ENCODE2 pipeline v1 specifications (single end only)

QC metrics reported in ENCODE2:

- http://encodeproject.org/ENCODE/qualityMetrics.html#definitions
- http://encodeproject.org/ENCODE/qualityMetrics.html

0. Filter out low quality reads - optional, was not specified

- FastQC can show the distribution of sequence quality scores and to help set an appropriate cutoff: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/3%20Per%20Sequence%20Quality%20Scores.html)
- The -q parameter in BWA does read trimming so it can also be used to remove bad parts of reads

1a. Read alignment (BWA aligner)

Single-End ChIP-seq parameters

- With dynamic read trimming q = 5
- seed length l = 32
- max. mismatches in seed k = 2

Program(s)	• BWA version ? (ask Anshul which one has bug)
Input(s)	• FASTQ file • hg19 male or female or mm9 reference sequence
Output(s)	• SAM file • (SAM index file)
Commands	• `bwa aln -q 5 -l 32 -k 2 -t [NUM_THREADS] ref.fa SE.fastq.gz > SE.sai` • `bwa samse ref.fa SE.sai SE.fastq.gz > final_SE.sam`
QC to report	None.

1b. Post-alignment filtering

- Remove unmapped reads (using bit 3 in flag with -f)
- Remove multi-mapped reads (i.e. those with MAPQ = 0, using -q in SAMtools)
    - http://samtools.sourceforge.net/
    - This would output alignments in BAM format that are not unmapped and have MAPQ > 1
- Remove PCR duplicates (using Picard's MarkDuplicates)
    - http://picard.sourceforge.net/command-line-overview.shtml#MarkDuplicates

Program(s)	• SAMtools (latest version 0.1.19) • MarkDuplicates (Picard - latest version 1.110)
Input(s)	• SAM file
Output(s)	• Filtered BAM file, (BAM index file), metrics file from MarkDuplicates
Commands	• `samtools view -b -F 4 -q 1 file.sam > mapped.bam` • `java -jar MarkDuplicates.jar I=sortedInput.bam O=sortedDeduped.bam M=Dedup_metrics.txt VALIDATION_STRINGENCY=LENIENT QUIET=true REMOVE_DUPLICATES=true`

QC to report	• # of uniquely mapped reads (N_uniq map reads)
	• PCR bottleneck coefficient (PBC) = N1/Nd, where
	  ○ N1 = number of genomic locations to which EXACTLY one unique mapping read maps
	  ○ Nd = the number of genomic locations to which AT LEAST one unique mapping read maps, i.e. the number of non-redundant, unique mapping reads
	  ○ Provisionally, 0-0.5 is severe bottlenecking, 0.5-0.8 is moderate bottlenecking, 0.8-0.9 is mild bottlenecking, while 0.9-1.0 is no bottlenecking.

2a. Convert BAM to tagAlign (BED 3+3 format)

Program(s)	• SAMTools (latest version 0.1.19)
Input(s)	• Filtered BAM file
Output(s)	• tagAlign file (gzip only necessary for space saving, not for subsequent steps)
Commands	• `samtools view -F 0x0204 -o - <bamFile> | awk 'BEGIN{OFS="\t"}{if (and($2,16) > 0) {print`

	`$3,($4-1),($4-1+length($10)),"N","1000","-"} else {print $3,($4-1),($4-1+length($10)),"N","1000","+"} }' | gzip -c > <gzip_TagAlignFileName>`
QC to report	None.

2b. Calculate NSC/RSC

- Code package: https://code.google.com/p/phantompeakqualtools/
- Dependencies: unix, bash, R-2.10 and above, awk, samtools, boost C++ libraries, R packages: SPP, caTools, snow

Program(s)	• phantompeakqualtools (v1.1)
Input(s)	• tagAlign or BAM file
Output(s)	• outFile containing NSC/RSC results in tab-delimited file of 11 columns (same file can be appended to from multiple runs) • cross-correlation plot
Commands	• `Rscript run_spp.R -c=<tagAlign/BAMfile> -p=<num parallel processing nodes> -savp=<cross-correlation plot> -out=<outFile>` • -out=<outFile> will create and/or append to a file named <outFile> several important characteristics of the dataset.
QC to report	• Normalized strand cross-correlation coefficient (NSC) = col9 in outFile • Relative strand cross-correlation coefficient (RSC) = col10 in outFile • A check that NSC > 1.1 and RSC > 1, flag otherwise • Quality tag, a thresholded version of RSC (codes: -2:veryLow,-1:Low,0:Medium,1:High,2:veryHigh) = col11 in outFile • Estimated fragment length = col3 in outFile, take the top value • Important columns highlighted, but all/whole file can be stored for display

2c. Generate pseudoreplicates for self-consistency tests

- For each replicate, use this shell script:
    - ```
      fileName='chipSampleRep1.tagAlign.gz' # input tagAlign file name
      outputDir='/mapped/selfPseudoReps' # output directory for pseudoreplicate files
      outputStub='chipSampleRep1' # prefix name for pseudoReplicate files

      nlines=$( zcat ${fileName} | wc -l ) # Number of reads in the tagAlign file
      nlines=$(( (nlines + 1) / 2 )) # half that number
      ```
      ```

```
zcat "${fileName}" | shuf | split -d -l ${nlines} -
"${outputDir}/${outputStub}" # This will shuffle the lines in the file and
split it into two parts
gzip "${outputDir}/${outputStub}00"
gzip "${outputDir}/${outputStub}01"
mv "${outputDir}/${outputStub}00.gz"
"${outputDir}/${outputStub}.pr1.tagAlign.gz"
mv "${outputDir}/${outputStub}01.gz"
"${outputDir}/${outputStub}.pr2.tagAlign.gz"
```

- Pool relevant control files into one: (unclear on rule to decide which controls to pool, the species specific IgG was matched where possible to the Ab in Snyder experiments, but Myers sometimes had many controls and not all were pooled)
    - `zcat controlSampleRep1.tagAlign.gz controlSampleRep2.tagAlign.gz controlSampleRep3.tagAlign.gz | gzip -c > controlSampleRep0.tagAlign.gz`
- Pool replicates into one:
    - `zcat chipSampleRep1.tagAlign.gz chipSampleRep2.tagAlign.gz chipSampleRep3.tagAlign.gz | gzip -c > chipSampleRep0.tagAlign.gz`
- Make pseudoreplicates from pooled replicates as above

4. Peak calling - SPP

- To call peaks at the relaxed threshold for IDR:
    - Note: “For most ChIP-seq datasets, it is good to use the exclusion parameter (-x) with run_spp.R which will help the code skip any cross-correlation peak in the exclusion zone (values that are impossible as fragment lengths). I recommend that you use -x=-500:85 it is relatively unlikely that your fragment lengths will be in that range.” (This was assuming the old ENCODE2 36bp or 50bp SE reads).
    - Can use the estimated fragment length from column 3 in step 2b’s output file:
        - `sed -r 's/,[^\t]+//g' <outFile> > <newOutFile>`

Program(s)	• Anshul-modified SPP (v1.0) in phantompeakqualtool
Input(s)	• RepN ChIP, pooled control BAM/tagAlign • Pooled replicate, pooled control BAM/tagAlign • RepN pseudoreplicate 1 vs. pooled control BAM/tagAlign • RepN pseudoreplicate 2 vs. pooled control BAM/tagAlign • Pooled replicate pseudoreplicate 1 vs. pooled control BAM/tagAlign • Pooled replicate pseudoreplicate 2 vs. pooled control BAM/tagAlign
Output(s)	• RepN_vs_pooled_control peaks • Pooled_replicate_vs_pooled_control peaks • RepN_pseudoreplicate_1_vs_pooled_control peaks • RepN_pseudoreplicate_2_vs_pooled_control peaks

	• Pooled_replicate_pseudoreplicate_1_vs_pooled_control peaks • Pooled_replicate_pseudoreplicate_2_vs_pooled_control peaks • Output peak files are in narrowPeak format
Commands	• `Rscript run_spp_nodups.R -c=<ChIP_tagalign/BAM_file> -i=<control_tagalign/BAM_file> -npeak=300000 -odir=<peak_call_output_dir> -savr -savp -rf -x=-500:85 -out=<resultFile>`
QC to report	• Number of peaks called for each set of replicates

4a. IDR analysis - original replicates

- Code: https://sites.google.com/site/anshulkundaje/projects/idr
- R scripts batch-consistency-analysis.r and batch-consistency-plot.r

Program(s)	• batch-consistency-analysis.r
Input(s)	• narrowPeak files from SPP peak caller, run for each biological replicate combinations (if more than 2 replicates)
Output(s)	• The output from EM fitting: suffixed by -em.sav • The output for plotting empirical curves: suffixed by -uri.sav Note: 1 and 2 are objects that can be loaded back to R for plotting or other purposes (e.g. retrieve data) • The parameters estimated from EM and the log of consistency analysis, suffixed by -Rout.txt • The number of peaks that pass specific IDR thresholds for the pairwise analysis: suffixed by npeaks-aboveIDR.txt • The full set of peaks that overlap between the replicates with local and global IDR: suffixed by overlapped-peaks.txt
Commands	• `Format: Rscript batch-consistency-analysis.r [peakfile1] [peakfile2] [peak.half.width] [outfile.prefix] [min.overlap.ratio] [is.broadpeak] [ranking.measure]` • `Command for SPP: Rscript batch-consistency-analysis.r [peakfile1] [peakfile2] -1 [outfile.prefix] 0 F signal.value`
Parameters	• [peakfile1] and [peakfile2] are the peak calls for the pair of replicates in narrowPeak format. They must be uncompressed files. e.g. /peaks/reps/chipSampleRep1_VS_controlSampleRep0.narrowPeak AND /peaks/reps/chipSampleRep2_VS_controlSampleRep0.narrowPeak • [peak.half.width]: Set this to -1 if you want to use the reported peak width in the peak files. (-1 is the only value that works for this parameter right now) • [outfile.prefix] is a prefix that will be used to name the output data for this pair of replicates. The prefix must also include the PATH to the directory where you

	want to store the output data. e.g. /consistency/reps/chipSampleRep1_VS_chipSampleRep2 • [min.overlap.ratio]: fractional bp overlap (ranges from 0 to 1) between peaks in replicates to be considered as overlapping peaks. Set to 0 if you want to allow overlap to be defined as >= 1 bp overlap. If set to say 0.5 this would mean that atleast 50% of the peak in one replicate should be covered by a peak in the other replicate to count as an overlap. IMPORTANT: This parameter has not been tested fully. It is recommended to set this to 0. • [is.broadpeak]: Is the peak file format narrowPeak or broadPeak. Set to F if it is narrowPeak/regionPeak or T if it is broadPeak. BroadPeak files do not contain Column 10. • [ranking.measure] is the ranking measure to use. It can take only one of the following values signal.value , p.value or q.value
QC to report	• Number of peaks above IDR thresholds • For each pairwise analysis, we have a *overlapped-peaks.txt file. The last column (Column 11) of the overlapped-peaks.txt file has the globa IDR score for each pair of overlapping peaks. • To get the number of peaks that pass an IDR threshold of T (e.g. 0.01 you simply find the number of lines that have a global IDR score <= T • `awk '$11 <= 0.01 {print $0}' [overlappedPeaksFileName] | wc -l`

Plot IDR consistency plots

Program(s)	• batch-consistency-plot.r
Input(s)	• narrowPeak files from SPP peak caller, run for each biological replicate combinations (if more than 2 replicates)
Output(s)	• Summary consistent plot in .ps format (suffixed -plot.ps)
Commands	• `Format: Rscript batch-consistency-plot.r [npairs] [output.prefix [input.file.prefix1] [input.file.prefix2] [input.file.prefix3] ....` • `Command: Rscript batch-consistency-plot.r 3 /consistency/reps/chipSampleAllReps /consistency/reps/chipSampleRep1_VS_chipSampleRep2 /consistency/reps/chipSampleRep1_VS_chipSampleRep3 /consistency/reps/chipSampleRep2_VS_chipSampleRep3`
Parameters	• [n.pairs] is the number of pairs of replicates that you want to plot on the same plot e.g. 1 or 3 or ... • The prefix must also include the PATH to the directory where you want to store the output data. e.g. /consistency/plots/chipSampleAllReps
QC to report	• IDR consistency plots between replicates

- Threshold at 0.02 (2%) IDR for true replicates and pseudoreplicates if # of peaks called is >100K
- Threshold at 0.05 (5%) IDR for true replicates and pseudoreplicates if # of peaks called is < 100K
- Threshold at 0.0025 (0.25%) or 0.005 (0.5%) for pooled pseudoreplicates if # of peaks called is > 100K
- Threshold at 0.01 (1%) IDR for pooled pseudoreplicates if # of peaks called < 100K
- Check self-consistency IDR (i.e. pseudoreplicates) to see if # of peaks passing IDR is similar (within a factor of 2), flag if not
- Select the true replicate with the most peaks passing IDR thresholds, call it **max_numPeaks_Rep**
- **max_numPeaks_Rep** and number of peaks passing IDR in pooled pseudoreplicates should be similar (within a factor of 2), flag if not

4b. IDR analysis - pseudoreplicates

- Perform as with real replicates, but comparing pseudoreplicate 1 vs pseudoreplicate 2 made from each of the real biological replicate peaks
- This gives the self-consistent IDR peaks

4c. IDR analysis - pooled pseudoreplicates

- Perform as with real replicates, but comparing pseudoreplicate 1 vs pseudoreplicate 2 made from the pooled biological replicate peaks

4d. Select final peak calls - conservative set

- Sort peaks from the pooled real replicates set (i.e. pooled reps vs. pooled control) and sort by signal value (column 7) in descending order
- Select the top **max_numPeaks**
- Filter using black list: http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeMapability/wgEncodeDacMapabilityConsensusExcludable.bed.gz

4e. Select final peak calls - optimal set

- Sort peaks from the pooled real replicates set (i.e. pooled reps vs. pooled control) and sort by signal value (column 7) in descending order
- Select the top max(**max_numPeaks**, # peaks passing IDR from pooled replicates)
- Filter using black list: http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeMapability/wgEncodeDacMapabilityConsensusExcludable.bed.gz

5. Make signal files - WIGGLER (Anshul needs to clarify combining replicates and lab datasets)

- Code package:
  - https://sites.google.com/site/anshulkundaje/projects/wiggler
  - https://code.google.com/p/align2rawsignal/
- Dependencies:
  - MATLAB runtime library: http://www.broadinstitute.org/~anshul/softwareRepo/MCR2010b.bin
  - SAMtools
  - At least 2 GB of memory - the more, the faster it runs
  - Appropriate sex and length matched mappability files: http://www.broadinstitute.org/~anshul/projects/umap/
- Install MCR into a directory of choice: `./MCR2010b.bin -console`
- Modify .bashrc:
  - `MCRROOT=<MCR_ROOT>/v714`
  - `LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${MCRROOT}/runtime/glnxa64`
  - `LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${MCRROOT}/bin/glnxa64`
  - `LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${MCRROOT}/sys/os/glnxa64`
  - `MCRJRE=${MCRROOT}/sys/java/jre/glnxa64/jre/lib/amd64`
  - `LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${MCRJRE}/native_threads`
  - `LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${MCRJRE}/server`
  - `LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${MCRJRE}`
  - `XAPPLRESDIR=${MCRROOT}/X11/app-defaults`
  - `export LD_LIBRARY_PATH`
  - `export XAPPLRESDIR`
- or .cshrc:
  - `setenv MCRROOT <MCR_ROOT>/v714`
  - `setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${MCRROOT}/runtime/glnxa64`
  - `setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${MCRROOT}/bin/glnxa64`
  - `setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${MCRROOT}/sys/os/glnxa64`
  - `setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${MCRROOT}/sys/java/jre/glnxa64/jre/lib/amd64/native_threads`
  - `setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${MCRROOT}/sys/java/jre/glnxa64/jre/lib/amd64/server`
  - `setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:${MCRROOT}/sys/java/jre/glnxa64/jre/lib/amd64`
  - `setenv XAPPLRESDIR ${MCRROOT}/X11/app-defaults`

- Generates genome-wide normalized (but not control normalized for ChIP-seq) signal tracks in bedGraph and/or bigWig format

Program(s)	• WIGGLER
Input(s)	• tagAlign/BAM files • directory of all fasta sequences by chromosome (hg19 male/female) • mappability files from http://www.broadinstitute.org/~anshul/projects/umap/
Output(s)	• Wiggle or bedGraph signal files
Commands	• `Usage: align2rawsignal -i=<alignFname> -s=<seqDir> -of=bg -l=<fragLen> -mm=<memory>`
Parameters	• -i=<alignFname> (MANDATORY, MULTIPLE ALLOWED). One or more tagAlign/BAM files (replicates) as input. • -s=<seqDir> (MANDATORY). Full path to directory containing chromosome fasta files (eg. chr1.fa ..) The file names MUST match the chromosome names used in the tagAlign files • -u=<uMapDir> (MANDATORY). Full path to directory containing binary mappability tracks. The directory name must be of the form [PATH]/globalmap_k<min>tok<max> • -o=<oFname> (OPTIONAL). Full path and name of output signal file. • -of=<outputFormat> (OPTIONAL). Output signal file format wiggle (wig) or bedGraph (bg) or matfile (mat) • -l=<fragLen> (OPTIONAL, MULTIPLE ALLOWED). Fragment-length == 2*Tag-shift. Default: 1 (no extension)* • -w=<smoothingWindow> (OPTIONAL). Smoothing window size for signal. Default: mean(1.5*fragLen) • -k=<smoothingKernel> (OPTIONAL). Smoothing kernel to use. Valid kernels (rectangular,triangular,epanechnikov,biweight,triweight,cosine,gaussian,tukey) Default: tukey (with taper ratio of max(0.25 , min (0.5,max(w-mean(l),0)/(2*mean(l)))) • -mm=<memory> (OPTIONAL). Total memory to use in GB. Default: 2
QC to report	• None.

- *NOTE: Multiple arguments of this type are allowed. In such a case, number of fragLen arguments MUST BE == no. of Align files. The ORDER of these arguments is important.
- e.g. The first <fragLen> is matched with the first tagAlign/BAM file and so on.
- It might make sense to use the estimated fragment lengths reported in 2b here (to be verified with Anshul).
- Use bedGraphToBigWig from the Kent source tree to convert into bigWig format

6. HOTSPOT/SPOT - QC measure calculating the percentage of tags (reads) that fall into hotspots (Anshul needs to clarify the run details).

Program(s)	• HOTSPOT/SPOT

Input(s)	● tagAlign/BAM files ● Mappability file (unclear if this is the same as those needed by WIGGLER)
Output(s)	● minimally thresholded hotspots BED file ● FDR thresholded hotspots BED file ● FDR thresholded peaks BED file ● FDR thresholded smoothed density at peaks file ● file listing z-scores for each hotspot containing peak ● file listing binomial p-values for each hotspot containing peak ● SPOT score in spot.out file
Commands	● ./runhotspot ● Parameters are specified in the runall.tokens.txt file
Parameters	● Set _DUPOK_ = F for ChIP-seq to not allow duplicate reads ● Set _K_ to the appropriate read length corresponding to mappability file ● Set _OMIT_REGIONS_ to include those in blacklist(?) ● Set _FDR_ = N to just calculate SPOT score (unclear if this relies on Hotspot output from a previous full run)
QC to report	● SPOT score

TFs to use as possible example data:

- GATA1 in K562 (“We found 2785 reproducible GATA1 ChIP-seq peaks” - Gerstein et al., suppl).
- RAD21 in SK-N-SH_RA: wgEncodeHaibTfbsSknshraRad21V0416102AlnRep1 - this has the highest SPOT value after CTCF and Pol II (and of all the TFs)
- GABP in K562: wgEncodeHaibTfbsK562GabpV0416101AlnRep4 - this had PCR bottleneck problems (~0.4), but it could also be because it binds very specifically to a small number of places in the genome. It has high NSC and RSC
- KAP1 in U2OS: wgEncodeSydhTfbsU2osKap1UcdAlnRep2 - this had relatively low NSC (1.02) and low RSC (0.78) which may mean it was a bad experiment. This was flagged (This was not imported or in Anshul’s list)
- CFOS in GM12878: CFOSwgEncodeSydhTfbsGm12878CfosStdAlnRep1 - this was flagged for low number of uniquely mapped reads. It also has 3 replicates