Characterization and correction of ATAC-seq Tn5 sequence bias

Jacob Wolpe and Michael J. Guertin

July 10, 2020

Contents

List of Figures

1 Processing ATAC-seq reads

1.1 Retrieving FASTA and FASTQ files

All the files should be named with the cell type, conditions, description, the replicate number. No spaces. This makes it so that downstream analyses can be automated. The ATAC-seq data is GSE92674 (Martins *et al.*, 2018).

`fastq-dump` is an SRA tool (Kodama *et al.*, 2011).
I explicitly indicate software dependencies preceding the relevant chunks (let me know if I miss any). Software installs can be very annoying. My advice: read the `README` files.
Software installations:

`sra-tools`: `https://github.com/ncbi/sra-tools`

Code: `https://raw.githubusercontent.com/guertinlab/Tn5bias/master/section1_1.sh`

```
cd ~/Desktop/atac_test

#Rivanna
#module load sratoolkit/2.9.1
#cd /scratch/mjg7y/
fasterq-dump SRR5123141
fasterq-dump SRR5123142

gzip *fastq

mv SRR5123141_1.fastq.gz C1_gDNA_rep1_PE1.fastq.gz
mv SRR5123141_2.fastq.gz C1_gDNA_rep1_PE2.fastq.gz
mv SRR5123142_1.fastq.gz C1_gDNA_rep2_PE1.fastq.gz
mv SRR5123142_2.fastq.gz C1_gDNA_rep2_PE2.fastq.gz

#you also need to download the human genome:
#one can use the following command on a Linux:
wget http://hgdownload.cse.ucsc.edu/goldenPath/hg38/bigZips/hg38.fa.gz
wget https://hgdownload.cse.ucsc.edu/goldenpath/hg38/chromosomes/chrM.fa.gz

#unzip
gunzip hg38.fa.gz
gunzip chrM.fa.gz
```

1.2 Processing ATAC reads

The first task is to build the genome indices with `bowtie2`. This only has to be performed once per genome. We trim the first few bases with `fastp` (Chen *et al.*, 2018) . We want to exclude the chrM reads because ATAC enriches for chrM DNA and there are regions of the nuclear genome that are mitochondrial in origin. This causes reads to pile up in these nuclear regions. We align to the `chrM` genome using `bowtie2` (Langmead *et al.*, 2009). Next we use a few `samtools` (Li *et al.*, 2009) commands to manipulate the files with the end result being two `FASTQ` files that lack `chrM`-aligned reads and have the first four bases trimmed. **Note that the trimming of the first four bases is based upon my understanding of the molecular biology, but this may need to change based upon the results of this project.** We align the remaining reads to the `hg38` genome using `bowtie2`. Then we convert the `sam` file to the compressed and sorted `BAM` format using `samtools` (Li *et al.*, 2009). The `fixmate`, `sort`, and `markdup` steps removes paired reads in which the alignments were previously seen. These are likely PCR duplicate amplicons, and unlikely to arise independently. I ran it through `seqOutBias` (Martins *et al.*, 2018) , but I don't think it is strictly necessary, as we are going to process the paired end `BAM` files and the *plus* and *minus*-aligned reads separately so that we can track what is going on.

Software installations:

`fastp`: `https://github.com/OpenGene/fastp/blob/master/README.md`

`bowtie2`: `http://bowtie-bio.sourceforge.net/bowtie2/index.shtml`

`samtools`: `http://www.htslib.org`

`seqOutBias`: `https://github.com/guertinlab/seqOutBias`

Code: `https://raw.githubusercontent.com/guertinlab/Tn5bias/master/section1_2.sh`

```
#build index genome
bowtie2-build hg38.fa hg38
bowtie2-build chrM.fa chrM

for fq in C1*PE1.fastq.gz
do
    name=$(echo $fq | awk -F"_PE1.fastq.gz" '{print $1}')
    echo $name
    ./fastp -f 4 -F 4 -i ${name}_PE1.fastq.gz -I ${name}_PE2.fastq.gz \
       -o ${name}_PE1.trimmed.fastq.gz -O ${name}_PE2.trimmed.fastq.gz
    bowtie2 -x chrM -1 ${name}_PE1.trimmed.fastq.gz -2 ${name}_PE2.trimmed.fastq.gz \
       | samtools view -b - | samtools sort - -o $name.sorted.chrM.bam
# I cannot figure out how to pipe the previous line with the next two lines.
    samtools index $name.sorted.chrM.bam
    samtools view -b $name.sorted.chrM.bam '*' | samtools sort -n - \
       | bamToFastq -i - -fq ${name}_PE1.chrM.fastq -fq2 ${name}_PE2.chrM.fastq
    gzip *.chrM.fastq
    rm $name.sorted.chrM.bam
    bowtie2 --maxins 500 -x hg38 -1 ${name}_PE1.chrM.fastq.gz -2 ${name}_PE2.chrM.fastq.gz \
       | samtools view -bS - | samtools sort -n - | samtools fixmate -m - - \
       | samtools sort - | samtools markdup -r - $name.bam
    seqOutBias hg38.fa $name.bam --no-scale --bw=${name}.bigWig --shift-counts --read-size=30
done
```

2 Characterizing the sequence bias directly from ATAC reads

I have already run through the analysis with the BAM file "CEM-C7_untreated_rep2.bam", but the results are confusing. The only way that I can think to make sense of the spacing of the sequence bias is to treat PE1 and PE2 reads separately and reads that align to the plus and minus strand separately. Then we can empirically determine whether the seqLogo spacing is offset in these analyses and if the Hill Climbing- determined k-mer mask is offset between PE and strand status. If the seqLogo and mask are offset the same degree in the two analyses, then it may mean that the plus and minus reads need to be scaled separately and then we can build on the the molecular biology illustration the sequence bias and see if this makes sense on the Tn5 structure, if it is known how the structure corresponds to precise insertion/cut site.

2.1 Python script: `bedToOneEntryBed.py`

The file is at `https://raw.githubusercontent.com/guertinlab/Tn5bias/master/bedToOneEntryBed.py` and it converts a bed to a bed with one entry per instance, so that a FASTA entry is present for each ATAC insertion event.

```
#! /sw/bin/python
import re
import string
import sys
import getopt
import os
import itertools
import glob

def function(fqFileName):
    infile=open(fqFileName, 'r')
    outfile=open(string.split(fqFileName,'.bed')[0]+'.oneentry.bed', 'w')
    while 1:
        line=infile.readline()
        if not line: break
        for i in range(int(line.split()[4])):
            outfile.write('%s'%(line))
    infile.close()
    outfile.close()


def main(argv):
    try:
        opts, args = getopt.getopt(argv, "i:h", ["infile=","help"])
    except getopt.GetoptError, err:
        print str(err)
        sys.exit(2)
    infile = False
    for opt, arg in opts:
        if opt in ('-i', '--infile'):
            infile = arg
        elif opt in ('-h', '--help'):
            print '\n./bedToOneEntryBed.py -i test_PE1_plus_not_scaled.bed'
            sys.exit()
    if infile:
        print infile
        function(infile)
if __name__ == "__main__":
    main(sys.argv[1:])
```

2.2 Separate BAM by strand and PE status and retrieve the FASTA sequence

Next we will split PE and minus/plus reads and get their sequences. I don't know if `--shift-counts` appropriately shifts the BED out of minus aligned reads. The way to test this is to run with and without this option and look at the output and determine if it changes. If this does not shift the BED, then the `awk` code that extends the window needs to be modified with a 1 base shift. Let me know once you test this and we should discuss what has to change in the `awk` lines below. **this section is hard coded for the CEM-C7_untreated_rep2 sample, but you can change the name of the file and proceed to determine whether the shift counts option applies to the BED output in seqOutBias. Once you determine that, it will be a good excercise to incorporate this code into a loop that operates on all aligned BAM files.**

Code: `https://raw.githubusercontent.com/guertinlab/Tn5bias/master/section2_2.sh`

```
#comments are for Rivanna (in progress)
wget https://raw.githubusercontent.com/guertinlab/Tn5bias/master/bedToOneEntryBed.py
#module load gcc/7.1.0
#module load seqoutbias/1.2.0
#module load bedtools/2.26.0
#module load intel/20.0
#module load intelmpi/20.0
#module load python/3.7.7
#ijob -A guertinlab -c 1 -p standard -t 10:00:00

for i in C1_*_PE1_plus.bam
do
    name=$(echo $i | awk -F"_PE1_plus.bam" '{print $1}')
    echo $name
#separate PE1 and PE2 reads
    samtools view -b -f 0x0040 ${name}.bam > ${name}_PE1.bam
    samtools view -b -f 0x0080 ${name}.bam > ${name}_PE2.bam
#separate plus and minus strand aligning reads:
    samtools view -bh -F 20 ${name}_PE1.bam > ${name}_PE1_plus.bam
    samtools view -bh -f 0x10 ${name}_PE1.bam > ${name}_PE1_minus.bam
    samtools view -bh -F 20 ${name}_PE2.bam > ${name}_PE2_plus.bam
    samtools view -bh -f 0x10 ${name}_PE2.bam > ${name}_PE2_minus.bam
#run seqOutBias to get the bed
    seqOutBias hg38.fa ${name}_PE1_plus.bam --no-scale --out=no_scale.tbl --shift-counts \
                                --bed=${name}_PE1_plus_shift_counts.bed \
                                --bw=${name}_PE1_plus_shift_counts.bigWig --read-size=30
    seqOutBias hg38.fa ${name}_PE1_minus.bam --no-scale --out=no_scale.tbl --shift-counts \
                                --bed=${name}_PE1_minus_shift_counts.bed \
                                --bw=${name}_PE1_minus_shift_counts.bigWig --read-size=30
    seqOutBias hg38.fa ${name}_PE2_plus.bam --no-scale --out=no_scale.tbl --shift-counts \
                                --bed=${name}_PE2_plus_shift_counts.bed \
                                --bw=${name}_PE2_plus_shift_counts.bigWig --read-size=30
    seqOutBias hg38.fa ${name}_PE2_minus.bam --no-scale --out=no_scale.tbl --shift-counts \
                                --bed=${name}_PE2_minus_shift_counts.bed \
                                --bw=${name}_PE2_minus_shift_counts.bigWig --read-size=30
    python bedToOneEntryBed.py ${name}_PE1_plus_shift_counts.bed
    python bedToOneEntryBed.py ${name}_PE1_minus_shift_counts.bed
    python bedToOneEntryBed.py ${name}_PE2_plus_shift_counts.bed
    python bedToOneEntryBed.py ${name}_PE2_minus_shift_counts.bed
#these bed files can be used to get the sequence flanking ALL Tn5 insertion sites,
#so we can see the precise nature of the sequence bias for each PE/strand combination
    awk '{$2 = $2 - 9; print}' ${name}_PE1_plus_shift_counts.oneentry.bed | \
       awk '{OFS="\t";} {$3 = $2 + 20; print}' | grep -v - | \
       fastaFromBed -fi hg38.fa -s -bed stdin -fo ${name}_PE1_plus_shift_counts.fasta
    awk '{$2 = $2 - 9; print}' ${name}_PE1_minus_shift_counts.oneentry.bed | \
       awk '{OFS="\t";} {$3 = $2 + 20; print}' |  grep -v - | \
       fastaFromBed -fi hg38.fa -s -bed stdin -fo ${name}_PE1_minus_not_scaled_shift_counts.fasta
    awk '{$2 = $2 - 9; print}' ${name}_PE2_plus_shift_counts.oneentry.bed | \
       awk '{OFS="\t";} {$3 = $2 + 20; print}' | grep -v - | \
       fastaFromBed -fi hg38.fa -s -bed stdin -fo ${name}_PE2_plus_shift_counts.fasta
    awk '{$2 = $2 - 9; print}' ${name}_PE2_minus_shift_counts.oneentry.bed | \
       awk '{OFS="\t";} {$3 = $2 + 20; print}' |  grep -v - | \
       fastaFromBed -fi hg38.fa -s -bed stdin -fo ${name}_PE2_minus_shift_counts.fasta
done


for i in C1_*_PE1_plus.bam
do
    name=$(echo $i | awk -F"_PE1_plus.bam" '{print $1}')
    echo $name
    seqOutBias hg38.fa ${name}_PE1_plus.bam --no-scale --out=no_scale.tbl \
                                --bed=${name}_PE1_plus_no_shift.bed \
                                --bw=${name}_PE1_plus_no_shift.bigWig --read-size=30
    seqOutBias hg38.fa ${name}_PE1_minus.bam --no-scale --out=no_scale.tbl \
                                --bed=${name}_PE1_minus_no_shift.bed \
                                --bw=${name}_PE1_minus_no_shift.bigWig --read-size=30
    seqOutBias hg38.fa ${name}_PE2_plus.bam --no-scale --out=no_scale.tbl \
                                --bed=${name}_PE2_plus_no_shift.bed \
                                --bw=${name}_PE2_plus_no_shift.bigWig --read-size=30
    seqOutBias hg38.fa ${name}_PE2_minus.bam --no-scale --out=no_scale.tbl \
                                --bed=${name}_PE2_minus_no_shift.bed \
                                --bw=${name}_PE2_minus_no_shift.bigWig --read-size=30
    python bedToOneEntryBed.py ${name}_PE1_plus_no_shift.bed
    python bedToOneEntryBed.py ${name}_PE1_minus_no_shift.bed
    python bedToOneEntryBed.py ${name}_PE2_plus_no_shift.bed
    python bedToOneEntryBed.py ${name}_PE2_minus_no_shift.bed
#these bed files can be used to get the sequence flanking ALL Tn5 insertion sites,
#so we can see the precise nature of the sequence bias for each PE/strand combination
    awk '{$2 = $2 - 9; print}' ${name}_PE1_plus_no_shift.oneentry.bed | \
       awk '{OFS="\t";} {$3 = $2 + 20; print}' | grep -v - | \
       fastaFromBed -fi hg38.fa -s -bed stdin -fo ${name}_PE1_plus_no_shift.fasta
    awk '{$2 = $2 - 9; print}' ${name}_PE1_minus_no_shift.oneentry.bed | \
       awk '{OFS="\t";} {$3 = $2 + 20; print}' |  grep -v - | \
       fastaFromBed -fi hg38.fa -s -bed stdin -fo ${name}_PE1_minus_no_shift.fasta
    awk '{$2 = $2 - 9; print}' ${name}_PE2_plus_no_shift.oneentry.bed | \
       awk '{OFS="\t";} {$3 = $2 + 20; print}' | grep -v - | \
```

```
       fastaFromBed -fi hg38.fa -s -bed stdin -fo ${name}_PE2_plus_no_shift.fasta
    awk '{$2 = $2 - 9; print}' ${name}_PE2_minus_no_shift.oneentry.bed | \
       awk '{OFS="\t";} {$3 = $2 + 20; print}' |  grep -v - | \
       fastaFromBed -fi hg38.fa -s -bed stdin -fo ${name}_PE2_minus_no_shift.fasta
done
```

2.3 Visualizing the sequence bias in R

The next step is to visualize the preferred sequence of Tn5 by anchoring the sequence on the beginning of the trimmed read and asking how frequent A,C, G, and T occur at each position. The code below does this, but there are a few excercises that may help you understand the data and how to generaize a function in R. First, as coded below, which position in the PSWM corresponds to the first base of the trimmed FASTQ file? Is this the same for PE and strand status? Second, this function is hard coded for a 20mer, but can you try to change it to accept and FASTA file with a constant number of bases per sequence entry? Then expand it to 40+ bases. I woudl hope that the IC as you get further away from the cut site shoudl diminsh to zero. Third, I run the same process four times, which probably means I should make the process a function and simply call the function four times. I realize that the minus strand reads shoudl be reverse complemented. In teh function you can include a rc = FALSE, option, then a conditional: if rc == TRUE, then reverse complement to motif. Can you try to do this? I understand that this is a lot, but I will be coding it up in parallel, if you need help.

```
source('https://raw.githubusercontent.com/guertinlab/seqOutBias/master/docs/R/seqOutBias_functions.R')
library(ggseqlogo)
library(universalmotif)
pswm.func.2 <- function(x.ligation, out = 'outfilename') {
    a = lapply(strsplit(as.character(x.ligation), ''), "[", 1)
    b = lapply(strsplit(as.character(x.ligation), ''), "[", 2)
    c = lapply(strsplit(as.character(x.ligation), ''), "[", 3)
    d = lapply(strsplit(as.character(x.ligation), ''), "[", 4)
    e = lapply(strsplit(as.character(x.ligation), ''), "[", 5)
    f = lapply(strsplit(as.character(x.ligation), ''), "[", 6)
    g = lapply(strsplit(as.character(x.ligation), ''), "[", 7)
    h = lapply(strsplit(as.character(x.ligation), ''), "[", 8)
    i = lapply(strsplit(as.character(x.ligation), ''), "[", 9)
    j = lapply(strsplit(as.character(x.ligation), ''), "[", 10)
    k = lapply(strsplit(as.character(x.ligation), ''), "[", 11)
    l = lapply(strsplit(as.character(x.ligation), ''), "[", 12)
    m = lapply(strsplit(as.character(x.ligation), ''), "[", 13)
    n = lapply(strsplit(as.character(x.ligation), ''), "[", 14)
    o = lapply(strsplit(as.character(x.ligation), ''), "[", 15)
    p = lapply(strsplit(as.character(x.ligation), ''), "[", 16)
    q = lapply(strsplit(as.character(x.ligation), ''), "[", 17)
    r = lapply(strsplit(as.character(x.ligation), ''), "[", 18)
    s = lapply(strsplit(as.character(x.ligation), ''), "[", 19)
    t = lapply(strsplit(as.character(x.ligation), ''), "[", 20)
    col.matrix = cbind(a,b,c,d,e,f, g,h,i,j,k,l,m,n,o,p,q,r,s,t)
    a.nuc = sapply(1:20, function(x) sum(col.matrix[,x] == "A"))
    t.nuc = sapply(1:20, function(x) sum(col.matrix[,x] == "T"))
    c.nuc = sapply(1:20, function(x) sum(col.matrix[,x] == "C"))
    g.nuc = sapply(1:20, function(x) sum(col.matrix[,x] == "G"))
    #pswm = cbind(a.nuc*(0.25/.2), c.nuc*(0.25/.3), g.nuc*(0.25/.3), t.nuc*(0.25/.2))
    pswm = cbind(a.nuc, c.nuc, g.nuc, t.nuc)
    print(pswm)
    outfile = file(paste0(out, '.txt'))
    on.exit(close(outfile))
    writeLines(c("MEME version 4", "ALPHABET= ACGT", "strands: + -", " ",
                 "Background letter frequencies (from uniform background):",
                 "A 0.20000 C 0.30000 G 0.30000 T 0.20000", paste("MOTIF", out), " ",
                 "letter-probability matrix: alength= 4 w= 20"), outfile)
    pswm = pswm/rowSums(pswm)
    write.table(pswm, file = paste0(out, '.txt'), append = TRUE, quote=FALSE, row.names =FALSE, col.names = FALSE)
#the followign line can be uncommented if ceqlogo is installed and in your $PATH
#    system(paste('ceqlogo -i ', out, '.txt -m 1 > ', out, '.eps', sep=''))
    return(pswm)
}

plot.seqlogo.func <- function(x, outfile = "ATAC-kmer_optimization_all_test.pdf") {
    w =  0.663 + (ncol(x) + 1)*0.018 + (ncol(x)+2)* .336
    pdf(outfile, useDingbats=FALSE, width=w, height=2.695)
    print(ggseqlogo(x,  facet = "wrap", font = 'helvetica_bold'))
    dev.off()
}

dir = paste0(path.expand("~"),'/Desktop/atac_test/')
setwd(dir)
```

```
#I am running the same scripts on four files because it was easiest to copy and paste
#How can you clean this up by making a function and running this function on a list of input files?

pe1.minusATAC = read.table('CEM-C7_untreated_rep2_PE1_minus_shift_counts.fasta',
                           comment.char = '>')
pe1.minusATAC[,1] = as.character(pe1.minusATAC[,1])
pe1.minusATAC = data.frame(lapply(pe1.minusATAC, function(v) {
if (is.character(v)) return(toupper(v))
else return(v)
}))

pe1.plusATAC = read.table('CEM-C7_untreated_rep2_PE1_plus_shift_counts.fasta',
                          comment.char = '>')
pe1.plusATAC[,1] = as.character(pe1.plusATAC[,1])
pe1.plusATAC = data.frame(lapply(pe1.plusATAC, function(v) {
if (is.character(v)) return(toupper(v))
else return(v)
}))

pe2.minusATAC = read.table('CEM-C7_untreated_rep2_PE2_minus_shift_counts.fasta',
                           comment.char = '>')
pe2.minusATAC[,1] = as.character(pe2.minusATAC[,1])
pe2.minusATAC = data.frame(lapply(pe2.minusATAC, function(v) {
if (is.character(v)) return(toupper(v))
else return(v)
}))

pe2.plusATAC = read.table('CEM-C7_untreated_rep2_PE2_plus_shift_counts.fasta',
                          comment.char = '>')
pe2.plusATAC[,1] = as.character(pe2.plusATAC[,1])
pe2.plusATAC = data.frame(lapply(pe2.plusATAC, function(v) {
if (is.character(v)) return(toupper(v))
else return(v)
}))


pswm.pe1.minus = pswm.func.2(pe1.minusATAC[,1], 'ATAC_bias_pe1_minus')
pswm.pe1.plus = pswm.func.2(pe1.plusATAC[,1], 'ATAC_bias_pe1_plus')
pswm.pe2.minus = pswm.func.2(pe2.minusATAC[,1], 'ATAC_bias_pe2_minus')
pswm.pe2.plus = pswm.func.2(pe2.plusATAC[,1], 'ATAC_bias_pe2_plus')

save(pswm.pe1.minus, pswm.pe1.plus, pswm.pe2.minus, pswm.pe2.plus, file ='pswm.Rdata')



pswm.pe1.minus.trans = t(pswm.pe1.minus)
rownames(pswm.pe1.minus.trans) = c('A', 'C', 'G', 'T')

pswm.pe1.plus.trans = t(pswm.pe1.plus)
rownames(pswm.pe1.plus.trans) = c('A', 'C', 'G', 'T')

pswm.pe2.minus.trans = t(pswm.pe2.minus)
rownames(pswm.pe2.minus.trans) = c('A', 'C', 'G', 'T')

pswm.pe2.plus.trans = t(pswm.pe2.plus)
rownames(pswm.pe2.plus.trans) = c('A', 'C', 'G', 'T')


#the minus strand may need to be reverse complemented...
#does it make sense to reverse complement the minus (I don't know the answer--
#i will need to refer to Tn5 molcular biology, which is complicated).

plot.seqlogo.func(pswm.pe1.minus.trans, outfile='pswm_pe1_minus_trans.pdf')
plot.seqlogo.func(pswm.pe1.plus.trans, outfile='pswm_pe1_plus_trans.pdf')

plot.seqlogo.func(pswm.pe2.minus.trans, outfile='pswm_pe2_minus_trans.pdf')
plot.seqlogo.func(pswm.pe2.plus.trans, outfile='pswm_pe2_plus_trans.pdf')
```

I had an idea that we did not implement in the `seqOutBias` paper, we can optimize the mask at sites that most strongly conform to the Tn5 bias. This finds those sites in the genome. The rationale is that the bias will be extreme at these sites, so the most appropriate k-mer mask sites will be revealed. The later code is designed to use 20 PSWMs, but we should alter to code to run only on sites containing the sites that conform best to the Tn5 preferred site.

```
fimo --thresh 0.00005 --text ATAC_bias_pe1_minus.txt hg38.fa > ATAC_bias_pe1_minus_00005_fimo.txt
fimo --thresh 0.00005 --text ATAC_bias_pe1_plus.txt hg38.fa > ATAC_bias_pe1_plus_00005_fimo.txt
fimo --thresh 0.00005 --text ATAC_bias_pe2_minus.txt hg38.fa > ATAC_bias_pe2_minus_00005_fimo.txt
fimo --thresh 0.00005 --text ATAC_bias_pe2_plus.txt hg38.fa > ATAC_bias_pe2_plus_00005_fimo.txt
```

2.4 seqLogos

I am just putting these seqLogos in here as a place holder...

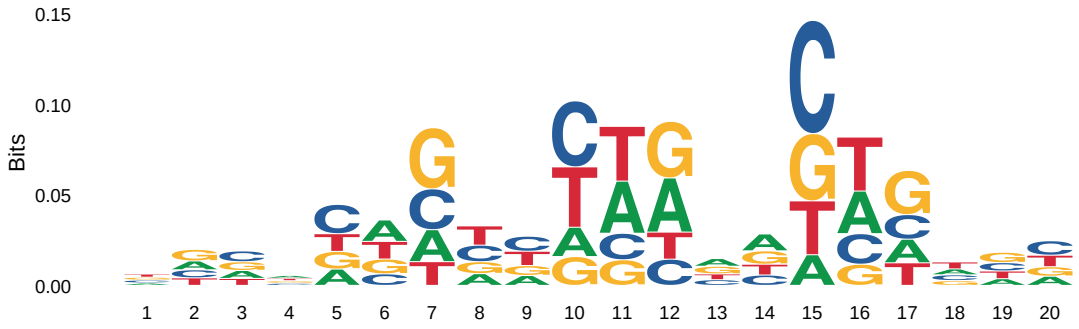

Figure 1: PE1 minus PSWM.

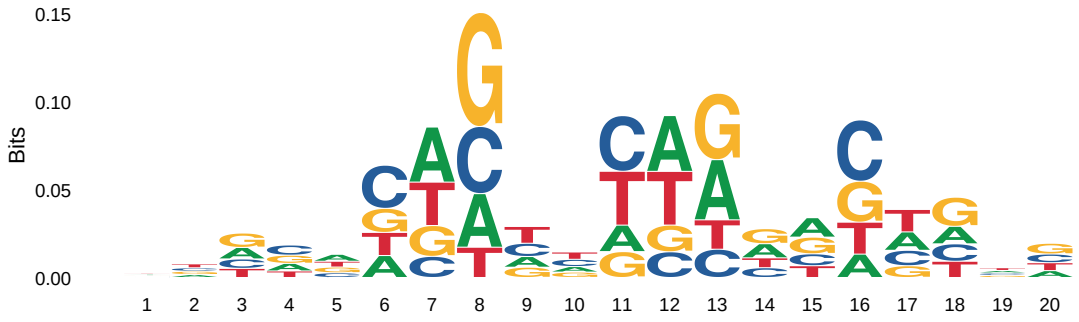

Figure 2: PE1 plus PSWM.

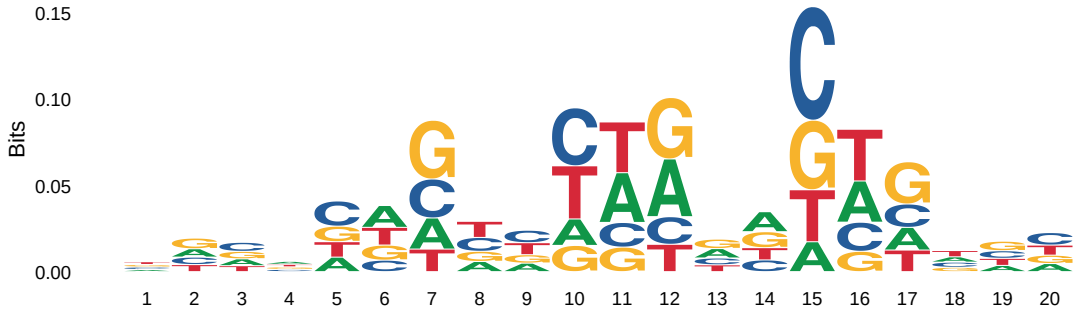

Figure 3: PE2 minus PSWM.

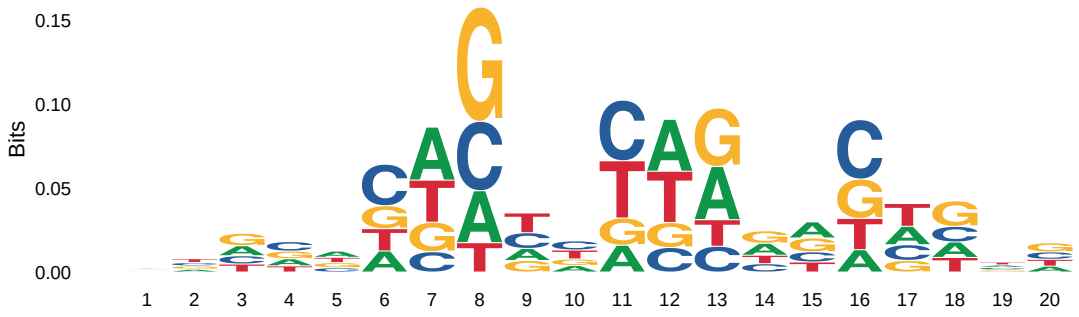

Figure 4: PE2 plus PSWM.

3 Selecting a k-mer mask

The sequence bias was first recognized by composite profiles of DNase-seq data centered on Transcription Factor binding sites. Therefore, we aim to use a diverse set of PSWMs representing many composite TFBSs and empirically determine which mask positions are most influential at flattening these profiles.

3.1 Hill climbing optimization

I have the code below for exclusively the PE1 plus data, but you should run it for all four combinations of PE and strand. This takes a long time to run and the output files are big. The method takes a starting k-mer mask, a set of site tables (one table per TF), and at each step turns an masked position into an unmasked position, choosing the position that results in the lowest score. It iterates until there are no more unmasked positions.

because I plan to change the input sites, the hc.search and/or run.cutmask should be modified to look for the file and if the files exists, then skip making the bigWig and proceed directly to the evaluation: eval.cutmask. Running the seqOutBias is what is time consuming, so if all the bigWig files are there, then the processes have the potential to run more quickly.

I have this running on Rivanna with 20 + cores. It takes too long to run locally. Remind me to update this code if you get here.

```
#make sure you ar ein the directoy with all teh relevant files
#hill climbing determinatino of a mask

#
#module load gcc/7.1.0
#module load openmpi/3.1.4
#module load R/4.0.0
#make a slurm and couple with the previous chunk
#ijob -A guertinlab -c 20 -p standard -t 48:00:00


source("https://raw.githubusercontent.com/guertinlab/seqOutBias/master/docs/R/seqOutBias_hcsearch.R")

#I changed it so that there is no more than 300k sites per fimo file
#All the motif conformity scores are greater than the 300,000th score
load.sites <- function(filenames) {
    lapply(filenames, function(filename) {
        fimo = read.table(filename, skip = 1, sep ='\t')
        len.vec.values = length(fimo[,7])
        if (length(fimo[,7]) > 3000000 ){
            subset.len = len.vec.values - 300000
            sort.sub = sort(fimo[,7], partial=subset.len)[subset.len]
            top.index = which(fimo[,7] > sort.sub)
            fimo.small = fimo[top.index,]
            bed6 = fimo.small[, c(3,4,5,2,7,6)]
        } else {
             bed6 = fimo[, c(3,4,5,2,7,6)]
        }
        bed6
    })
}

#this helps a little especially for the first position in the mask, or subsequent positions if they are the saem
hc.search.mods <- function(sites, start_mask, seqoutbias.args, prefix = "run_",
                           bw.split = FALSE, cleanup = TRUE, sqcmd = "seqOutBias", mc.cores = 2) {
    neighbors <- function(vecmask) {
                                        # generate list of neighboring masks that differ
                                        # by the addition of a single unmasked position
        result <- vector(mode="list", length=length(vecmask))
        n <- 0
        for (k in 1:length(vecmask)) {
            if (vecmask[k] == -1) { # X
                n <- n + 1
                vi = vecmask
                vi[k] = 1 # N
                result[[n]] <- vec.to.mask(vi)
            }
        }

        if (n == 0) return(NULL)
        result[1:n]
    }                                   #
    mask = character()
    score = numeric()
    rtable = data.frame(c(score, mask), stringsAsFactors=FALSE)
```

```
    nextlst = neighbors(mask.to.vec(start_mask))

        while (!is.null(nextlst)) {
            print(nextlst)
            scores = mclapply(nextlst, function(cutmask) {
                if (file.exists(paste0('runhc_', cutmask, '.bw'))) {
                    bw.plus = load.bigWig(paste0('runhc_', cutmask, '.bw'))
                    bw.minus = load.bigWig(paste0('runhc_', cutmask, '.bw'))
                } else {
                    bw.paths = run.cutmask(cutmask, seqoutbias.args, sqcmd=sqcmd, prefix=prefix)
                    bw.plus = load.bigWig(bw.paths[1])
                    bw.minus = load.bigWig(bw.paths[2])
                }
                    score = eval.cutmask(sites, bw.plus, bw.minus)
                    data.frame(score = score, mask = cutmask, stringsAsFactors=FALSE)
            }, mc.cores = mc.cores)
            scores.values = sapply(scores, function(pair) pair$score)

                                    # pick the neighbor with the best score
            k = which.min(scores.values)[1]
            mask = scores[[k]]$mask
            cat(scores.values[k], mask, "\n")
            rtable = rbind(rtable, scores[[k]])
            print(rtable)
                                    # next
            nextlst <- neighbors(mask.to.vec(mask))
        }
    rtable
}


#sites = load.sites(c("CTCF_fimo.txt", "Elf1_fimo.txt", "Gata3_fimo.txt", "Max_fimo.txt",
#          "SP1_fimo.txt", "EBF1_fimo.txt", "REST_fimo.txt"))
#names(sites) = c("CTCF", "Elf1", "Gata3", "Max", "SP1", "EBF1", "REST")

system('mkdir pe1_plus')
setwd('pe1_plus')

seqOutBias.args = "hg38.fa C1_gDNA_rep1.bam --shift-counts --read-size=30"
seqOutBias.cmd = "seqOutBias"

#I am starting here with a 10mer window, but depending on how much IC is outside this region when you
#change the awk command in sectino 2.2 and carry it forward, you may want to make a wider window.
initial.mask = "XXXXXXXXXXCXXXXXXXXXX"


#This approach requires many kmer-mask evaluations, which correspond to full runs of seqOutBias.
#for some reason I can only run this with mc.core > 1 in an R terminal session and not an R environment:
#this invokes run.cutmask, but this only needs to be implemented once per BAM file.
#result.table = hc.search(sites, initial.mask, seqOutBias.args, prefix = "runhc_", sqcmd = seqOutBias.cmd, mc.cores = 4)

sites = load.sites(c("ATAC_bias_pe1_minus_00005_fimo.txt", "ATAC_bias_pe1_plus_00005_fimo.txt",
                     "ATAC_bias_pe2_minus_00005_fimo.txt", "ATAC_bias_pe2_plus_00005_fimo.txt"))

#use more mc.cores if you have more than 20 X in the mask
result.table = hc.search.mods(sites, initial.mask,
                              seqOutBias.args, prefix = "runhc_", sqcmd = seqOutBias.cmd, mc.cores = 20)
```

Since the results.table above takes a long time to run, exploring parallel options in Rivanna is a good idea. mc.cores runs a mask instance on each core. If pe1 minus is identical to plus, then the offset in the PSWM may be an artifact of how the bed file was made. The *hc.atac.cutmasks* are empirically determined based upon the previous code chunk. Look at the first 11 *N* instances. Here we are evaluating the k-mer masks over the top eleven positions, and we are loading the bigWigs that were previously generated.

```
system(paste('seqOutBias ', seqOutBias.args,
   ' --no-scale --out=runhc_XXXXXXXXXXCXXXXXXXXXX.tbl --bw=runhc_XXXXXXXXXXCXXXXXXXXXX.bw'))


hc.atac.cutmasks = c('XXXXXXXXXXCXXXXXXXXXX',
                     'XXXXXXXXXXCXXXNXXXXXX',
                     'XXXXXXXXNXCXXXNXXXXXX',
                     'XXXXXXXXNXCNXXNXXXXXX',
                     'XXXXXNXXNXCNXXNXXXXXX',
                     'XXXNXNXXNXCNXXNXXXXXX',
                     'XNXNXNXXNXCNXXNXXXXXX',
                     'XNXNXNXXNXCNXXNXXXXNX',
                     'XNXNXNXXNXCNNXNXXXXNX',
                     'XNXNXNXXNXCNNXNXNXXNX',
                     'XNXNXNXNNXCNNXNXNXXNX',
                     'NNXNXNXNNXCNNXNXNXXNX')

#get runhc_**X*X*X**XC**X*X*XX*X.bw
```

```
em.scores.atac.all = mclapply(hc.atac.cutmasks, function(cutmask) {
#    bw.paths = run.cutmask(cutmask, seqOutBias.args, sqcmd=seqOutBias.cmd, clean = FALSE, prefix = "runhc_")
    bw.plus = load.bigWig(paste0('runhc_', cutmask, '.bw'))
    bw.minus = load.bigWig(paste0('runhc_', cutmask, '.bw'))
    eval.cutmask(sites, bw.plus, bw.minus)
}, mc.cores = 20)


save(em.scores.atac.all, file = 'em.scores.atac.all.ATAC-kmer_optimization_CEM-C7_untreated_rep2_ATACbias_seq.Rdata')

hc.atac.cutmasks = factor(hc.atac.cutmasks, levels=hc.atac.cutmasks)

pdf("ATAC-kmer_optimization_CEM-C7_untreated_rep2_ATACbias_seq.pdf", useDingbats=FALSE, width=4, height=6)
dotplot(as.numeric(em.scores.atac.all) ~ hc.atac.cutmasks,
        pch = 19,
        cex =1,
        col = 'black',
        main = "Hill Climbing derived k-mer masks",
        xlab = 'Masked Positions',
#        ylim = c(0, 31900),
        scales=list(x=list(rot=45)),
        ylab = expression(paste(Sigma, ' SDs between PSWM positions')))
#for each set of TF PSWMs we sum the intensity of signal at each position, #then we take the standard deviation between positions.
#The final metric is a sum of these standard deviations.
dev.off()

#does figure 4 of this give the correct sequence bias?:
#https://www.biorxiv.org/content/10.1101/525808v1.full.pdf
#what about this one:
#https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1642-2

source('https://raw.githubusercontent.com/guertinlab/seqOutBias/master/docs/R/seqOutBias_functions.R')

#needed to update this with the latest fimo version output
parse.fimo <- function(file) {
    fimo.data = read.table(file, skip = 1, sep ='\t')
    res = fimo.data[, c(3,4,5,8,9,6, 2)]
    colnames(res) = c('chr', 'start', 'end', 'score', 'pval', 'strand', 'motif')
    return(res)
}


#THIS NEEDS WORK, LET ME KNOW IF THIS COMMENT IS STILL HERE WHEN YOU GET HERE

#need to move the relevant bw files to a folder and change extensino to .bigWig
all.composites.ATAC = cycle.fimo.new.not.hotspots(path.dir.fimo = '~/Desktop/atac_test', rand.rows = 200000,
                  path.dir.bigWig = '/Users/guertinlab/Desktop/atac_test/no_shift_bigWigs/', window = 30, exp = 'ATAC')

#runhc_NNXNXNXNNXCNNXNXNXXNX.bigWig
#runhc_XNXNXNXXNXCNXXNXXXXX.bigWig
#runhc_XXXXXXXXNXCNXXNXXXXX.bigWig
#runhc_XXXXXXXXXXCXXXXXXXXX.bigWig
########
all.composites.ATAC = cycle.fimo.new.not.hotspots(path.dir.fimo = '~/Desktop/atac_test',
                  path.dir.bigWig = '/Users/guertinlab/Desktop/atac_test/CEM-C7_untreated_rep2_PE1_plus_bigWigs_Atac_bias/final/',
                  window = 30, exp = 'ATAC')


all.composites.ATAC.bias = cycle.fimo.new.not.hotspots(path.dir.fimo = '/Users/guertinlab/Desktop/atac_test/pe1_plus',
                  path.dir.bigWig = '/Users/guertinlab/Desktop/atac_test/CEM-C7_untreated_rep2_PE1_plus_bigWigs_Atac_bias/final/',
                  window = 30, exp = 'ATAC')

composites.func.panels.naked.chromatin(all.composites.ATAC.bias,
                                       fact = paste('ATACbias', sep= ' '), summit = 'Motif',
                                       num = 24, col.lines = rev(c(rgb(0,0,1,1/2),
                                  rgb(0,0,0,1/2), rgb(1,1,0,1/2), rgb(1,0,1,1/2) )),
                                  fill.poly = rev(c(rgb(0,0,1,1/4), rgb(0,0,0,1/4),
                                                    rgb(1,1,0,1/4), rgb(1,0,1,1/4) )))

atac_fimo_bias
unique(all.composites.ATAC$cond)
all.composites.ATAC$cond = gsub("runhc_XXXXXXXXXXCXXXXXXXXX", "Raw", all.composites.ATAC$cond)
all.composites.ATAC$cond = gsub("runhc_XXXXXXXXNXCNXXNXXXXX", "Corrected", all.composites.ATAC$cond)

composites.func.panels.naked.chromatin(all.composites.ATAC,
                                       fact = paste('ATACbias_motifs', sep= ' '), summit = 'Motif',
                                       num = 24, col.lines = rev(c(rgb(0,0,1,1/2),
                                  rgb(0,0,0,1/2), rgb(1,1,0,1/2), rgb(1,0,1,1/2) )),
                                  fill.poly = rev(c(rgb(0,0,1,1/4), rgb(0,0,0,1/4),
                                                    rgb(1,1,0,1/4), rgb(1,0,1,1/4) )))
```

3.2 Testing with TFBS motifs

I need to update this, but it is largely lifted from seqOutBias. I think it is sufficient to load individual PSWM files to github and provide direct links to the files. Many of these TF binding sites are exhaustively characterized, so it is not necessary to show how we generated the PSWM files if they are just provided. Also, below there is a complication that we are using ATAC-seq data from cells and seeing how the profiles are flattened. I think this is a good test of the masks, but initially I support using the naked DNA data that we published in seqOutBias to optimize the mask on all genome-wide instances of a motif (FIMO) at some threshold, then go to published ATAC-seq data with corresponding ChIP-seq data (MAST) to exclusively focus on motifs within ATAC peaks to *test* the determined mask. I want to pick 20 motifs, with some AT-rich (like FoxA1 and HSF) and CG rich (like SP1 and CTCF). Choosing the 20 motifs wisely is important, because the next section takes a long time to run. I also want to generate a simple function in R that loads a minimal MEME PSWM file and outputs a PDF seqLogo, so that we don't have to use ceqLogo...this is easy, the minimal meme file needs to be loaded into R and parsed to look like `pswm.pe1.minus.trans` from a previous code chunk.

```
#This merged BAM can be the input for section 7.1 of the seqOutBias vignette
#I do not believe there is any need to separate the plus and minus reads from the BAM, but it is probably a good coherence check


#these chip-seq data come from either MCF7 or GM12878, so analysis is not precise at the moment.
#it is best to have binding data and accessibiltiy data from the same cells.
#I use genomic fimo below anyways, so it does not matter

url=http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeHaibTfbs/
wget ${url}wgEncodeHaibTfbsMcf7Elf1V0422111PkRep1.broadPeak.gz
wget ${url}wgEncodeHaibTfbsMcf7Gata3V0422111PkRep1.broadPeak.gz
wget ${url}wgEncodeHaibTfbsMcf7MaxV0422111PkRep1.broadPeak.gz
#wget ${url}wgEncodeHaibTfbsMcf7CtcfcV0422111PkRep1.broadPeak.gz

url=http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeSydhTfbs/
wget ${url}wgEncodeSydhTfbsGm12878Corestsc30189IggmusPk.narrowPeak.gz
wget ${url}wgEncodeSydhTfbsGm12878Ebf1sc137065StdPk.narrowPeak.gz
wget ${url}wgEncodeSydhTfbsGm12878Ctcfsc15914c20StdPk.narrowPeak.gz

url=http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeHaibTfbs/
wget ${url}wgEncodeHaibTfbsGm12878Sp1Pcr1xPkRep1.broadPeak.gz

wget http://hgdownload.cse.ucsc.edu/goldenPath/hg19/liftOver/hg19ToHg38.over.chain.gz
gunzip hg19ToHg38.over.chain.gz


for peak in *Mcf7*Rep1.broadPeak.gz
do
    name=$(echo $peak | awk -F"wgEncodeHaibTfbsMcf7" '{print $NF}' | awk -F"V0422111PkRep1.broadPeak.gz" '{print $1}')
    unz=$(echo $peak | awk -F".gz" '{print $1}')
    echo $name
    gunzip $peak
    echo $unz
    liftOver $unz hg19ToHg38.over.chain $name.hg38.broadPeak $name.hg38.unmapped.txt -bedPlus=6
    fastaFromBed -fi hg38.fa -bed $name.hg38.broadPeak -fo $name.hg38.fasta
    gzip *broadPeak
done

for peak in *Gm12878*Peak.gz
do
    name=$(echo $peak | awk -F"TfbsGm12878" '{print $NF}' | awk -F"." '{print $1}')
    unz=$(echo $peak | awk -F".gz" '{print $1}')
    echo $name
    gunzip $peak
    echo $unz
    liftOver $unz hg19ToHg38.over.chain $name.hg38.narrowPeak $name.hg38.narrow.unmapped.txt -bedPlus=6
    fastaFromBed -fi hg38.fa -bed $name.hg38.narrowPeak -fo $name.hg38.fasta
    gzip ${name}*Peak
done

#manually changing these names
mv Ctcfsc15914c20StdPk.hg38.fasta CTCF.hg38.fasta
mv Sp1Pcr1xPkRep1.hg38.fasta SP1.hg38.fasta
mv Ebf1sc137065StdPk.hg38.fasta EBF1.hg38.fasta
mv Corestsc30189IggmusPk.hg38.fasta REST.hg38.fasta


wget http://meme-suite.org/meme-software/Databases/motifs/motif_databases.12.12.tgz
tar -xvf motif_databases.12.12.tgz
head -9 motif_databases/JASPAR/JASPAR_CORE_2016_vertebrates.meme > header_meme_temp.txt
grep -i -A 14 'MOTIF MA0058.3 MAX' motif_databases/JASPAR/JASPAR_CORE_2016.meme > max_temp.txt
grep -i -A 16 'MOTIF MA0473.2 ELF1' motif_databases/JASPAR/JASPAR_CORE_2016.meme > elf1_temp.txt
grep -i -A 12 'MOTIF MA0037.2 GATA3' motif_databases/JASPAR/JASPAR_CORE_2016.meme > gata3_temp.txt
grep -i -A 23 'MOTIF MA0139.1 CTCF' motif_databases/JASPAR/JASPAR_CORE_2016.meme > CTCF_temp.txt
```

```
grep -i -A 25 'MOTIF MA0138.2 REST' motif_databases/JASPAR/JASPAR_CORE_2016.meme > REST_temp.txt
grep -i -A 18 'MOTIF MA0154.3 EBF1' motif_databases/JASPAR/JASPAR_CORE_2016.meme > EBF1_temp.txt
grep -i -A 15 'MOTIF MA0079.3 SP1' motif_databases/JASPAR/JASPAR_CORE_2016.meme > SP1_temp.txt

for i in *_temp.txt
do
    name=$(echo $i | awk -F"_temp" '{print $1}')
    cat header_meme_temp.txt $i > ${name}_minimal_meme.txt
done

rm *temp.txt

#fimo takes a while to run
for meme in *.hg38.fasta
do
    name=$(echo $meme | awk -F".hg38.fasta" '{print $1}')
    echo $name
    mast ${name}_minimal_meme.txt $meme -hit_list -mt 0.0005 > ${name}_mast.txt
    fimo --thresh 0.0001 --text ${name}_minimal_meme.txt hg38.fa > ${name}_fimo.txt
    grep -v chrM ${name}_fimo.txt > ${name}_noM_fimo.txt
    rm ${name}_fimo.txt
    mv ${name}_noM_fimo.txt ${name}_fimo.txt
    ceqlogo -i1 ${name}_minimal_meme.txt -o ${name}_logo.eps -N -Y
done
```

After all this, I believe we will fully understand Tn5 bias and how to correct it and the next step is to test it with the GM12878 ATAC and ChIP data, as we did in seqOutBias. We should use PSWM and ChIP-seq data for factors that were not used in the Hill Climbing optimization as a true test set. I have some experience using pymol for Sathyan's paper and it woudl be great to layer the understanding fromt eh genomics onto the Tn5 structure...

References

Chen S, Zhou Y, Chen Y, Gu J (2018). "fastp: an ultra-fast all-in-one FASTQ preprocessor." *Bioinformatics*, **34**(17), i884–i890.

Kodama Y, Shumway M, Leinonen R (2011). "The Sequence Read Archive: explosive growth of sequencing data." *Nucleic acids research*, **40**(D1), D54–D56.

Langmead B, Trapnell C, Pop M, Salzberg S (2009). "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome." *Genome Biology*, **10**(3), R25. ISSN 1465-6906. doi: 10.1186/gb-2009-10-3-r25. URL `http://genomebiology.com/2009/10/3/R25`.

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, *et al.* (2009). "The sequence alignment/map format and SAMtools." *Bioinformatics*, **25**(16), 2078–2079.

Martins AL, Walavalkar NM, Anderson WD, Zang C, Guertin MJ (2018). "Universal correction of enzymatic sequence bias reveals molecular signatures of protein/DNA interactions." *Nucleic acids research*.