Computational analysis of ChIP-seq and RNA-seq data

Michael J. Guertin

September 9, 2019

This vignette covers preliminary analysis of genomic binding and expression data. I believe that the best way for experimental biologists to learn bioinformatics is to generate their own data and start analyzing it.[1]

Contents

1 Retrieving Raw Sequencing Data **3**

2 Find the Appropriate Genome File **4**

3 Aligning Raw Data with Bowtie2 **4**
3.1 Build a set of genome index files with bowtie2 . . . 5
3.2 Align a single file with bowtie2 . . . 5
3.3 Align multiple files with bowtie2 . . . 5
3.4 Sam file format . . . 5

4 Calling Peaks with MACS2 **7**
4.1 Calling peaks for one file . . . 7
4.2 Calling peaks for multiple files . . . 7
4.3 Modifying peak files for uploading into the UCSC browser . . . 7
4.4 Visualize peaks and bedGraph files with the UCSC browser . . . 8
4.5 Select peaks that overlap in each replicate . . . 9

5 De novo motif analysis with MEME **11**
5.1 Manuipulating the summits file . . . 11
5.2 Selecting a subset of coordinates for MEME input . . . 11
5.3 Extracting the sequence for each genomic coordinate . . . 11
5.4 Running MEME . . . 12
5.5 Other MEME tools . . . 12

6 RNA-seq Analysis **13**
6.1 Aquiring data from ENCODE . . . 13
6.2 Aquiring gene annotations from UCSC . . . 13
6.3 Aligning with Tophat2 . . . 14
6.4 Counting Reads in Genes with HTSeq . . . 14
6.5 Converting files to upload into UCSC . . . 14
6.6 Differential expression with DESeq2 . . . 15

7 Post-class contact information **17**

List of Figures

1 GEO website. . . . 3
2 UCSC downloads. . . . 4
3 UCSC homepage. . . . 8

[1] address questions to Michael Guertin, guertin@virginia.edu

4 UCSC add custom tracks. 9
5 This screen shot of Hsp90 shows the peaks called in each replicate. The spurious 'peaks' suggests that the peak calling criteria were too relaxed. 9
6 The HSF1 motif looks nearly identical to previously published HSE (Vihervaara *et al.*, 2013). 12
7 MAplot from DESeq analysis. 16

1 Retrieving Raw Sequencing Data

Sequencing files will come off the machine in FASTQ format, but they are stored are SRA (short read archive) files in data bases such as GEO (Gene Expression Omnibus) (`http://www.ncbi.nlm.nih.gov/geo/`) (Figure 1). We will look at the Linquist's Lab's HSF data: GSE45852

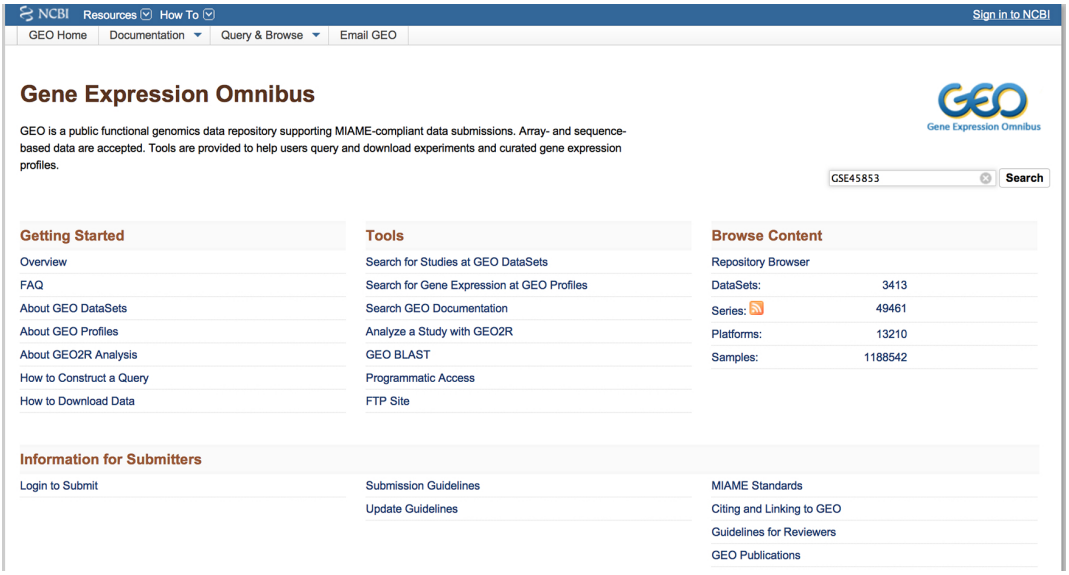


Figure 1: GEO website.

Drop the *.sra files into the 'kurs' folder. Convert the binary SRA files to FASTQ files with the sratool kit.

```
mkdir kurs
cd kurs
fastq-dump SRR822360.sra
fastq-dump SRR822356.sra
fastq-dump SRR822357.sra
```

Use the mv command to name the files something descriptive.

```
mv SRR822360.fastq Preimmune_MCF7_DMSO.fastq
mv SRR822356.fastq HSF1_MCF7_DMSO_rep1.fastq
mv SRR822357.fastq HSF1_MCF7_DMSO_rep2.fastq
```

Next we can look at the first 12 lines of the fastq files. Note that each sequencing read has four lines associated with it:
Line 1 begins with a '@' and is followed by a unique identifier
Line 2 is the raw nucleotide sequence.
Line 3 begins with a '+' and is optionally followed by the same sequence identifier.
Line 4 encodes the quality scores for the sequence in Line 2, and must contain the same number of symbols as letters in the sequence. This line is a measure of how confident the base is called correctly.

```
head -12 HSF1_MCF7_DMSO_rep1.fastq
```

As sequencing becomes cheaper, many labs use longer and longer reads, but for ChIP, we do not need reads longer than 50bp. In fact, sequencing quality deteriorates with longer reads–although now we can sequence at least 100bp of high quality sequence. I often trim the reads and assess the quality of the base calling using the fastx toolkit (`http://hannonlab.cshl.edu/fastx_toolkit/commandline.html`).

The fastx toolkit has many great tools that I suggest looking into–but I will not cover their usage in this vignette.
Free up space by zipping the files; this may take considerable time.

```
gzip *.fastq
```

2 Find the Appropriate Genome File

The first task is to retrieve the relevant genome from UCSC (Karolchik *et al.*, 2014): (`http://hgdownload.soe.ucsc.edu/downloads.html`) (Figure 2.). We want the hg38.fa.gz file. This is a zipped fasta file of the entire human genome.

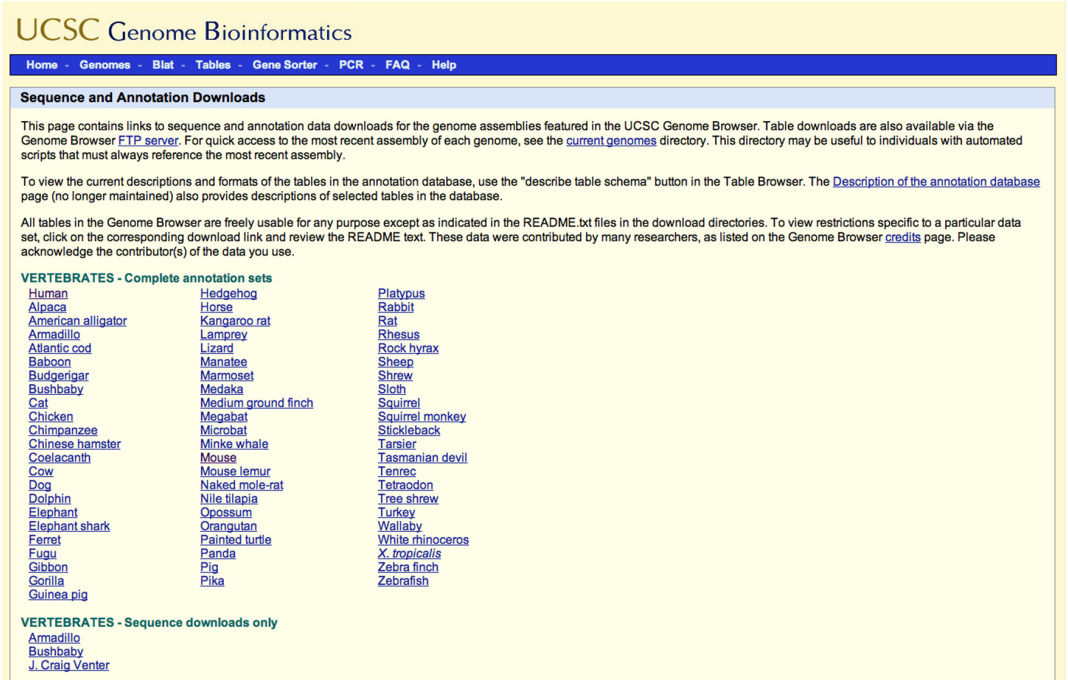


Figure 2: UCSC downloads.

```
cd ~/kurs
#one can use the following command on a Linux:
wget http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.fa.gz
#use curl on a Mac:
curl -OL http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.fa.gz

#unzip the file
gunzip hg38.fa.gz
```

Look at the format of a fasta file. Most of the sequence is either A,C,T, or G–but toward the telomeres there are many N's, due to the fact that assemblying repetitive DNA is non-trivial.

```
head hg38.fa
```

3 Aligning Raw Data with Bowtie2

Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters, and particularly good at aligning to relatively long (e.g. mammalian) genomes.

3.1 Build a set of genome index files with bowtie2

The first task is to build the genome index with bowtie2 (Langmead *et al.*, 2009) `http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml#the-bowtie2-build-indexer`. This only has to be performed once per genome.

```
cd ~/kurs
bowtie2-build hg38.fa hg38
```

3.2 Align a single file with bowtie2

Now we can align the fastq files to the indexed bowtie genomes using the bowtie2 command `http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml#the-bowtie2-aligner`.

The syntax below is the minimal syntax you would want to use to align single-end data. See the link above for full usage and options.

```
#how many processors does my Mac have?
sysctl hw.ncpu | awk '{print $2}'


## 8
```

I will leave one processor free by specifying the -p option as 7:

```
bowtie2 -p 7 -x hg38 -U HSF1_MCF7_DMSO_rep1.fastq.gz -S HSF1_MCF7_DMSO_rep1.sam
```

3.3 Align multiple files with bowtie2

As opposed to running each command individually, we can write a loop to cycle through the files in a directory and automatically name the ouput files.
The syntax for assigning the name variable does not require a full explanation. In short, the code splits each file on the '/' character, then takes the last string. This is useful if you are looping through files in a nested directory. The resultant string is split on the '.fastq.gz' string and the variable 'name' is assigned to the first string.

```
for fq in *.fastq.gz
do
  name=$(echo $fq | awk -F"/" '{print $NF}' | awk -F".fastq.gz" '{print $1}')
  echo $name
  bowtie2 -p 7 -x hg38 -U $fq -S $name.sam
done
```

3.4 Sam file format

Sam files retain all the information from the fastq files, but include additional information, including alignment coordinates (`http://samtools.github.io/hts-specs/SAMv1.pdf`). The header has all the chromosome size information from the hg39.fa file.

```
head HSF1_MCF7_DMSO_rep1.sam
```

The header is 457 lines, se we should also look at the read data using a combination of head and tail.

```
head -465 HSF1_MCF7_DMSO_rep1.sam | tail
```

Next convert the sam file to the compressed BAM format using samtools (Li *et al.*, 2009). The output files are in binary format, so you cannot see the file structure with head/tail.

```
for sam in *.sam
do
  name=$(echo $sam | awk -F"/" '{print $NF}' | awk -F".sam" '{print $1}')
  echo $name
  samtools view -bS $sam > $name.bam
done
```

4 Calling Peaks with MACS2

MACS captures the influence of genome complexity to evaluate the significance of enriched ChIP regions, and MACS improves the spatial resolution of binding sites through combining the information of both sequencing tag position and orientation. MACS can be easily used for ChIP-Seq data alone, or with control sample with the increase of specificity (Zhang *et al.*, 2008). The MACS documentation is found at: `https://github.com/taoliu/MACS/`.

4.1 Calling peaks for one file

I always recommend using a control for calling peaks. The best control is an IP with the same antibody, in cells that are depleted or knocked out for the factor (Guertin and Lis, 2010). Otherwise, a preimmune IP is second best. A mock IP or non-specific IgG is also acceptable. I strongly advise against using input for the background (control) data set (Auerbach *et al.*, 2009).

```
macs2 callpeak -t HSF1_MCF7_DMSO_rep1.bam -c Preimmune_MCF7_DMSO.bam -f BAM -n test -g hs
```

4.2 Calling peaks for multiple files

If the treatment files have the same control, you can loop through the files in the directory to call peaks. Note that you will call peaks in the control file as well if it is in the directory, but these files should not contain peaks.

```
ctrl=Preimmune_MCF7_DMSO.bam
for i in *bam
do
    name=$(echo $i | awk -F"/" '{print $NF}' | awk -F".bam" '{print $1}')
    echo $name
    macs2 callpeak -t $i -c $ctrl -f BAM --keep-dup 10 -n $name -g hs -B -m 10 100
done
```

4.3 Modifying peak files for uploading into the UCSC browser

MACS2 is still in development, so some of the features from earlier versions of MACS are not present. I like to put headers on my bedGraph and peaks files. This way, I can upload them into the UCSC genome browser.
This will put the header on the peaks file:

```
for narrow in *.narrowPeak
do
    name=$(echo $narrow | awk -F"/" '{print $NF}' | awk -F"_peaks" '{print $1}')
    echo $name
    touch  tempnarrow.txt
    echo "track type=narrowPeak name=$name.peaks" >> tempnarrow.txt
    cat tempnarrow.txt $narrow > $name.narrowPeak
    rm $narrow
    rm tempnarrow.txt
done
```

This will put the header on the bedGraph file:

```
for bdg in *treat_pileup.bdg
do
    name=$(echo $bdg | awk -F"/" '{print $NF}' | awk -F"_treat_pileup" '{print $1}')
    echo $name
    touch temp.txt
    echo "track type=bedGraph name=$name" >> temp.txt
    cat temp.txt $bdg > $name.bedGraph
    rm temp.txt
    gzip $name.bedGraph
done
```

As always, I suggest using a combination of head and tail to familiarize yourself with the file format associated with each fiel extension. MACS2 has several output files (*.bed, *.narrowPeak, *.bdg, *.broadPeak, *.r, etc).

4.4 Visualize peaks and bedGraph files with the UCSC browser

I prefer the UCSC browser to visual my data and for exporting publication-quality figures (`https://genome.ucsc.edu`) (Karolchik *et al.*, 2014). Click 'Genomes' in the upper left corner (Figure 3).

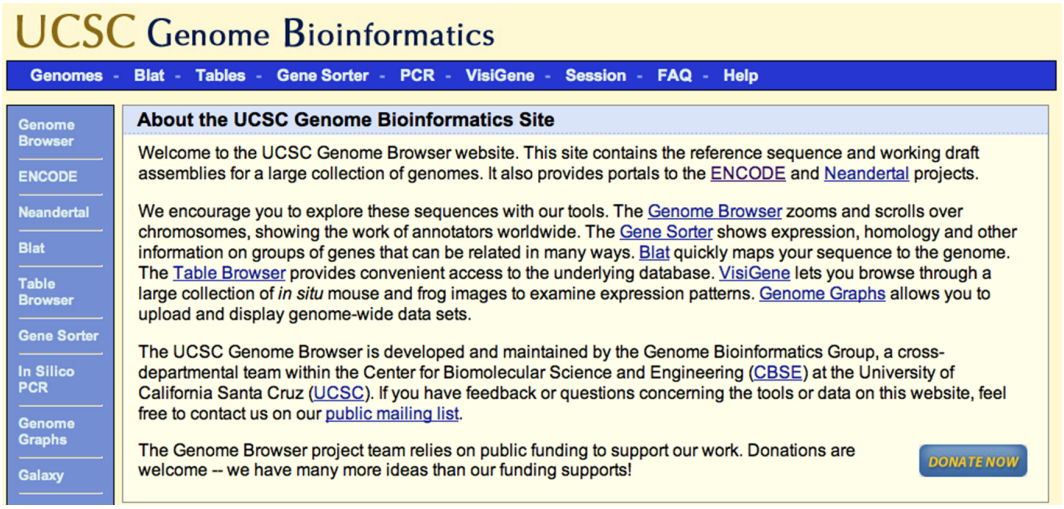


Figure 3: UCSC homepage.

Make sure you have the correct assembly, we are using the latest: hg38. Then click 'add custom tracks' (Figure 4. Use the GUI to navigate to the *.narrowPeaks and *.bedGraph.gz files. Upload each file. You probably want to register and save sessions if you have your own data. Then you can save sessions and you will not need to upload the data more than once.

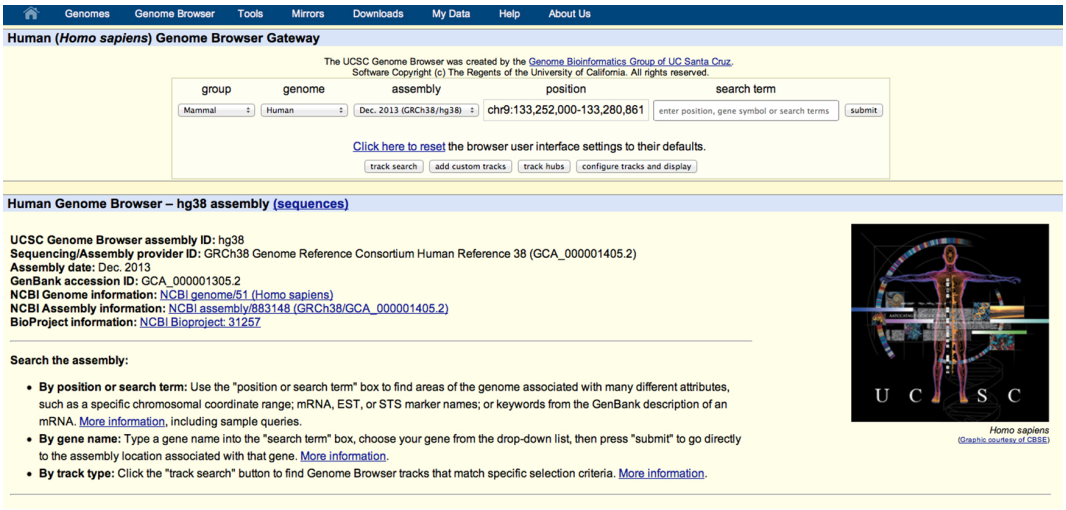


Figure 4: UCSC add custom tracks.

You can use the UCSC browser to scan your tracks and get an idea as to whether you peak-calling criteria were too stingent or relaxed (Figure 5). I like the 'continuous' tracks (bedGraph) to be in 'full' mode and the peaks in 'dense'. The default when uploading is dense, so you have to select full from the drop-down menu, if you want to change it. You can also manage your custom tracks and access UCSC tracks from the manage custom tracks link below the browser.

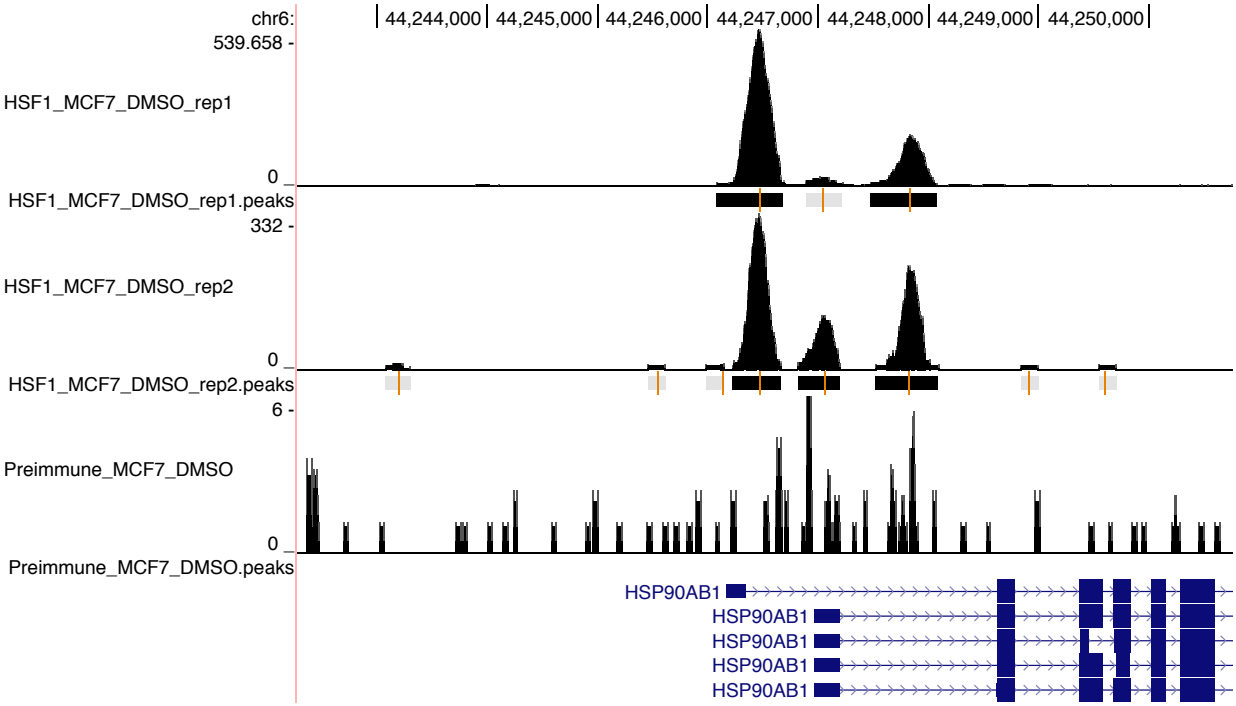


Figure 5: This screen shot of Hsp90 shows the peaks called in each replicate. The spurious 'peaks' suggests that the peak calling criteria were too relaxed.

4.5 Select peaks that overlap in each replicate

If you want a set of high confidence peaks, you can select the peak coordinates that overlap from each replicate. You will use the bedtools module (`http://bedtools.readthedocs.org`) to perform genomic arithmetic, such as the intersection of the two narrowPeaks files (`http://bedtools.readthedocs.org/en/latest/content/tools/intersect.html`).

```
intersectBed -a HSF1_MCF7_DMSO_rep1.narrowPeak -b HSF1_MCF7_DMSO_rep2.narrowPeak > HSF1_MCF7_DMSO_intersect.bed
```

Bedtools (Quinlan and Hall, 2010) is quite powerful, so explore the options of each command.

5 De novo motif analysis with MEME

A motif is a sequence pattern that occurs repeatedly in a group of related protein or DNA sequences. MEME (Bailey *et al.*, 2006) represents motifs as position-dependent letter-probability matrices which describe the probability of each possible letter at each position in the pattern. Individual MEME motifs do not contain gaps. Patterns with variable-length gaps are split by MEME into two or more separate motifs. MEME takes as input a group of DNA or protein sequences and outputs as many motifs as requested. MEME uses statistical modeling techniques to automatically choose the best width, number of occurrences, and description for each motif (`http://meme.nbcr.net/meme/`).

5.1 Manuipulating the summits file

Sequence-specific transcription factors tend to bind near the summit of ChIP-seq peaks. So we can focus on the region around the summit for de novo motif calling. This also works well because MEME handles smaller sequence inputs much more efficiently than larger inputs. In later examples we will subset the data so that MEME can run efficiently.

First we can reduce each peak to a 100bp window centered on the peak summit:

```
for i in *summits.bed
do
    name=$(echo $i | awk -F"/" '{print $NF}' | awk -F"_summits.bed" '{print $1}')
    echo $name
    awk '{$2=$2-50; print}' $i | awk -v OFS='\t' '{$3=$3+50; print}' > $name.window.bed
done
```

We will take the intersect of the *summits.window.bed files for a set of high confidence peak summits for MEME input.

```
intersectBed -a HSF1_MCF7_DMSO_rep1.window.bed -b HSF1_MCF7_DMSO_rep2.window.bed > HSF1_MCF7_shared_summits.bed
#next we can select those that overlap at least 20bp
awk '($3 -$2) > 20 {print $0}' HSF1_MCF7_shared_summits.bed > HSF1_MCF7_shared20bp_summits.bed
```

5.2 Selecting a subset of coordinates for MEME input

I usually implement de novo motif analysis to see if my ChIP 'worked', I first take a random sample. If I get the motif I expect, then I am reluctantly happy. But hope is not lost if you get a jibberish motif. Perhaps your criteria for peak calling were too relaxed, so you may have to compromise sensitivity for specificity. So, you can always take a non-random sample of the summits file, by sorting them based on summit intensity.

```
#sort randomly, and take the first 1000 rows
perl -MList::Util -e 'print List::Util::shuffle <>' HSF1_MCF7_shared20bp_summits.bed | head -1000 > HSF1_MCF7.ran.1K.bed

#sort based on summit intensity, take 1000 rows
sort -k5 -r -n HSF1_MCF7_shared20bp_summits.bed | head -1000 > HSF1_MCF7_shared.top.1K.bed
```

5.3 Extracting the sequence for each genomic coordinate

Now we can use bedtools again to extract the sequences from the original 'hg38.fa' file. The command is fastaFromBed (`http://bedtools.readthedocs.org/en/latest/content/tools/getfasta.html`).

```
fastaFromBed -fi hg38.fa -bed HSF1_MCF7.ran.1K.bed -fo HSF1_MCF7_shared.ran.1K.fasta
fastaFromBed -fi hg38.fa -bed HSF1_MCF7_shared.top.1K.bed -fo HSF1_MCF7_shared.top.1K.fasta

#can you write a loop to do the same thing?
```

5.4 Running MEME

For we will run MEME on the random 1000 peaks and the 1000 most intense peaks (Bailey *et al.*, 2006).

```
meme HSF1_MCF7_shared.ran.1K.fasta -oc HSF1_MCF7_shared.ran.1K.meme -dna -nmotifs 2 -minw 5 -maxw 15 -revcomp
meme HSF1_MCF7_shared.top.1K.fasta -oc HSF1_MCF7_shared.top.1K.meme -dna -nmotifs 2 -minw 5 -maxw 15 -revcomp
```

The meme.html output provides a nice visual output from the analysis. You can compare the motifs to the expected Heat Shock Element (Guertin and Lis, 2010; Vihervaara *et al.*, 2013). This is ad hoc, but a subsequent section will go through systematic ways to figure out what motif resembles your de novo motif. Given these two results, would you change any of the MACS parameters and options when calling peaks originally?

5.5 Other MEME tools

We can find all the instances of the motif in the genome using FIMO, a tool in the MEME suite. There are many clever things one can do with bedtools and a set of motifs. You can determine the fraction of peaks that have a motif, get a list of motifs that are not within peaks, etc.

```
fimo -oc HSF1_MCF7_shared.top.1K.fimo HSF1_MCF7_shared.top.1K.meme/meme.txt hg38.fa
```

Next we can query the motif against a known database using TOMTOM. The first task is to get the data base. You can download many from the MEME suite, but I will use one that I made (Guertin *et al.*, 2014), because I know that it contains the heat shock element.

```
#get the motif data base
#Linux:
wget http://press.endocrine.org/doi/suppl/10.1210/me.2014-1130/suppl_file/me-14-1130-2.txt

#use curl on a Mac:
curl -OL http://press.endocrine.org/doi/suppl/10.1210/me.2014-1130/suppl_file/me-14-1130-2.txt

tomtom -no-ssc -oc HSF1_MCF7_shared.top.1K.tomtom -thresh 0.01 -eps HSF1_MCF7_shared.top.1K.meme/meme.txt me-14-1130-2.txt
```

The 'tomtom.html' file has a nice GUI output. You can export publishable seqLogo graphics (Figure 6), find where the motif was originally published, etc.

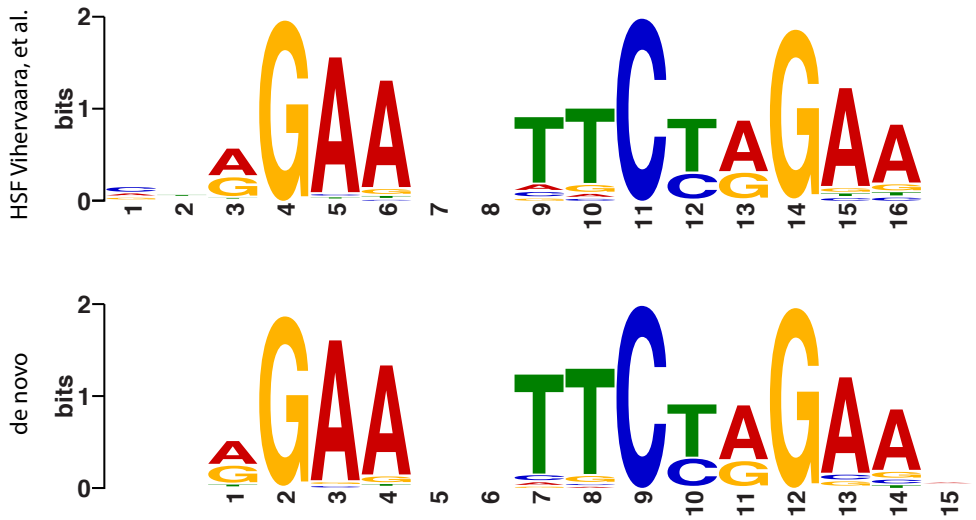


Figure 6: The HSF1 motif looks nearly identical to previously published HSE (Vihervaara *et al.*, 2013).

6 RNA-seq Analysis

In this section we will retrieve RNA-seq ENCODE data via UCSC, align to the genome and splice junctions with Tophat, count reads in genes with HTSeq, and assess differential expression with the DESeq package in R.

6.1 Aquiring data from ENCODE

The ENCODE consortium generated thousands of molecular genomic data sets that are publically available (`https://www.encodedcc.org`). UCSC houses most of the raw and processed data for download, or it can be accessed from ENCODE directly. I prefer the UCSC layout (`https://genome.ucsc.edu/cgi-bin/hgFileUi?db=hg19&g=wgEncodeHaibRnaSeq`). I already copied the URL for the data we will use, but I encourage you to use different data and navigate the UCSC or ENCODE websites yourself.
Tophat2 takes a long time to run, so in the interest of time, we will reduce the file size for the inputs and run tophat as a proof-of-principle. The complete alignments were performed already and can be accessed in the 'kurs' folder. Why do we want to subset the data so that the number of lines is a multiple of four?

```
#curl does not permit 'wildcards': *
curl -OL  http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeHaibRnaSeq/wgEncodeHaibRnaSeqT47dDm002p4hRawDataRep1.fastq.gz
curl -OL  http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeHaibRnaSeq/wgEncodeHaibRnaSeqT47dDm002p4hRawDataRep2.fastq.gz
curl -OL  http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeHaibRnaSeq/wgEncodeHaibRnaSeqT47dEstradia4hRawDataRep1.fastq.gz
curl -OL  http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeHaibRnaSeq/wgEncodeHaibRnaSeqT47dEstradia4hRawDataRep2.fastq.gz

gunzip wg*fastq.gz

for i in wg*.fastq
do
    name=$(echo $i | awk -F"wgEncodeHaibRnaSeq" '{print $NF}' | awk -F".fastq" '{print $1}')
    echo $name
    head -400000 $i > $name.subset.test.fastq
    gzip $name.subset.test.fastq
done
```

6.2 Aquiring gene annotations from UCSC

We already built indexed genomes for bowtie2, but now we will need to acquire and index either a GTF of GFF file. I prefer to use the Ensembl genes GTF files (`http://useast.ensembl.org/info/data/ftp/index.html` or GENCODE (`http://www.gencodegenes.org/releases/current.html`).

```
x = read.table('Homo_sapiens.GRCh38.81.gtf', sep ='\t')
x[,1] = paste('chr', x[,1], sep='')
write.table(x, file ='Homo_sapiens.GRCh38.81.chr.gtf', quote=FALSE, sep = '\t', row.names=FALSE, col.names=FALSE)

#or
system("wget ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_human/release_23/gencode.v23.annotation.gtf.gz")
system("gunzip gencode.v23.annotation.gtf.gz")
```

6.3 Aligning with Tophat2

Tophat (Trapnell *et al.*, 2009) is a fast splice junction mapper for RNA-Seq reads. It aligns RNA-Seq reads to mammalian-sized genomes using the ultra high-throughput short read aligner Bowtie, and then analyzes the mapping results to identify splice junctions between exons. Tophat information is found here: `http://ccb.jhu.edu/software/tophat/index.shtml`.

Recall, that we are working with truncated data, but the following code is written for the raw data. You need to **modify** the code below so that it uses the truncated files and names the output directory appropriately.

```
for i in wg*.fastq.gz
do
    name=$(echo $i | awk -F"wgEncodeHaibRnaSeq" '{print $NF}' | awk -F".fastq.gz" '{print $1}')
    echo $name
    tophat2 -o $name.tophat -G Homo_sapiens.GRCh38.81.chr.gtf -p 3 hg38 $i
done
```

The following code can be used for paired-end strand specific RNA-seq data. Unfortunately, I am not aware of great tools to generate strand-specific "browser" files (bigWig or bedGraph), so I came up with an ad hoc method for getting full coverage in "browser form from stranded paired-end RNA-seq data. We align the entire file, but also the first and second read for the strand-specific bedGraphs, used later. The "both" paired-end aligned bam file will be used to count tags in genomic features, the 'pe1' and 'pe2' files are generated as a means to create files that can be uploaded into a browser.

```
for i in *_PE.1.fastq.gz
do
    name=$(echo $i | awk -F"/" '{print $NF}' | awk -F"_PE." '{print $1}')
    echo $name
    tophat2 -o $name.both.tophat --library-type fr-firststrand -G Homo_sapiens.GRCh38.81.chr.gtf -p 15 hg38 $i ${name}_PE.2.fastq.gz
    tophat2 -o $name.pe1.tophat --library-type fr-firststrand -G Homo_sapiens.GRCh38.81.chr.gtf -p 15 hg38 $i
    tophat2 -o $name.pe2.tophat --library-type fr-firststrand -G Homo_sapiens.GRCh38.81.chr.gtf -p 15 hg38 ${name}_PE.2.fastq.gz
done
```

6.4 Counting Reads in Genes with HTSeq

HTSeq (Anders *et al.*, 2014) is a Python package that provides infrastructure to process data from high-throughput sequencing assays (`http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html`). This process also takes a bit of time, so the output text files were already generated. However, I encourage you to run this on your own time, perhaps with the truncated tophat2 alignments.

```
for i in *.tophat/accepted_hits.bam
do
    name=$(echo $i |  awk -F".tophat" '{print $1}')
    echo $name
    samtools sort -n accepted_hits.bam $name.accepted_hits.sorted
    htseq-count -f bam --stranded=no $name.accepted_hits.sorted.bam Homo_sapiens.GRCh38.81.chr.gtf > $name.gene.counts.txt
done
```

For the strand-specific paired-end RNA-seq you can use the tophat output that results from aligning both read pairs for counting with HTSeq.

```
for i in *both.tophat/accepted_hits.bam
do
    name=$(echo $i |  awk -F".both.tophat" '{print $1}')
    echo $name
    samtools sort -n $i $name.accepted_hits.sorted
    htseq-count -f bam --order=name --stranded=reverse $name.accepted_hits.sorted.bam Homo_sapiens.GRCh38.81.chr.gtf > $name.gene.counts.txt
done
```

6.5 Converting files to upload into UCSC

You can also upload RNA-seq data to view in UCSC, see section 4.4. You will need a file that has the chromosome sizes for your genome assembly. You can use the UCSC tool 'fetchChromSizes' (`http://`

hgdownload.cse.ucsc.edu/admin/exe/) to generate the file for the chromosome sizes, but I uploaded the 'hg38.chrom.sizes' file here: https://drive.google.com/file/d/0ByeQUT6Ge3xrandKN3o5Wng2aEE/edit?usp=sharing. You will use the sorted BAM file from section 6.4. Then use the bedtools function 'genomeCoverageBed' with the following options for RNA-seq data. The zipped bedGraph can be uploaded to UCSC.

```
for i in *.accepted_hits.sorted.bam
do
    name=$(echo $i | awk -F"/" '{print $NF}' | awk -F".accepted" '{print $1}')
    echo $name
    genomeCoverageBed -bg -split -trackline -trackopts name=$name -ibam $i -g hg38.chrom.sizes > $name.bedGraph
    gzip $name.bedGraph
done
```

If you have paired-end data you may want to normailze by library depth and average replicate data. Above, you aligned each end (pe1 and pe2) separtely. This script normalizes by coverage per 10 million reads, combines paired-end data, and normalizes replicates. There are many intermediate files that I usually delete within this script, but I left out those 'rm' commands so you can troubleshoot and view intermediate files. The python I wrote to normalize can be found at https://drive.google.com/open?id=0ByeQUT6Ge3xrMkl4c0M1WDNPT1E.

```
for i in *pe1.tophat/accepted_hits.bam
do
    name=$(echo $i |  awk -F".pe1.tophat" '{print $1}')
    echo $name
    samtools sort $i $name.pe1.accepted_hits.sorted
    samtools sort $name.pe2.tophat/accepted_hits.bam $name.pe2.accepted_hits.sorted
    normpe1=$(head -3 $name.pe1.tophat/align_summary.txt | tail -1 | awk '{ print $3 }'|bc)
    scalepe1=$(python -c "print 10000000.0/$normpe1")
    normpe2=$(head -3 $name.pe2.tophat/align_summary.txt | tail -1 | awk '{ print $3 }'|bc)
    scalepe2=$(python -c "print 10000000.0/$normpe2")
    echo $normpe1
    echo $scalepe1
    echo $normpe2
    echo $scalepe2
    genomeCoverageBed -bg -strand + -scale $scalepe1 -split -ibam $name.pe1.accepted_hits.sorted.bam -g hg38.chrom.sizes > $name.pe1.minus.bedGraph
    genomeCoverageBed -bg -strand - -scale $scalepe1 -split -ibam $name.pe1.accepted_hits.sorted.bam -g hg38.chrom.sizes > $name.pe1.plus.bedGraph
    genomeCoverageBed -bg -strand - -scale $scalepe2 -split -ibam $name.pe2.accepted_hits.sorted.bam -g hg38.chrom.sizes > $name.pe2.minus.bedGraph
    genomeCoverageBed -bg -strand + -scale $scalepe2 -split -ibam $name.pe2.accepted_hits.sorted.bam -g hg38.chrom.sizes > $name.pe2.plus.bedGraph
    bedGraphToBigWig $name.pe2.plus.bedGraph hg38.chrom.sizes $name.pe2.plus.bigWig
    bedGraphToBigWig $name.pe2.minus.bedGraph hg38.chrom.sizes $name.pe2.minus.bigWig
    bedGraphToBigWig $name.pe1.plus.bedGraph hg38.chrom.sizes $name.pe1.plus.bigWig
    bedGraphToBigWig $name.pe1.minus.bedGraph hg38.chrom.sizes $name.pe1.minus.bigWig
    bigWigMerge $name.pe2.plus.bigWig $name.pe1.plus.bigWig $name.plus.bedGraph
    bigWigMerge $name.pe2.minus.bigWig $name.pe1.minus.bigWig $name.minus.bedGraph
    touch temp.txt
    echo "track type=bedGraph name=$name.plus color=255,0,0 visibility=2" >> temp.txt
    cat temp.txt $name.plus.bedGraph > $name.plus.temp.bedGraph
    rm temp.txt
    touch temp.txt
    echo "track type=bedGraph name=$name.minus color=0,0,255 visibility=2" >> temp.txt
    cat temp.txt $name.minus.bedGraph > $name.minus.temp.bedGraph
    rm temp.txt
    rm $name.plus.bedGraph
    rm $name.minus.bedGraph
    mv $name.plus.temp.bedGraph $name.plus.bedGraph
    mv $name.minus.temp.bedGraph $name.minus.bedGraph
    python normalize_bedGraph.py -i $name.plus.bedGraph -o $name.plus.norm.bedGraph -n 0.5
    python normalize_bedGraph.py -i $name.minus.bedGraph -o $name.minus.norm.bedGraph -n 0.5
    gzip $name.plus.norm.bedGraph
    gzip $name.minus.norm.bedGraph
done
```

6.6 Differential expression with DESeq2

DESeq (Anders and Huber, 2010; Love *et al.*, 2014) is an R library with the tools for differential gene expression analysis based on the negative binomial distribution (http://www.bioconductor.org/packages/release/bioc/html/DESeq2.html).
First we need to load the DESeq2 library and my favorite graphics library, lattice (Sarkar, 2008). Then I define the plot function. Refer to the DESeq2 vignette (http://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.pdf) for an explanation of the processes and statistics. Then plot the data (Figure 7).

```
#load libraries
library(DESeq2);
library(lattice);

#define plotting function
plot.ma.lattice <- function(ma.df, filename = 'file.name', p = 0.01, title.main = "Differential Expression",log2fold =0.5) {;
   pdf(paste("MA_plot", filename, ".pdf", sep=''), width=3.83, height=3.83);
   print(xyplot(ma.df$log2FoldChange ~ log(ma.df$baseMean, base=10),
                groups=(ma.df$padj < p & abs(ma.df$log2FoldChange) > log2fold & !is.na(ma.df$padj)),
```

```
                col=c("black","red"), main=title.main, scales="free", aspect=1, pch=20, cex=0.5,
                ylab=expression("log"[2]~"RNA fold change"), xlab=expression("log"[10]~"Mean of Normalized Counts"),
                par.settings=list(par.xlab.text=list(cex=1.1,font=2), par.ylab.text=list(cex=1.1,font=2))));
   dev.off()
}

#deseq2 processes
directory = "~/kurs";
htseq.files = grep("gene.counts",list.files(directory),value=TRUE);
htseq.files;
sample.conditions = c("untreated","untreated", "treated","treated");
counts.table = data.frame(sampleName=htseq.files, fileName=htseq.files, condition=sample.conditions);
counts.table;
deseq.counts.table = DESeqDataSetFromHTSeqCount(sampleTable=counts.table, directory=directory, design=~condition);

#The levels in colData are important because they are used in the fold change calculations;
#set untreated first so that the direction of the change makes sense.
colData(deseq.counts.table)$condition<-factor(colData(deseq.counts.table)$condition, levels=c("untreated","treated"));

dds = DESeq(deseq.counts.table);
res = results(dds);
res = res[order(res$padj),];
head(res);

#plot figure
plot.ma.lattice(res, filename = 'T47-D_E2', p=0.1, title.main = 'Differential Expression');
```

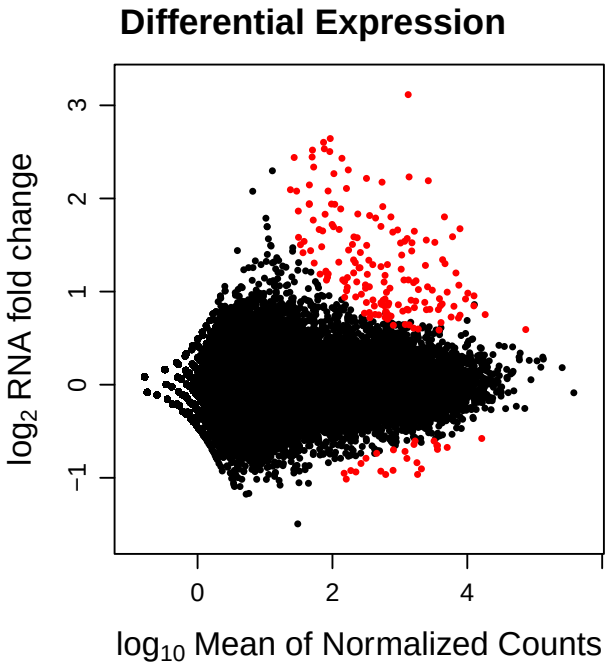


Figure 7: MAplot from DESeq analysis.

These genes all have Ensembl gene indentifiers, but it is useful to have gene names. To retrieve a 'key' to convert as the Ensembl ID to gene name do the following.

```
ensembl.all = read.table('~/kurs/Homo_sapiens.GRCh38.81.chr.gtf', sep='\t', header =F);
ensembl.gene.names = data.frame(sapply(strsplit(sapply(strsplit(as.character(ensembl.all[,9]),'gene_name '), "[", 2), ";"), "[", 1));
ensembl.id.names = data.frame(sapply(strsplit(sapply(strsplit(as.character(ensembl.all[,9]),'gene_id '), "[", 2), ";"), "[", 1));

ensembl.code = cbind(ensembl.gene.names, ensembl.id.names);
ensembl.code = ensembl.code[!duplicated(ensembl.code[,2]),];
rownames(ensembl.code) = ensembl.code[,2];
colnames(ensembl.code) = c('gene', 'id');

res.names = merge(res, ensembl.code, by="row.names", all.x=F);
res.names = res.names[,c(1:8)];
```

Next we may want to select specific rows. We can seach our favorite gene, or find genes that pass our adjusted p-value threshold. Note, that all NA values will be returned unless we specify '!is.na()'.

```
esr1.gene = res.names[res.names$gene == 'ESR1',];
print(esr1.gene);
low.padj = res.names[res.names$padj < 0.1 & !is.na(res.names$padj),];
print(low.padj);
```

7 Post-class contact information

Please email me with any questions that are specific to this tutorial, or if you find that something could be clearer–or even if you find that something is misspelled.

I would also be happy to help you analyze your own data, so feel free to email me with general questions.

I am compiling a course focusing on analyzing genomic data in more depth using R, email me if you are interested.

If you are share an interest in transcription, chromatin, or cancer biology and you want to generate and analyze your own molecular genomics data, contact me about potential openings in the Guertin lab at the University of Virginia.

guertin@virginia.edu

References

Anders S, Huber W (2010). "Differential expression analysis for sequence count data." *Genome biol*, **11**(10), R106.

Anders S, Pyl PT, Huber W (2014). "HTSeq–A Python framework to work with high-throughput sequencing data." *bioRxiv*.

Auerbach RK, Euskirchen G, Rozowsky J, Lamarre-Vincent N, Moqtaderi Z, Lefrançois P, Struhl K, Gerstein M, Snyder M (2009). "Mapping accessible chromatin regions using Sono-Seq." *Proceedings of the National Academy of Sciences*, **106**(35), 14926–14931.

Bailey TL, Williams N, Misleh C, Li WW (2006). "MEME: discovering and analyzing DNA and protein sequence motifs." *Nucleic acids research*, **34**(suppl 2), W369–W373.

Guertin MJ, Lis JT (2010). "Chromatin landscape dictates HSF binding to target DNA elements." *PLoS genetics*, **6**(9), e1001114. URL `http://www.plosgenetics.org/article/info%3Adoi%2F10.1371%2Fjournal.pgen.1001114`.

Guertin MJ, Zhang X, Coonrod SA, Hager GL (2014). "Transient ER binding and p300 redistribution support a squelching mechanism for E2-repressed genes." *Molecular Endocrinology*. URL `http://press.endocrine.org/doi/abs/10.1210/me.2014-1130`.

Karolchik D, Barber GP, Casper J, Clawson H, Cline MS, Diekhans M, Dreszer TR, Fujita PA, Guruvadoo L, Haeussler M, *et al.* (2014). "The UCSC genome browser database: 2014 update." *Nucleic acids research*, **42**(D1), D764–D770.

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.

Love MI, Huber W, Anders S (2014). "Moderated estimation of fold change and dispersion for RNA-Seq data with DESeq2." *bioRxiv*.

Quinlan AR, Hall IM (2010). "BEDTools: a flexible suite of utilities for comparing genomic features." *Bioinformatics*, **26**(6), 841–842.

Sarkar D (2008). *Lattice: Multivariate Data Visualization with R*. Springer, New York. ISBN 978-0-387-75968-5, URL `http://lmdvr.r-forge.r-project.org`.

Trapnell C, Pachter L, Salzberg SL (2009). "TopHat: discovering splice junctions with RNA-Seq." *Bioinformatics*, **25**(9), 1105–1111.

Vihervaara A, Sergelius C, Vasara J, Blom MA, Elsing AN, Roos-Mattjus P, Sistonen L (2013). "Transcriptional response to stress in the dynamic chromatin environment of cycling and mitotic cells." *Proceedings of the National Academy of Sciences*, **110**(36), E3388–E3397.

Zhang Y, Liu T, Meyer CA, Eeckhoute J, Johnson DS, Bernstein BE, Nusbaum C, Myers RM, Brown M, Li W, *et al.* (2008). "Model-based analysis of ChIP-Seq (MACS)." *Genome Biol*, **9**(9), R137.