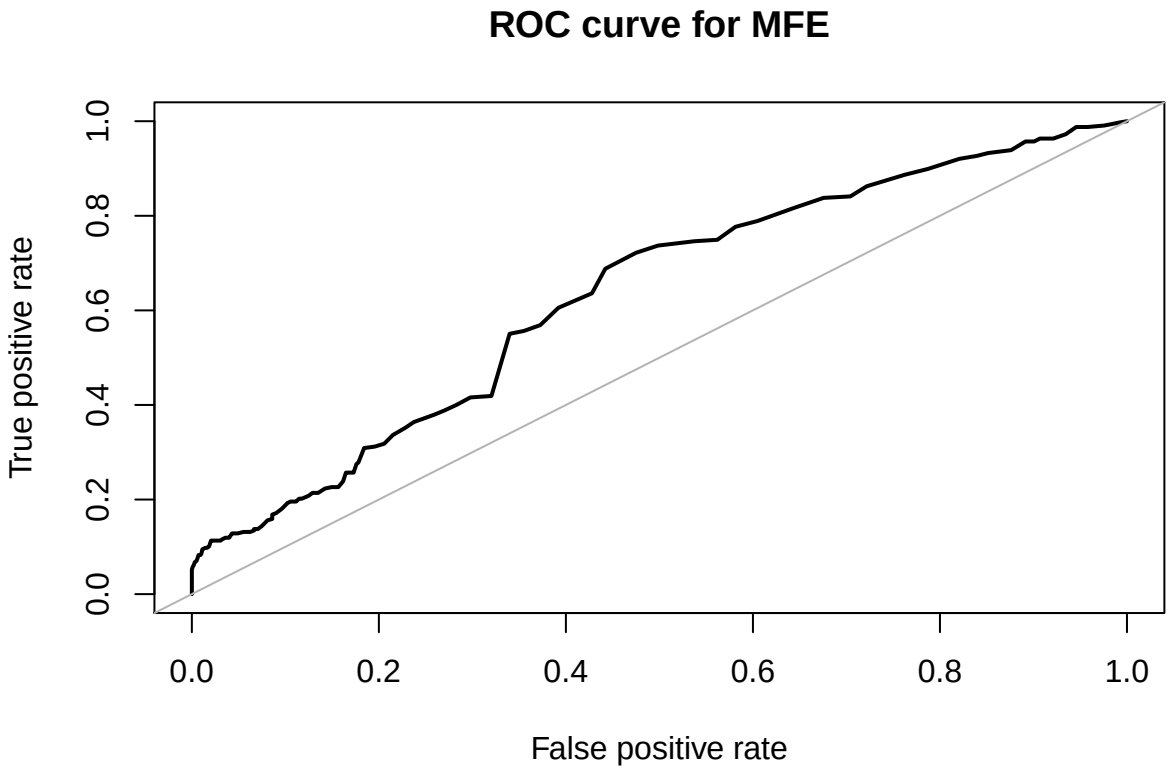

```
## Area under the curve (AUC): 0.758
```

```
pcAlifold<- read.table("~/phd/RNASeq/srna_seqs/version_1/positive_control/positive_control.alifold", hea
ncAlifold<- read.table("~/phd/RNASeq/srna_seqs/version_1/negative_control/negative_control.alifold", hea

colnames(pcAlifold) <- c( "From",      "To",    "Strand",    "Native.MFE",    "Mean.MFE",     "STDV",
colnames(ncAlifold) <- c( "From",      "To",    "Strand",    "Native.MFE",    "Mean.MFE",     "STDV",

ncAlifold <- ncAlifold %>% filter(grepl(pattern = "GCA_", ID))
pcAlifold <- pcAlifold %>% filter(grepl(pattern = "GCA_", ID))

pcAlifoldMean <- pcAlifold %>% group_by(ID) %>% summarise(z_mean = mean(as.numeric(Z), na.rm = T))
pcAlifoldMax <- pcAlifold %>% group_by(ID) %>% summarise(z_max = max(as.numeric(Z), na.rm = T))

ncAlifoldMean <- ncAlifold %>% group_by(ID) %>% summarise(z_mean = mean(as.numeric(Z), na.rm = T))
ncAlifoldMax <- ncAlifold %>% group_by(ID) %>% summarise(z_max = max(as.numeric(Z), na.rm = T))

pcAlifold <- pcAlifoldMean %>% full_join(pcAlifoldMax, by = "ID")
ncAlifold <- ncAlifoldMean %>% full_join(ncAlifoldMax, by = "ID")


save(pcAlifold, file = "~/bin/r_git/R/pcAlifold.Rda")
save(ncAlifold, file = "~/bin/r_git/R/ncAlifold.Rda")
```

ncRNA motifs

```
load("~/bin/r_git/R/pcMotif.Rda")
load("~/bin/r_git/R/ncMotif.Rda")
```

```
load("~/bin/r_git/R/predMotif.Rda")

ncMotif <- ncMotif %>% mutate(response = 0)
pcMotif <- pcMotif %>% mutate(response = 1)

rocData <- pcMotif %>% bind_rows(ncMotif)

roc.curve(response = rocData$response, predicted = rocData$max_score,
          main="ROC curve for MFE")
```




```
## Area under the curve (AUC): 0.631
```

```
pcMotif <- read.table("~/phd/RNASeq/srna_seqs/version_1/positive_control/positive_control.rmfam", sep =
ncMotif <- read.table("~/phd/RNASeq/srna_seqs/version_1/negative_control/negative_control.rmfam", sep =

predMotif <- read.table("~/phd/RNASeq/srna_seqs/version_1/predicted/predicted.rmfam", sep = "", comment

colnames(pcMotif) <- c("seqname", "source", "feature", "start", "end", "score", "strand", "frame", "atti
colnames(ncMotif) <- c("seqname", "source", "feature", "start", "end", "score", "strand", "frame", "atti
colnames(predMotif) <- c("seqname", "source", "feature", "start", "end", "score", "strand", "frame", "a

pcMotifMean <- pcMotif %>% group_by(ID) %>% summarise(mean_score = mean(score))
pcMotifMax <- pcMotif %>% group_by(ID) %>% summarise(max_score = max(score))

pcMotif <- pcMotifMean %>% full_join(pcMotifMax, by = "ID")


ncMotifMean <- ncMotif %>% group_by(ID) %>% summarise(mean_score = mean(score))
ncMotifMax <- ncMotif %>% group_by(ID) %>% summarise(max_score = max(score))
ncMotif <- ncMotifMean %>% full_join(ncMotifMax, by = "ID")
```

```
predMotifMean <- predMotif %>% group_by(ID) %>% summarise(mean_score = mean(score))
predMotiffMax <- predMotif %>% group_by(ID) %>% summarise(max_score = max(score))

predMotif <- predMotifMean %>% full_join(predMotiffMax, by = "ID")


save(pcMotif, file = "~/bin/r_git/R/pcMotif.Rda")
save(ncMotif, file = "~/bin/r_git/R/ncMotif.Rda")
save(predMotif, file = "~/bin/r_git/R/predMotif.Rda")
```

RandomForest

```
pcMFE <- read.table("~/phd/RNASeq/srna_seqs/version_1/positive_control/positive_control.rnaalifold", sep
ncMFE <- read.table("~/phd/RNASeq/srna_seqs/version_1/negative_control/negative_control.rnaalifold", sep

pcMFE <- pcMFE %>% separate(V1, into = c("ID_1", "ID_2", "t1"), remove = T, extra = "drop", sep = "\\."
ncMFE <- ncMFE %>% separate(V1, into = c("ID_1", "ID_2", "t1"), remove = T, extra = "drop", sep = "\\."

pcGC <- read.table("~/phd/RNASeq/srna_seqs/version_1/positive_control.gc", sep = "\t", comment.char = "#
ncGC <- read.table("~/phd/RNASeq/srna_seqs/version_1/negative_control_no_shuffle.gc", sep = "\t", commen

pcGC <- pcGC %>% group_by(V1) %>% summarise(gc_score = mean(V2)) %>% separate(V1, into = c("ID", "t1"),
ncGC <- ncGC %>% group_by(V1) %>% summarise(gc_score = mean(V2)) %>% separate(V1, into = c("ID", "t1"),

load("maxDistsPC.Rda") #variablename: distsPositive
load("maxDistsNC.Rda") #variablename: distsNegative

ncReadDepths <- read.table("~/phd/RNASeq/srna_seqs/version_1/negative_read_depths.txt", header = T, sep
pcReadDepths <- read.table("~/phd/RNASeq/srna_seqs/version_1/positive_control_read_depths.txt", header =

load("pcCovariation.Rda") #variablename: pcCov
load("ncCovariation.Rda") #variablename: ncCov

pcCov <- pcCov %>% dplyr::rename(mean_cov = mean_score, min_eval_cov = min_eval)
ncCov <- ncCov %>% dplyr::rename(mean_cov = mean_score, min_eval_cov = min_eval)

load("pcMotif.Rda") #variablename: pcMotif
load("ncMotif.Rda") #variablename: ncMotif

pcMotif <- pcMotif %>% dplyr::rename(mean_motif = mean_score, max_motif = max_score)
ncMotif <- ncMotif %>% dplyr::rename(mean_motif = mean_score, max_motif = max_score)

load("pcAlifold.Rda") #variablename: pcAlifold
load("ncAlifold.Rda") #variablename: ncAlifold

pcDat <- pcMFE %>%
  full_join(pcGC, by = "ID") %>%
  full_join(distsPositive, by = "ID") %>%
  full_join(pcReadDepths, by = "ID") %>%
  full_join(pcCov, by = "ID") %>%
  full_join(pcMotif, by = "ID")%>%
```

```
  full_join(pcAlifold, by = "ID") %>%
  mutate(group = "Positive Control")


ncDat <- ncMFE %>%
  full_join(ncGC, by = "ID") %>%
  full_join(distsNegative, by = "ID") %>%
  full_join(ncReadDepths, by = "ID") %>%
  full_join(ncCov, by = "ID") %>%
  full_join(ncMotif, by = "ID")%>%
  full_join(ncAlifold, by = "ID") %>%
  mutate(group = "Negative Control")


dat <- pcDat %>% bind_rows(ncDat)%>%
  select(-mean_median, -mean_max, -median_mean, -median_median, -median_max, -max_mean, -max_median, -I

dat <- dat[,c(4, 1:3, 5:12)]

dat$mfe_score[is.na(dat$mfe_score)] <- 0
dat$gc_score[is.na(dat$gc_score)] <- 50
dat$max_dist[is.na(dat$max_dist)] <- 0
dat$mean_mean[is.na(dat$mean_mean)] <- 0
dat$max_max[is.na(dat$max_max)] <- 0
dat$mean_cov[is.na(dat$mean_cov)] <- 0
dat$min_eval_cov[is.na(dat$min_eval_cov)] <- 10
dat$mean_motif[is.na(dat$mean_motif)] <- 0
dat$max_motif[is.na(dat$max_motif)] <- 0
dat$z_mean[is.na(dat$z_mean)] <- 10
dat$z_max[is.na(dat$z_max)] <- 10
randomNum <- runif(n = nrow(dat), min = 0, max = 1)

dat$random <- randomNum
dat2 <- dat %>% mutate(group = ifelse(group == "Positive Control", 1, 0)) #%>% select(-na_count)

dat2$group <- as.factor(dat2$group)

data_set_size <- floor(nrow(dat2)/2)
indexes <- sample(1:nrow(dat2), size = data_set_size)

training <- dat2[indexes,]
validation1 <- dat2[-indexes,]

rf_classifier = randomForest(group ~ ., data=training, ntree=100, importance=TRUE)
rf_classifier
varImpPlot(rf_classifier)
prediction_for_table <- predict(rf_classifier,validation1[,-1])
table(observed=validation1[,1],predicted=prediction_for_table)
prediction_for_roc_curve <- predict(rf_classifier,validation1[,-1],type="prob")
dat3 <- dat %>% select(-group)
corMat <- cor(dat3, method = "spearman")
round(corMat, 2)
  get_lower_tri<-function(cormat){
```

```
    cormat[upper.tri(cormat)] <- NA
    return(cormat)
  }
  get_upper_tri <- function(cormat){
    cormat[lower.tri(cormat)]<- NA
    return(cormat)
  }
upper_tri <- get_upper_tri(corMat)

melted_cormat <- melt(upper_tri, na.rm = TRUE)
p <- ggplot(data = melted_cormat, aes(Var2, Var1, fill = value))+
 geom_tile(color = "white")+
 scale_fill_gradient2(low = "blue", high = "red", mid = "white",
   midpoint = 0, limit = c(-1,1), space = "Lab",
   name="Pearson\nCorrelation") +
  theme_minimal()+
 theme(axis.text.x = element_text(angle = 45, vjust = 1,
    size = 12, hjust = 1))+
 coord_fixed()
p
```