TITLE: Identification of histone mark location using ChiP-seq
COURSE: MED263, "Bioinformatics Applications to Human Disease"
PREPARER: John Doe (jdoe@ucsd.edu)

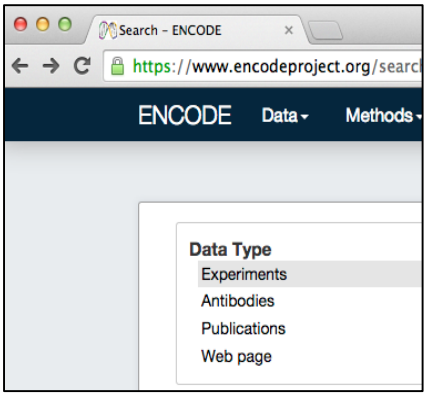
1) INTRODUCTION

In this practical, you are going to use linux command lines tools and web-service applications to identify histone mark location from Chromatin ImmunoPrecipitation and Sequencing (ChiP-Seq) data. The use of ChIP-Seq in histone modification is popular to delineate gene regulation mechanism. You will use a public ChiP-Seq data from Encyclopedia of DNA Elements (ENCODE). After this tutorial, you should be able to

- search and download ChIP-Seq data from ENCODE project
- conduct QC diagnosis and filtering for raw FASTQ data
- align short reads to a reference genome
- call enriched peaks
- annotate peak regions
- visualize ChiP-Seq findings on UCSC Genome Browser
- perform data manipulation with basic commandline in a linux shell

2) DATA

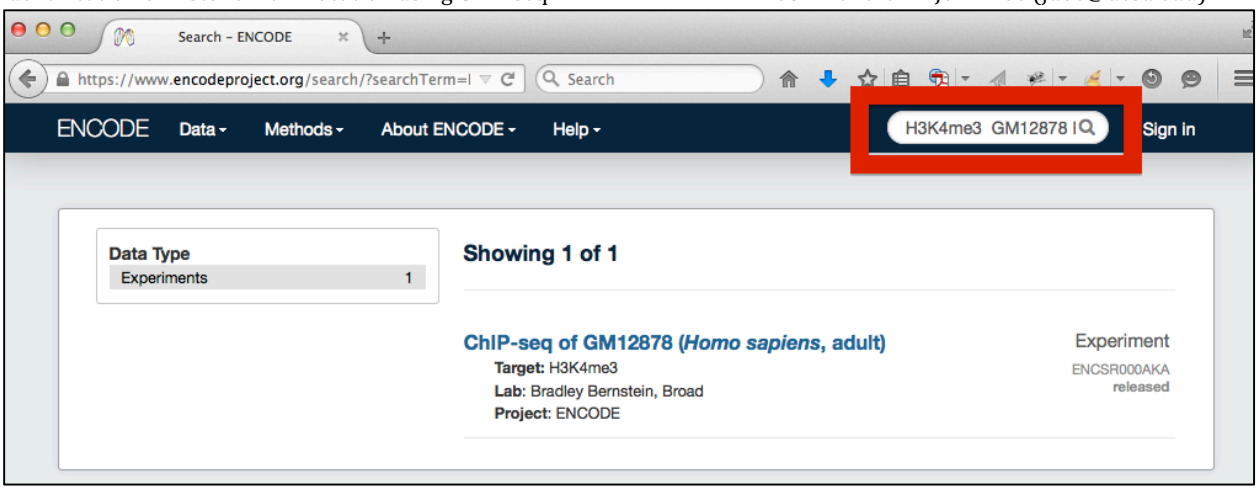
Open a browser, go to ENCODE Project web-site (https://www.encodeproject.org). On its top right window, type '**chip-seq**' in a search box. Then the number of retrieved records will show up on the top and breakdown of counts per Data Type (Experiments/Antibodies/Publication/Web page) on the left such as Figure X.



Q1. How many chip-seq experiments are there in total in ENCODE Project web-site? And what are the numbers of 'chip-seq' records records per-data type for Experiments, Antibodies, Publication, and Web page?

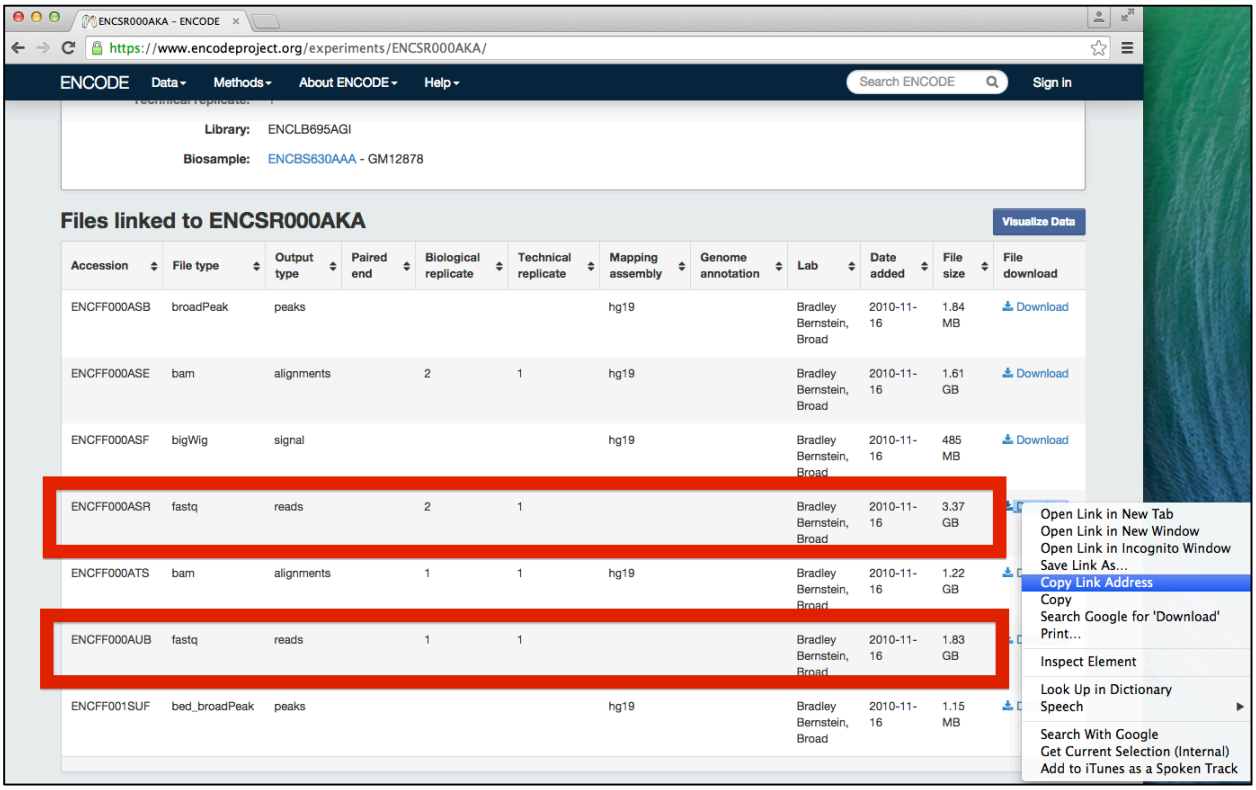
A1. 2872; Experiments=2467,Publications=229, Antibodies=150, and Web page=13.

In a search box, type four key words '**H3K4me3 GM12878 Bernstein Broad**' as a screenshot below.



Click the link of experiment,<ChIP-seq of GM12878 (Homo sapiens, adult)> to retrieve the experiment summary for accession ENCSR000AKA. Scroll down to find to two FASTQ files; replicate1 and replicate2.

For each FASTQ file, right-mouse click Download to copy a link address to FASQT file (red rectangle).

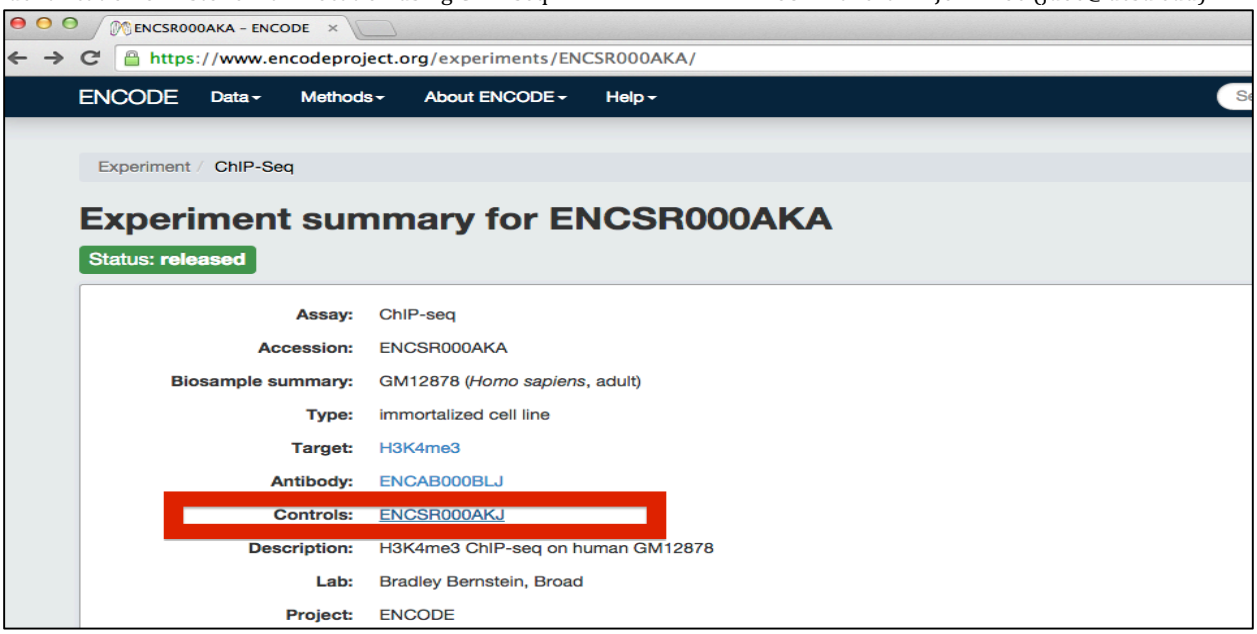


Then paste each link to a terminal window after typing 'wget' command in the terminal.

```
## download two antibody chip-Seq samples from ENCODE; replicate1 and replicate2
$ wget https://www.encodeproject.org/files/ENCFF000AUB/@@download/ENCFF000AUB.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF000ASR/@@download/ENCFF000ASR.fastq.gz
```

In addition to antibody chip-seq data, you also need to download sequencing control assays to be used in background noise elimination for reliable identification of enriched region in ChiP-Seq (need a reference here).

To obtain controls, scroll up and click the companion control assay with accession ID '**ENCSR000AKJ**' (red rectangle). The controls also have two replicates.



Similarly as in antibody chip-seq data, copy and paste links to two FASTQ files.

```
## download two control chip-seq samples; replicate1 and replicate2
$ wget https://www.encodeproject.org/files/ENCFF000ARK/@@download/ENCFF000ARK.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF000ARO/@@download/ENCFF000ARO.fastq.gz
```

Next extract compressed files and rename those to more intuitive names.

```
## extract all compressed files;  '*.gz' means all files whose names end with '.gz'
$ gunzip *.gz

### rename files to more intuitive names
$ mv ENCFF000ASR.fastq HistoneRep1.raw.fastq
$ mv ENCFF000AUB.fastq HistoneRep2.raw.fastq
$ mv ENCFF000ARK.fastq HistoneCtlRep1.raw.fastq
$ mv ENCFF000ARO.fastq HistoneCtlRep2.raw.fastq
```

An accurate data analysis starts from the right data. Be sure to check the md5 hash value of downloaded FASTQ files. You can easily get md5 hash value using '**md5sum**' command in a terminal. Below are md5 values of four FASTQ files.

```
## check md5sum values of FASTQ files
$ md5sum Histone*.raw.fastq
65bef62327caf739f0b1c2494d7a9d30  HistoneCtlRep1.raw.fastq
b5de39bd1fa91fbbf0a05a6f91f6b3b9  HistoneCtlRep2.raw.fastq
bed70332600ec3a2b4ffa280abddf1fd  HistoneRep1.raw.fastq
c51c6eeea2849620ed0c502f3fdf55dd  HistoneRep2.raw.fastq
```

Q2. Do your downloaded FASTQ files have identical md5 values as above? Use '**md5sum**' command. (If you happened to have different md5 sum value, do not move on to next steps as you might get incorrect answers in all downstream analysis. Try downloading raw data again.)

A2. Yes, all my FASTQ files had identical md5 values as in the tutorial.

```
## check md5sum values of FASTQ files
$ md5sum Histone*.raw.fastq
65bef62327caf739f0b1c2494d7a9d30  HistoneCtlRep1.raw.fastq
b5de39bd1fa91fbbf0a05a6f91f6b3b9  HistoneCtlRep2.raw.fastq
bed70332600ec3a2b4ffa280abddf1fd  HistoneRep1.raw.fastq
c51c6eeea2849620ed0c502f3fdf55dd  HistoneRep2.raw.fastq
```

3) QUALITY CONTROL

We will use FASTX-Toolkit (http://hannonlab.cshl.edu/fastx_toolkit) for quality control of FASTQ files. It is a collection of command line tools for FASTQ file preprocessing.

Create FASTQ quality statistics report for each FASTQ file.

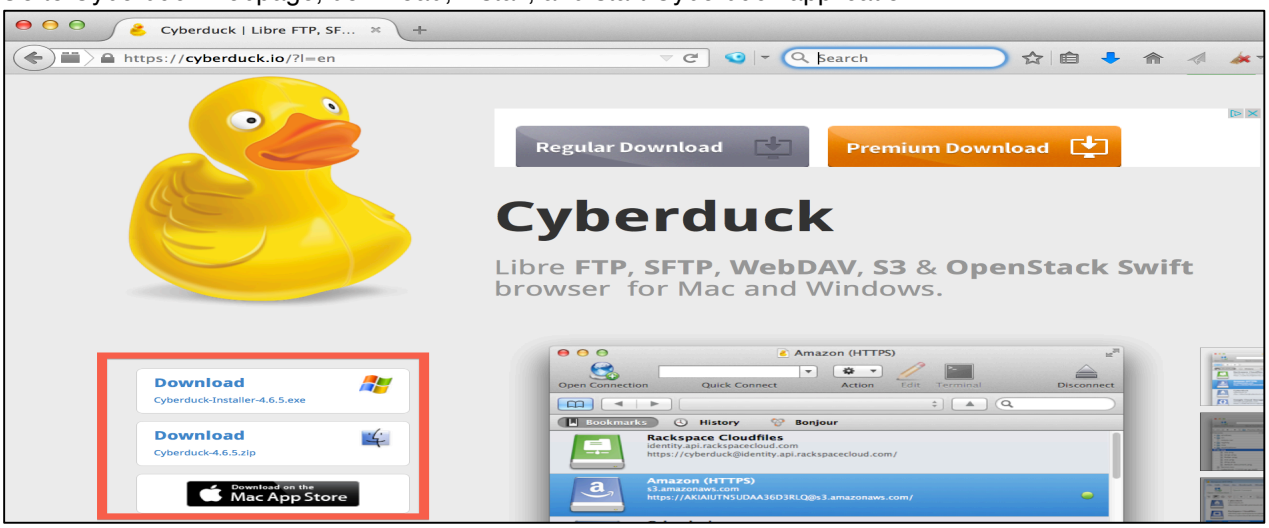
```
## create a quality control report
fastx_quality_stats -Q 33 -i HistoneRep1.raw.fastq -o HistoneRep1.qc.stats
fastx_quality_stats -Q 33 -i HistoneRep2.raw.fastq -o HistoneRep2.qc.stats
fastx_quality_stats -Q 33 -i HistoneCtlRep1.raw.fastq -o HistoneCtlRep1.qc.stats
fastx_quality_stats -Q 33 -i HistoneCtlRep2.raw.fastq -o HistoneCtlRep2.qc.stats
```

For each FASTQ file, draw box-plots of quality scores along read position using '**fastq_quality_boxplot_graph.sh**' command from .qc.stats file generated above.

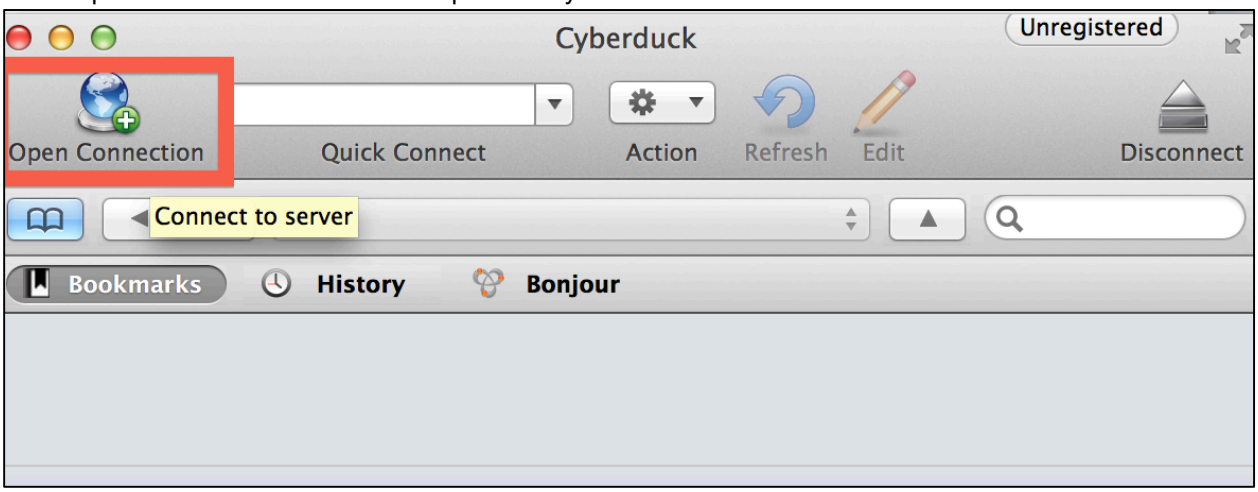
```
## draw a fastq quality chart
$ fastq_quality_boxplot_graph.sh -i HistoneRep1.qc.stats -o HistoneRep1.boxplot.png
$ fastq_quality_boxplot_graph.sh -i HistoneRep2.qc.stats -o HistoneRep2.boxplot.png
$ fastq_quality_boxplot_graph.sh -i HistoneCtlRep1.qc.stats -o HistoneCtlRep1.boxplot.png
$ fastq_quality_boxplot_graph.sh -i HistoneCtlRep1.qc.stats -o HistoneCtlRep1.boxplot.png

@LocalComputer$ scp username@hostname:/home/username/run/*.png .
```

Use Cyberduck (https://cyberduck.io/) to download generated .png image files to a local computer. Be sure to run this in your local computer.

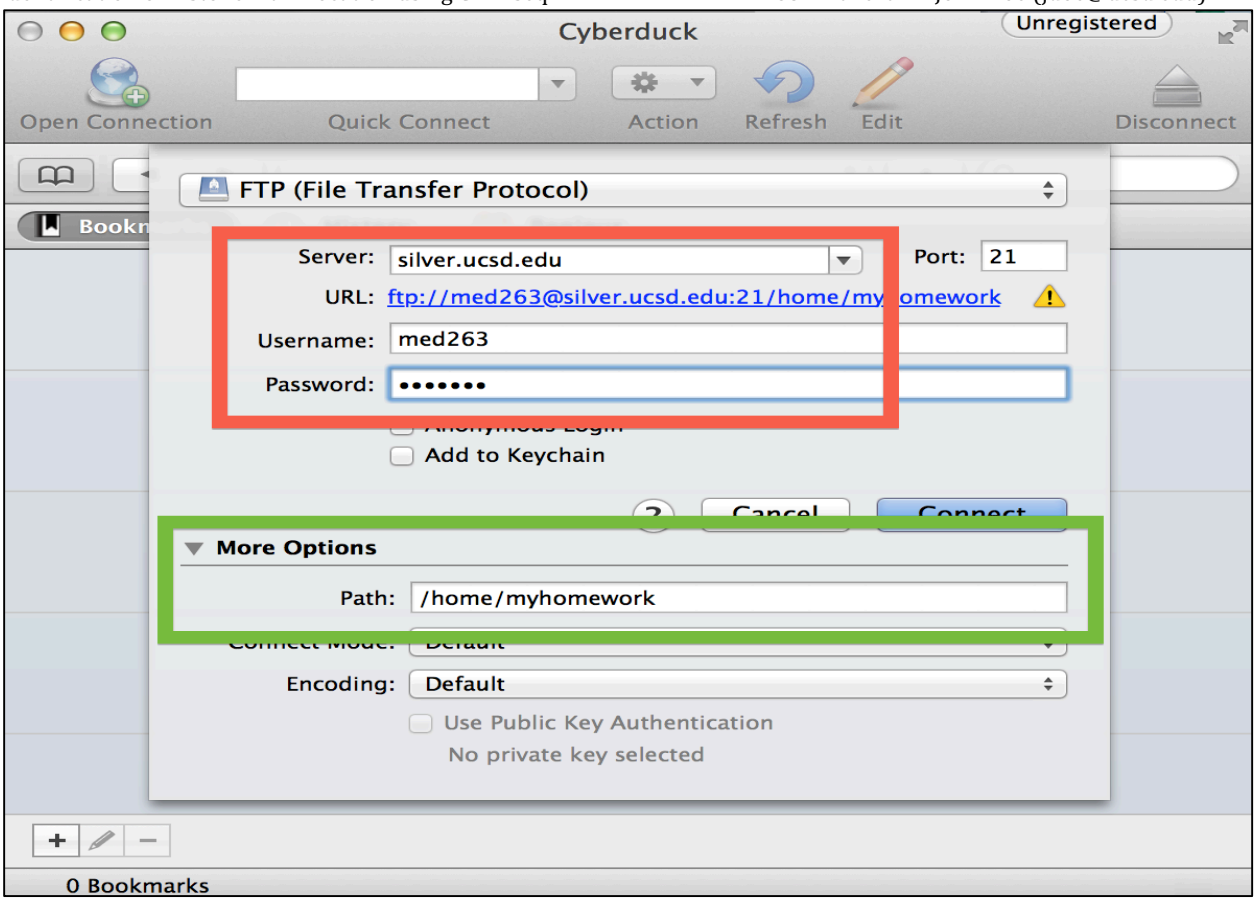
Go to Cyberduck webpage, download, install, and start Cyberduck application.



Click 'Open Connection' icon on the top left in Cyberduck.



Cyberduck will pop up a window asking for a FTP server information. Type server/username/password for the server (red rectangle). Click 'More Options' to type a path to a directory in your local computer (green).

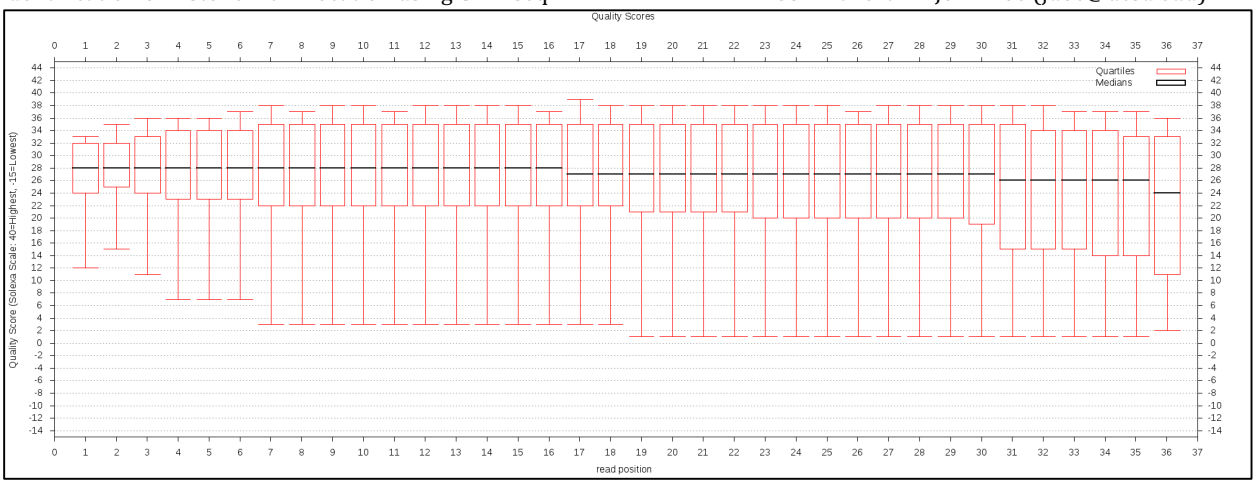


Browse direcotories and drag and drop files from the silver.ucsd.edu server down to your local computer (e.g. your laptop computer).

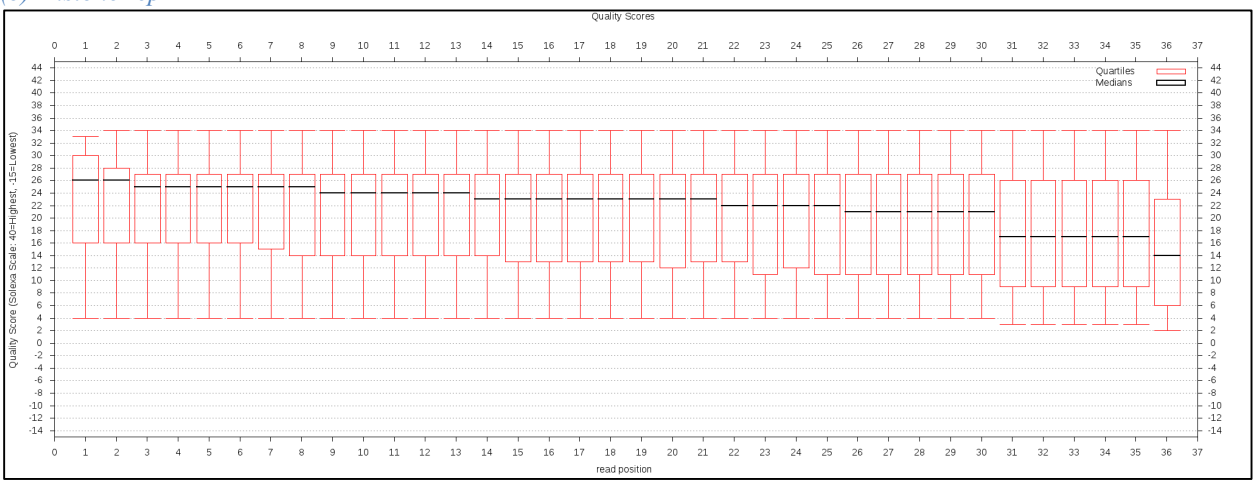
Q3. View downloaded boxplot image files in .png format. Does any FASTQ file have a read-position whose median quality score is below 15? In Phred quality score, how Q=30 can be linked to base call accuracy in percentage?

A3. Yes, there was one FASTQ fils with a read position whose median quality score was below 15. The median quality score for read position 36 in HistoneRep2 was 14. In Phred quality score, $Q=-10\log_{10}(p)$ where p is the base-calling probability. Hence, p=10^(-Q/100 = 10^(-30/10) = 0.001 and base calling accuracy is 100(1-p)=99.9%.*

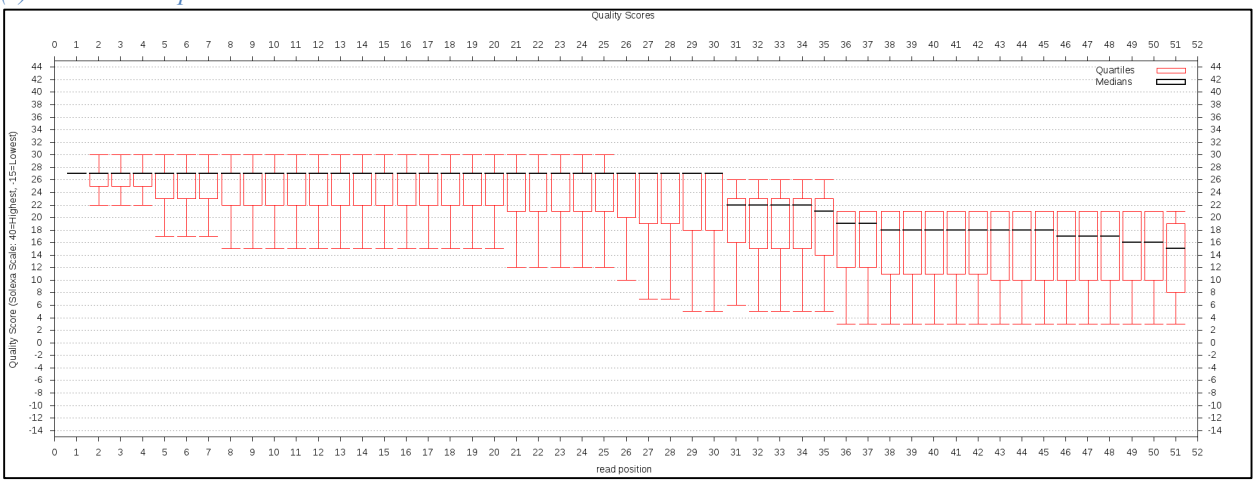
(a) HistoneRep1



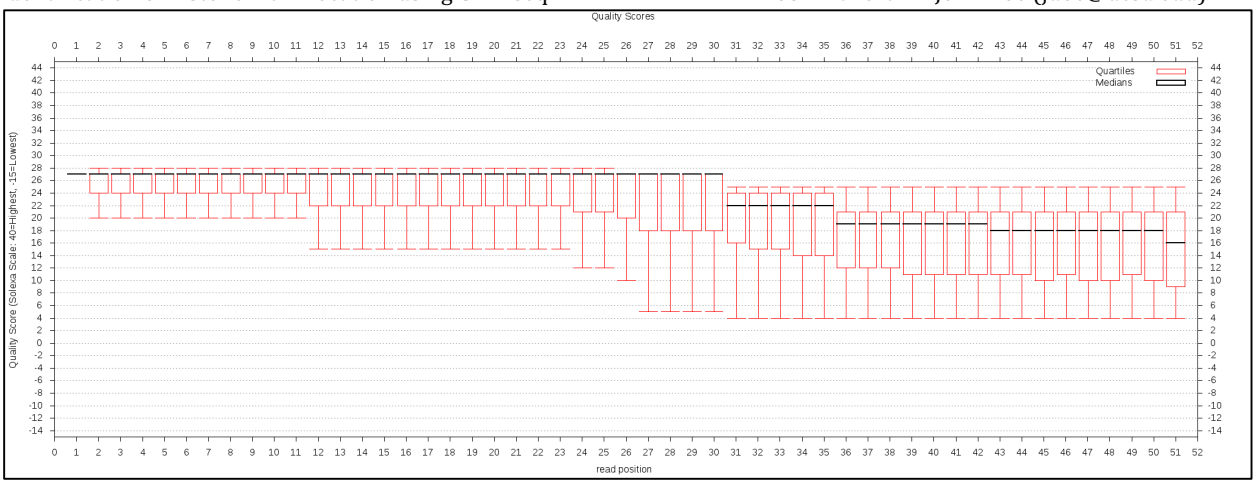
(b) HistoneRep2



(c) HistoneCtlRep1



(d) HistoneCtlRep2



Apply '**fastq_quality_filter**' command to keep only high-quality reads and save it as a new FASTQ file. The parameters are

-Q 33 = Illumina encoding for quality score
-q 20 = minimum quality score to keep
-p 80 = minimum percent of bases that must have [-q] quality

```
## filter reads
$ fastq_quality_filter -Q 33 -q 20 -p 80 -i HistoneRep1.raw.fastq -o HistoneRep1.qc.fastq
$ fastq_quality_filter -Q 33 -q 20 -p 80 -i HistoneRep2.raw.fastq -o HistoneRep2.qc.fastq
$ fastq_quality_filter -Q 33 -q 20 -p 80 -i HistoneCtlRep1.raw.fastq -o HistoneCtlRep1.qc.fastq
$ fastq_quality_filter -Q 33 -q 20 -p 80 -i HistoneCtlRep2.raw.fastq -o HistoneCtlRep2.qc.fastq
```

Q4. Use '**wc -l filename**', a linux line counting command to answer this question. What is the percentage of survived reads after applying '**fastq_quality_filter**' linux command for each FASTQ file? Which FASTQ file had the lowest survival percentage?

A4. HistoneCtlRep1 39.96%, HistoneCtlRep1 47.47%, HistoneRep1 62.16%, and HistoneRep2 32.49%. HistoneRep2 had the lowest survival percentage.

```
## word count for all .fastq files
$ wc -l *.fastq
17063580 HistoneCtlRep1.qc.fastq
42704640 HistoneCtlRep1.raw.fastq
18932840 HistoneCtlRep2.qc.fastq
39887484 HistoneCtlRep2.raw.fastq
237952504 HistoneRep1.qc.fastq
382814744 HistoneRep1.raw.fastq
63779712 HistoneRep2.qc.fastq
196332636 HistoneRep2.raw.fastq
```

4) ALIGNMENT

You now have the high-quality reads (FASTQ format). You will align these to the reference genome. You can build a bowtie index from a faw FASTQ file (http://bowtie-bio.sourceforge.net/manual.shtml#the-bowtie-build-indexer), but to save some time, let's download a pre-built index of hg19 human reference genome (hg19).

```
## download a file from web
$ wget ftp://ftp.ccb.jhu.edu/pub/data/bowtie_indexes/hg19.ebwt.zip

## extract a compressed file
$ unzip hg19.ebwt.zip
```

hg19 is the build version of human reference genome. The more recent version will have a bigger number. For example, the build dates are December 2013 for hg38, February, 2009 for hg19, and March 2006 for hg18.

Now you are ready to map short reads. The input is filter-passed reads whose filename ending with .qc.fastq from Part 4). The output will be in a SAM format and specified as the last parameter in **bowtie** run. The used parameters are

-m 1 = suppress all alignments mapped to more than one location, to keep unique-mapped reads
--sam = write output in a SAM format
-q = input file is in FASTQ format
hg19 = use the reference genome with a prefix 'hg19' located in a current directory

```
## call bowtie to align reads to a reference genome hg19
$ bowtie -m 1 --sam -q hg19  HistoneRep1.qc.fastq  HistoneRep1.sam
$ bowtie -m 1 --sam -q hg19  HistoneRep2.qc.fastq  HistoneRep2.sam
$ bowtie -m 1 --sam -q hg19  HistoneCtlRep1.qc.fastq  HistoneCtlRep1.sam
$ bowtie -m 1 --sam -q hg19  HistoneCtlRep2.qc.fastq  HistoneCtlRep2.sam
```

Next you will compress .sam format file into .bam. The used parameters are

-b = write output in BAM format
-S = input format is auto-detected

```
## convert .sam to .bam
$ samtools view -bS HistoneRep1.sam > HistoneRep1.raw.bam
$ samtools view -bS HistoneRep2.sam > HistoneRep2.raw.bam
$ samtools view -bS HistoneCtlRep1.sam > HistoneCtlRep1.raw.bam
$ samtools view -bS HistoneCtlRep2.sam > HistoneCtlRep2.raw.bam
```

Merge two replicates in bam format into one for antibody chip-seq and control chip-seq respectively.

```
## merge the replicate bam files
$ samtools merge Histone.raw.bam  HistoneRep1.raw.bam  HistoneRep2.raw.bam
$ samtools merge HistoneCtl.raw.bam  HistoneCtlRep1.raw.bam  HistoneCtlRep2.raw.bam
```

5) PEAK CALLING

You will now perform peak calling with MACS. The used parameters are

-t = input file name of treatment chip-seq; in our case histone
-c = input file name of control chip-seq
-g = genome size; 'hs' is human short-cut for 2.7e9 and 'mm' is mouse for 1.87e9
-n = output file prefix '`/home/username/Histone`' with a leading directory, where output will be created.
-B = create extended fragment pileup at every base-pair into a bedGraph file
-S = generate a single bedGraph file for treatment input.
--call-subpeaks = call external program called **PeakSplitter** for downstream processing.

The character '\' represents the current command is extended to a next line. Without it, the terminal will run '`macs14`' ignoring '`--call-subpeaks`' parameter below.

```
$ macs14 -t Histone.raw.bam -c HistoneCtl.raw.bam –g hs -n /home/username/Histone -B –S \
 --call-subpeaks
```

The primary MACS output is '_peaks.xls', in our case, Histone_peaks.xls as '`/home/username/Histone`' was used as a prefix for output. Its first 30 lines using '`head –n 30`' looks like the one below.

```
$ head -n 30 Histone_peaks.xls
# This file is generated by MACS version 1.4.2 20120305
# ARGUMENTS LIST:
# name = /mnt/mydata/histone/Histone
# format = AUTO
# ChIP-seq file = Histone.raw.bam
# control file = HistoneCtl.raw.bam
# effective genome size = 2.70e+09
# band width = 300
# model fold = 10,30
# pvalue cutoff = 1.00e-05
# Large dataset will be scaled towards smaller dataset.
# Range for calculating regional lambda is: 1000 bps and 10000 bps

# tag size is determined as 36 bps
# total tags in treatment: 56174382
# tags after filtering in treatment: 41144332
# maximum duplicate tags at the same position in treatment = 1
# Redundant rate in treatment: 0.27
# total tags in control: 6912716
# tags after filtering in control: 6856541
# maximum duplicate tags at the same position in control = 1
```

```
# Redundant rate in control: 0.01
# d = 188
chr   start end   length      summit      tags  -10*log10(pvalue) fold_enrichment   FDR(%)
chr1  892574      894987      2414  1530  413   465.20      286.96      0.11
chr1  895511      897434      1924  1462  142   92.86 62.84 0.49
chr1  901538      902970      1433  717   119   74.88 62.06 0.54
chr1  934491      937726      3236  1623  231   119.08      60.28 0.38
chr1  947965      951314      3350  1194  774   670.13      117.02      0.12
chr1  955455      957317      1863  1196  145   102.85      64.93 0.45
```

It comes with nine columns; chr, start, end, length, summit, tags, -10*log10(pvalue), fold_enrichment, and FDR (%). You can confirm this by running a '**grep**' command below. **grep** will search and return the lines meeting the given pattern. In the command line above, used parameters are

-P = interpret pattern as a Perl regular expression

"^chr\s+" = pattern of line starting with 'chr' followed by at least one space or tab character

```
$ grep -P "^chr\s+" Histone_peaks.xls
chr     start   end     length  summit  tags    -10*log10(pvalue)      fold_enrichment        FDR(%)
```

With **grep** and help with other commands, you can search the top 5 most significant regions of called peaks by MACS. The parameters are

-P = interpret pattern as a Perl regular expression

"^chr\W+" = pattern of line staring with 'chr' followed by at least one character of numeric, alphabet or underscore

Then **sort** command is used to sort lines by descending order of the column -10*log10(pvalue).

-r = sort in reverse ordr

-n = sort as numeric value

-k 7 = sort by 7th column; in this case, -10*log10(pvalue)

Lastly, **head** command is used to show only first 5 results.

Regular expression is not easy to understand at first time, but it is quite handy and quick tool for bioinformatics. Let's run following command to search for p-value cutoff applied to Histone_peaks.xls. Let's just run the command below. This will give you 1.00e-05, the default pvalue cutoff for MACS.

```
$ grep 'pvalue cutoff' Histone_peaks.xls
# pvalue cutoff = 1.00e-05
```

Let's just run the commands below to answer the question Q5.

```
## display the top 5 most significant called peaks
$ grep -P "^chr\w+"  Histone_peaks.xls | wc –l

## display the top 5 most significant called peaks
$ grep -P "^chr\w+"  Histone_peaks.xls | sort -rnk7 | head -n 5
```

The command 'wc –l' returns the number of lines of a file. And the vertical bar 'l' is referred to as a 'pipe' and used to direct the output of the first command before pipe into the input for the second command coming after pipe. To see how the pipe works, run above command with and without pipe to see how output results change.

Q5. How many peak regions are there at the significant level 1.00e-5 by MACS? What is the genomic coordinate of the most significant called peak by MACS? What was the p-value and FDR(%)?

*A5. The total number of peak regions was 24616.The most significant called peak by MACS was chr5:88166116-88187447 with -10*log10(pvalue) = 3323.6. Hence, p-value = 10^(-332.36) and FDR(%) was 0.00.*

```
## count the number of peak regions
$ grep -P "^chr\w+"  Histone_peaks.xls | wc –l
24616

## display the top 5 most significant called peaks
$ grep -P "^chr\w+"  Histone_peaks.xls | sort -rnk7 | head -n 5
chr5    88166116        88187447        21332   12435   5519    3233.06 280.92  0.00
chr1    234734273       234745565       11293   8115    3011    3126.77 265.96  0.00
chr3    121792093       121802792       10700   4991    3102    3119.62 164.01  0.00
chr2    158288229       158302959       14731   12118   4260    3100.00 208.33  37.50
chr15   45001245        45011939        10695   5217    3178    3100.00 225.70  37.50
```

6) ANNOTATION

We found a list of significant peak regions but what are their functions? Homer (http://homer.salk.edu) provides a convenient tool called `annotatePeaks.pl` to answer these annotation related questions.

```
## annotate peas
$ annotatePeaks.pl Histone_peaks.bed hg19 -annStats Histone_HomerStat.txt \
 -go Histone_GO > Histone_HomerAnno.txt
```

Extract the first 13 lines of the Histone_HomerStat.txt file to create a subset Histone_peaks_barplot.txt

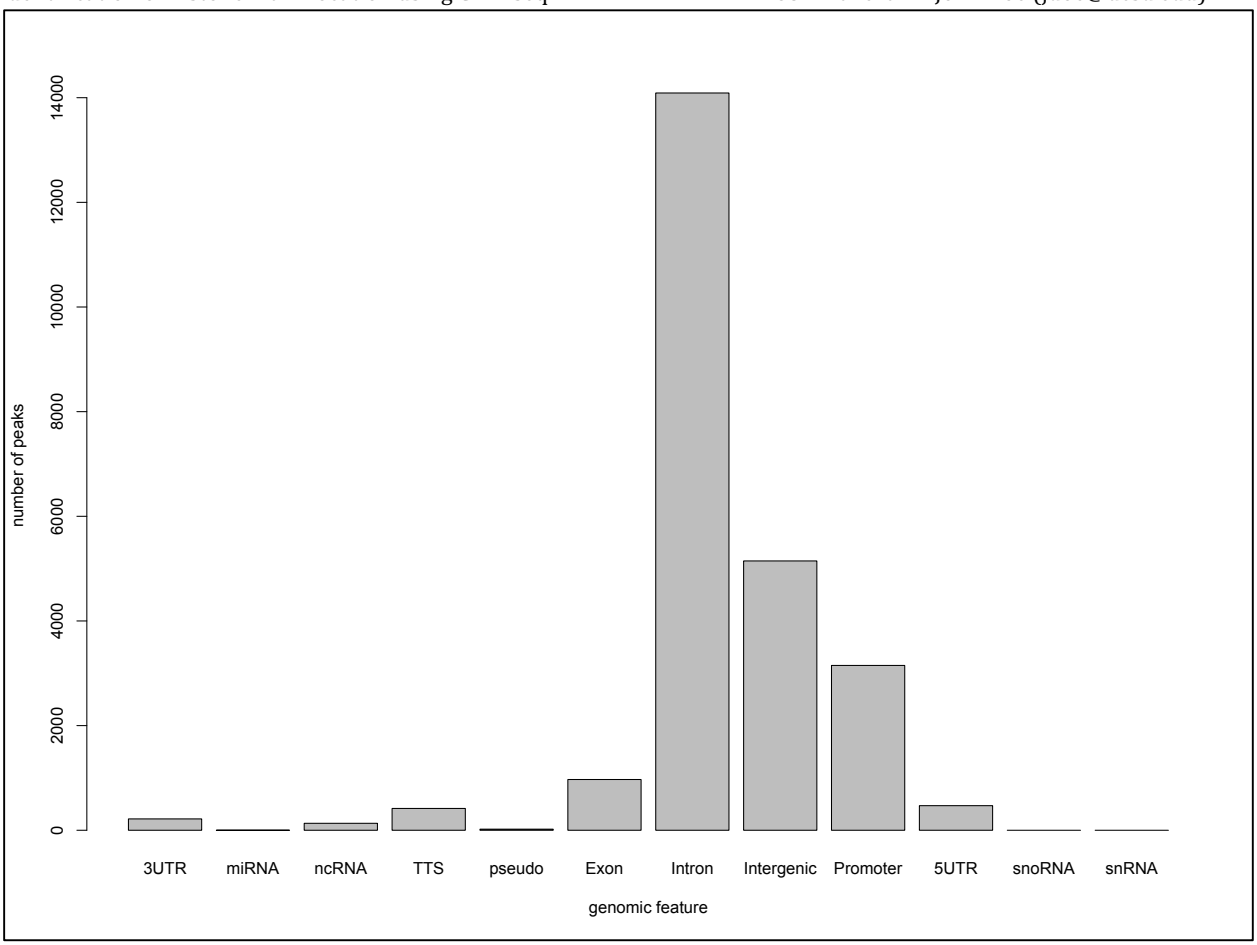
```
$ head -n 13 Histone_HomerStat.txt >  Histone_peaks_barplot.txt
```

Type `'$ R'` in the terminal to start R, a statistical computing tool. Then type following R commands in R console to draw a barplot of number of peaks across different genomic features. Save the plot as a pdf file.

```
$ R
R version 3.1.1 (2014-07-10) -- "Sock it to Me"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

    Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> pdf("Histone_peaks_barplot.pdf")
> da = read.table("Histone_peaks_barplot.txt", header=T, sep="\t", row.names=1)
> peaks = da[,1]
> names(peaks) = rownames(da)
> barplot(peaks, xlab="genomic feature", ylab="number of peaks")
> dev.off()
```

Q6. Which genome feature had the most peak regions called by MACS?

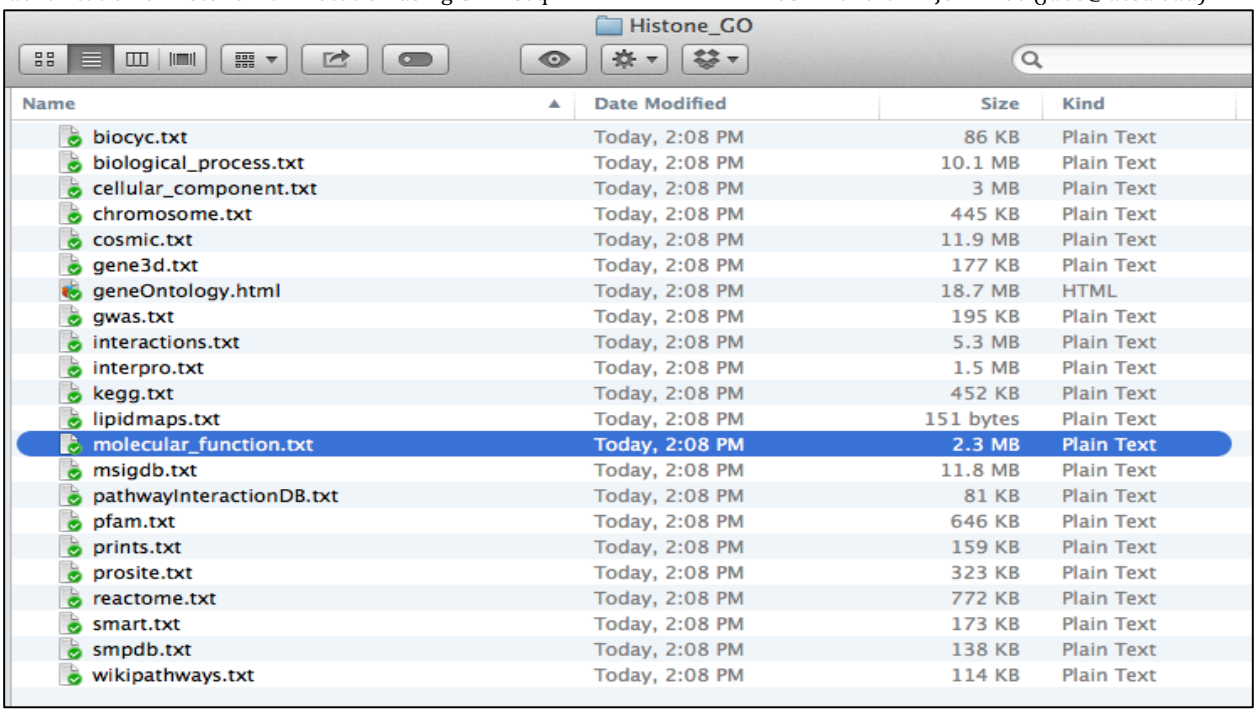
A6. Intron had the most peak regions.



Q7. In Part 5, you identified the genomic coordinate of the most significant peak region. Open Histone_HomerAnno.txt, take a look the first row after header row. What is the annotation information(=column H)? What is the gene name (=Column P)? What is the gene type (=Column S)?

A7. Annotation information intron (NM_001193347, intron 2 of 11), gene name is MEF2C-AS1, and gene type is ncRNA.

Previous Homer annotation also performed Gene Ontology (GO) enrichment analysis of peak regions with '`-go Histone_GO`' generating enrichment results in the 'Histone_GO' directory. Open a directory and you will see a lost files as below. Among these, you will open 'molecular_function.txt'. file (highlighted in blue).



Histone_GO

Name	Date Modified	Size	Kind
biocyc.txt	Today, 2:08 PM	86 KB	Plain Text
biological_process.txt	Today, 2:08 PM	10.1 MB	Plain Text
cellular_component.txt	Today, 2:08 PM	3 MB	Plain Text
chromosome.txt	Today, 2:08 PM	445 KB	Plain Text
cosmic.txt	Today, 2:08 PM	11.9 MB	Plain Text
gene3d.txt	Today, 2:08 PM	177 KB	Plain Text
geneOntology.html	Today, 2:08 PM	18.7 MB	HTML
gwas.txt	Today, 2:08 PM	195 KB	Plain Text
interactions.txt	Today, 2:08 PM	5.3 MB	Plain Text
interpro.txt	Today, 2:08 PM	1.5 MB	Plain Text
kegg.txt	Today, 2:08 PM	452 KB	Plain Text
lipidmaps.txt	Today, 2:08 PM	151 bytes	Plain Text
molecular_function.txt	Today, 2:08 PM	2.3 MB	Plain Text
msigdb.txt	Today, 2:08 PM	11.8 MB	Plain Text
pathwayInteractionDB.txt	Today, 2:08 PM	81 KB	Plain Text
pfam.txt	Today, 2:08 PM	646 KB	Plain Text
prints.txt	Today, 2:08 PM	159 KB	Plain Text
prosite.txt	Today, 2:08 PM	323 KB	Plain Text
reactome.txt	Today, 2:08 PM	772 KB	Plain Text
smart.txt	Today, 2:08 PM	173 KB	Plain Text
smpdb.txt	Today, 2:08 PM	138 KB	Plain Text
wikipathways.txt	Today, 2:08 PM	114 KB	Plain Text

Instead of downloading and opening molecular_function.txt file to the local computer, you can check this out simply by running command below.

```
$ head -n 2 Histone_GO/molecular_function.txt | awk '{print $1,$2,$4}'
```

Q8. What is the Gene Ontology TermID, Term, and logP (columns A, B, C) of the most significant one (=the first row)?

A8. For the most significant GO, TermID was GO:0005488, Term was 'binding', and logP was -216.568

```
$ head -n 2 Histone_GO/molecular_function.txt | awk '{print $1,$2,$4}'
TermID Term logP
GO:0005488 binding -216.567726236506
```

7) UCSC GENOME BROWSER

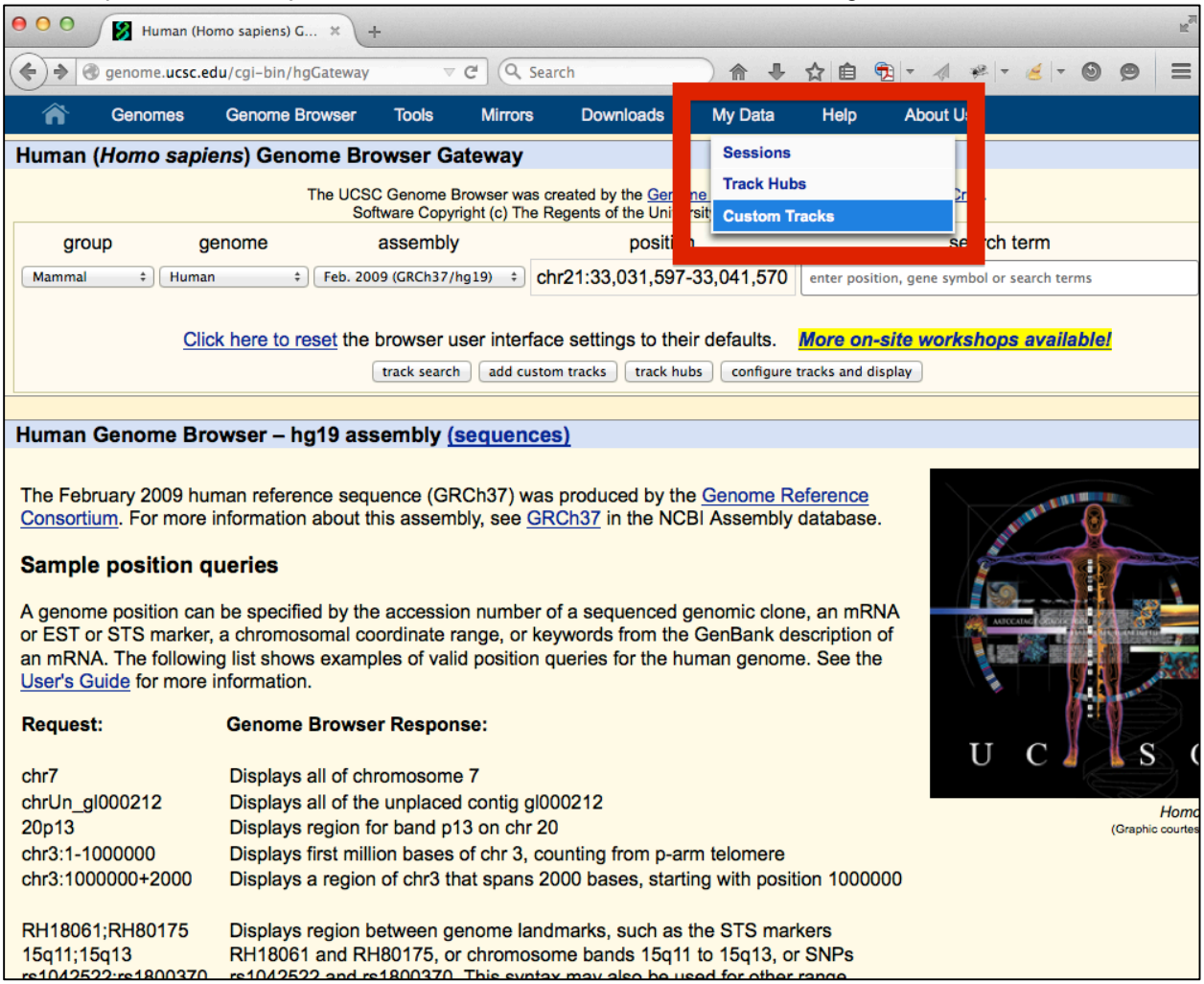
Lastly, you will process .bdg files generated by MACS to generate visualization file to be uploaded tos UCSC Genome Browser(http://genome.ucsc.edu). Both input and output file format is bedGraph and its detailed specification can be found at http://genome.ucsc.edu/goldenpath/help/bedgraph.html. Simply run the command line below by copying and pasting the whole contents in a box below into the terminal.

```
export chrom=chr1
export spos=32000000
export epos=32800000
export outfile=${chrom}.bdg

echo -e "browser hide all\nbrowser pack refGene"  > ${outfile}

echo "track type=bedGraph name=Histone_control description=Histone_control visibility=full
autoScale=off viewLimits=0:200 color=51,153,255" >> ${outfile}

grep "^$chrom\t" Histone_control_afterfiting_all.bdg | awk -v spos=$spos -v epos=$epos '{if($2 >
spos && $3 < epos ) print $0}'  >> ${outfile}

echo  "track type=bedGraph name=Histone_treat description=Histone_treat visibility=full
autoScale=off viewLimits=0:200 color=255,153,51" >> ${outfile}

grep "^$chrom\t" Histone_treat_afterfiting_all.bdg | awk -v spos=$spos -v epos=$epos '{if($2 >
spos && $3 < epos ) print $0}'  >> ${outfile}

echo "track type=bedGraph name=MACS_peak description=MACS_peak visibility=pack color=255,102,102"
>> ${outfile}

grep "^$chrom\t" Histone_peaks.bed | awk -v spos=$spos -v epos=$epos '{if($2 > spos && $3 < epos)
```

```
print $1,$2,$3,$5 }' >> ${outfile}
```

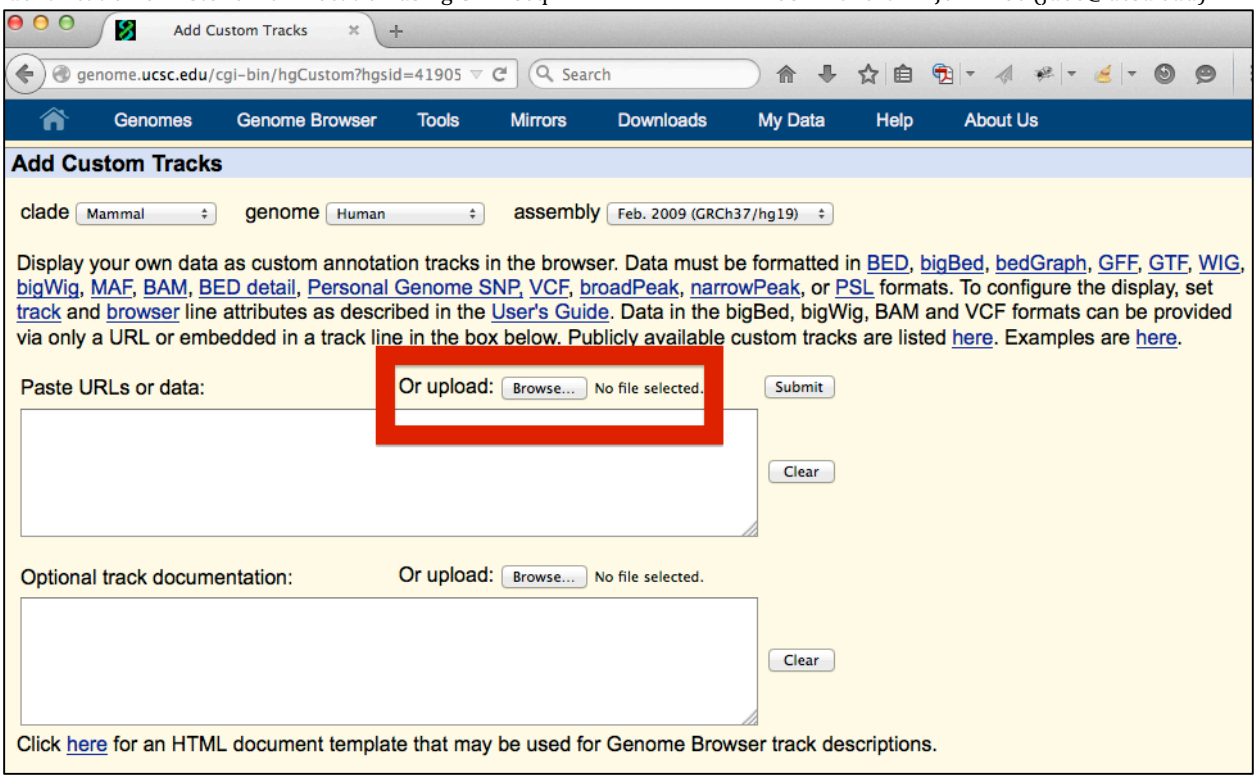
Then the file chr1.bdg will be created. Download this file into your local computer using CyberDuck.

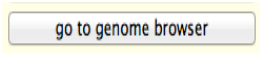
Open a browser and go to UCSC Genome Browser Gateway (http://genome.ucsc.edu/cgi-bin/hgGateway).
On the top menu, click 'My Data' and 'Custom Tracks' as a red box in the figure below.



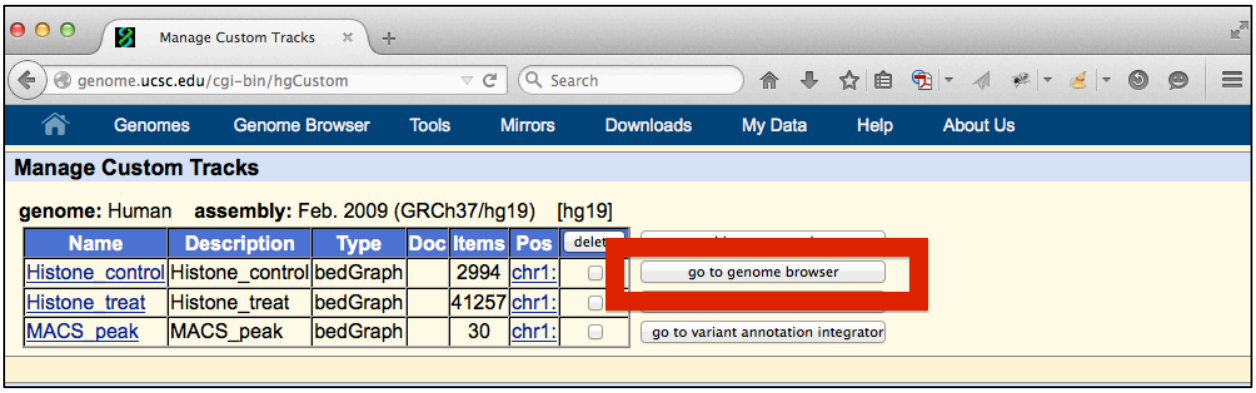
Click 'Browse' button 

, select downloaded chr1.bdg file, and press submit button.

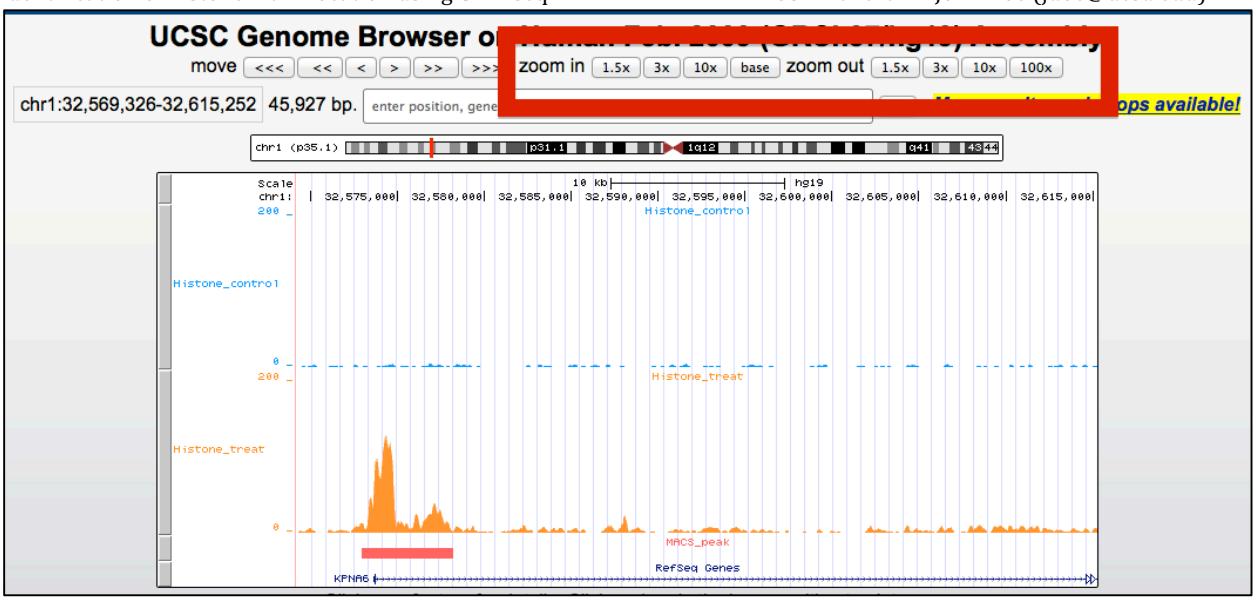


Click 

'go to genome browser' button (red rectangle).

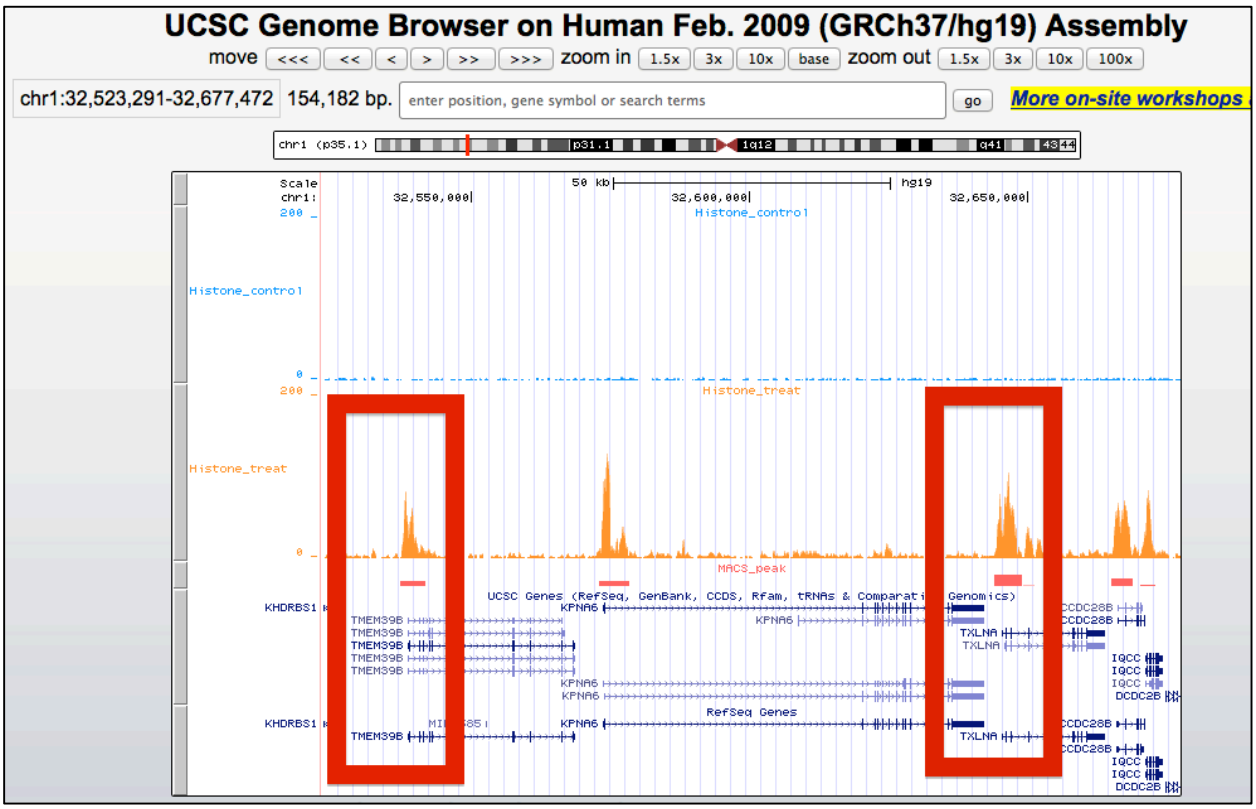


Try 'zoom in' and 'zoom out' buttons to find high peaks in Histone_treat track (orange color). You can also type in gene name such as 'KPNA6' to get the screen shot below. Click the plot with left mouse click, then you will able to move left and right.



Q9. From the UCSC Genome Browser Custom track above, find two other nearest genes (in both 3' and 5' direction) around 'KPNA6' in the interval of chr1:32000000-32800000. Use zoom in/out and move around to show all three genes similar to above figure with KPNA6 gene alone.

A9. Two genes, TMEM39B on the left and TXLNA on the right, were found and their peaks are shown below.

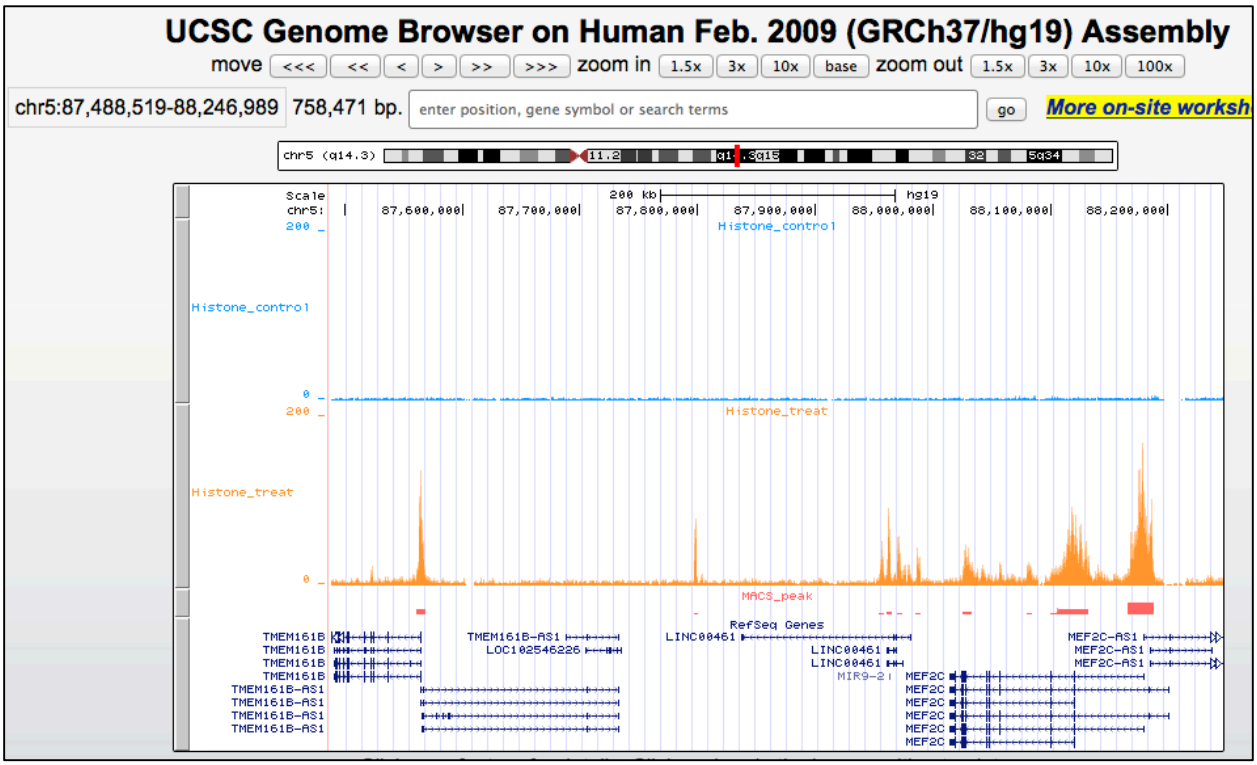


Q10. Recall the genomic coordinate of the most significant peak region by MACS from Q5. Expand this region with subtracting 100,000 bp from the start point and adding 100,000 bp to the end point. In the Part 9 script for creation of input to UCSC Genome Browser, use this new interval and new chromosome. Run a script in the tutorial **with different genomic cooridates** to generate .bdg file and upload it the Custom Track of UCSC Genome Browser. Get a screenshot of the peak region using the interval in Q5 and report the neighbor genes with high MACS peaks near this region.

A10. In Q5, the genomic coordinate of the most significant peak region was chr5:88166116-88187447.+/- 100,000 bp was applied.

```
export chrom=chr5
export spos=87166116
export epos=89187447
export outfile=${chrom}.bdg

echo -e "browser hide all\nbrowser pack refGene"  > ${outfile}

echo "track type=bedGraph name=Histone_control description=Histone_control visibility=full
autoScale=off viewLimits=0:200 color=51,153,255" >> ${outfile}

grep "^$chrom\t" Histone_control_afterfiting_all.bdg | awk -v spos=$spos -v epos=$epos '{if($2 >
spos && $3 < epos ) print $0}'  >> ${outfile}

echo  "track type=bedGraph name=Histone_treat description=Histone_treat visibility=full
autoScale=off viewLimits=0:200 color=255,153,51" >> ${outfile}

grep "^$chrom\t" Histone_treat_afterfiting_all.bdg | awk -v spos=$spos -v epos=$epos '{if($2 >
spos && $3 < epos ) print $0}'  >> ${outfile}

echo "track type=bedGraph name=MACS_peak description=MACS_peak visibility=pack color=255,102,102"
>> ${outfile}

grep "^$chrom\t" Histone_peaks.bed | awk -v spos=$spos -v epos=$epos '{if($2 > spos && $3 < epos)
print $1,$2,$3,$5 }' >> ${outfile}
```

The most significant regions in Q5 was in MEF2C-AS1 gene. Its neighbor genes with MACS peaks are TMEM161B and MEF2C.



8) APPENDIX: MACHINE PROVISONING

This part is Appendix and has no questions to be answered. Installation and configuration of software tools are beyond this course MED263 and required a special privilege such as system administrator in some cases. You can assume that all the tools and necessary dependent linux packages for this Practical are already installed in a dedicate machine. If you want to run tools in this tutorial in other environment, you need to have 'sudo' privilege or you should contact your system administrator to run provisioning script below.

```
## set environment variables
export IH=/opt/bin

## install dependent ubuntu packages
apt-get install -y gnuplot g++ libncurses5-dev libncursesw5-dev  make
apt-get install –y software-properties-common wget zip zlib1g-dev

## install bowtie
cd ${IH}
wget http://sourceforge.net/projects/bowtie-bio/files/bowtie/1.1.1/bowtie-1.1.1-linux-x86_64.zip
unzip bowtie-1.1.1-linux-x86_64.zip

## install FASTX-Toolkit
cd ${IH}
wget
http://hannonlab.cshl.edu/fastx_toolkit/fastx_toolkit_0.0.13_binaries_Linux_2.6_amd64.tar.bz2
tar jxvf fastx_toolkit_0.0.13_binaries_Linux_2.6_amd64.tar.bz2
mv bin fastx_toolkit_0.0.13

## install homer
cd ${IH}
wget http://homer.salk.edu/homer/configureHomer.pl
perl configureHomer.pl -install

## install MACS
cd ${IH}
wget https://github.com/downloads/taoliu/MACS/macs_1.4.2_python2.7.deb
dpkg -i macs_1.4.2_python2.7.deb

## install samtools
cd ${IH}
wget https://github.com/samtools/samtools/releases/download/1.2/samtools-1.2.tar.bz2
tar jxvf samtools-1.2.tar.bz2
cd samtools-1.2
make

# set path
echo -e "export PYTHONPATH=${IH}/macs/lib/python2.7/site-packages:$PYTHONPATH" >> /etc/profile
echo -e "${IH}/bowtie.1.1:${IH}/fastx_toolkit_0.0.13:$PATH" >> /etc/profile
echo -e "${IH}/fastx_toolkit_0.0.13:$PATH" >> ~/.bashrc
echo -e "${IH}/PeakSplitter_Cpp/PeakSplitter_Linux64:$PATH" >> /etc/profile
```

References

1. Feng J, Liu T, Qin B, Zhang Y, Liu XS. Identifying ChIP-seq enrichment using MACS. Nat Protoc. 2012 Sep;7(9):1728-40.
2. ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. Nature. 2012 Sep 6;489(7414):57-74.
3. Heinz S, Benner C, Spann N, Bertolino E, Lin YC, Laslo P, Cheng JX, Murre C, Singh H, Glass CK. Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities. Mol Cell. 2010 May 28;38(4):576-89.
4. Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol. 2009;10(3):R25.
5. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R; 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009 Aug 15;25(16):2078-9.
6. Zhang Y, Liu T, Meyer CA, Eeckhoute J, Johnson DS, Bernstein BE, Nusbaum C, Myers RM, Brown M, Li W, Liu XS. Model-based analysis of ChIP-Seq (MACS). Genome Biol. 2008;9(9):R137.