Automated quality control and reproducible peak calling for transcription factor ChIP-seq data

Jin Wook Lee[2,*], Youngsook L. Jung[3,*], Qunhua Li[4,*], Georgi K. Marinov[2,*], Nathan Boley[2,*], J. Seth Strattan[2], Peter V. Kharchenko[3,5], Yuchun Guo[11], Arif Harmanci[12], Dongjun Chung[13], Tao Liu[19], Robert E. Thurman[21], Noam Shoresh[20], Russell Bonneville[14], Florencia Pauli-Behn[15], Trupti Kawli[2], Joel Rozowsky[6], James B. Brown[7], Cricket A. Sloan[2], Benjamin C. Hitz[2], Esther T. Chan[2], Jason A. Hilton[2], Arend Sidow[2,9], Serafim Batzoglou[1], Michael P. Snyder[2], Richard M. Myers[15], Peggy J. Farnham[16], Victor X. Jin[17], Mark Gerstein[12], David Gifford[11], Sunduz Keles[18], Barbara J. Wold[8], Peter J. Bickel[10], J. Michael Cherry[2], Peter J. Park[3,#], and Anshul Kundaje[1,2,*,#]

[1] *Department of Computer Science, Stanford University, Stanford, California, 94305, USA*
[2] *Department of Genetics, Stanford University School of Medicine, Stanford, CA 94305, USA*
[3] *Department of Biomedical Informatics, Harvard Medical School, Boston, Massachusetts, 02115, USA*
[4] *Department of Statistics, Penn State University, University Park, Pennsylvania, 16802, USA*
[5] *Division of Hematology Oncology, Boston Children's Hospital, Massachusetts, 02115, USA*
[6] *Department of Molecular Biophysics and Biochemistry, Yale University, Yale, Connecticut, 06520, USA*
[7] *Lawrence Berkeley Laboratory, Berkeley, California, 94710, USA*
[8] *Division of Biology and Beckman Institute, California Institute of Technology, California, 91125, USA*
[9] *Department of Pathology, Stanford University, Stanford, California, 94305, USA*
[10] *Department of Statistics, University of California at Berkeley, Berkeley, California, 94710, USA*
[11] *Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*
[12] *Program in Computational Biology and Bioinformatics, Yale University, New Haven, Connecticut, 06520, USA*
[13] *Division of Biostatistics and Bioinformatics, Medical University of South Carolina, Charleston, South Carolina, 29425, USA*
[14] *Comprehensive Cancer Center, The Ohio State University College of Medicine, Columbus, Ohio, 43210, USA*
[15] *HudsonAlpha Institute for Biotechnology, Huntsville, Alabama, 35806, USA*
[16] *Department of Biochemistry and Molecular Medicine, University of Southern California, Los Angeles, California, 90089, USA*
[17] *Department of Molecular Medicine, University of Texas Health Science Center, San Antonio, Texas, 78229, USA*
[18] *Department of Biostatistics and Medical Informatics, University of Wisconsin, Madison, Wisconsin, 53705, USA*
[19] *Department of Biochemistry, The State University of New York, Buffalo, New York, 14203, USA*
[20] *Broad Institute, Cambridge, Massachusetts, 02142, USA*
[21] *Department of Genome Sciences, University of Washington, Seattle, Washington, 98195, USA*
[*] *These authors contributed equally to this work*
[#] *Correspondence should be addressed to A.K. (akundaje@stanford.edu) and P.J.P. (peter_park@harvard.edu).*

Abstract

Chromatin immunoprecipitation with high-throughput sequencing (ChIP-seq) is the predominant assay for generating genome-wide maps of protein-DNA interactions. We present AQUAS (Automated QUality control and reproducibility Analysis of ChIP-Seq data), an end-to-end protocol for robust data processing and quality control of transcription factor ChIP-seq experiments. AQUAS supports several complementary data quality indicators, including a strand cross-correlation enrichment measure, and uses the Irreproducibility Discovery Rate (IDR) framework for identifying reproducible binding events from replicate experiments. We provide standalone and cloud-based software implementations that support resumable distributed workflows, user-friendly analysis reports and visualization via genome-browser sessions. The protocol has been used by the ENCODE project to analyze over 6,500 ChIP-seq datasets. AQUAS enables investigators to critically assess and process their TF ChIP-seq data in concordance with ENCODE standards. The workflow takes ∼1 day to process a typical pair of single-end or paired-end TF ChIP-seq replicate experiments.

Introduction

Chromatin immunoprecipitation with high-throughput sequencing (ChIP-seq) is an assay for profiling protein-DNA interactions and chromatin modifications genome-wide. It was one of the first applications of next-generation sequencing technologies for functional genomics and remains one of the most widely used assays for regulatory and epigenomic profiling of genomes[1,2]. However, ChIP-seq experiments are sensitive to a myriad experimental parameters as well as the properties of the target protein/marker. They can thus exhibit significant variation in data quality, and default analysis workflows can often result in unstable or poorly calibrated results[3]. As the number of ChIP-seq datasets continues to grow rapidly in individual laboratories[4] and large coordinated consortia[5–8], there is a need for robust protocols for automated large-scale data processing and objective, calibrated measures of data quality that enable systematic critical evaluation of datasets. Here, we present AQUAS (Automated QUality control and reproducibility Analysis of ChIP-Seq data), an end-to-end protocol with stand-alone and cloud-based software implementations for large-scale, automated data processing and quality control of ChIP-seq datasets targeting proteins with punctate binding landscapes.

A typical ChIP-seq experiment consists of a series of steps including antibody selection and validation, sample preparation, co-immunoprecipitation, DNA library preparation, and sequencing[9,10]. Critical factors affecting data quality are antibody specificity and efficiency, the quality of sonication, the size selection protocol and the resulting distribution of immunoprecipitated DNA fragment sizes, library complexity, the presence of PCR artifacts, sequencing read quality, and sequencing depth[11,12]. Downstream computational analysis involves mapping of sequenced reads to the genome, creating genome-wide signal coverage tracks, and identifying regions of enrichment (i.e., peaks), often in comparison to a corresponding control experiment (i.e., input DNA). An ideal ChIP-seq experiment should have high specificity and sensitivity, i.e., significant clustering of immunoprecipitated (IP) fragments centered at the genuine binding (or target) sites of the protein (or chromatin mark) of interest with a low background signal throughout the rest of the genome. A truly comprehensive ChIP-seq experiment must also be sufficiently "saturated" so as to allow discovery of the majority of true target sites. Replicate experiments should also be performed to ensure the identification of reproducible target sites. Biological replicates are strongly preferred, since most of the variation comes from variability in biological conditions and lack of specificity for the antibody.

Users have frequently resorted to visualizing the processed data in a genome browser, such as the UCSC Genome Browser[13], WashU Epigenome Browser[14], and IGV[15], in order to make a subjective judgment on data quality and reproducibility between replicates. While this practice is often useful, a systematic and automated analysis is necessary to ensure that the assessment is unbiased, especially when less is known about the protein or modification of interest. Objective measures also allow researchers to compare and evaluate samples generated using different reagents and by different laboratories. The need for such analyses is more acute in projects that generate a large number of ChIP-seq experiments, since it is not feasible to visually inspect each profile.

A large number of tools commonly referred to as "peak callers" have been developed for identifying *in vivo* binding events of transcription factors (which typically have punctate, sharp peaks of enrichment) and other chromatin-associated proteins as well as the genomic landscapes of histone variants and modifications (which typically exhibit broad regions of enrichment)[16,17]. However, the choice of peak caller and its parameters can influence analysis results significantly. Default analysis parameters can produce vastly differing results from different peak callers applied to the same data. Moreover, default parameters of peak callers often cannot generalize to the diversity of binding landscapes of DNA binding proteins in terms of numbers of binding sites and signal-to-noise ratios. Users are often left with no choice but to manually tune parameters to satisfy subjective definitions of optimality – an approach that is not scalable. Hence, there is a need for automated methods for robust and reliable peak calling that can adapt to the properties to the data.

The protocol we describe here has been developed as part of the Encyclopedia of DNA Elements (ENCODE) Project Consortium. ENCODE has been generating a massive reference compendium of ChIP-seq data for a wide range of target proteins and histone modifications across multiple cellular contexts and organisms[5–7,18], necessitating the development of a robust and automated quality control and statistical analysis pipeline. Many of the guidelines and practices of the Consortium with respect to ChIP-seq data have been summarized previously[11]. Here we describe the quality control and reproducibility steps in greater detail together with the further development of the automated pipeline for applying them to ChIP-seq datasets targeting transcription factors and other regulatory DNA binding proteins with punctate binding events.

The overview of the AQUAS protocol is shown in Figure 1. Briefly, single-end or paired-end sequenced reads from two or more replicates of a TF ChIP-seq experiment and matched controls are mapped to a reference genome using one of two popular short-read aligners. Read mapping quality control (QC) statistics are recorded. QC measures evaluating the molecular complexity of sequencing libraries are computed. Strict filtering criteria are used to discard poor quality and unreliably mapped reads. Putative binding events are identified as regions of enrichment (peaks) of ChIP signal relative to control using one of several peak callers. The Irreproducible Discovery Rate (IDR) framework is used to adaptively threshold and retain peaks that are reproducible and rank-concordant across replicates.

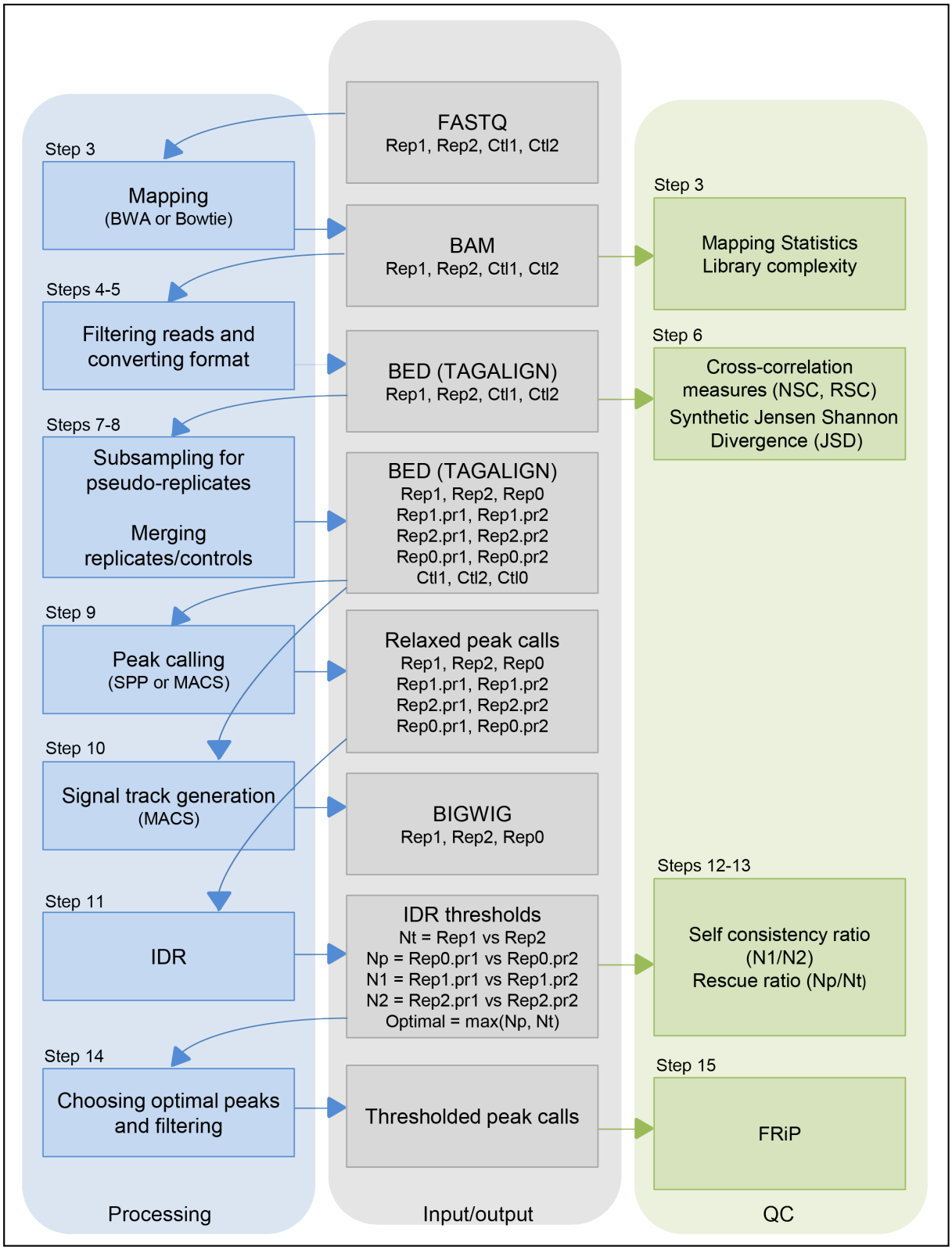


Figure 1: The workflow of the AQUAS pipeline. Cross-correlation measures and IDR analysis in AQUAS are integrated with other quality control measures. `Rep0` is the merged file from replicates 1 (`Rep1`) and 2 (`Rep2`); `Ctl0` is the merged file from control replicates 1 (`Ctl1`) and 2 (`Ctl2`); `pr1` or `pr2` stands for pseudo-replicate 1 or 2. N_t and N_p are the replicate and the pooled-pseudoreplicate thresholds, respectively. N_1 and N_2 are self-consistency thresholds for Rep1 and Rep2, respectively (see the Experimental Design section for further details).

Box 1 | Irreproducible Discovery Rate (IDR)

IDR is a statistical method to assess the reproducibility of entries on rank lists[19]. Given a set of peak calls, the peaks can be ranked based on a criterion of significance, such as the p-value, the q-value, or the fold enrichment. Significant peaks generally are ranked more consistently across the replicates than the peaks with low significance. This change of consistency provides an indicator of the transition from real signal to noise. IDR quantifies this transition by modeling the peaks as a mixture of a reproducible and an irreproducible group using a copula mixture model, where the peaks in the reproducible group are ranked higher and more consistently across replicates than the ones in the irreproducible group. It assigns each peak a reproducibility index, called the local IDR, to estimate its probability of being reproducible, and also reports the expected rate of irreproducible discoveries in the selected peaks (called IDR) in a fashion analogous to that of false discovery rate (FDR). Similar to FDR, peaks with lower IDR are more reliable. Hence, an IDR threshold can be used as a reproducibility-based criterion to determine the rank threshold of the peaks at which signal transits to noise. The IDR thresholds exhibit significantly more stable behavior than FDR thresholds provided by peak callers. A single IDR threshold can be used comparably across datasets as it reflects the inherent sampling reproducibility of a dataset and is not significantly affected by antibodies, labs, platforms and analysis protocols[11]. When using IDR, a relatively relaxed peak-calling threshold is advised because the IDR algorithm requires sampling of both signal and noise distributions to assess the reproducibility of peaks.

QC measures of reproducibility of peaks within and across replicates are computed[19]. Genome browser-friendly signal tracks recording the fold enrichment of ChIP signal over control as well as the statistical significance of enrichment are computed for individual replicates and pooled reads. Several QC measures estimating signal-to-noise ratio and ChIP enrichment are computed. Strand cross-correlation QC metrics capture the degree of fragment clustering in a dataset by accounting for the asymmetric, stranded structure of read distributions expected near true binding sites. The Jensen-Shannon Divergence is an information-theoretic measure of signal-to-noise ratio. Both of these measures can be efficiently computed after read alignment but before peak calling, thereby providing early feedback on data quality independent of peak calling algorithms and parameters. The fraction of reads that fall within reproducible peaks is also reported as a peak-calling dependent measure of enrichment. This protocol has been applied to thousands of diverse TF ChIP-seq datasets from the ENCODE and modENCODE[5–7,18] consortia, as well as other independent data sets[3,20,21].

Applications of the method

This protocol was developed and tested primarily for ChIP-seq data. However, parts of it can be used for other data types or other purposes. The cross-correlation function and metrics can be applied to assess the fragment length distribution and the extent of fragment clustering of a wide variety of data types, while the peak calling procedures centered around reproducibility analysis can be effectively used to identify consensus sets of peaks in chromatin accessibility profiling assays such as DNase-seq[22], FAIRE-seq[23] and ATAC-seq[24].

Comparison with other methods

There are several methods for evaluating the quality of high-throughput sequencing experiments[25–27]. However, many measures, while useful, are not specific to ChIP-seq and provide only limited information about the overall quality of an experiment. For example, the distribution of quality scores along read positions or the fraction of reads that map confidently to the target genome are often used to determine sequencing quality[25] but provide no information about the success of the ChIP reaction. A number of ChIP-seq-specific quality metrics have been developed[28–30]. Diaz et al. estimated the strength of ChIP signals by decomposing them into real IP signal and background signal, and calculating the percentage of the IP signals[28]. Planet et al. assessed IP enrichment using the measure of the dispersion of read density by estimating the standard deviation and the Gini index[29]. Mendoza-Parra et al. estimated the stability of peaks by resampling[30]. However, cross-correlation measures have numerous advantages over these approaches: they directly evaluate the extent of fragment clustering, they provide information about the fragment length distribution of a dataset, and they can be easily implemented as an intermediate step in a typical peak calling pipeline as we demonstrate here. In addition, researchers can compare the cross-correlation measures with those for the large collection of TFs that have been profiled by the ENCODE and modENCODE consortia (`http://genome.ucsc.edu/ENCODE/qualityMetrics.html` and `https://www.encodeproject.org`).

To ensure that experimental results are reproducible, one may examine reproducibility either at the level of reads or identified peaks, or both[11]. The reproducibility of read coverage is often measured by computing the Pearson correlation coefficient of the (mapped) read counts at each genomic position[31]. However, this quantity can be dominated by a small number of very highly enriched regions.

In our approach, we assess the reproducibility of identified peaks between replicates using IDR analysis [19] (Box 1). The IDR pipeline reports the number of peaks that pass a user-specified reproducibility threshold (e.g., IDR = 0.05), allowing one to select a reproducible set of peaks and to identify experiments with low reproducibility. A major advantage of the IDR method is that it is rank-based; it is therefore independent of peak-calling algorithms and can be applied to a range of significance criteria across labs and platforms. It has been shown to produce a stable threshold that is more consistent across laboratories, antibodies, and analysis protocols (e.g. peak callers) than FDR measures [19].

Experimental Design

Aligning reads. Raw sequencing data in FASTQ format can be obtained from a sequencing facility or from public repositories, such as GEO (`http://www.ncbi.nlm.nih.gov/geo/`) and the ENCODE portal (`encodeproject.org`). Including a large number of reads with poor quality scores in an analysis can potentially result in mapping artifacts. It is therefore advisable to perform a QC step at the level of the raw reads. Read quality histograms are a simple way to summarize overall read quality. One can compute the histogram of PHRED scores per position in all reads that map to some genomic location as well as the mean score per position. High quality sequenced reads from the Illumina sequencing platform tend to have average PHRED scores >30. Average PHRED scores <20 reflect problems with read quality. Reads can then be mapped against a reference genome using a variety of aligners, such as BWA [32], Bowtie [33] and Bowtie2 [34].

Mapping and sequencing statistics. For each dataset, we compute (i) the total number of reads, (ii) the number and fraction of reads that map to the genome, and (iii) the number (N_u) and fraction (f_u) of uniquely aligned reads. Human ENCODE datasets with $N_u < 8$ million are flagged as having suboptimal sequencing depth. Datasets with abnormally low f_u values (<60%) tend to have poor read quality or other data quality issues (although exceptions are possible, for example, a highly enriched ChIP-seq experiment targeting repressive histone modificatons associated with repetitive elements can exhibit a high fraction of multimapping reads, and, accordingly, a low fraction of uniquely mappable reads).

Post-alignment filtering. All analyses are subsequently performed on alignment files containing only uniquely mapped reads. We remove alignments that are either unmapped or a non-primary alignment using `samtools`.

Library complexity measures. If a sequencing library has an inadequate representation of unique DNA molecules, a single molecule may give rise to a very large number of clones during the PCR amplification process. This PCR "bottlenecking" results in many more reads mapping to the same location than expected from a library with greater complexity, which can potentially lead to peak calling and other analytical artifacts. We use two metrics to assess library complexity:

- PCR Bottleneck Coefficient (PBC). Let M_0, M_1, and M_2 be the numbers of distinct genomic locations to which at least one, exactly one, and exactly two reads map uniquely, respectively. We define two simple heuristic PCR Bottleneck Coefficients ($PBC1$ and $PBC2$) as follows:

$$PBC1 = M_1/M_0 \tag{1}$$

$$PBC2 = M_1/M_2 \tag{2}$$

  High $PBC1$ and $PBC2$ scores ($PBC1 \in [0,1]$, $PBC2 \geq 1$) indicate that the dataset has low bottlenecking (i.e. high-molecular complexity), while low scores imply that it has high bottlenecking (i.e. low complexity).

- Non-Redundant Fraction (NRF). Let U_P be the set of genomic positions to which $5'$ ends of reads map uniquely. Let U_R be the total number of uniquely mapped reads. We define the Non-Redundant Fraction (NRF) as follows:

$$NRF = U_P/U_R \tag{3}$$

  A high NRF score ($NRF \in (0,1)]$) indicates that the dataset has a high molecular complexity.

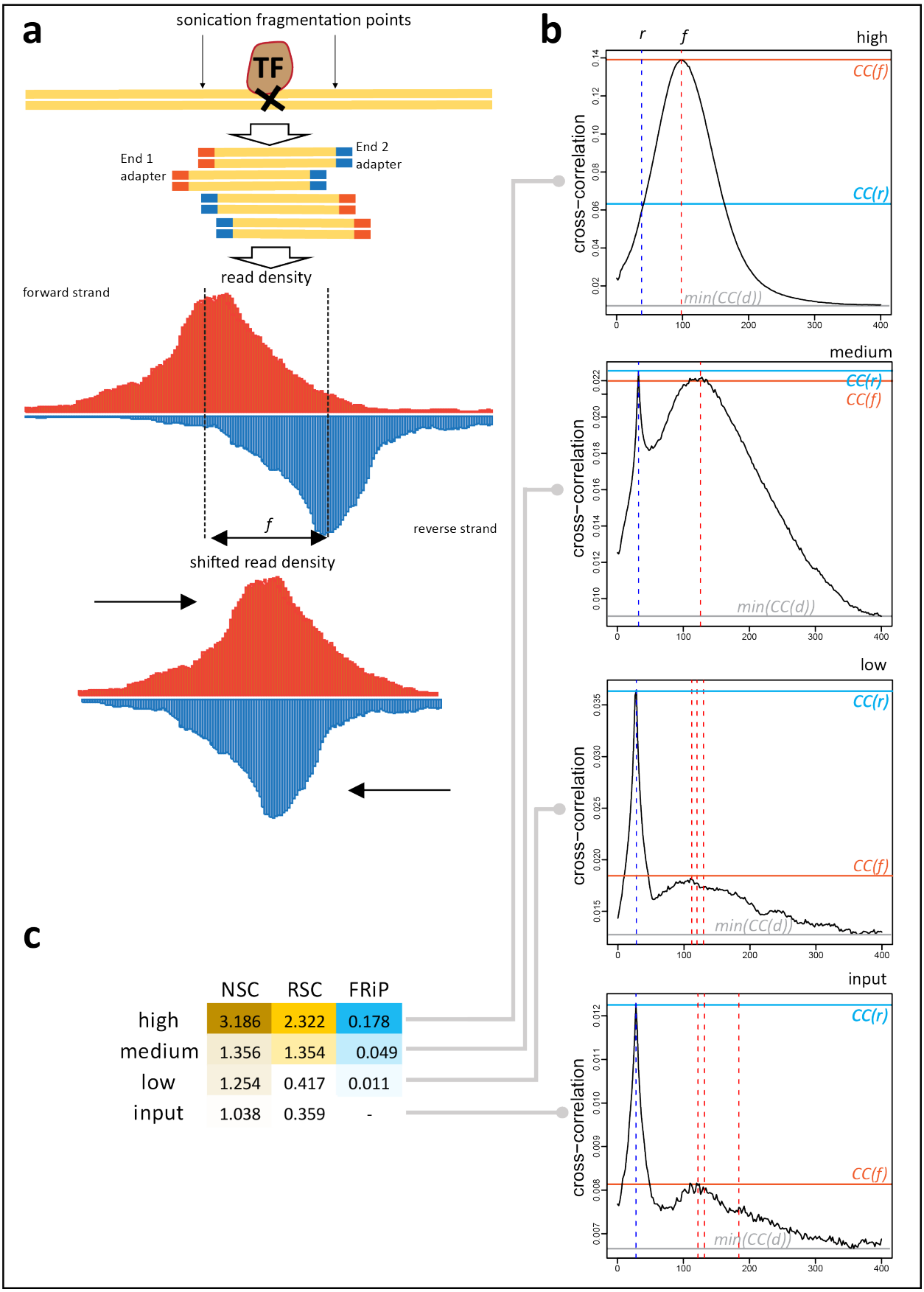
Converting file formats. Although the cross-correlation analysis can take BAM files, the current implementation of the pipeline converts them into the tagAlign file format as it is more convenient for the downstream IDR analysis.

FRiP metric. FRiP (Fraction of Reads in Peaks) is the simplest metric for evaluating the extent of enrichment in a ChIP dataset [11]. It is defined as:

$$FRiP = \frac{|\text{mapped reads within called peaks}|}{|\text{all mapped reads}|} \tag{4}$$

FRiP values can range from 0 to 1 (although in practice values significantly above 0.5 are very rarely observed in mammalian-sized genomes), with higher values indicating greater IP enrichment. FRiP's main shortcoming is that it is highly dependent on the particulars of the peak calling algorithm applied, as the number of peaks called and the width/narrowness of their definition greatly impact the final FRiP value. Nevertheless, if peak calling has been carried out in a uniform way, FRiP can be an informative metric for comparing datasets. We compute FRiP scores for post-IDR output as part of the AQUAS pipeline (see further below for more details).

Strand cross-correlation measures. To calculate cross-correlation metrics, stranded coverage vectors are first


a
sonication fragmentation points
TF
End 1
adapter
End 2
adapter
read density
forward strand
reverse strand
f
shifted read density
b
r
f
high
CC(f)
CC(r)
min(CC(d))
cross−correlation
medium
low
input
c
NSC RSC FRiP
high 3.186 2.322 0.178
medium 1.356 1.354 0.049
low 1.254 0.417 0.011
input 1.038 0.359 -

computed for each chromosome by calculating the number of uniquely mapped reads at each genomic location on the positive and negative strands (Figure 2). We mask genomic locations that show artifactual stacking of reads, i.e., singular genomic locations with very high read counts as compared to surrounding locations[17]. The strand cross-correlation profile (CCP) for each chromosome is computed as the Pearson correlation coefficient between coverage vectors of the positive and negative strands shifted by specific distances towards (positive shifts) or away (negative shifts) from each other. We typically use strand shifts ranging from −500 bp to 1500 bp at steps of 5 bp. We then compute a weighted average of the CCPs over all chromosomes, using the fraction of reads mapped to each chromosome as weights. All computations are performed using `R` scripts in AQUAS based on the functions in the `SPP` peak caller package[17].

We define a Normalized Strand Cross-correlation coefficient (NSC) as the peak height of the CCP divided by a background cross-correlation value which is defined as the minimum cross-correlation over a sufficiently wide range of strand shifts (typically −500 bp to ∼3 times the expected size of DNA fragments based on sonication and size selection protocols; see also Box 2). NSC represents the enrichment of localized clustering of relatively fixed-size IP fragments at potential target sites, and the normalization with respect to the background makes the measure comparable across datasets. The smallest possible NSC value is 1 and larger values represent greater enrichment. Based on calibration using a comprehensive analysis of control datasets (input DNA), for the human genome, NSC values lower than 1.1 typically represent datasets with low enrichment. Datasets with tight fragment size distributions and a large number of narrow peaks with strong signal compared to background show higher NSC values.

Poorly-enriched or failed datasets typically exhibit a low degree of localized fragment clustering. In the extreme case, we observe a predominant peak in the CCP at a strand shift equal to the read length and a significantly weaker peak at a strand shift equal to the predominant fragment length (Figure 2b, lower two panels). This "read-length peak" has been previously noted and attributed to variable local mappability of the genome[35]. To elaborate, if reads map uniquely to a genomic position (chr, i) on the positive strand then this implies that reads can also map uniquely to the position $(chr, i + RL)$, where RL is the read length, on the negative strand. Hence, in the absence of any significant fragment clustering and variable mappability across the genome, at random, on average, there is a higher probability of observing reads mapping to positions on the positive and negative strand separated by a distance equal to the read length than any other strand shift when restricting analysis to uniquely mappable reads only.

This read-length cross-correlation peak can serve as an internal reference point that can be used to normalize the fragment-length peak. Thus, we define the Relative Strand Cross-correlation coefficient (RSC) as the height of the fragment length peak divided by the height of the read-length peak, after adjusting for the background from each (See Box 2). Highly enriched ChIP-seq datasets show RSC values greater than 1 (a dominant fragment-length peak); RSC

Figure 2 *(preceding page)*: The anatomy of a cross-correlation profile (CCP) for a ChIP-seq dataset. **(a)** Computation of the cross-correlation function and the biochemical rationale behind its application to ChIP-seq. Transcription factors generally bind to specific sequence motifs precisely positioned within DNA. Crosslinking and sonication result in DNA fragments centered around the site of transcription factor binding (as well as a background population of fragments from the rest of the genome). Sequencing library preparation adapters are randomly ligated to the two ends of these fragments, which results in the typical asymmetric read distribution on the forward and reverse DNA strands around the transcription factor binding site after immunoprecipitation, with the peaks on the two strands being separated by the average DNA fragment size f. The cross-correlation function is computed by shifting the read distributions on the two strands relative to each other and computing the cross-correlation function at each shift value. The bottom panel shows that two peaks exhibit maximum overlap when reads on the reverse strand are shifted towards the positive strand ($5'$ to $3'$ direction) by f. **(b)** Cross-correlation profiles (CCP) computed as the Pearson correlation between the positive strand profiles and negative strand profiles at different strand shift distances d (for detailed description, see the Experimental Design section and Box 2). Typical CCPs are shown for high-quality, medium-quality, low-quality, and input datasets, respectively, from top to bottom. Blue dashed vertical lines: d = read-length (r); blue solid horizontal lines: the magnitude of the read-length peak; red vertical lines: $d = f$; orange solid horizontal lines: the magnitude of the estimated fragment length peaks; gray solid horizontal lines: minimum values of CCP in the given range. Note that the estimated fragment length peak becomes stronger relative to the read length peak for datasets with greater enrichment. Also note that in the actual implementation of the pipeline cross-correlation is computed over a range of shift values $d \in [-500, 1500]$, but the narrower $d \in [0, 400]$ range is shown here for clarity. The high-quality dataset is BAM file ID ENCFF901DWV from ENCODE experiment ID ENCSR000BMY, the medium-quality one is ENCFF757ZWI from ENCSR000EBQ, the low-quality one is ENCFF000VSZ from ENCSR000EYZ, the control is ENCFF487SIK from ENCSR000EAF. **(c)** NSC (Normalized Cross-correlation Coefficient) and RSC (Relative Cross-correlation Coefficient) are measures of data quality based on parameters derived from CCPs. FRiP (Fraction of Reads in Peaks) is another measure of ChIP enrichment. The values of these metrics for the datasets shown in (b)) are displayed here.

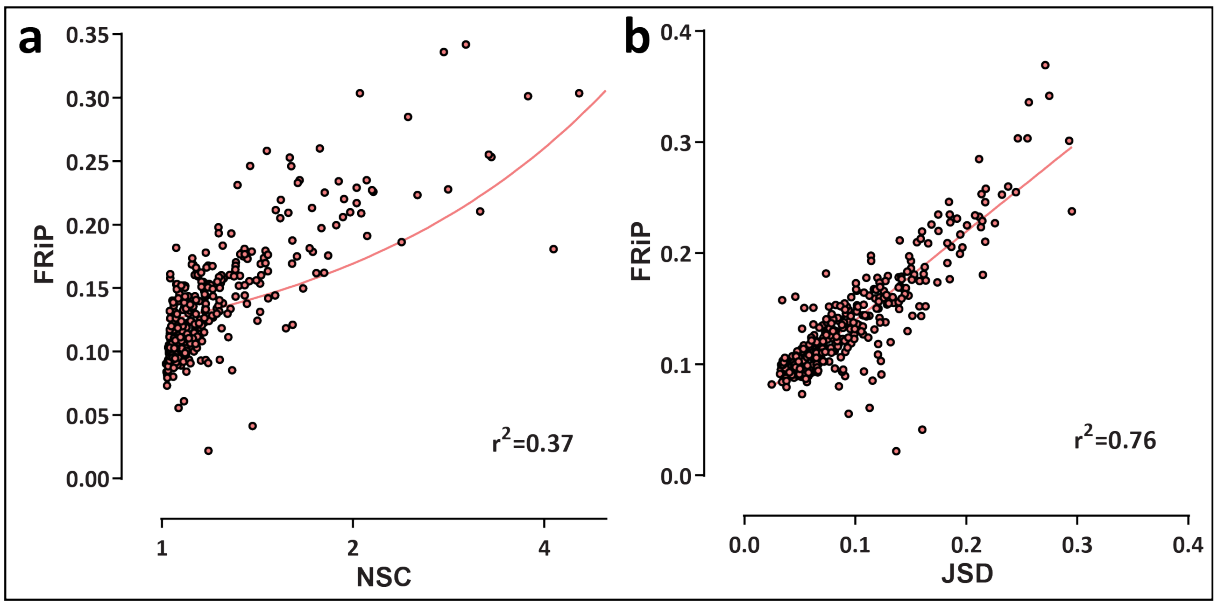


Figure 3: Correlation between quality control metrics used in AQUAS. A set of ∼500 ENCODE transcription factor ChIP-seq datasets were processed using the AQUAS pipeline and quality control metrics were calculated for each. **(a)** Cross-correlation ChIP quality control metrics (NSC) compared to FRiP. **(b)** Synthetic Jensen-Shannon Divergence JSD metric compared to FRiP.

Box 2 | The definitions of cross-correlation measures (NSC, RSC and Qtag)

Let $CC(d)$ be the strand cross-correlation function at a shift d (see Figure 2). It is calculated as follows:

$$CC(d) = \sum_{c \in C} \frac{|R_c|}{\sum_{c \in C} |R_c|} CC(d)_c$$

where C is the set of all individual chromosomes c, and R_c is the set of all reads uniquely mapping to a chromosome c.

The normalized strand cross-correlation (NSC) coefficient is defined as:

$$NSC = \frac{max_{d \in (d_{min}, r-\Delta) \cup (r+\Delta, d_{max})}(CC_d)}{min_{d \in (d_{min}, d_{max})}(CC_d)}$$

where r is the read length, Δ is a flanking region near the read length, d_{min} and d_{max} are the minimum and maximum value of strand shift examined, respectively. This coefficient represents the peak height of the CCP ignoring any peak at a strand shift equal to the read length, normalized by the minimum CCP value over a large range of strand shifts. NSC represents the enrichment of localized clustering of relatively fixed-sized IP fragments and is an effective continuous measure of ChIP-seq data quality.

The relative strand cross-correlation coefficient (RSC) is defined as:

$$RSC = \frac{max_{d \in (d_{min}, r-\Delta) \cup (r+\Delta, d_{max})}(CC_d) - min_{d \in (d_{min}, d_{max})}(CC_d)}{CC(r) - min_{d \in (d_{min}, d_{max})}(CC_d)}$$

where r is the read length, Δ is a flanking region near the read length, d_{min} and d_{max} are the minimum and maximum value of strand shift examined, respectively. RSC represents the relative value of the cross-correlation maximum at the estimated fragment length (excluding the read-length peak) to the value at the read length with respect to the minimum cross-correlation across all strand shifts. For convenience, we also define a discretized version we call a quality tag (Qtag) as follows:

If $RSC \geq 1.5 \rightarrow Qtag = 2$ (Very High)
If $RSC \in [1, 1.5) \rightarrow Qtag = 1$ (High)
If $RSC \in [0.5, 1) \rightarrow Qtag = 0$ (Medium)
If $RSC \in [0.25, 0.5) \rightarrow Qtag = -1$ (Low)
If $RSC < 0.25 \rightarrow Qtag = -2$ (Very Low).

Box 3 | Cross-correlation analysis in AQUAS

1. Input file format
   (1) BAM format
   This is a binary alignment format specified in `https://genome.ucsc.edu/goldenPath/help/bam.html`
   (2) TagAlign files
   This a text-based BED3+3 alignment format that is easier to manipulate. It contains 6 tab delimited columns.

   COL1: `chrom` `string` Name of the chromosome
   COL2: `chromStart` `int` The starting position of the feature in the chromosome. The first base in a chromosome is numbered 0.
   COL3: `chromEnd` `int` The ending position of the feature in the chromosome or scaffold. The `chromEnd` base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as `chromStart` = 0, `chromEnd` = 100, and span the bases numbered 0–99.
   COL4: `sequence` `string` Sequence of this read
   COL5: `score` `int` Indicates uniqueness or quality (preferably 1000/`alignmentCount`).
   COL6: `strand` `char` Orientation of the read (+ or -)

2. Output file
   The output is a tab-delimited text file containing the following columns.

   COL1: `Filename` : tagAlign/BAM filename
   COL2: `numReads` : effective sequencing depth i.e. total number of mapped reads in input file
   COL3: `estFragLen` : comma separated strand cross-correlation peak(s) in decreasing order of correlation.
   The top 3 local maxima locations that are within 90
   In almost all cases, the top (first) value in the list represents the predominant fragment length.
   If it is desired to keep only the top value simply run
   `sed -r 's/,[^\t]+//g' <outFile> > <newOutFile>`
   COL4: `corr_estFragLen` : comma-separated strand cross-correlation value(s) in decreasing order (col2 follows the same order)
   COL5: `phantomPeak` : read length/phantom peak strand shift
   COL6: `corr_phantomPeak`: correlation value at phantom peak
   COL7: `argmin_corr` : strand shift at which cross-correlation is lowest
   COL8: `min_corr` : minimum value of cross-correlation
   COL9: : normalized strand cross-correlation coefficient (NSC) = COL4 / COL8
   COL10: : relative strand cross-correlation coefficient (RSC) = (COL4 - COL8) / (COL6 - COL8)
   COL11: `QualityTag` : quality tag based on thresholded RSC (codes: -2:veryLow, -1:Low, 0:Medium, 1:High, 2:very-High)

3. Parameter options
   Optional arguments:

   `-s=<min>:<step>:<max>`, strand shifts at which cross-correlation is evaluated, default=`-500:5:1500`
   `-speak=<strPeak>`, user-defined cross-correlation peak strandshift
   `-x=<min>:<max>`, strand shifts to exclude (This is mainly to avoid region around phantom peak) default=`10:(readlen+10)`
   `-p=<nodes>`, number of parallel processing nodes, default=0
   `-fdr=<falseDisoveryRate>`, false discovery rate threshold for peak calling
   `-npeak=<numPeaks>`, threshold on number of peaks to call
   `-tmpdir=<tempdir>`, temporary directory (if not specified R function `tempdir()` is used)
   `-filtchr=<chrnamePattern>`, pattern to use to remove tags that map to specific chromosomes e.g. `_\verb` will remove all tags that map to chromosomes with _ in their name

   Output arguments

   `-odir=<outputDirectory>`, name of output directory (if not set explicitly, the directory of the ChIP file is used)
   `-savn=<narrowpeakfilename>` OR `-savn`, narrowPeak file name (fixed width peaks)
   `-savr=<regionpeakfilename>` OR `-savr`, regionPeak file name (variable width peaks with regions of enrichment around peak summits)

`-savd=<rdatafile>` OR `-savd`, save Rdata file
`-savp=<plotdatafile>` OR `-savp`, save cross-correlation plot
`-out=<resultfile>`, append peakshift/phantomPeak results to a file
`-rf`, if a plot, rdata or narrowPeak file exists, replace it. If not used, then the run is aborted if the plot, Rdata or narrowPeak files exist
`-clean`, if used, it will remove the original ChIP and control files after reading them in.

values markedly smaller than 1 (a dominant read-length peak) are a sign of potential problems with ChIP enrichment or sub-optimal sequencing depths. RSC scores can be discretized into easy to interpret integer scores, which we refer to as "Qtag" (see See Box 3). High-quality datasets tend to display higher values of NSC, RSC or Qtag as well as a larger number of detected peaks and a higher fraction of reads residing within the peaks compared to low quality datasets (Figure 2c).

Additional enrichment metrics. In addition to cross-correlation, we also compute the Jensen-Shannon distance (JSD) for a sample as previously defined[36] and implemented by the `deepTools`[37] package. The JSD metric is defined as follows:

$$JS = \sqrt{JSD(P||Q)}$$

Where JSD is the Jensen-Shannon divergence:

$$JSD(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M)$$

where

$$M = \frac{1}{2}(P+Q)$$

and

$$D_{KL}(X||Y) = \sum_i X_i log\left(\frac{X_i}{Y_i}\right)$$

Where D_{KL} is the Kullback-Leibler divergence, and P and Q are the two statistical distributions; in this case the JS metric evaluates the difference between the evaluated sample and a "perfect" (i.e. a Poisson probability distribution with λ equal to the mean coverage of the sample) input sample sequenced to the same depth. The distributions compared are the following: for both the actual and the ideal input sample, the genome is split into bins $b_1, ..., n$, and for each bin b_i, the ratio of sequencing reads for this bin relative to the bin with the highest coverage, i.e. $y_i = b_i/max_{1,...,n}(b_i)$ is calculated; bins are then ranked and the resulting rank versus fractional coverage y_i curves are used to evaluate the JS divergence and calculate the synthetic JS distance.

Both cross-correlation and JSD metrics exhibit positive correlation with FRiP values (Figure 3).

Determining reproducible peaks using IDR analysis. The IDR framework (Boxes 1, 4 and 5; Figure 4) is used to obtain more stable, objective estimates of the number of peaks or regions of enrichment based on the reproducibility across biological replicates[11,19,38]. To evaluate reproducibility across biological replicates, a peak caller is used to call peaks with relaxed thresholds on each biological replicate. If a peak in one replicate overlaps with a peak in another replicate by at least 1 bp, then the pair is considered as calling the same region. The significance scores of such pairs (peak signal score for SPP and peak p-value for MACS) are then used as input to the IDR framework to evaluate reproducibility. The number of peaks that pass a user-specified IDR threshold is recorded and is used to determine the number of reproducible peaks to keep.

For ENCODE datasets, we use SPP[17] and MACS[16] with FDR = 0.9 for SPP and p-value ≤ 0.01 for MACS, to obtain 150,000 to 300,000 peaks. We use the default peak width for SPP and a fixed peak width of +/- 200 bp around the peak summit for MACS since MACS tends to call wider peaks at relaxed peak calling thresholds. We determine the number of reproducible peaks (N) at IDR=0.05 and keep the most significant N peaks on each replicate sample. As shown in Figures 4b and 4c, IDR eliminates much of the variation in peak calling that arises due to the specifics of individual peak callers.

Flagging poorly reproducible datasets using IDR analysis. In the event that one replicate is significantly worse than the other replicate, the reproducibility between the replicates is low and the number of reproducible peaks is lower bounded by the worse replicate. To differentiate such cases from the scenario in which the targeted protein genuinely has a small number of occupancy sites, we pool the uniquely aligned reads from the biological replicates and divide the pooled reads into two randomly sampled, non-overlapping sets, referred to as "pseudo-replicates". We then assess reproducibility between pseudo-replicates using IDR as described above with some modifications (See Box 4). If the number of reproducible peaks identified on the pooled pseudo-replicates (N_p) is significantly more than the number of reproducible peaks identified on the biological replicates (N_t), the dataset is flagged for poor reproducibility. For ENCODE data, a dataset is flagged if $N_p/N_t > 2$.

Calculating FRiP values. As a final step in the AQUAS pipeline, FRiP values are calculated on the set of post-IDR peak calls, and datasets with low enrichment are flagged, based on a self-consistency FRiP threshold of 1%, which we have found to work well for human ENCODE datasets, and to show consistency with NSC and RSC thresholds.

Box 4 | Code description for IDR analysis

1. Input file formats

(1) Peak files (specified with `--samples [PEAK_FILE_1] [PEAK_FILE_2]`)

Peak files must be in narrowPeak format. NarrowPeak files are in BED6+4 format. It consists of 10 tab-delimited columns

COL1: `chrom` `string`: name of the chromosome

COL2: `chromStart` `int` : the starting position of the feature in the chromosome. The first base in a chromosome is numbered 0.

COL3: `chromEnd` `int` : the ending position of the feature in the chromosome or scaffold. The chromEnd base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as `chromStart=0`, `chromEnd=100`, and span the bases numbered 0-99.

COL4: `name` `string`: name given to a region (preferably unique). Use '.' if no name is assigned

COL5: `score` `int` : indicates how dark the peak will be displayed in the browser (1-1000). If '0', the DCC will assign this based on signal value. Ideally average `signalValue` per base spreads between 100-1000.

COL6: `strand` `char` : +/- to denote strand or orientation (whenever applicable). Use '.' if no orientation is assigned.

COL7: `signalValue` `float` : measurement of overall (usually, average) enrichment for the region.

COL8: `pValue` `float` : measurement of statistical signficance ($-log_{10}$). Use -1 if no `pValue` is assigned.

COL9: `qValue` `float` : measurement of statistical significance using false discovery rate ($-log_{10}$). Use -1 if no `qValue` is assigned.

COL10: `peak` `int` : point-source summit called for this peak; 0-based offset from `chromStart`. Use -1 if no point-source summit is called.

The p-value and q-value columns should be at $-log_{10}$ scale.

The narrowPeak format has 3 columns that can be used to rank peaks (`signal.value`, `p.value` ($-log_{10}$) and `q.value` ($-log_{10}$)).

The peak summit column must have values relative to the start coordinate of the peaks.

Any of these columns can be used to as a ranking measure but the measure used has to be relatively continuous without too many ties. A measure can be chosen with `--rank [RANK_MEASURE]`. We recommend using `signal.value`, `p.value` and `q.value` for the SPP, MACS2 and PeakSeq peak callers, respectively.

(2) Oracle peak list (specified with `--peak-list [ORACLE_PEAK_LIST]`)

Oracle peak lists are usually peaks called from merged replicates. For each oracle peak a single peak from each replicate is chosen that overlaps the oracle peak. If there are multiple peaks that overlap the oracle, then ties are broken by applying the following criteria in order: 1) choose the replicate peak with a summit closest to the oracle peak's summit 2) choose the replicate peak that has the largest overlap with the oracle peak 3) choose the replicate peak with the highest score. If an oracle peak list is not given, peaks are grouped by overlap and then merged. The merged peak aggregate value is determined by `--peak-merge-method`. Peaks that don't overlap another peak in every other replicate are not included unless `--use-nonoverlapping-peaks` is set.

2. Output file formats

(1) IDR peaks

The output format mimics the input file type, with some additional fields. The first 6 columns are a standard BED6, the first 10 columns are a standard narrowPeak. Also, for columns 7-10, only the score that the IDR code used for ranking will be set – the remaining two columns will be set to -1. `--idr-threshold [IDR_THRESHOLD]` only returns peaks with a global IDR threshold below this value while `--soft-idr-threshold [IDR_THRESHOLD]` reports statistics (including plots) for peaks with a global IDR below this value but return all peaks.

COL11: `localIDR` `float`: $-log_{10}$(local IDR value)

COL12: `globalIDR` `float`: $-log_{10}$(global IDR value)

COL13: `rep1_chromStart` `int` : the starting position of the feature in the chromosome or scaffold for common replicate 1 peaks, shifted based on offset. The first base in a chromosome is numbered 0.

COL14: `rep1_chromEnd` `int` : the ending position of the feature in the chromosome or scaffold for common replicate 1 peaks. The chromEnd base is not included in the display of the feature.

COL115: `rep1_signalValue` `float`: signal measure from replicate 1. Note that this is determined by the `--rank` option. e.g. if `--rank` is set to `signal.value`, this corresponds to the 7th column of the narrowPeak, whereas if it is set to `p.value` it corresponds to the 8th column.

COL16: `rep1_summit` `int` : the summit of this peak in replicate 1.

(2) Plots

Four plots are generated in a single `.png` file. These plots are informative about the quality and similarity of the replicates.

Upper left: replicate 1 peak ranks versus replicate 2 peak ranks. Peaks that do not pass the specified IDR threshold are colored orange.

Upper right: replicate 1 log_{10} peak scores versus replicate 2 log_{10} peak scores. Peaks that do not pass the specified IDR threshold are colored orange.

Bottom row: Peak ranks versus IDR scores are plotted in black. The overlaid boxplots display the distribution of IDR values in each 5% quantile. The IDR values are thresholded at the optimization precision $1e-6$ by default.

Study cases

We illustrate the application of the AQUAS pipeline for identifying poor quality and poorly reproducible datasets with a couple case studies extracted from the experience of the ENCODE Consortium.

Study case 1:

We demonstrate how cross-correlation analysis can be used to provide feedback to experimentalists using two pairs of ChIP-seq experiments targeting the transcriptional coactivator p300 in the HepG2 cell line (Figure 5). Cross-correlation measures flagged one of the initial pair of replicates (Figure 5b) as being suboptimal. The production lab later generated a new pair of datasets, which showed high NSC/RSC scores, exceeding the stronger replicate in the older pair of datasets (Figures 5b and 5c). The newer pair replaced the old one in the ENCODE repository (experiment ID ENCSR000BLW). Hence, cross-correlation analysis can be used to provide data producers with feedback about data quality, and to effectively flag potentially low-quality datasets for replacement resulting in higher-quality datasets.

Study case 2:

We illustrate how IDR can be used to identify poorly reproducible datasets with an ENCODE ChIP-seq dataset targeting the GATA2 transcription factor in K562 cells, generated using an eGFP-GATA2 fusion and an α-GFP antibody (Figure 6). IDR analysis returned 9,116 reproducible peaks when true replicates are compared, but 20,996 when pooled pseudo-replicates are considered. This in turn appears to be the result of the first replicate returning a much higher number of peaks than the second (25,073 vs. 2,772; based on IDR analysis of individual pseudoreplicates). The overal dataset (experiment ID ENCSR000DKA) was flagged for having reproducibility issues in the ENCODE data matrix.

Limitations

The quality control metrics and IDR analysis described here have been largely tested on TFs and on active histone marks that have a localized binding pattern, such as H3K4me3 and H3K27ac. Due to the nature of broad histone modifications, determining enriched regions is challenging compared to peak calling for TFs and is not yet a fully solved problem. Thus, this protocol is not optimized for broad marks. We are in the process of developing analysis pipelines for broad histone modifications.

We also note that cross-correlation can be quite sensitive to the content of the input files on which it is run. For example, if the read length for a dataset is within the range of the predominant fragment length, then the read-length cross-correlation peak can be hard to distinguish from the fragment-length cross-correlation peak.

The evaluation of read clustering by cross-correlation also relies on the assumption that reads on opposite strands are independent of each other. This assumption is violated if paired-end data is to be analyzed by cross-correlation, as in such cases there is always another read at on average a mean fragment length distance away from each read; as a result cross-correlation scores are always very high on paired-end data and provide little meaningful information regarding ChIP quality. Of note, this issue cannot be resolved by only considering the first end in mapped read pairs if reads have been mapped in a paired-end format. This is because, as discussed above, the phantom peak arises due to local mappability variations in the genome; when reads are mapped as pairs, the additional information provided by the second read in a pair makes many of the repetitive regions in the genome that give rise to the phantom peak in the cross-correlation function uniquely mappable, thus suppressing the phantom peak relative to the fragment-length peak.

Another consideration to keep in mind is that if reads are trimmed to a variable length by removing adapters, and this results in a significant proportion of alignments being of less than the maximum length, the phantom peak will again be lowered compared to the fragment length peak, for the reasons concerning unique mappability discussed above.

The phantom peak is also artificially suppressed if duplicate reads have been removed, as in such cases only a limited number of aligned reads remain present in the regions of variable mappability that give rise to it.

Finally, the assumptions behind cross-correlation analysis are also violated if multimapping reads are included in the input alignment files, and it should therefore only be carried out on unique alignments alone for the purposes of ChIP-seq quality control.

For these reasons, we calculate cross-correlation on single-end alignments of reads trimmed to at most 50-mers

Box 5 | Interpretation of the output of the IDR statistical analysis

The output of the IDR statistical analysis consists of two main elements, a graphical tool and the irreproducible discovery rate for each entry on the rank lists.

The graphical tool provides a quick illustration for visualizing and inspecting the reproducibility of the replicates and the transition from reproducible to irreproducible signals, without making any model assumptions. This tool is convenient for diagnosis and quality control. This tool provides two curves, namely, the correspondence curve and the change of correspondence curve. For the correspondence curve, a perfect rank consistency between peaks shows as a straight line of slope 1, and lack of consistency shows as departures from the diagonal line, such as curvature bending toward the x-axis. If almost no association is present, the curve shows a parabolic shape. For the change of correspondence curve, a perfect consistency shows as a straight line of slope 0 with intercept 1, and lack of consistency shows as a line with a positive slope. The transition of the shape of the curves, if present, indicates the breakdown of consistency, which can be used for determining the threshold. Prototypical plots and detailed interpretation of the two curves can be found in Li et al.[19]. The patterns in real datasets usually are more complicated than the prototypical ones. Yet the prototypical plots are still helpful for interpreting the actual patterns.

Both the local IDR and global IDR are reported in the output for peaks that are commonly identified on replicates. To illustrate the relationship between the IDR thresholds and the number of top peaks, an IDR curve is produced to plot the numbers of selected peaks at various IDR cutoffs. Reproducible peaks can be selected according to a user-specified IDR threshold.

using only the first end of read pairs, with all multireads removed and without deduplication of alignments. Subsequent pipeline steps (peak calling, reproducibility analysis, signal track generation, etc.) are carried out on alignments utilizing the full length of input reads and in paired-end format (if the library was sequenced as paired-end).

Due to these considerations, when working with single-end reads longer than 50 bp, and when working with paired-end reads, we carry out peak calling, IDR analysis, and signal track generation on the full-length reads, but we map reads separately in a single-end 50 bp format for the purposes of cross-correlation analysis.

Another important consideration is that the thresholds presented in this protocol are mainly focused on mammalian ENCODE datasets. Other genomes can exhibit very different properties (due to smaller or larger size, a very different repetitive element and unique mappability structure, and other factors), and the parameters used for mammalian genomes may not be appropriate in such cases. Careful calibration for each new genome needs to be carried out in such situations.

Finally, the quality control metrics described here are designed with conventional ChIP-seq datasets in mind. In recent years, novel methods for ChIP library preparation, such as ChIPmentation[39], TCL[40], and others, novel variations of the ChIP assay, such as ChIP-exo[41] and ChIP-nexus[42], and alternatives to the ChIP assay, such as ChEC-seq[43] and CUT&RUN[44] have been described. While the general IDR peak calling and reproducibility framework is applicable to such datasets, the correspondence between fragment ends and protein occupancy sites in the libraries generated by these assays may not conform to the assumptions of the cross-correlation quality control analysis as designed for conventional ChIP libraries, therefore cross-correlation results should be interpreted with caution with respect to the degree of target enrichment in such datasets.

Materials

Equipment

Operating system

This protocol assumes using a Unix-like operating system (e.g., Linux). Alternatively, instruction for using the DNAnexus platform are also provided.

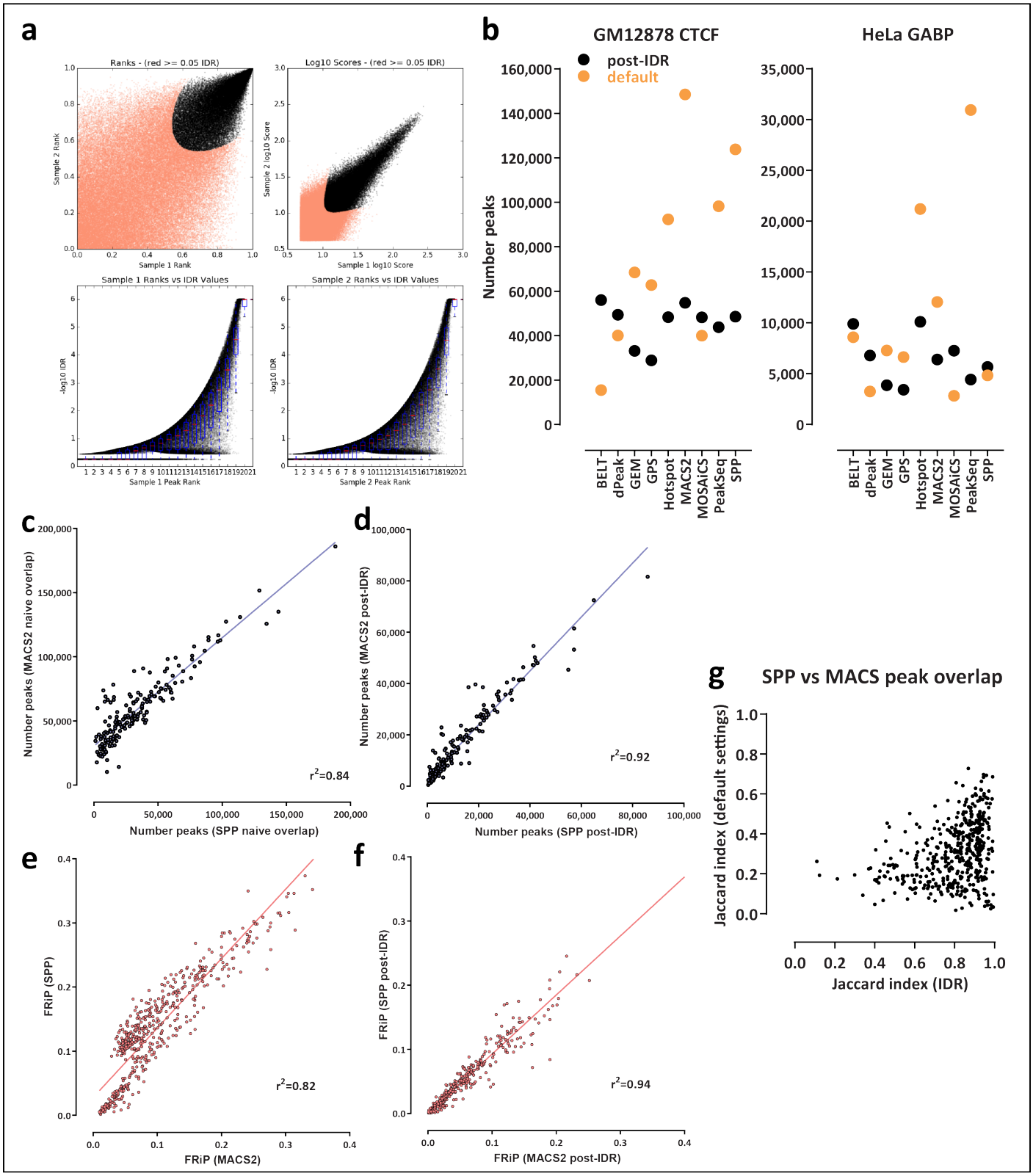
Hardware requirement

A system with high-speed internet connection, $\geq$ 8 Intel/AMD 64-bit CPU cores, $\geq$ 32 GB memory, and $\geq$ 200 GB disk space is assumed. If the pipeline described in this protocol is to be run on a cloud platform, such as DNAnexus, Google Cloud Platform and Amazon Web Service, then a system with much less computational resources will also suffice.

Software

We provide a workflow description language (WDL) script to run our pipeline in this protocol in an automated way. Any WDL runner can be used to run it but we chose Cromwell (`https://github.com/broadinstitute/cromwell`) and its wrapper Caper (`https://github.com/ENCODE-DCC/caper`) for demonstration purposes in this paper. There are three methods to run our pipeline in this protocol.

- DNAnexus platform (`https://www.dnanexus.com`). We have built out workflow templates for each ref-


a
Ranks - (red >= 0.05 IDR)
Sample 2 Rank
Sample 1 Rank
Log10 Scores - (red >= 0.05 IDR)
Sample 2 log10 Score
Sample 1 log10 Score
Sample 1 Ranks vs IDR Values
-log10 IDR
Sample 1 Peak Rank
Sample 2 Ranks vs IDR Values
Sample 2 Peak Rank
b
GM12878 CTCF
HeLa GABP
post-IDR
default
Number peaks
BELT
dPeak
GEM
GPS
Hotspot
MACS2
MOSAiCS
PeakSeq
SPP
c
Number peaks (MACS2 naive overlap)
Number peaks (SPP naive overlap)
r2=0.84
d
Number peaks (MACS2 post-IDR)
Number peaks (SPP post-IDR)
r2=0.92
e
FRiP (SPP)
FRiP (MACS2)
r2=0.82
f
FRiP (SPP post-IDR)
FRiP (MACS2 post-IDR)
r2=0.94
g
SPP vs MACS peak overlap
Jaccard index (default settings)
Jaccard index (IDR)

erence genome. Users can simply copy one of the templates into their own project according to the reference genome of interest and then choose FASTQ files to be processed.

- Caper. Caper is a convenient Python wrapper for Cromwell. Caper supports multiple platforms such as SLURM, PBS, Sun GridEngine, Amazon Web Service and Google Cloud platform. Caper can automatically make a cached database for all required files defined in an input JSON file so that users do not need to manually download or localize each data file on a chosen platform. In this paper, we demonstrate how to run it locally with Docker or Singularity.

- Bash command line interface (CLI). This protocol can also be run without WDL runners by manually running each subtask in a correct order with a Bash CLI.

Continuous integration

The AQUAS pipeline resides on GitHub as a repository maintained with continuous integration through CircleCI (`https://www.circleci.com`). As we are continuously working on improving the pipeline and adding new features to it, it is important to have automated testing process to ensure robustness and reproducibility. Any new commits pushed to the repository are immediately noticed by CircleCI and a testing procedure kicks in, including two levels of testing – task-level for unit tests and workflow-level for integration tests. For task-level tests, each task defined in pipeline's WDL file is tested separately with given inputs and expected outputs. For workflow-level tests, downsampled real ENCODE experiments are used for different pipeline types (TF ChIP-seq and histone ChIP-seq) and different input types (single-end and paired-end reads). In both test regimes, MD5 hashes of outputs are compared with expected values. Pipeline versions follow semantic versioning with three numbers (major, minor and patch). Major version numbers are for rare structural changes of the pipeline, minor numbers are for any changes of outputs in terms of MD5 hash, and patch numbers are for bug fixes. Pipeline documentation is also maintained on the same GitHub repository, in Markdown format.

Requirements for the DNAnexus platform method

- A computer with a web browser connected to the internet.

Requirements for the Caper method

- **AQUAS pipeline**: this protocol
- **Java** (JRE $\geq$ 1.8 or JDK $\geq$ 1.8)
- **Caper** ($\geq$ 2.3.0). Caper will automatically download Cromwell. Users can choose one among Docker and Singularity to run the pipeline.
- **Docker**: Caper pulls a Docker image automatically from `quay.io/encode-dcc/chip-seq-pipeline:v1.3.6`. Users need to ask their system administrators to install Docker on their system.
- **Singularity**: Singularity is a second option for users without a super-user privilege to install Docker on their system. Caper has a built-in backend for Singularity and it takes care of building a Singularity container from `docker://quay.io/encode-dcc/chip-seq-pipeline:v1.3.6`. Once the local container is built and cached, then Caper uses a cached one instead of building one for every run.
- **Reference genome data**: We provide a reference genome database builder and downloader for several reference genomes (`hg19`, `mm9`, `hg38` and `mm10`). Users can download them or build one for the particular genome they are working with. Users can also build a reference genome database for any custom genome if they have a FASTA file for it.
- **Croo**: Croo (Cromwell output organizer) is an optional tool to organize raw outputs from Cromwell (or from its wrapper Caper). Croo organizes outputs in a different directory structure and provides an HTML report that has a table describing all output files, task graphs, and UCSC genome browser tracks. See Figure 7 for more details.

Figure 4 *(preceding page)*: The irreproducible discovery rate (IDR) framework for assessing reproducibility of ChIP-seq data sets. (a) Example (ENCODE experiment ID ENCSR000DYI) of the output of the IDR (version 2.0). **(b–f)** The IDR framework improves consistency between peak calls, minimizes variation and eliminates systematic differences caused by the specific parameter and algorithmic choices implemented by different peak calling packages. (b) Example of the wide variation in the output of different peak callers (ran with default settings) (orange) and the stabilization of peak calls after running IDR (black) using ENCODE ChIP-seq datasets for CTCF in GM12878 cells and for GABP in HeLa cells. (c and d) A set of ~500 ENCODE transcription factor ChIP-seq datasets were processed using the MACS2 and SPP peak callers individually and also using the IDR framework; the number of peaks called with each of the two peak callers is shown in **(b)**, and the number of peaks after applying IDR is shown in **(c)**. (e and f) Concordance between FRiP values for the two peak callers run with default settings **(e)**, and after applying IDR **(f)**.

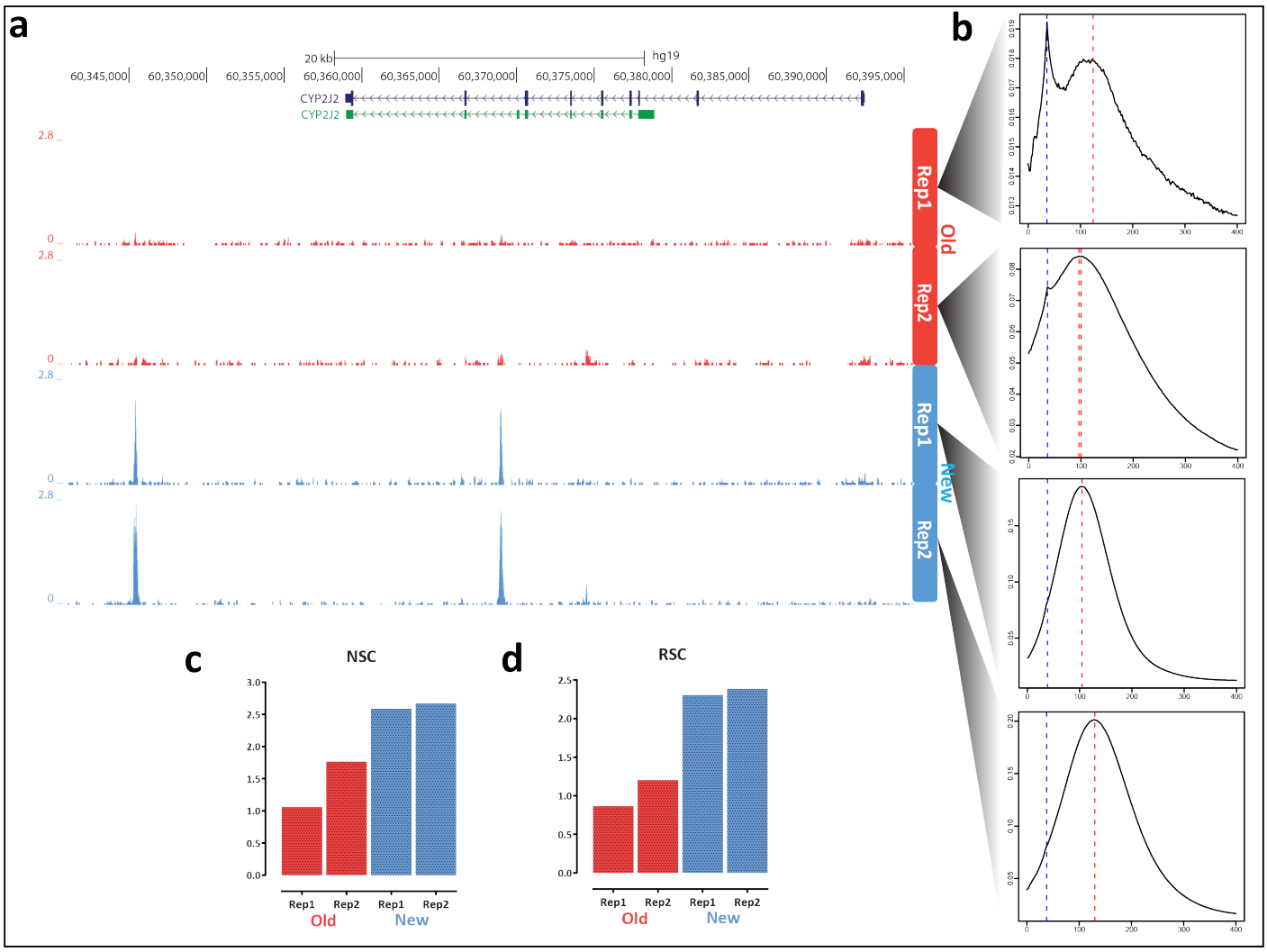


Figure 5: Study case 1: Identifying and replacing lower quality p300 ENCODE ChIP-seq datasets resulting in significant improvement in data quality. (a) Genome browser snapshot of normalized ChIP-seq signal for four p300 ChIP-seq datasets in the human HepG2 cell line. The upper two tracks represent replicates where one replicate (old Rep1) was found to show significantly lower NSC and RSC scores significantly weaker ChIP-seq peak signal compared to the other replicate (old Rep2). These datasets were replaced by the production group with new replicates (new Rep1 and Rep2). **(b)** CCPs generated from datasets in (a). Replaced datasets show substantially higher fragment length peaks compared to the old datasets (in particular Rep1). **(c and d)** Corresponding RSC and NSC. A significant increase in the quality control metrics in the replacement datasets indicates improvement in data quality.

Requirements for the Bash CLI method

- **AQUAS pipeline**: this protocol
- **Singularity**: Singularity $\geq$ 2.6.0
- **Reference genome data**

Equipment Setup

This section includes installation instructions for all software packages. Users should selectively install these packages according to their choice for a WDL runner described in the section "Software".

AQUAS pipeline: The AQUAS pipeline `v1.3.6` can be obtained as follows:

```
$ cd $HOME
$ git clone --branch v1.3.6 https://github.com/encode-dcc/chip-seq-pipeline2
```

Java: Super-user privileges are needed to install Java; alternatively, it can be installed locally. Make sure that the

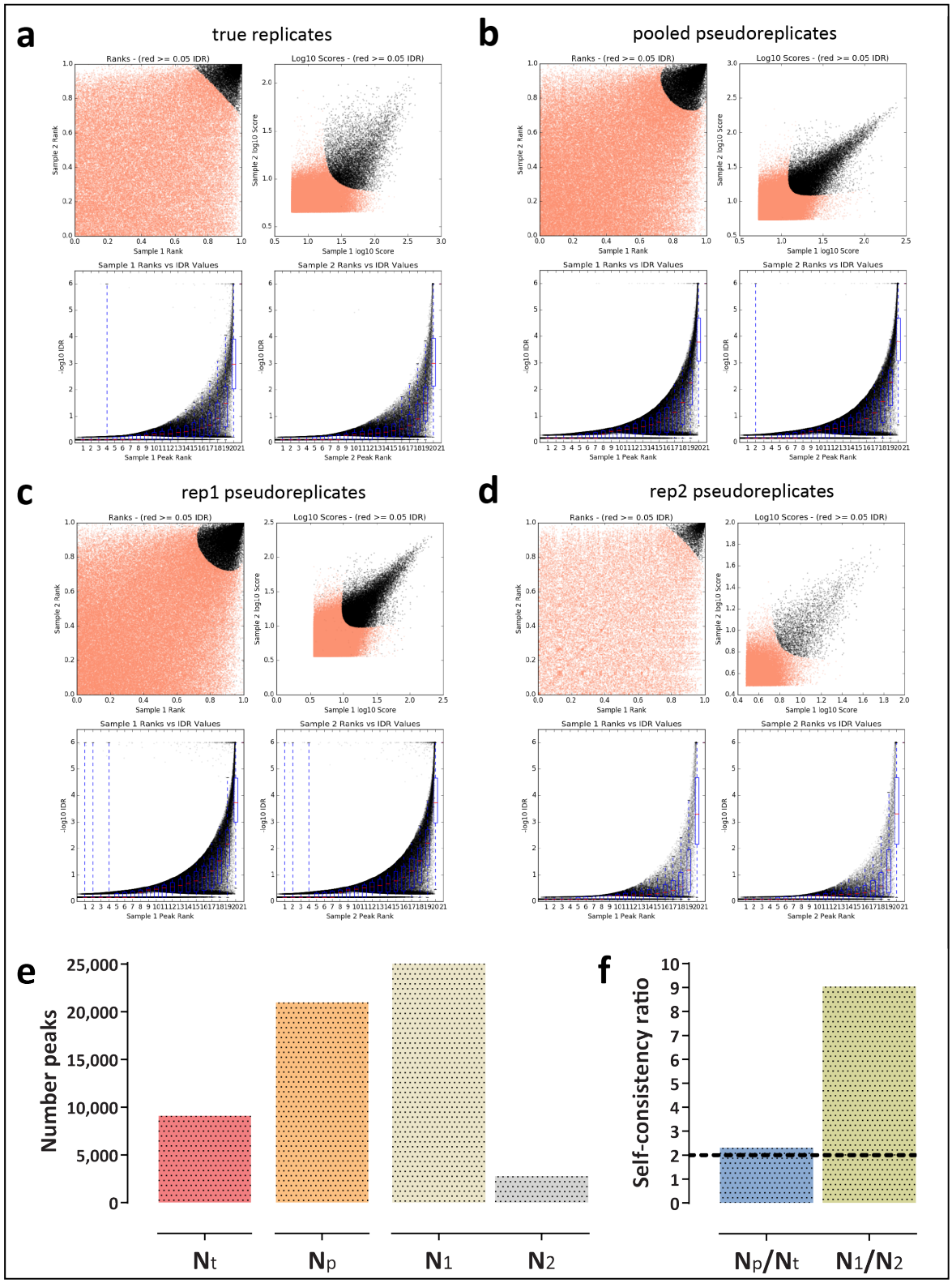


Figure 6: Study case 2: Identification of ChIP-seq replicates with poor reproducibility using IDR. ENCODE experiment ENCSR000DKA is used as an example (eGFP-tagged GATA2 in human K562 cells). **(a)** IDR results for comparison between true replicates. **(b)** IDR results for comparison between pooled pseudoreplicates. **(c)** IDR results for comparison between first replicate pseudoreplicates. **(d)** IDR results for comparison between second replicate pseudoreplicates. **(e)** Number of peaks for each IDR comparison **(f)** Self-consistency and rescue ratios. As the experiment shows poor reproducibility based on the self-consistency and rescue ratios, it was flagged in the ENCODE data matrix.

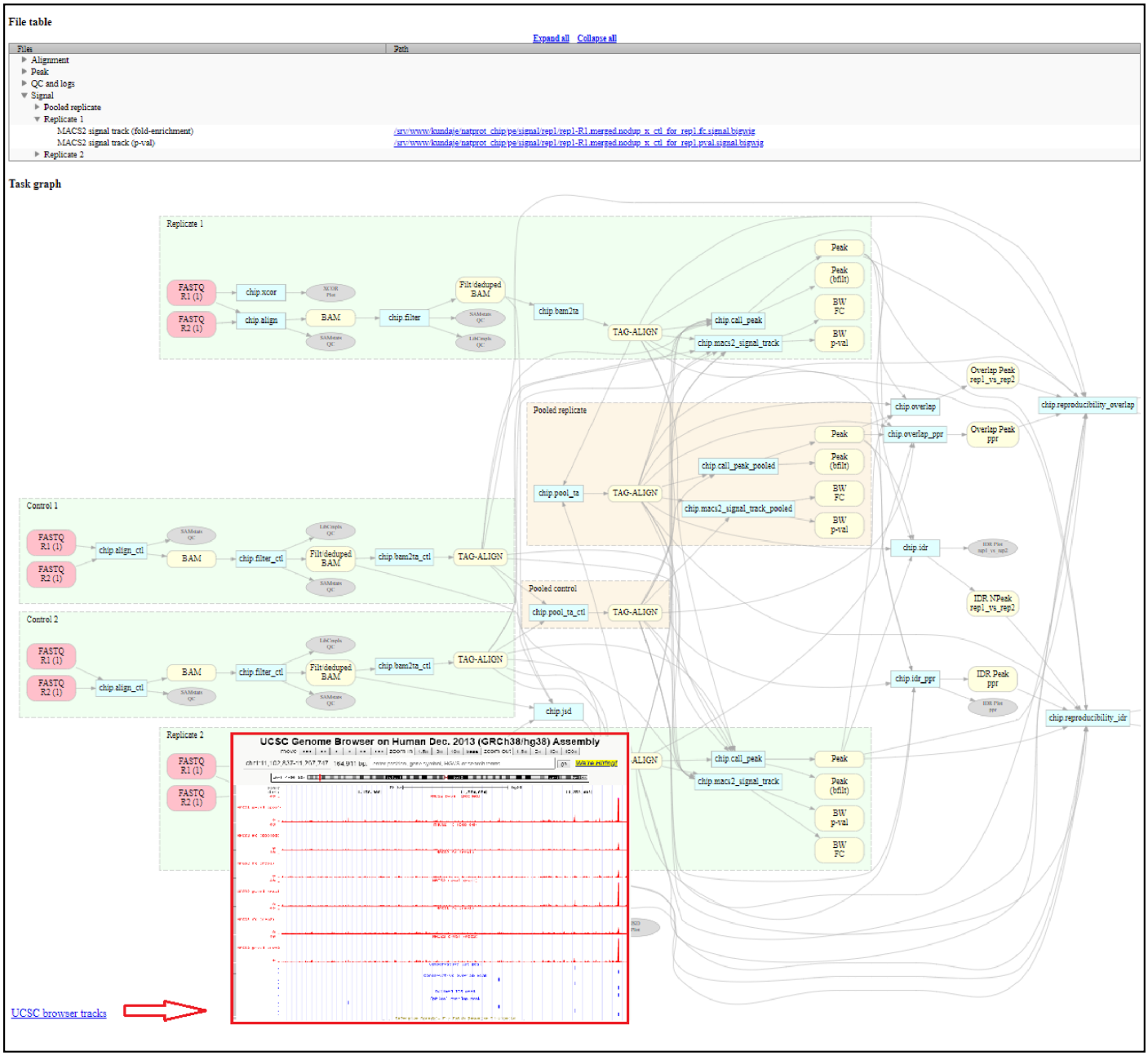


Figure 7: Croo HTML report including a table for organized outputs, task graph and UCSC genome browser tracks.

Java binary path is appended to the `$PATH` for all login/interactive Bash shells.

- For Debian/Ubuntu-based (18.04 or higher) Linux:

  ```
  $ sudo apt-get install git default-jre
  ```

- For Fedora/Red-Hat-based Linux:

  ```
  $ sudo sudo yum install java-11-openjdk
  ```

Box 6 | Recommendation for the IDR thresholds

For self-consistency and comparison of true replicates:

- If starting with ∼150 to 300K relaxed pre-IDR peaks for large genomes (human/mouse), then a threshold of 0.01 or 0.02 generally works well.
- If starting with < 100K pre-IDR peaks for large genomes (human/mouse), then a threshold of 0.05 is more appropriate. This is because IDR sees a smaller noise component and IDR scores become weaker. This is typically for use with peak callers that are unable to be adjusted to call large number of peaks (e.g., PeakSeq).
- For smaller genomes, such as worms and flies, if starting with ∼15K to 40K peaks then once again IDR thresholds of 0.01 or 0.02 work well.
- For self-consistency analysis of datasets with shallow sequencing depths, an IDR threshold as relaxed as 0.1 can be used if starting with < 100K pre-IDR peaks.
- For pooled-consistency analysis, if starting with ∼150 to 300K relaxed pre-IDR peaks for large genomes (human/mouse), then thresholds of 0.0025 or 0.005 generally works well. We use a tighter threshold for pooled-consistency analysis since pooling and subsampling equalizes the pseudo-replicates in terms of data quality. So we err on the side of caution and use more stringent thresholds. The equivalence between a pooled-consistency threshold of 0.0025 and original replicate consistency threshold of 0.01 was calibrated based on a gold-standard pair of high-quality replicate datasets for the CTCF transcription factor in human.
- One can also use 0.01 for less stringent thresholds.
- If starting with < 100K, pre-IDR peaks for large genomes (human/mouse), then a threshold of 0.01 is more appropriate. Since the IDR sees a smaller noise component and the IDR scores become weaker, relaxing the thresholds is advised. This is typically for use with peak callers that are unable to be adjusted to call large number of peaks (e.g. PeakSeq)
- For smaller genomes, such as worms and flies, if starting with ∼15K to ∼40K, an IDR threshold of 0.01 works well.

Caper: Make sure that you have Python 3.7 (or higher) installed on your system. Caper can be installed as follows:

```
$ pip install caper
```

Docker: Super-user privileges are needed to install Docker. Users have to be added to the Docker group in order to have access to the Docker daemon running on a given system.

- For Debian/Ubuntu (18.04 or higher) based Linux:

  ```
  $ sudo apt-get install docker.io
  ```

- For Fedora/Red-Hat based Linux:

  ```
  $ sudo yum install docker
  ```

Singularity (for Debian based Linux): Follow instructions at `https://neuro.debian.net/install_pkg.html?p=singularity-container`.

Singularity (non-Debian based Linux): Singularity can be installed without root privileges in most cases. However, some libraries still need root access.

```
$ VERSION=2.5.2
$ mkdir -p $HOME/.singularity && cd $HOME/.singularity
$ wget
  https://github.com/singularityware/singularity/releases/download/$VERSION/singularity-$VERSION.tar.gz
$ tar xvf singularity-$VERSION.tar.gz
$ cd singularity-$VERSION
$ ./configure --prefix=$HOME/.singularity --disable-suid
$ make && make install
```

If an error related to the `libarchive` library is encountered while installing Singularity, it can be installed by the system admin.

- For Debian/Ubuntu (18.04 or higher) based Linux:

```
$ sudo apt-get install libarchive-dev
```

Box 7 | Configuration input JSON files for two test cases (ENCSR000DYI for SE and ENCSR936XTK for PE).

```
ENCSR000DYI.json
{
    "chip.pipeline_type" : "tf",
    "chip.genome_tsv" : "hg38/hg38.tsv",
    "chip.fastqs_rep1_R1" : ["rep1.fastq.gz"],
    "chip.fastqs_rep2_R1" : ["rep2.fastq.gz"],
    "chip.ctl_fastqs_rep1_R1" : ["ctl1.fastq.gz"],
    "chip.ctl_fastqs_rep2_R1" : ["ctl2.fastq.gz"],
    "chip.paired_end" : false,
    "chip.always_use_pooled_ctl" : true,
    "chip.title" : "ENCSR000DYI",
    "chip.description" : "CEBPB ChIP-seq on human A549 produced by the Snyder lab",
    "chip.align_cpu" : 2,
    "chip.call_peak_cpu" : 2
}
ENCSR936XTK.json
{
    "chip.pipeline_type" : "tf",
    "chip.genome_tsv" : "hg38/hg38.tsv",
    "chip.fastqs_rep1_R1" : ["rep1-R1.fastq.gz"],
    "chip.fastqs_rep1_R2" : ["rep1-R2.fastq.gz"],
    "chip.fastqs_rep2_R1" : ["rep2-R1.fastq.gz"],
    "chip.fastqs_rep2_R2" : ["rep2-R2.fastq.gz"],
    "chip.ctl_fastqs_rep1_R1" : ["ctl1-R1.fastq.gz"],
    "chip.ctl_fastqs_rep1_R2" : ["ctl1-R2.fastq.gz"],
    "chip.ctl_fastqs_rep2_R1" : ["ctl2-R1.fastq.gz"],
    "chip.ctl_fastqs_rep2_R2" : ["ctl2-R2.fastq.gz"],
    "chip.paired_end" : true,
    "chip.always_use_pooled_ctl" : true,
    "chip.title" : "ENCSR936XTK",
    "chip.description" : "ZNF143 ChIP-seq on human GM12878",
    "chip.align_cpu" : 2,
    "chip.call_peak_cpu" : 2
}
```

- For Fedora/Red-Hat based Linux:

```
$ sudo yum install libarchive-devel
```

Once the installation for Singularity is done successfully, the following has to be added to the user's shell startup script (`$HOME/.bashrc` or `$HOME/.bash_profile`).

```
export PATH=$PATH:$HOME/.singularity/bin
```

Box 8 | Software requirements in AQUAS

Aligner

- `bowtie2` $\geq$ 2.3.4.3

Samtools, SAMstats and bedtools

- `samtools` ($\geq$ 1.9), `htslib`, `SAMstats` ($\geq$ 0.2.1) and `bedtools` ($\geq$ 2.29.0)

UCSC tools

- `ucsc-fetchchromsizes`, `ucsc-wigtobigwig`, `ucsc-bedgraphtobigwig`, `ucsc-bigwiginfo`, `ucsc-bedclip`, `ucsc-bedtobigbed` and `ucsc-twobittofa`

PICARD tools

- `picard` (= 2.20.7)

Python and its packages

- Python3 and packages: `python3` (≥ 3.6), `numpy`, `matplotlib`, `macs2` (= 2.2.4), `deeptools` (= 3.3.1) `pyfaidx` (= 0.5.5.2) and `idr` (≥ 2.0.4.2).

R and its packages

- `R` (≥ 3.2.2), `r-snow`, `r-snowfall`, `r-bitops`, `r-catools`, `r-spp` (= 1.14) and `bioconductor-rsamtools`.

Others

- `boost`, `openssl` (= 1.0.2), `libtool`, `ghostscript` and `zlib`

Reference genome data: This protocol provides a genome database downloader for various species and their versions (`mm9`, `hg19`, `hg38` and `mm10`). Use `hg38` and `mm10` for genome data associated with the ENCODE project. The genome database includes reference sequences, `bowtie2` indexes, chromosome sizes files and blacklisted regions for each species. The following command automatically generates genome data and a corresponding `.tsv` file in a specified destination directory `[DEST_DIR]`, which includes all genome-specific data file paths:

```
$ cd $HOME/chip-seq-pipeline2
$ bash scripts/download_genome_data.sh hg38 ./hg38
```

The `.tsv` file `"chip.genome_tsv":"[DEST_DIR]/[GENOME].tsv"` is to be used in the input `.json` file for the Cromwell runner (and its wrapper Caper).

Timing

The execution of the protocol depends on the sequencing depth of the datasets and the size of the genome of the organism being studied. In general, for reasonably deeply sequenced mammalian ChIP-seq datasets (i.e. two control replicates and two ChIP replicates at 20–30 $\times 10^6$ mapped reads each), the whole pipeline should take less than 24-36 hours.

Procedure and Anticipated Results

As described in the Software section, the AQUAS pipeline can be run using three different methods (DNAnexus platform, Caper and Bash shell CLI).

Procedure for DNAnexus platform method

In order for the AQUAS protocol to be run on the DNAnexus platform, users must first sign up for a DNAnexus account and create an empty project (Figure 8). Choose one of the regions (Azure and non-Azure) and make sure to match the region with our repository project later.

The instructions shown here are based on the classic version of the DNAnexus user interface. Click on `Search` on the top right corner of the web site. Search for `ENCODE Uniform Processing Pipelines`. Choose one of the found projects (Azure or non-Azure). Make sure to match the region (Azure or non-Azure) with the specific project being worked on. Navigate into a directory `ChIP-seq2/workflows/v1.3.6-dockerhub/` and right-click on `test_ENCSR000DYI` (single-end test sample). Click on `Copy` and another window will pop up. Choose the single-end test sample DNAnexus project and click on `Copy into this folder` in the bottom right corner of the window. Repeat it for the paired-end test sample `test_ENCSR936XTK` too.

Box 9 | Parameters for AQUAS

category	parameter	type	description
	`chip.fastqs_repi_Rj`	`file array`	Definition for FASTQ file paths/URIs for replicate i and read end j : `[merge_id]`
	`chip.bams`	`array`	Raw (unfiltered) BAM file paths/URIs : `[rep_id]`

input files

	chip.nodup_bams	file array	Filtered/deduped BAM file paths/URIs : [rep_id]
	chip.tas	array	tagAlign file paths/URIs : [rep_id]
	chip.ctl_fastqs_repi_Rj	file array	Definition for FASTQ file paths/URIs for control i and read end j : [merge_id]
	chip.ctl_bams	array	Raw (unfiltered) control BAM file path/URI : [rep_id]
	chip.ctl_nodup_bams	array	Filtered (deduped) control BAM file path/URI : [rep_id]
	chip.ctl_tas	array	Control tagAlign file path/URI : [rep_id]
general	chip.pipeline_type	str	tf for TF ChIP-Seq or histone for Histone ChIP-Seq
	chip.paired_end	bool	Input files are paired-end
	chip.align_only	bool	Disable all downstream analysis after mapping
	chip.true_rep_only	bool	Disable all analyses related to pseudo replicates
alignment	chip.crop_length	bool	Crop IP and control FASTQs to this length with using Trimmomatic. Reads shorter than this will be removed from output BAMs and all downstream analyses. 0 by default, which means disabled.
	chip.xcor_trim_bp	bool	Number of basepairs to trim paired-end FASTQ (for cross-corr. analysis only)
	chip.dup_marker	str	Dup marker. picard or sambamba
	chip.mapq_thresh	int	Threshold for low MAPQ reads removal
	chip.no_dup_removal	bool	No dup reads removal when filtering BAM
	chip.filter_chrs	str array	Filter out chromosomes in a filtering step
	chip.subsample_reads	int	Number of reads to subsample tagAlign. Subsampled TAGALIGN will be used for all downstream analyses
	chip.xcor_subsample_reads	int	Number of reads to subsample tagAlign (for cross-corr. analysis only)
peak calling	chip.ctl_depth_ratio	float	if control read depth ratio is higher than this use pooled control for all exp rep
	chip.always_use_pooled_ctl	bool	Always use pooled control
	chip.cap_num_peak_macs2	int	Cap number of raw peaks called from MACS2
	chip.pval_thresh	float	p-value threshold for MACS2
	chip.cap_num_peak_spp	int	Cap number of raw peaks called from SPP
	chip.fdr_thresh	float	FDR threshold for SPP
	chip.idr_thresh	float	IDR threshold
resources	chip.align_cpu	int	Number of cores for Bowtie2 mapping
	chip.align_mem_mb	int	Maximum memory limit in MB for Bowtie2 mapping
	chip.align_time_hr	int	Walltime for Bowtie2 mapping
	chip.filter_cpu	int	Number of cores for filtering BAMs
	chip.filter_mem_mb	int	Maximum memory limit in MB for filtering BAMs
	chip.filter_time_hr	int	Walltime for filtering BAMs
	chip.bam2ta_cpu	int	Number of cores for converting BAMs into TAG-ALIGNs
	chip.bam2ta_mem_mb	int	Maximum memory limit in MB for converting BAMs into TAG-ALIGNs
	chip.bam2ta_time_hr	int	Walltime for converting BAMs into TAG-ALIGNs
	chip.xcor_cpu	int	Number of cores for cross-correlation analysis
	chip.xcor_mem_mb	int	Maximum memory limit in MB for cross-correlation analysis
	chip.xcor_time_hr	int	Walltime for cross-correlation analysis
	chip.call_peak_cpu	int	Number of cores for SPP peak-calling (fixed at 1 for MACS2)
	chip.call_peak_mem_mb	int	Maximum memory limit in MB for SPP/MACS2 peak-calling
	chip.call_peak_time_hr	int	Walltime for SPP/MACS2 peak-calling
QC report	chip.title	str	Name of sample
	chip.description	str	Description for sample

Two directories, test_ENCSR000DYI and test_ENCSR936XTK, will be generated in the root of the DNAnexus project. For each directory, a workflow named chip will be present. Click on it (Figure 9).

Set the output directory for the workflow and click on the launch button (Figure 10).

The workflow the automatically redirects to a monitor page (Figure 11).

Once analysis has finished on the monitor page, go to the specified output folder, where the final output QC JSON file qc.json and an HTML report qc.html can be found (Figure 12).

Preparation for Caper and Bash CLI methods

Move to the pipeline `git` repo directory of the AQUAS pipeline.

```
$ cd $HOME/chip-seq-pipeline2
```

Please make sure that Java, Singularity and the dependencies described in the previous sections have been installed. For this demonstration, download raw reads for the two replicates of the ChIP-seq experiments with ENCODE IDs ENCSR936XTK and ENCSR000DYI:

```
$ wget https://www.encodeproject.org/files/ENCFF960TNP/@@download/ENCFF960TNP.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF640CBP/@@download/ENCFF640CBP.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF246DIP/@@download/ENCFF246DIP.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF616WSS/@@download/ENCFF616WSS.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF000VOL/@@download/ENCFF000VOL.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF000VON/@@download/ENCFF000VON.fastq.gz
```

Download replicates for the corresponding controls experiments ENCSR398JTO and ENCSR496AXR:

```
$ wget https://www.encodeproject.org/files/ENCFF002EFT/@@download/ENCFF002EFT.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF002EFQ/@@download/ENCFF002EFQ.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF002EFU/@@download/ENCFF002EFU.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF002EFS/@@download/ENCFF002EFS.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF000VPI/@@download/ENCFF000VPI.fastq.gz
$ wget https://www.encodeproject.org/files/ENCFF000VPK/@@download/ENCFF000VPK.fastq.gz
```

There are two replicates for each ChIP and each controls, which we rename for convenience for the purposes of this demonstration:

```
$ mv ENCFF960TNP.fastq.gz rep1-R1.fastq.gz && mv ENCFF640CBP.fastq.gz rep1-R2.fastq.gz
$ mv ENCFF246DIP.fastq.gz rep2-R1.fastq.gz && mv ENCFF616WSS.fastq.gz rep2-R2.fastq.gz
$ mv ENCFF002EFT.fastq.gz ctl1-R1.fastq.gz && mv ENCFF002EFQ.fastq.gz ctl1-R2.fastq.gz
$ mv ENCFF002EFU.fastq.gz ctl2-R1.fastq.gz && mv ENCFF002EFS.fastq.gz ctl2-R2.fastq.gz
$ mv ENCFF000VOL.fastq.gz rep1.fastq.gz
$ mv ENCFF000VON.fastq.gz rep2.fastq.gz
$ mv ENCFF000VPI.fastq.gz ctl1.fastq.gz
$ mv ENCFF000VPK.fastq.gz ctl2.fastq.gz
```

Procedure using Caper with Docker (recommended)

All subtasks described in the next subsection are run in the correct order with a single command line. Make sure that Docker is installed. A WDL script requires an input JSON file to specify all important files and parameters. As described in Box 7, make a JSON file for each experiment ENCSR000DYI (single-end) and ENCSR936XTK (paired-end) save them as `ENCSR000DYI.json` and `ENCSR936XTK.json`, respectively, on the pipeline's git directory. Save the contents shown in Box 7 into each JSON file. You can adjust number of CPU cores used for two tasks (`align` and `call_peak`) according to your system's condition. Caper will not work without those input JSON files.

The flag `chip.always_use_pooled_ctl` forces the pipeline to use pooled control replicates for relaxed peaks called with the SPP peak caller. There are more parameters that can be tuned to customize the pipeline, the details of which parameters are described in Box 9.

Caper tries to maximize parallelization for running tasks such that all replicates will be processed at the same time. To disable such parallelization for a computer with limited resources, add `--max-concurrent-tasks 1` to the command line argument.

- For the single-end test data set:

  ```
  $ cd $HOME/chip-seq-pipeline2
  $ caper run chip.wdl -i ENCSR000DYI.json --docker --max-concurrent-tasks 1
  ```

Box 10 | Criteria to choose control in AQUAS

It is important to choose an appropriate control replicate for each IP replicate when calling peaks (using peak-caller SPP) and generating signal tracks (using peak-caller MACS2). There are five criteria.

- If the flag `chip.always_use_pooled_ctl` is activated in an input JSON file, choose pooled control replicate for all IP replicates. Such flag is activated for the demonstration test cases.
- If there is only one control replicate provided, choose it for all IP replicates.
- If number of reads in any control replicate differ by a factor of the control depth ratio (1.2 by default), choose pooled control replicate for all IP replicates.
- If there are fewer reads in control i than IP replicate i, choose pooled control replicate for IP replicate i.
- Otherwise, choose control replicate i for IP replicate i.

- For the paired-end test data set:

```
$ cd $HOME/chip-seq-pipeline2
$ caper run chip.wdl -i ENCSR936XTK.json --docker --max-concurrent-tasks 1
```

Procedure using Caper with Singularity

Run pipelines for test samples. Caper will automatically build a local Singularity container before running a pipeline.

- For the single-end test data set:

```
$ cd $HOME/chip-seq-pipeline2
$ caper run chip.wdl -i ENCSR000DYI.json --singularity --max-concurrent-tasks 1
```

- For the paired-end test data set:

```
$ cd $HOME/chip-seq-pipeline2
$ caper run chip.wdl -i ENCSR936XTK.json --singularity --max-concurrent-tasks 1
```

Organizing outputs for Caper methods

This section describes how to organize raw outputs from Caper methods. Caper is a wrapper for Cromwell and Cromwell generates outputs in a directory structure based on the name of a task. All raw outputs are written to `./chip/[UUID_FOR_PIPELINE]/` and it's not very user-friendly. Croo organizes such raw outputs and makes a table describing all organized output files, task graph and UCSC genome browser tracks as shown in Figure 7.

Install Croo first.

```
$ pip install croo
```

Croo's input is a final output of Caper `metadata.json`, which can be found on the bottom of Caper's screen log. Let us call it `METADATA_JSON`. For this demonstration, it is not possible to see UCSC genome browser tracks in Croo's HTML report since pipelines are run locally without a web server hosting files to be visualized and the UCSC genome browser can only take public URLs.

```
$ mkdir -p croo_out
$ croo [METADATA_JSON] --out-dir croo_out
```

Check an HTML file on `croo_out` directory with your web browser. In order to see a final HTML report for the test sample (ENCSR000DYI or ENCSR936XTK), find `QC and logs/Final HTML report` in the file table and click on it.

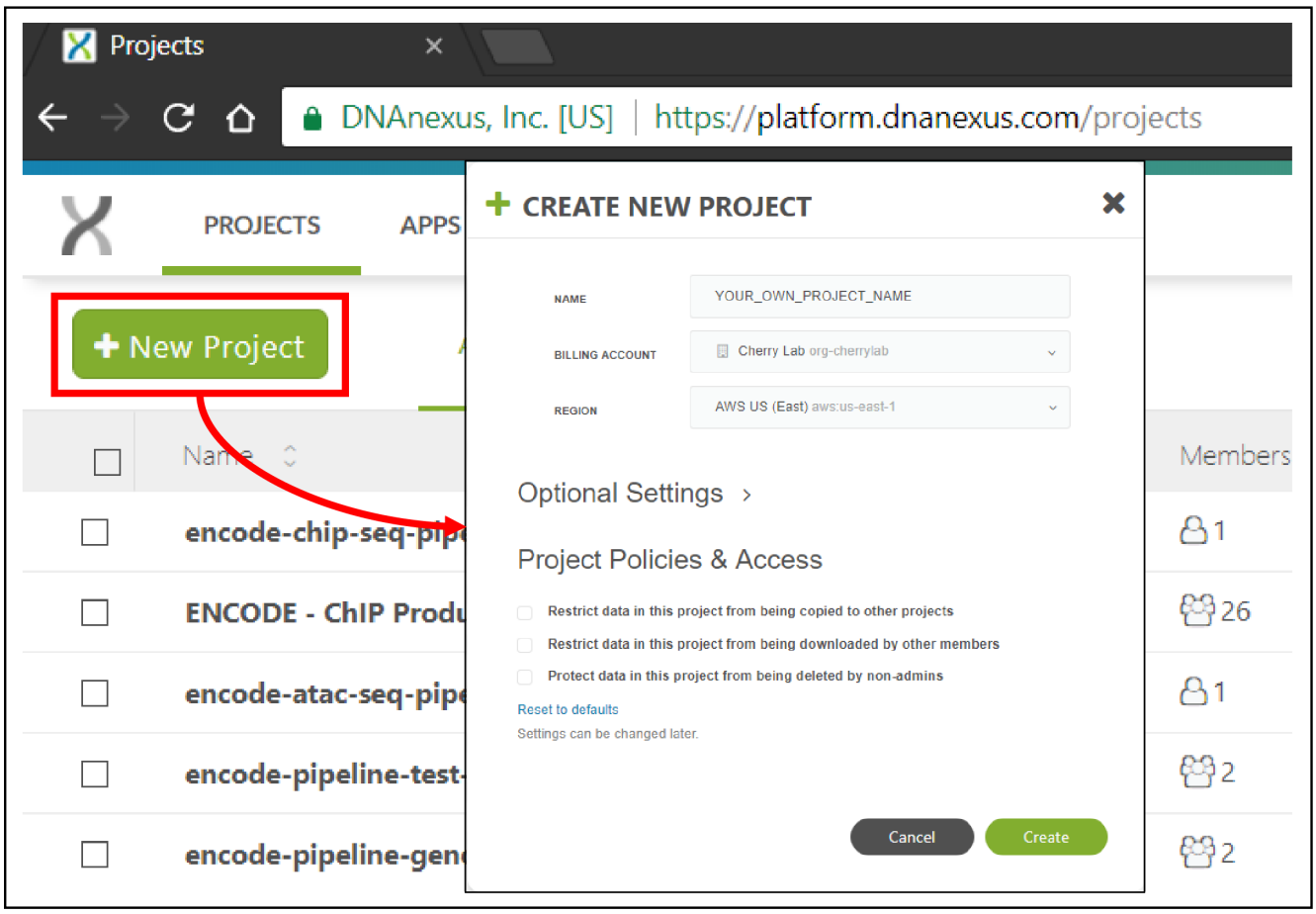


Figure 8: Creating a new DNAnexus project on the DNAnexus website .

Procedure using Bash shell commands

This section describes all subtasks in the AQUAS protocol, with each individual shell command line in each subtask.

1. *Create a directory structure for outputs.* Create directories to store mapped/filtered BAM files, tagAlign files, and quality metrics.

```
$ cd $HOME/chip-seq-pipeline2
$ mkdir align; mkdir qc
```

Create directories to store called peaks.

```
$ mkdir peak; mkdir peak/reps; mkdir peak/pooled;
$ mkdir peak/self_pseudo_reps; mkdir peak/pooled_pseudo_reps
```

Create directories to store signal tracks

```
$ mkdir signal; mkdir signal/reps; mkdir signal/pooled;
```

Create directories to store IDR peaks.

```
$ mkdir idr; mkdir idr/true_reps; mkdir idr/pooled;
$ mkdir idr/self_pseudo_reps; mkdir idr/pooled_pseudo_reps
```

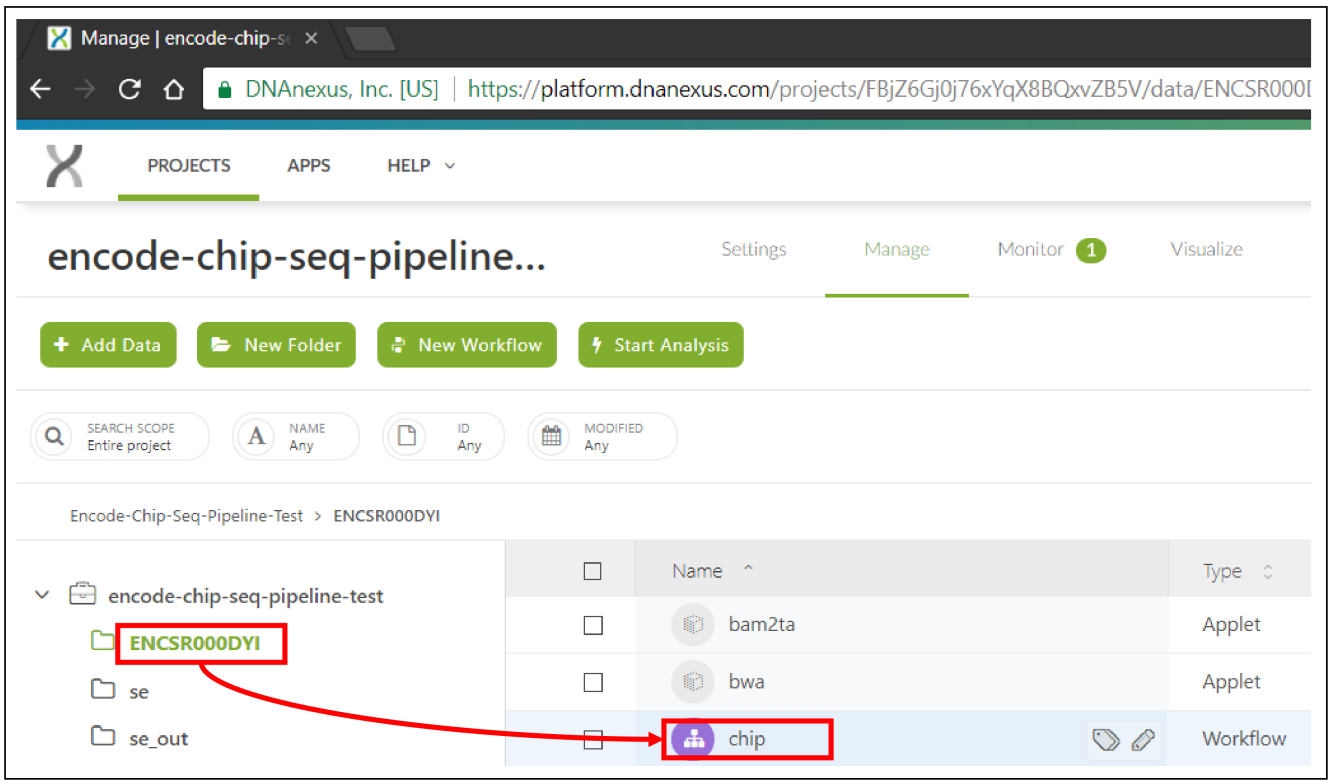


Figure 9: Creating a AQUAS ChIP workflow on the DNAnexus website .

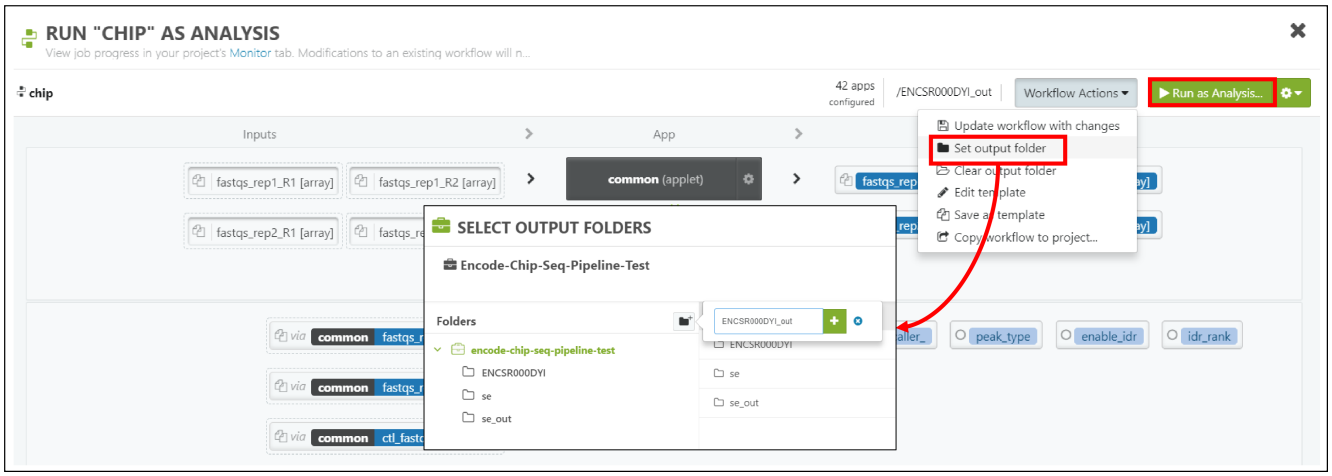


Figure 10: Setting up an output directory for the AQUAS ChIP workflow on the DNAnexus website .

2. *Run Singularity to load a containerized environment for the pipeline.*
   Please make sure that `Singularity>` is shown as the shell prompt after loading the containerized environment.

   ```
   $ singularity run --cleanenv --no-home docker://quay.io/encode-dcc/chip-seq-pipeline:v1.3.6
   ```

   Map raw reads: TIMING ~16 h (when fully parallelized)

3. *Map raw reads in FASTQ format and save the alignments as BAM files.* Use two threads per FASTQ. Set an environment variable for the `bowtie2` index.

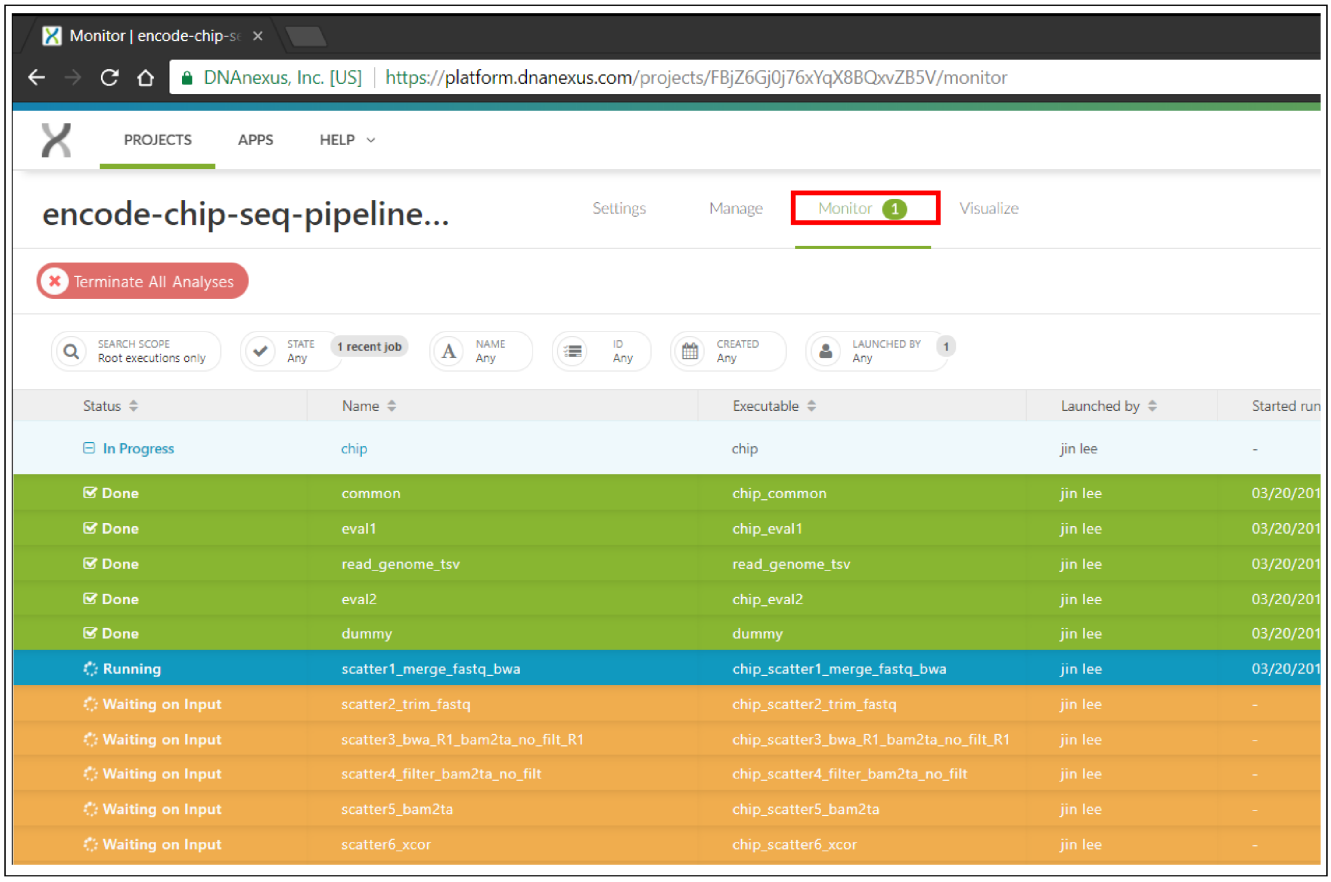


Figure 11: Example of the monitor page on the DNAnexus website .

```
$ NTH=2; BOWTIE2_IDX=hg38/bowtie2_index/GRCh38_no_alt_analysis_set_GCA_000001405.15.fasta
```

Untar the index TAR file and get back to the pipeline directory.

```
$ cd hg38/bowtie2_index/
$ tar xvf GRCh38_no_alt_analysis_set_GCA_000001405.15.fasta.tar
$ cd ../../
```

Start with ChIP replicate 1.

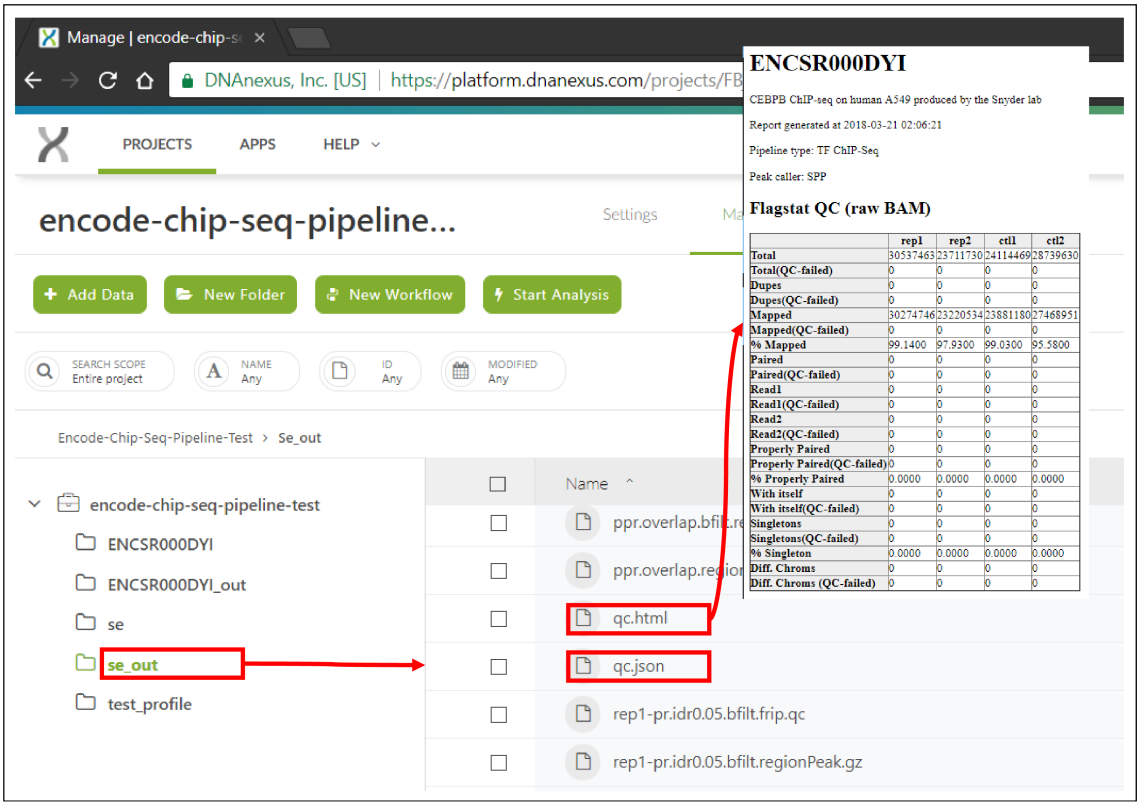
```
$ REP=rep1
```

For single-end datasets:

```
$ bowtie2 --mm --threads $NTH -x $BOWTIE2_IDX -U $REP.fastq.gz | samtools view -Su
   /dev/stdin | samtools sort /dev/stdin -o align/$REP.bam -T align/$REP
$ samtools index align/$REP.bam
$ samtools sort -n align/$REP.bam -T align/$REP -O sam | SAMstats --sorted_sam_file -
   --outf qc/$REP.samstat.qc
```

There is an additional alignment step for cross correlation analysis only. This step is for IP replicates only. A FASTQ file for a IP replicate must be trimmed to 50bp by default and aligned separately.

```
$ python $(which trimfastq.py) $REP.fastq.gz 50 | gzip -nc > align/$REP.trim_50bp.fastq.gz
```



	rep1	rep2	ctl1	ctl2
Total	30537463	23711730	24114469	28739630
Total(QC-failed)	0	0	0	0
Dupes	0	0	0	0
Dupes(QC-failed)	0	0	0	0
Mapped	30274746	23220534	23881180	27468951
Mapped(QC-failed)	0	0	0	0
% Mapped	99.1400	97.9300	99.0300	95.5800
Paired	0	0	0	0
Paired(QC-failed)	0	0	0	0
Read1	0	0	0	0
Read1(QC-failed)	0	0	0	0
Read2	0	0	0	0
Read2(QC-failed)	0	0	0	0
Properly Paired	0	0	0	0
Properly Paired(QC-failed)	0	0	0	0
% Properly Paired	0.0000	0.0000	0.0000	0.0000
With itself	0	0	0	0
With itself(QC-failed)	0	0	0	0
Singletons	0	0	0	0
Singletons(QC-failed)	0	0	0	0
% Singleton	0.0000	0.0000	0.0000	0.0000
Diff. Chroms	0	0	0	0
Diff. Chroms (QC-failed)	0	0	0	0

Figure 12: Output and final reports from the AQUAS ChIP workflow on the DNAnexus website .

```
$ bowtie2 --mm --threads $NTH -x $BOWTIE2_IDX -U align/$REP.fastq.gz |
   samtools view -Su /dev/stdin | samtools sort /dev/stdin
   -o align/$REP.trim_50bp.bam -T align/$REP.trim_50bp
$ samtools index align/$REP.trim_50bp.bam
```

However, our test single-end dataset is already trimmed to 50bp so we skip trimming for this specific test dataset.

```
$ cp align/$REP.bam align/$REP.trim_50bp.bam
$ samtools index align/$REP.trim_50bp.bam
```

For paired-end datasets, reads are mapped as paired-end:

```
$ bowtie2 -X2000 --mm --threads $NTH -x $BOWTIE2_IDX -1 $REP-R1.fastq.gz
   -2 $REP-R2.fastq.gz | samtools view -Su /dev/stdin | samtools sort
   /dev/stdin -o align/$REP.bam -T align/$REP
$ samtools index align/$REP.bam
$ samtools sort -n align/$REP.bam -T align/$REP -O sam | SAMstats
   --sorted_sam_file - --outf qc/$REP.samstat.qc
```

Additionally the first end is mapped separately after trimming (this step is done for cross correlation analysis only, as discussed above). This step is for IP replicates only.:

```
$ python $(which trimfastq.py) $REP-R1.fastq.gz 50 | gzip -nc > $REP-R1.trim_50bp.fastq.gz
$ bowtie2 --mm --threads $NTH -x $BOWTIE2_IDX -U
  align/$REP-R1.trim_50bp.fastq.gz | samtools view -Su /dev/stdin
  | samtools sort /dev/stdin -o align/$REP.trim_50bp.bam
  -T align/$REP.trim_50bp
$ samtools index align/$REP.trim_50bp.bam
```

Repeat for other IP/control replicates (`REP=rep2`, `REP=ctl1` and `REP=ctl2`).

Remove temporary files.

```
$ rm -f align/*.trim_50bp.fastq.gz
```

Quality metrics for Bowtie2 mapping for each IP/control replicate are generated as shown in Figure 13.

SAMstat (raw unfiltered BAM)

	rep1	rep2	ctl1	ctl2
Total Reads	69122852	86590386	90753054	86825372
Total Reads (QC-failed)	0	0	0	0
Duplicate Reads	0	0	0	0
Duplicate Reads (QC-failed)	0	0	0	0
Mapped Reads	68199569	83912929	89404845	85411524
Mapped Reads (QC-failed)	0	0	0	0
% Mapped Reads	98.7	96.89999999999999	98.5	98.4
Paired Reads	69122852	86590386	90753054	86825372
Paired Reads (QC-failed)	0	0	0	0
Read1	34561426	43295193	45376527	43412686
Read1 (QC-failed)	0	0	0	0
Read2	34561426	43295193	45376527	43412686
Read2 (QC-failed)	0	0	0	0
Properly Paired Reads	67803592	83381260	88446836	84530314
Properly Paired Reads (QC-failed)	0	0	0	0
% Properly Paired Reads	98.1	96.3	97.5	97.39999999999999
With itself	67888798	83525240	88825402	84840910
With itself (QC-failed)	0	0	0	0
Singletons	310771	387689	579443	570614
Singletons (QC-failed)	0	0	0	0
% Singleton	0.4	0.4	0.6	0.7000000000000001
Diff. Chroms	22493	29958	109194	94732
Diff. Chroms (QC-failed)	0	0	0	0

Figure 13: Example of pipeline output summary. Alignment and mapping quality statistics.

Prepare sequences for AQUAS. TIMING ~4 hours:

4. *Filter multimapping reads.* If the BAM file contains reads aligned to multiple places in the genome, they can be filter using `samtools -q` (if MAPQ<30). Start with IP replicate 1:

```
$ REP=rep1
```

For single-end datasets:

```
$ samtools view -F 1804 -q 30 -u align/$REP.bam |
  samtools sort /dev/stdin -o align/$REP.filt.bam -T align/$REP.filt
```

Mark duplicate reads for single-end datasets:

```
$ export _JAVA_OPTIONS="-Xms256M -Xmx$12G -XX:ParallelGCThreads=1"
$ $(which picard) MarkDuplicates INPUT="align/$REP.filt.bam"
  OUTPUT="align/$REP.dupmark.bam" METRICS_FILE="qc/$REP.dup"
  VALIDATION_STRINGENCY=LENIENT ASSUME_SORTED=true REMOVE_DUPLICATES=false
```

Remove duplicate reads for single-end datasets:

```
$ samtools view -F 1804 -b align/$REP.dupmark.bam > align/$REP.nodup.bam
$ samtools index align/$REP.nodup.bam
$ samtools sort -n align/$REP.nodup.bam -T align/$REP.nodup -O sam
    | SAMstats --sorted_sam_file - --outf qc/$REP.nodup.samstat.qc
```

Compute library complexity for single-end datasets:

```
$ bedtools bamtobed -i align/$REP.dupmark.bam | awk 'BEGIN{OFS="\t"}{print $1,$2,$3,$6}'
  | grep -v 'chrM' | sort | uniq -c | awk 'BEGIN{mt=0;m0=0;m1=0;m2=0} ($1==1){m1=m1+1}
  ($1==2){m2=m2+1} {m0=m0+1} {mt=mt+$1} END{m1_m2=-1.0; if(m2>0) m1_m2=m1/m2;
  printf "%d\t%d\t%d\t%d\t%f\t%f\t%f\n",mt,m0,m1,m2,m0/mt,m1/m0,m1_m2}' > qc/$REP.pbc
```

For paired-end datasets:

```
$ samtools view -F 1804 -f 2 -q 30 -u align/$REP.bam
  | samtools sort -n /dev/stdin -o align/$REP.tmp.bam -T align/$REP.tmp
$ samtools fixmate -r align/$REP.tmp.bam align/$REP.fixmate.bam
$ samtools view -F 1804 -f 2 -u align/$REP.fixmate.bam
  | samtools sort /dev/stdin -o align/$REP.filt.bam -T align/$REP.filt
```

Mark duplicate reads for paired-end datasets:

```
$ export _JAVA_OPTIONS="-Xms256M -Xmx$12G -XX:ParallelGCThreads=1"
$ $(which picard) MarkDuplicates INPUT="align/$REP.filt.bam"
  OUTPUT="align/$REP.dupmark.bam" METRICS_FILE="qc/$REP.dup"
  VALIDATION_STRINGENCY=LENIENT ASSUME_SORTED=true REMOVE_DUPLICATES=false
```

Remove duplicate reads for paired-end datasets:

```
$ samtools view -F 1804 -f 2 -b align/$REP.dupmark.bam > align/$REP.nodup.bam
$ samtools index align/$REP.nodup.bam
$ samtools sort -n align/$REP.nodup.bam -T align/$REP.nodup -O sam
   | SAMstats --sorted_sam_file - --outf qc/$REP.nodup.samstat.qc
$ samtools sort -n align/$REP.dupmark.bam -o align/$REP.dupmark.tmp.bam
   -T align/$REP.dupmark.tmp
```

Compute library complexity for paired-end datasets:

```
$ bedtools bamtobed -bedpe -i align/$REP.dupmark.tmp.bam
  | awk 'BEGIN{OFS="\t"}{print $1,$2,$4,$6,$9,$10}' | grep -v 'chrM' | sort
  | uniq -c | awk 'BEGIN{mt=0;m0=0;m1=0;m2=0} ($1==1){m1=m1+1} ($1==2){m2=m2+1}
  {m0=m0+1} {mt=mt+$1} END{m1_m2=-1.0; if(m2>0) m1_m2=m1/m2;
  printf "%d\t%d\t%d\t%d\t%f\t%f\t%f\n",mt,m0,m1,m2,m0/mt,m1/m0,m1_m2}' > qc/$REP.pbc
```

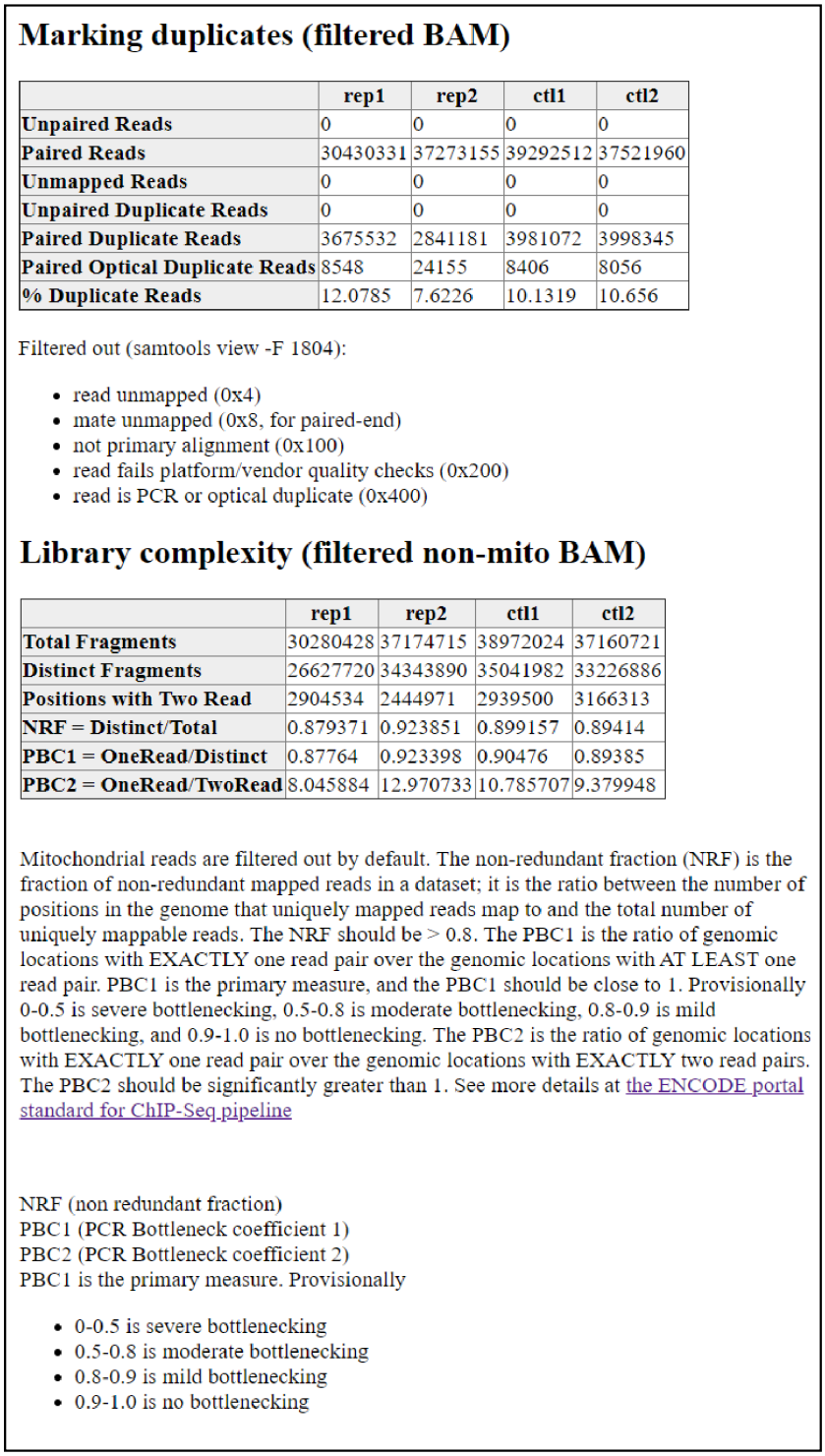
Repeat for other IP/control replicates (`REP=rep2`, `REP=ctl1` and `REP=ctl2`).

Additionally, filter out unmapped reads for cross-correlation analysis. This step is for IP replicates only. Repeat this for `REP=rep2`.

```
$ REP=rep1
$ samtools view -F 1804 -q 30 -u align/$REP.trim_50bp.bam |
  samtools sort /dev/stdin -o align/$REP.trim_50bp.filt.bam -T align/$REP.trim_50bp.filt
```

Remove temporary files.

```
$ rm -f align/*.tmp.bam* align/*.fixmate.bam* align/*.dupmark.bam* align/*.filt.bam*
```



Marking duplicates (filtered BAM)

	rep1	rep2	ctl1	ctl2
Unpaired Reads	0	0	0	0
Paired Reads	30430331	37273155	39292512	37521960
Unmapped Reads	0	0	0	0
Unpaired Duplicate Reads	0	0	0	0
Paired Duplicate Reads	3675532	2841181	3981072	3998345
Paired Optical Duplicate Reads	8548	24155	8406	8056
% Duplicate Reads	12.0785	7.6226	10.1319	10.656

Filtered out (samtools view -F 1804):

- read unmapped (0x4)
- mate unmapped (0x8, for paired-end)
- not primary alignment (0x100)
- read fails platform/vendor quality checks (0x200)
- read is PCR or optical duplicate (0x400)

Library complexity (filtered non-mito BAM)

	rep1	rep2	ctl1	ctl2
Total Fragments	30280428	37174715	38972024	37160721
Distinct Fragments	26627720	34343890	35041982	33226886
Positions with Two Read	2904534	2444971	2939500	3166313
NRF = Distinct/Total	0.879371	0.923851	0.899157	0.89414
PBC1 = OneRead/Distinct	0.87764	0.923398	0.90476	0.89385
PBC2 = OneRead/TwoRead	8.045884	12.970733	10.785707	9.379948

Mitochondrial reads are filtered out by default. The non-redundant fraction (NRF) is the fraction of non-redundant mapped reads in a dataset; it is the ratio between the number of positions in the genome that uniquely mapped reads map to and the total number of uniquely mappable reads. The NRF should be > 0.8. The PBC1 is the ratio of genomic locations with EXACTLY one read pair over the genomic locations with AT LEAST one read pair. PBC1 is the primary measure, and the PBC1 should be close to 1. Provisionally 0-0.5 is severe bottlenecking, 0.5-0.8 is moderate bottlenecking, 0.8-0.9 is mild bottlenecking, and 0.9-1.0 is no bottlenecking. The PBC2 is the ratio of genomic locations with EXACTLY one read pair over the genomic locations with EXACTLY two read pairs. The PBC2 should be significantly greater than 1. See more details at the ENCODE portal standard for ChIP-Seq pipeline

NRF (non redundant fraction)
PBC1 (PCR Bottleneck coefficient 1)
PBC2 (PCR Bottleneck coefficient 2)
PBC1 is the primary measure. Provisionally

- 0-0.5 is severe bottlenecking
- 0.5-0.8 is moderate bottlenecking
- 0.8-0.9 is mild bottlenecking
- 0.9-1.0 is no bottlenecking

Figure 14: Example of pipeline output summary. Duplicate filtering and library complexity/bottlenecking metrics.

Quality metrics for BAM filtering for each IP/control replicate are generated as shown in Figure 14.

5. *Convert file formats.* For the IDR rescue strategy later, the alignment files have to be shuffled and subsampled first. This is implemented in a `tagAlign` text file format using the UNIX `'shuf'` command. Covert `BAM` to `tagAlign` as follows:

```
$ REP=rep1
```

For single-end datasets:

```
$ bedtools bamtobed -i align/$REP.nodup.bam
  | awk 'BEGIN{OFS="\t"}{$4="N";$5="1000";print $0}'
  | gzip -nc > align/$REP.tagAlign.gz
$ bedtools bamtobed -i align/$REP.trim_50bp.filt.bam
  | awk 'BEGIN{OFS="\t"}{$4="N";$5="1000";print $0}'
  | gzip -nc > align/$REP.xcor.tagAlign.gz
```

For paired-end datasets:

```
$ samtools sort -n align/$REP.nodup.bam -o align/$REP.nodup.nmsrt.bam -T align/$REP.nodup.nmsrt
$ bedtools bamtobed -bedpe -mate1 -i align/$REP.nodup.nmsrt.bam
  | gzip -nc > align/$REP.bedpe
$ zcat align/$REP.bedpe | awk 'BEGIN{OFS="\t"}
  {printf "%s\t%s\t%s\tN\t1000\t%s\n%s\t%s\t%s\tN\t1000\t%s\n",$1,$2,$3,$9,$4,$5,$6,$10}'
  | gzip -nc > align/$REP.tagAlign.gz
$ bedtools bamtobed -i align/$REP.trim_50bp.filt.bam
  | awk 'BEGIN{OFS="\t"}{$4="N";$5="1000";print $0}'
  | gzip -nc > align/$REP.xcor.tagAlign.gz
```

Repeat for other IP/control replicates (`REP=rep2`, `REP=ctl1` and `REP=ctl2`).

Remove temporary files.

```
$ rm -f align/*.nodup.nmsrt.bam
```

Calculating cross-correlation metrics. TIMING ~30 min:

6. Subsample `tagAlign` and `BEDPE` files down to 1.5 $\times 10^7$ aligned reads, and compute cross-correlation measures (NSC and RSC). Start with IP replicate 1:

```
$ REP=rep1
```

For both single-end and paired-end datasets:

```
$ zcat align/$REP.xcor.tagAlign.gz | grep -v "chrM"
  | shuf -n 15000000 --random-source=<(openssl enc -aes-256-ctr -pass pass:$(zcat
  -f align/$REP.xcor.tagAlign.gz | wc -c) -nosalt </dev/zero 2>/dev/null) | gzip
  -nc > align/$REP.15M.xcor.tagAlign.gz
```

Estimate read length from first 100 reads.

```
$ zcat align/$REP.15M.xcor.tagAlign.gz > align/$REP.15M.xcor.tagAlign.gz.tmp
$ READ_LEN=$(head -n 100 align/$REP.15M.xcor.tagAlign.gz.tmp | awk 'function
  abs(v) {{return v < 0 ? -v : v}} BEGIN{{sum=0}} {{sum+=abs($3-$2)}}
  END{{print int(sum/NR)}}')
```

Determine exclusion range for fragment length estimation. Use a fixed lowerbound at -500. The upperbound `EXCLUSION_RANGE_MAX` is `max(read_len + 10, 50)` and `max(read_len + 10, 100)` for TF ChIP-Seq and Histone ChIP-Seq, respectively.

```
$ EXCLUSION_RANGE_MIN=-500
$ rm -f align/$REP.15M.xcor.tagAlign.gz.tmp
```

Run the `run_spp.R` R script to carry out cross-correlation analysis. For the details of the options and output format, please see Box 3.

```
$ Rscript $(which run_spp.R) -rf -c=align/$REP.15M.xcor.tagAlign.gz
  -filtchr=chrM -savp=qc/$REP.cc.pdf -out=qc/$REP.cc
  -x=${EXCLUSION_RANGE_MIN}:${EXCLUSION_RANGE_MAX}
$ sed -r 's/,[^\t]+//g' qc/$REP.cc > qc/$REP.cc.tmp
$ mv qc/$REP.cc.tmp qc/$REP.cc
```

Repeat for IP replicate 2 (`REP=rep2`).

Examine output results:

```
$ cat qc/*.cc
```

In the case of examples used here, the output is as follows.

For single-end datasets:

```
rep1.15M.xcor.tagAlign.gz 15000000 125 0.213726977759017 40
   0.186234 1500 0.1595652 1.339433 2.030905 2
rep2.15M.xcor.tagAlign.gz 15000000 105 0.234591645399676 40
   0.197753 1500 0.1552025 1.511519 1.865762 2
```

For paired-end datasets, as shown in Figure 15:

```
rep1.15M.xcor.tagAlign.gz 15000000 225 0.248815946337786 55
   0.1872084 1500 0.1387214 1.793638 2.270599 2
rep2.15M.xcor.tagAlign.gz 15000000 215 0.229553244602288 55
   0.1926292 1500 0.162315 1.414246 2.218041 2
```

Plots for cross-correlation analysis for each IP replicate are generated as shown in Figure 16.

7. Compute JSD metric

```
$ bedtools intersect -nonamecheck -v -abam align/rep1.nodup.bam -b
   hg38/hg38.blacklist.bed.gz > align/rep1.nodup.bfilt.bam
$ bedtools intersect -nonamecheck -v -abam align/rep2.nodup.bam -b
   hg38/hg38.blacklist.bed.gz > align/rep2.nodup.bfilt.bam
$ bedtools intersect -nonamecheck -v -abam align/ctl1.nodup.bam -b
   hg38/hg38.blacklist.bed.gz > align/ctl1.nodup.bfilt.bam
$ samtools index align/rep1.nodup.bfilt.bam
$ samtools index align/rep2.nodup.bfilt.bam
$ samtools index align/ctl1.nodup.bfilt.bam
$ LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8 plotFingerprint -b
  align/rep1.nodup.bfilt.bam align/rep2.nodup.bfilt.bam
  align/ctl1.nodup.bfilt.bam \
--JSDsample align/ctl1.nodup.bfilt.bam \
--labels rep1 rep2 ctl1 \
--outQualityMetrics qc/jsd.qc \
--minMappingQuality 30 \
-T "Fingerprints of different samples" \
--numberOfProcessors 4 \
--plotFile qc/jsd.png
```

JSD-based quality metrics are generated as shown in Figure 17.

Remove temporary files.

```
$ rm -f align/*.nodup.bfilt.bam
```

Strand cross-correlation measures (trimmed/filtered SE BAM)

	rep1	rep2
Number of Subsampled Reads	15000000	15000000
Estimated Fragment Length	225	210
Cross-correlation at Estimated Fragment Length	0.248728899064887	0.229513459729301
Phantom Peak	55	55
Cross-correlation at Phantom Peak	0.1872022	0.1929582
Argmin of Cross-correlation	1500	1500
Minimum of Cross-correlation	0.1387036	0.1622947
NSC (Normalized Strand Cross-correlation coeff.)	1.793241	1.414177
RSC (Relative Strand Cross-correlation coeff.)	2.268628	2.192146

Performed on subsampled (15000000) reads mapped from FASTQs that are trimmed to 50. Such FASTQ trimming and subsampling reads are for cross-corrleation analysis only. Untrimmed FASTQs are used for all the other analyses.

NOTE1: For SE datasets, reads from replicates are randomly subsampled to 15000000.
NOTE2: For PE datasets, the first end (R1) of each read-pair is selected and trimmed to 50 the reads are then randomly subsampled to 15000000.

- Normalized strand cross-correlation coefficient (NSC) = col9 in outFile
- Relative strand cross-correlation coefficient (RSC) = col10 in outFile
- Estimated fragment length = col3 in outFile, take the top value

Figure 15: Example of pipeline output summary. Cross-correlation analysis QC metrics.

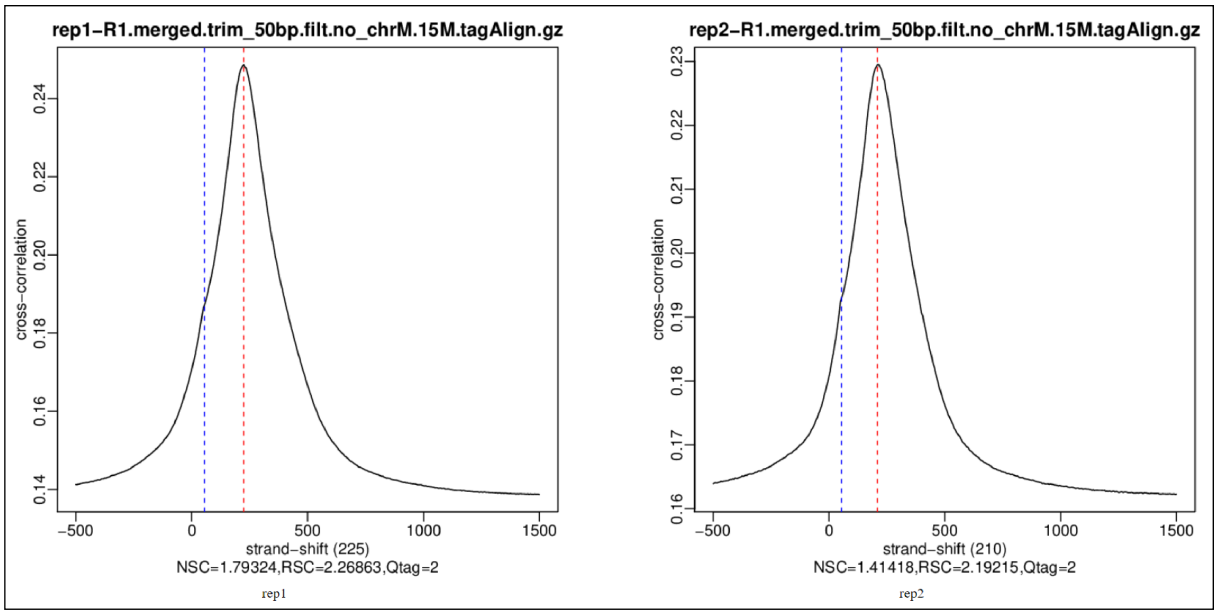


Figure 16: Example of pipeline output summary. Cross-correlation analysis plots.

Jensen-Shannon distance (filtered/deduped BAM)

	rep1	rep2
AUC	0.2536078629793387	0.2832876244897156
Synthetic AUC	0.495214842660931	0.49578043779370823
X-intercept	0.10873902873023172	0.0996921046844073
Synthetic X-intercept	0.0	0.0
Elbow Point	0.6563588379950817	0.6155387168362757
Synthetic Elbow Point	0.4998866984172783	0.5004638855712935
JS Distance	0.21810209741121928	0.17190962171426702
Synthetic JS Distance	0.35029970710339703	0.30244217373989163
% Genome Enriched	28.817002219245456	32.58951956334846
Diff. Enrichment	24.597780584105593	20.559528093285994
CHANCE Divergence	0.21000109692756722	0.17528583040824947

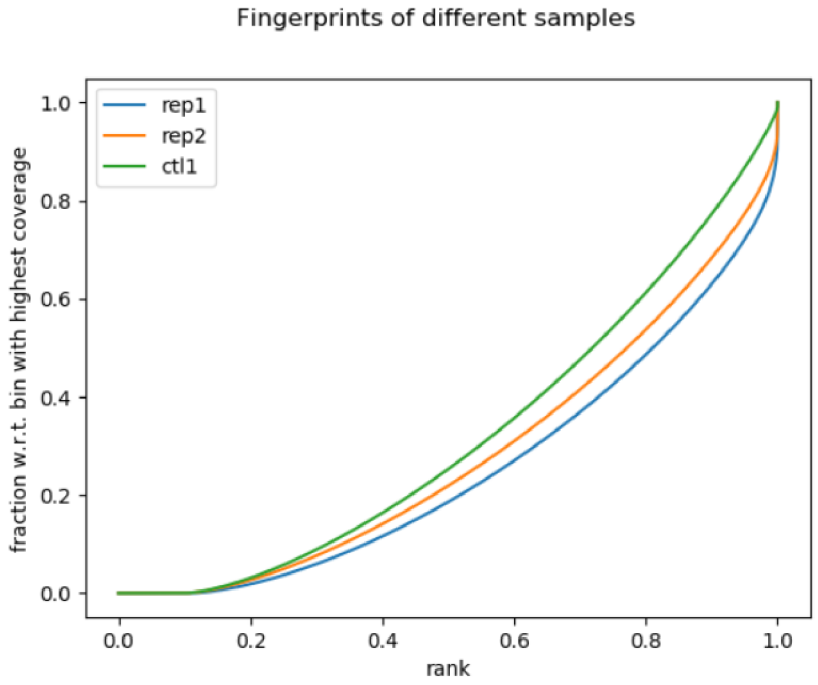


Figure 17: Example of pipeline output summary. Fingerprint and JS distance metrics.

Prepare input files for the IDR framework. TIMING < 20 minutes:

For self-consistency analysis, IDR analyses will be performed over each replicate, merged replicates (pooled data), pooled and individual pseudo-replicates.

8. Subsample to make pseudo-replicates. For self-consistency analysis, randomly split each IP replicate and pooled IP data. Start with IP replicate 1:

```
$ REP=rep1
```

For single-end datasets:

```
$ nlines=$( zcat align/$REP.tagAlign.gz | wc -l )
```

```
$ nlines=$(( (nlines + 1) / 2 ))
$ zcat align/$REP.tagAlign.gz
  | shuf --random-source=<(openssl enc -aes-256-ctr -pass pass:$(zcat -f align/$REP.tagAlign.gz
  | wc -c) -nosalt </dev/zero 2>/dev/null) | split -d -l $((nlines)) - align/$REP.bam_pr1.
$ gzip -nc align/$REP.bam_pr1.00 > align/$REP.pr1.tagAlign.gz
$ gzip -nc align/$REP.bam_pr1.01 > align/$REP.pr2.tagAlign.gz
$ rm -f align/$REP.bam_pr1.00 align/$REP.bam_pr1.01
```

For paired-end datasets:

```
$ nlines=$( zcat align/$REP.bedpe | wc -l )
$ nlines=$(( (nlines + 1) / 2 ))
$ zcat align/$REP.bedpe | shuf --random-source=align/$REP.bedpe
  | split -d -l $((nlines)) - align/$REP.bam_pr1.
$ awk 'BEGIN{OFS="\t"}
  {printf "%s\t%s\t%s\tN\t1000\t%s\n%s\t%s\t%s\tN\t1000\t%s\n",$1,$2,$3,$9,$4,$5,$6,$10}'
  "align/$REP.bam_pr1.00" | gzip -nc > align/$REP.pr1.tagAlign.gz
$ awk 'BEGIN{OFS="\t"}
  {printf "%s\t%s\t%s\tN\t1000\t%s\n%s\t%s\t%s\tN\t1000\t%s\n",$1,$2,$3,$9,$4,$5,$6,$10}'
  "align/$REP.bam_pr1.01" | gzip -nc > align/$REP.pr2.tagAlign.gz
$ rm -f align/$REP.bam_pr1.00 align/$REP.bam_pr1.01
```

Repeat for IP replicate 2 (`REP=rep2`).

9. Merge replicates (both for ChIP samples and control samples):

   For IP samples:

```
$ zcat align/rep1.tagAlign.gz align/rep2.tagAlign.gz | gzip -nc > align/rep0.tagAlign.gz
$ zcat align/rep1.pr1.tagAlign.gz align/rep2.pr1.tagAlign.gz
                                                    | gzip -nc > align/rep0.pr1.tagAlign.gz
$ zcat align/rep1.pr2.tagAlign.gz align/rep2.pr2.tagAlign.gz
                                                    | gzip -nc > align/rep0.pr2.tagAlign.gz
```

   For control samples:

```
$ zcat align/ctl1.tagAlign.gz align/ctl2.tagAlign.gz | gzip -nc > align/ctl0.tagAlign.gz
```

   IDR analysis: TIMING $<$ 1 day

10. Generate relaxed peak calls. For each IP replicate, pooled IP and its pseudo-replicates, perform peak calling with relaxed thresholds to call up to 300,000 peaks using the SPP peak caller ($<$ 1 day). Run the `run_spp.R` R script to carry out this step. For details regarding options and output formats, please see Box 3. Mean fragment length based on the fragment length for each of the original replicates (from the 3rd column of cross-correlation quality analysis log) is used for pooled replicates and pooled pseudo replicates:

```
$ fraglen_rep1=$(cat qc/rep1.cc | awk '{print $3}')
$ fraglen_rep2=$(cat qc/rep2.cc | awk '{print $3}')
$ fraglen_mean=$(( (fraglen_rep1+fraglen_rep2)/2 ))
```

    For convenience, we call peaks with a pooled control in this demonstration. However in the pipeline, an appropriate control is chosen according to the criteria described in Box 10. Use a flag `'-use_pooled_ctl'` to force the pipeline to use pooled control for calling peaks. Use 2 threads for each case. Use pooled control replicate for all cases.

```
$ NTH=2; CTL=ctl0
```

    Start with IP replicate 1. Convert scientific representation in genome coordinates to integer and set negative coordinates to 0.

Reproducibility QC and peak detection statistics

	overlap	idr
Nt	94929	27499
N1	83933	21946
N2	67259	14072
Np	98532	29761
N optimal	98532	29761
N conservative	94929	27499
Optimal Set	pooled-pr1_vs_pooled-pr2	pooled-pr1_vs_pooled-pr2
Conservative Set	rep1_vs_rep2	rep1_vs_rep2
Rescue Ratio	1.037954681920172	1.0822575366376959
Self Consistency Ratio	1.2479073432551777	1.55955088118249
Reproducibility Test	pass	pass

Reproducibility QC

- N1: Replicate 1 self-consistent peaks (comparing two pseudoreplicates generated by subsampling Rep1 reads)
- N2: Replicate 2 self-consistent peaks (comparing two pseudoreplicates generated by subsampling Rep2 reads)
- Ni: Replicate i self-consistent peaks (comparing two pseudoreplicates generated by subsampling RepX reads)
- Nt: True Replicate consistent peaks (comparing true replicates Rep1 vs Rep2)
- Np: Pooled-pseudoreplicate consistent peaks (comparing two pseudoreplicates generated by subsampling pooled reads from Rep1 and Rep2)
- Self-consistency Ratio: max(N1,N2) / min (N1,N2)
- Rescue Ratio: max(Np,Nt) / min (Np,Nt)
- Reproducibility Test: If Self-consistency Ratio >2 AND Rescue Ratio > 2, then 'Fail' else 'Pass'

Figure 18: Example of pipeline output summary. IDR analysis QC metrics.

Fraction of reads in peaks (FRiP)

FRiP for overlap peaks

	rep1_vs_rep2	rep1-pr1_vs_rep1-pr2	rep2-pr1_vs_rep2-pr2	pooled-pr1_vs_pooled-pr2
Fraction of Reads in Peaks	0.132298936569183	0.159811049225225	0.0964785957377872	0.13441311082053634

FRiP for IDR peaks

	rep1_vs_rep2	rep1-pr1_vs_rep1-pr2	rep2-pr1_vs_rep2-pr2	pooled-pr1_vs_pooled-pr2
Fraction of Reads in Peaks	0.09386920110985425	0.11405520183500538	0.06198790693789441	0.09597179606121735

For spp raw peaks:

- repX: Peak from true replicate X
- repX-prY: Peak from Yth pseudoreplicates from replicate X
- pooled: Peak from pooled true replicates (pool of rep1, rep2, ...)
- pooled-pr1: Peak from 1st pooled pseudo replicate (pool of rep1-pr1, rep2-pr1, ...)
- pooled-pr2: Peak from 2nd pooled pseudo replicate (pool of rep1-pr2, rep2-pr2, ...)

For overlap/IDR peaks:

- repX_vs_repY: Comparing two peaks from true replicates X and Y
- repX-pr1_vs_repX-pr2: Comparing two peaks from both pseudoreplicates from replicate X
- pooled-pr1_vs_pooled-pr2: Comparing two peaks from 1st and 2nd pooled pseudo replicates

Figure 19: Example of pipeline output summary. Fraction of reads in peaks (FRiP) QC metric.

```
$ REP=rep1; FRAGLEN=$fraglen_rep1; OUTDIR=peak/reps;
$ Rscript $(which run_spp.R) -c=align/$REP.tagAlign.gz -p=$NTH
  -i=align/$CTL.tagAlign.gz -npeak=300000 -odir=$OUTDIR -speak=$FRAGLEN -savr -savp -rf
```

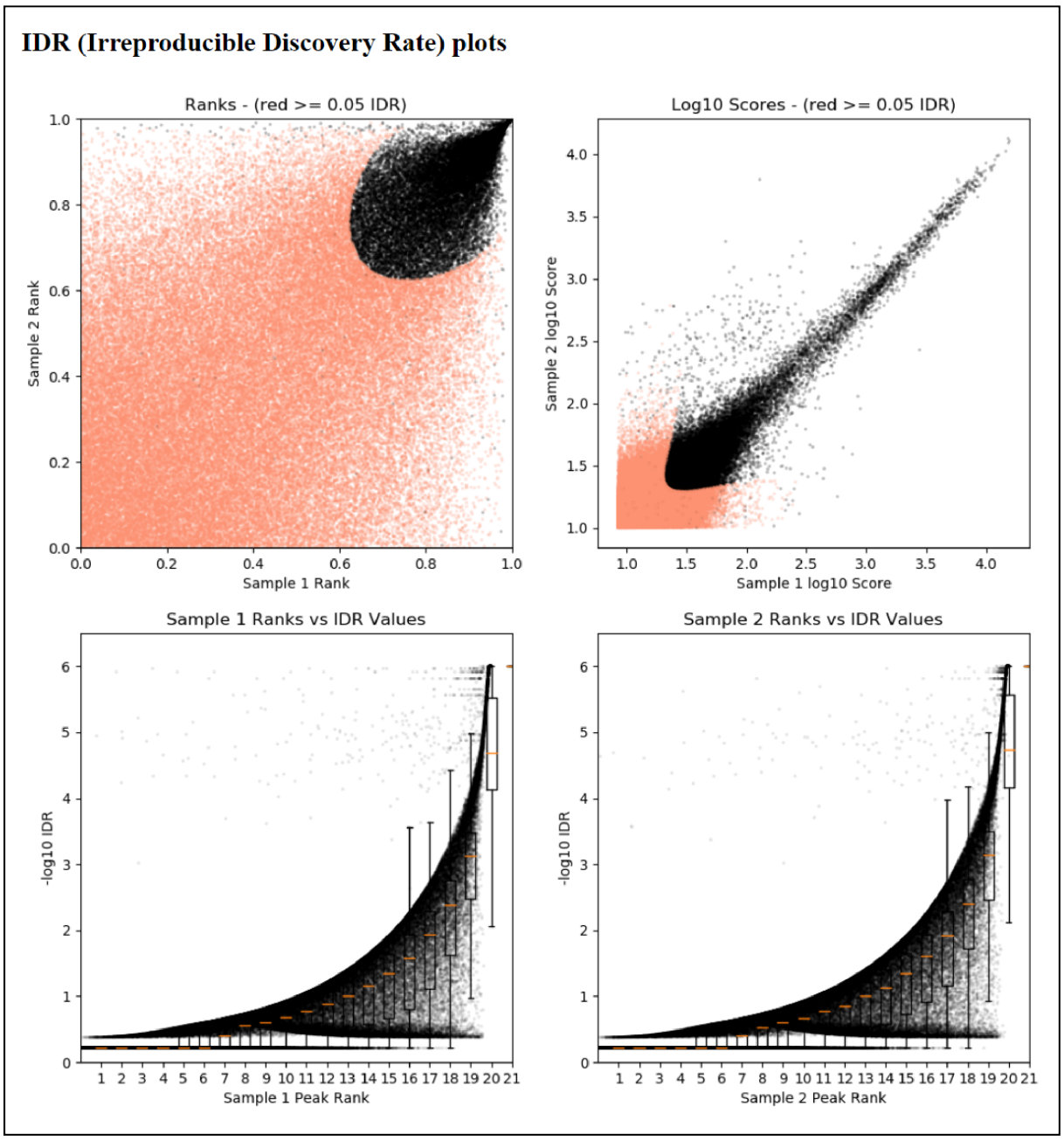


Figure 20: Example of pipeline output summary. IDR reproducibility analysis plots.

Convert possible scientific representations in genome coordinates to integers and set negative coordinates to 0.

```
$ RPEAKFILE_RAW=$OUTDIR/$REP.tagAlign_VS_$CTL.tagAlign.regionPeak.gz
$ RPEAKFILE=$OUTDIR/$REP.regionPeak.gz
$ zcat $RPEAKFILE_RAW | awk 'BEGIN{OFS="\t"}
  {if ($2<0) $2=0; print $1,int($2),int($3),$4,$5,$6,$7,$8,$9,$10;}' | gzip -f -nc > $RPEAKFILE
$ rm -f $RPEAKFILE_RAW
$ mv $OUTDIR/$REP.tagAlign.pdf qc/$REP.relaxed_peak.pdf
```

Filter out raw relaxed peaks in blacklisted region:

```
$ RPEAKFILE_FILT=$OUTDIR/$REP.filt.regionPeak.gz
```

```
$ bedtools intersect -v -a <(zcat -f $RPEAKFILE) -b
  <(zcat -f hg38/hg38.blacklist.bed.gz) | awk 'BEGIN{OFS="\t"}
  {if ($5>1000) $5=1000; print $0}' | grep -P 'chr[\dXY]+[ \t]' | gzip -nc > $RPEAKFILE_FILT
```

Remove temporary files:

```
$ rm -f $rpeakfile_raw
```

Repeat for:

IP replicate 2 (`REP=rep2`; `FRAGLEN=$fraglen_rep2`; `OUTDIR=peak/reps`),
pooled IP replicate (`REP=rep0`; `FRAGLEN=$fraglen_mean`; `OUTDIR=peak/pooled`),
the first pseudo replicate from IP replicate 1 (`REP=rep1.pr1`; `FRAGLEN=$fraglen_rep1`; `OUTDIR=peak/self_pseudo_reps`),
the second pseudo replicate from IP replicate 1 (`REP=rep1.pr2`; `FRAGLEN=$fraglen_rep1`; `OUTDIR=peak/self_pseudo_reps`),
the first pseudo replicate from IP replicate 2 (`REP=rep2.pr1`; `FRAGLEN=$fraglen_rep2`; `OUTDIR=peak/self_pseudo_reps`),
the second pseudo replicate from IP replicate 2 (`REP=rep2.pr2`; `FRAGLEN=$fraglen_rep2`; `OUTDIR=peak/self_pseudo_reps`),
the first pooled pseudo replicate (`REP=rep0.pr1`; `FRAGLEN=$fraglen_mean`; `OUTDIR=peak/pooled_pseudo_reps`),
and the second pooled pseudo replicate (`REP=rep0.pr2`; `FRAGLEN=$fraglen_mean`; `OUTDIR=peak/pooled_pseudo_reps`).

11. Signal track generation for each IP replicate and pooled IP. Perform peak calling with p-value threshold of 0.01 using MACS2 peak caller. Use pooled control for all IP replicate.

```
$ CTL=ctl0
```

Start with IP replicate 1:

```
$ REP=rep1; FRAGLEN=$fraglen_rep1; OUTDIR=signal/reps
```

Generate preliminary signal tracks:

```
$ export LC_COLLATE=C
$ macs2 callpeak -t align/$REP.tagAlign.gz align/$CTL.tagAlign.gz -f BED -n $REP -g hs
  -p 0.01 --nomodel --shift 0 --extsize $FRAGLEN --keep-dup all -B --SPMR
```

Generate fold enrichment signal tracks in `bedGraph` format:

```
$ macs2 bdgcmp -t "$REP"_treat_pileup.bdg -c "$REP"_control_lambda.bdg
                                                    --outdir . -o "$REP"_FE.bdg -m FE
```

Remove illegal coordinates outside the limits of specified chromosome sizes:

```
$ slopBed -i "$REP"_FE.bdg -g hg38/hg38.chrom.sizes -b 0
  | awk '{if ($3 != -1) print $0}'
  | bedClip stdin hg38/hg38.chrom.sizes $REP.fc.bedgraph
```

Convert `bedGraph` files to a `bigWig` format for fold enrichment signal tracks.

```
$ sort -k1,1 -k2,2n $REP.fc.bedgraph > signal/$REP.fc.srt.bedgraph
$ bedGraphToBigWig signal/$REP.fc.srt.bedgraph hg38/hg38.chrom.sizes
  signal/$REP.fc.signal.bigwig
```

Generate $-log_{10}(p\text{-value})$ signal tracks in `bedGraph` format.

```
$ chipReads=$(zcat align/$REP.tagAlign.gz | wc -l | awk '{printf "%f", $1/1000000}')
```

Compute $sval = min$(no. of reads in ChIP, no. of reads in control) / 1,000,000

```
$ controlReads=$(zcat align/$CTL.tagAlign.gz_tag | wc -l
  | awk '{printf "%f", $1/1000000}')
$ sval=$(echo "${chipReads} ${controlReads}"
  | awk '$1>$2{printf "%f",$2} $1<=$2{printf "%f",$1}')"
$ macs2 bdgcmp -t "$REP"_treat_pileup.bdg -c "$REP"_control_lambda.bdg
  --outdir . -o "$REP"_ppois.bdg -m ppois -S "${sval}"
```

Remove illegal coordinates outside the limits of specified chromosome sizes:

```
$ slopBed -i "$REP"_ppois.bdg -g hg38/hg38.chrom.sizes -b 0 | awk '{if ($3 != -1)
  print $0}' | bedClip stdin hg38/hg38.chrom.sizes $REP.pval.bedgraph
```

Convert `bedGraph` to `bigWig` for p-val signal tracks.

```
$ sort -k1,1 -k2,2n $REP.pval.bedgraph > signal/$REP.pval.srt.bedgraph
$ bedGraphToBigWig signal/$REP.pval.srt.bedgraph hg38/hg38.chrom.sizes
  signal/$REP.pval.signal.bigwig
```

Repeat for IP replicate 2 (`REP=rep2`; `FRAGLEN=$fraglen_rep2`; `OUTDIR=signal/reps`) and pooled replicate (`REP=rep0`; `FRAGLEN=$fraglen_mean`; `OUTDIR=signal/pooled_reps`). Remove temporary files.

```
$ rm -f *.bdg *.bedgraph signal/*.bedgraph *.xls *.narrowPeak *.bed
```

12. Perform IDR analysis.

    For each set of peak calls of IP replicate, its pseudo-replicates and pooled pseudo-replicates, perform IDR analysis. For the details of the input and output formats, see Box 4. Start with IP replicate 1.

```
$ PEAK1=peak/reps/rep1.regionPeak.gz; PEAK2=peak/reps/rep2.regionPeak.gz;
  PEAK_POOLED=peak/pooled/rep0.regionPeak.gz; OUTDIR=idr/true_reps; PREFIX=rep1_vs_rep2
$ idr --samples $PEAK1 $PEAK2 --peak-list $PEAK_POOLED --input-file-type narrowPeak
  --output-file $PREFIX.relaxed_peak.18-col.bed --rank signal.value
  --soft-idr-threshold 0.05 --plot --use-best-multisummit-IDR
$ idr_thresh_transformed=$(awk -v p=0.05 'BEGIN{print -log(p)/log(10)}')
$ awk 'BEGIN{OFS="\t"} $12>='"${idr_thresh_transformed}"' {if ($2<0) $2=0;
  print $1,$2,$3,$4,$5,$6,$7,$8,$9,$10}' $PREFIX.relaxed_peak.18-col.bed
  | sort | uniq | sort -k7n,7n | gzip -nc > $OUTDIR/$PREFIX.narrowPeak.gz
```

    Filter out peaks in blacklisted regions:

```
$ bedtools intersect -v -a <(zcat -f $OUTDIR/$PREFIX.narrowPeak.gz) -b <(zcat -f
  hg38/hg38.blacklist.bed.gz) | grep -P 'chr[\dXY]+[ \t]'
  | awk 'BEGIN{OFS="\t"} {if ($5>1000) $5=1000; print $0}'
  | gzip -nc > $OUTDIR/$PREFIX.filt.narrowPeak.gz
```

    Rename plot file names. IDR plots include the number of significant peaks and IDR values (Figure 20; see Boxes 4 and 5 for the interpretation of each panel).

```
$ mv $PREFIX.relaxed_peak.18-col.bed.noalternatesummitpeaks.png qc/IDR.$PREFIX.idr_plot.png
```

Repeat for pseudo-replicates of IP replicate 1
(`PEAK1=peak/self_pseudo_reps/rep1.pr1.regionPeak.gz;`
`PEAK2=peak/self_pseudo_reps/rep1.pr2.regionPeak.gz;`
`PEAK_POOLED=peak/reps/rep1.regionPeak.gz;`
`OUTDIR=idr/self_pseudo_reps; PREFIX=rep1_pr`),
pseudo-replicates of IP replicate 2
(`PEAK1=peak/self_pseudo_reps/rep2.pr1.regionPeak.gz;`
`PEAK2=peak/self_pseudo_reps/rep2.pr2.regionPeak.gz;`
`PEAK_POOLED=peak/reps/rep2.regionPeak.gz; OUTDIR=idr/self_pseudo_reps; PREFIX=rep2_pr`),
and pooled pseudo-replicates
(`PEAK1=peak/pooled_pseudo_reps/rep0.pr1.regionPeak.gz;`
`PEAK2=peak/pooled_pseudo_reps/rep0.pr2.regionPeak.gz;`
`PEAK_POOLED=peak/pooled/rep0.regionPeak.gz;`
`OUTDIR=idr/pooled_pseudo_reps; PREFIX=ppr`).

Remove temporary files:

```
$ rm -f *.bed
```

13. Calculate thresholds to truncate peak lists. We use IDR thresholds of 0.05 for the original replicates threshold, self-consistency thresholds and pooled-pseudoreplicate threshold. For the recommended IDR thresholds, see Box 6.

    We take the maximum value of N_t and N_p (see Figure 1).

    For the original (true) replicates:

    ```
    $ Nt=$(zcat idr/true_reps/rep1_vs_rep2.filt.narrowPeak.gz | wc -l)
    ```

    For the pooled-pseudoreplicates:

    ```
    $ Np=$(zcat idr/pooled_pseudo_reps/ppr.filt.narrowPeak.gz | wc -l)
    ```

    We also calculate the N_1 and N_2 values for individual (self) pseudoreplicates:

    ```
    $ N1=$(zcat idr/self_pseudo_reps/rep1_pr.filt.narrowPeak.gz | wc -l)
    $ N2=$(zcat idr/self_pseudo_reps/rep2_pr.filt.narrowPeak.gz | wc -l)
    ```

    These two values are the self-consistency thresholds (see Figure 1) and should be in a similar range. We use N_1 and N_2 to flag replicates with low consistency.

14. Flag replicates for low consistency. Typically, the self-consistency thresholds for all the original replicates should be reasonably similar (within a factor of 2). N_t (comparing original replicates) and N_p (comparing pooled pseudoreplicates) should be similar (within a factor of 2). We typically flag datasets that fail these criteria. We also flag datasets that fail either one of these criteria by a significant margin (i.e if the above ratios are $\gg 2$) as exhibiting severe replicate inconsistency.

    CRITICAL STEP: Other measures, such as the strand cross-correlation based quality metrics, sequencing depth differences and library complexity metrics for the replicates, should also be checked to see if they can provide further insights into the origins of the discrepancy.

15. Obtain the final set of peak calls. Generate a conservative and an optimal final set of peak calls for the dataset.

    For the optimal set, if $N_t > N_p$ choose IDR peaks (`idr/true_reps/rep1_vs_rep2.filt.narrowPeak.gz`) from true replicate. Otherwise choose IDR peaks (`idr/pooled_pseudo_reps/ppr.filt.narrowPeak.gz`) from pooled pseudo replicates.

    For the conservative set, always choose IDR peaks (`idr/true_reps/rep1_vs_rep2.filt.narrowPeak.gz`) from true replicate.

16. Calculate the fraction of reads in peaks (FRiP). For each replicate `tagAlign` and pooled `tagAlign`, the fraction of reads that fall within the N_p and N_t peak sets is calculated. Start with self pseudo replicates of IP replicate 1:

```
$ REP=rep1; FRAGLEN=$fraglen_rep1; PEAK=idr/self_pseudo_reps/rep1_pr.filt.narrowPeak.gz;
  FRiP=qc/IDR.rep1_pr.FRiP.qc
$ HALF_FRAGLEN=$(( (FRAGLEN+1)/2 ))
$ val1=$(bedtools slop -i align/$REP.tagAlign.gz -g hg38/hg38.chrom.sizes -s
  -l -$HALF_FRAGLEN -r $HALF_FRAGLEN | awk '{if ($2>=0 && $3>=0 && $2<=$3) print $0}'
  | bedtools intersect -a stdin -b <(zcat -f $PEAK) -wa -u | wc -l)
$ val2=$(zcat align/$REP.tagAlign.gz | wc -l)
$ awk 'BEGIN {print '${val1}'/'${val2}'}' > $FRiP
```

Repeat for self pseudo replicates of IP replicate 2
(`REP=rep2; FRAGLEN=$fraglen_rep2;`
`PEAK=idr/self_pseudo_reps/rep2_pr.filt.narrowPeak.gz;`
`FRiP=qc/IDR.rep2_pr.FRiP.qc`), pooled pseudo replicate
(`REP=rep0`; `FRAGLEN=$fraglen_mean;`
`PEAK=idr/pooled_pseudo_reps/ppr.filt.narrowPeak.gz;`
`FRiP=qc/IDR.ppr.FRiP.qc`),
and true IP replicates
(`REP=rep0`; `FRAGLEN=$fraglen_mean;`
`PEAK=idr/true_reps/rep1_vs_rep2.filt.narrowPeak.gz;`
`FRiP=qc/IDR.rep1_vs_rep2.FRiP.qc`).

FRiP and IDR quality metrics and plots for IDR analysis are generated as shown in Figures 18, 19 and 20.

Troubleshooting

The pipeline as described here is intended to work as an "out the box" solution and is expected to run smoothly beyond the particular situations discussed throughout the Procedure and Anticipated Results section, except in a variety of unforeseeable circumstances. Users experiencing such issues can seek support at the GitHub page of the ENCODE pipelines (`https://github.com/ENCODE-DCC/chip-seq-pipeline2`)

Acknowledgments

This work was supported by NIH grants NIH 1U01HG007037, NIH R01GM109453, NIH U01 HG007019, NIH U41HG006992, NIH U41HG007000, NIH U54HG006996, and NIH U54HG006998.

Author contributions

A.K. specified the protocol. A.K., J.W.L., G.K.M., Y.L.J., Q.L. performed analysis of case studies. A.K., P.J.P. supervised the analysis of case studies. J.W.L. developed the code implementing the standalone and DNAnexus pipelines. J.W.L., J.S.T., C.A.S., B.C.H., E.T.C., J.A.H., J.M.C. deployed the pipeline on DNAnexus and used it to process all ENCODE data at the ENCODE portal. A.K., P.V.K., Y.L.J. and P.J.P. developed the cross-correlation measures. Q.L., N.B., J.B.B, P.J.B. developed the IDR framework. N.B. provided an efficient Python implementation of the IDR framework. A.K., Q.L., N.B., P.J.B. specified the IDR protocol for the ChIP-seq pipeline. Y.G., A.H., D.C., T.L., R.E.T., N.S., R.B., J.R., V.J., M.G., D.G., S.K. provided analyses for peak caller comparisons. F.PB., T.K., M.P.S., R.M.M., P.J.F. provided datasets for case studies. B.J.W, P.J.B., A.S., S.B. contributed to the discussion during early protocol development. G.K.M., J.W.L., Y.L.J., Q.L., J.S.T., P.J.P., A.K. wrote the manuscript. All authors provided feedback and approved the manuscript.

Competing Financial Interests

A.S. and S.B. are equity holders and consultants for DNAnexus. S.B. is Vice President of Applied and Computational Biology at Illumina. R.M.M. is an equity holder and former Scientific Advisory Board member of DNAnexus. A.K. was a former Scientific Advisory Board member of DNAnexus. All other authors declare no competing financial interests.

References

1. Ren, B. et al. Genome-wide location and function of DNA binding proteins. *Science* **290**, 2306–2309 (2000).
2. Barski, A. et al. High-resolution profiling of histone methylations in the human genome. *Cell* **129**, 823–837 (2007).
3. Marinov, G. K., Kundaje, A., Park, P. J. & Wold, B. J. Large-scale quality analysis of published ChIP-seq data. *G3 (Bethesda)* **4**, 209–223 (2014).
4. Dréos R, et al. MGA repository: a curated data resource for ChIP-seq and other genome annotated data. *Nucleic Acids Res.* **46**, D175–D180.
5. ENCODE Project Consortium. An integrated encyclopedia of DNA elements in the human genome. *Nature* **489**, 57–74 (2012).

6. modENCODE Consortium et al. Identification of functional elements and regulatory circuits by *Drosophila* modENCODE. *Science* **330**, 1787–1797 (2010).
7. Gerstein, M. B. et al. Integrative analysis of the *Caenorhabditis elegans* genome by the modENCODE project. *Science* **330**, 1775–1787 (2010).
8. Roadmap Epigenomics Consortium et al. Integrative analysis of 111 reference human epigenomes. *Nature* **518**, 317–330 (2015)
9. Park, P. J. & Park, P. J. ChIPâ€“seq: advantages and challenges of a maturing technology. *Nat Rev Genet* **10**, 669–680 (2009).
10. Pepke, S., Wold, B. & Mortazavi, A. Computation for ChIP-seq and RNA-seq studies. *Nat. Meth.* **6**, S22–32 (2009).
11. Landt, S. G. et al. ChIP-seq guidelines and practices of the ENCODE and modENCODE consortia. *Genome Res.* **22**, 1813–1831 (2012).
12. Jung, Y. L. et al. Impact of sequencing depth in ChIP-seq experiments. *Nucleic Acids Res.* **42**, e74 (2014).
13. Tyner C, et al. The UCSC Genome Browser database: 2017 update. *Nucleic Acids Res.* **45**, D626–D634. (2017)
14. Zhou, X, et al. The Human Epigenome Browser at Washington University. *Nat. Methods* **8**(12), 989–990. (2011)
15. Thorvaldsdottir, H., Robinson, J. T. & Mesirov, J. P. Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Brief. Bioinform.* **14**, 178–192 (2013).
16. Zhang, Y. et al. Model-based analysis of ChIP-Seq (MACS). *Genome Biol.*. **9**, R137 (2008).
17. Kharchenko, P. V., Tolstorukov, M. Y. & Park, P. J. Design and analysis of ChIP-seq experiments for DNA-binding proteins. *Nat. Biotechnol.* **26**, 1351–1359 (2008).
18. Gerstein, M. B. et al. Architecture of the human regulatory network derived from ENCODE data. *Nature* **489**, 91–100 (2012).
19. Li, Q., Brown, J. B., Huang, H. & Bickel, P. J. Measuring reproducibility of high-throughput experiments. *Ann. Appl. Stat.* **5**, 1752–1779 (2011).
20. Kasowski, M. et al. Extensive Variation in Chromatin States Across Humans. *Science* **342**, 750–752 (2013).
21. Motamedi, F. J. et al. WT1 controls antagonistic FGF and BMP-pSMAD pathways in early renal progenitors. *Nat. Commun.* **5**, 4444 (2014).
22. Crawford, G. E. et al. Genome-wide mapping of DNase hypersensitive sites using massively parallel signature sequencing (MPSS). *Genome Res.* **16**, 123–131 (2006).
23. Giresi, P. G., Kim, J., McDaniell, R. M., Iyer, V. R. & Lieb, J. D. FAIRE (Formaldehyde-Assisted Isolation of Regulatory Elements) isolates active regulatory elements from human chromatin. *Genome Res.* **17**, 877–885 (2007).
24. Buenrostro, J. D., Giresi, P. G., Zaba, L. C., Chang, H. Y. & Greenleaf, W. J. Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. *Nat. Meth.* **10**, 1213–1218 (2013).
25. Patel, R. K. & Jain, M. NGS QC Toolkit: a toolkit for quality control of next generation sequencing data. *PLoS ONE* **7**, e30619 (2012).
26. Daley, T. & Smith, A. D. Predicting the molecular complexity of sequencing libraries. *Nat. Meth.* **10**, 325–327 (2013).
27. Yang, X. et al. HTQC: a fast quality control toolkit for Illumina sequencing data. *BMC Bioinform.* **14**, 33 (2013).
28. Diaz, A., Nellore, A. & Song, J. S. CHANCE: comprehensive software for quality control and validation of ChIP-seq data. *Genome Biol.* **13**, R98 (2012).
29. Planet, E., Attolini, C. S.-O., Reina, O., Flores, O. & Rossell, D. htSeqTools: high-throughput sequencing quality control, processing and visualization in R. *Bioinformatics* **28**, 589–590 (2012).
30. Mendoza-Parra, M.-A., Van Gool, W., Mohamed Saleem, M. A., Ceschin, D. G. & Gronemeyer, H. A quality control system for profiles obtained by ChIP sequencing. *Nucleic Acids Res.* **41**, e196 (2013).
31. Bardet, A. F., He, Q., Zeitlinger, J. & Stark, A. A computational pipeline for comparative ChIP-seq analyses. *Nat. Protoc.* **7**, 45–61 (2012).
32. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25**, 1754–1760 (2009).
33. Langmead, B., Trapnell, C., Pop, M. & Salzberg, S. L. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.* **10**, R25 (2009).
34. Langmead, B. & Salzberg, S.L. Fast gapped-read alignment with Bowtie 2. *Nat. Methods* **9**, 357–359 (2012)
35. Sasaki, S. et al. Chromatin-associated periodicity in genetic variation downstream of transcriptional start sites. *Science* **323**, 401–404 (2009).
36. Diaz, A., et al. Normalization, bias correction, and peak calling for ChIP-seq. *Stat Appl Genet Mol Biol* **11**, Article 9 (2012).
37. Ramírez F, Ryan DP, Grüning B, Bhardwaj V, Kilpert F, Richter AS, Heyne S, Dündar F, Manke T. 2016. deepTools2: a next generation web server for deep-sequencing data analysis. *Nucleic Acids Res* **44**(W1):W160–165.
38. Bailey, T. et al. Practical guidelines for the comprehensive analysis of ChIP-seq data. *PLoS Comput. Biol.* **9**, e1003326 (2013).
39. Schmidl, C., et al. ChIPmentation: fast, robust, low-input ChIP-seq for histones and transcription factors. *Nat Methods* **12**, 963–965 (2015).

40. Zarnegar, M.A., et al. Targeted chromatin ligation, a robust epigenetic profiling technique for small cell numbers. *Nucleic Acids Res* **45¡** e153 (2017).

41. Rhee, H.S., & B.F. Pugh. Comprehensive genome-wide protein-DNA interactions detected at single-nucleotide resolution. *Cell* **147**, 1408–1419 (2011).

42. He, Q., et al. ChIP-nexus enables improved detection of in vivo transcription factor binding footprints. *Nat Biotechnol* **33**, 395–401 (2015)

43. Zentner, G.E., et al. ChEC-seq kinetics discriminates transcription factor binding sites by DNA sequence and shape in vivo. *Nat Commun* **6**, 8733 (2015).

44. Skene, P.J., & S. Henikoff. An efficient targeted nuclease strategy for high-resolution mapping of DNA binding sites. *Elife* **6**, pii: e21856 (2017).

Supplementary Materials

Supplementary Methods

Commands used to run GPS/GEM

For optimal peaks (default threshold):

```
java -Xmx15G -jar gem.jar --g hg19.info --d Read_Distribution_default.txt --s 2400000000
     --expt wgEncodeSydhTfbsGm12878Ctcfsc15914c20StdAlnRep1.bam
     --expt wgEncodeSydhTfbsGm12878Ctcfsc15914c20StdAlnRep2.bam
     --ctrl wgEncodeSydhTfbsGm12878InputIggrabAlnRep1.bam
     --f SAM --out CTCF_GM12878 --genome /yourpath/hg19 --k_min 6 --k_max 13 --outNP
```

(2) For relaxed peaks (for use with IDR)

```
java -Xmx15G -jar gem.jar --g hg19.info --d Read_Distribution_default.txt --s 2400000000
     --expt wgEncodeSydhTfbsGm12878Ctcfsc15914c20StdAlnRep1.bam
     --expt wgEncodeSydhTfbsGm12878Ctcfsc15914c20StdAlnRep2.bam
     --ctrl wgEncodeSydhTfbsGm12878InputIggrabAlnRep1.bam --f SAM --out CTCF_GM12878
     --genome /yourpath/hg19 --k_min 6 --k_max 13 --outNP --q 0
```

Detailed description of options:

- `--g [path]`, the path to a genome information file. The file contains tab-delimited chromosome name/length pairs.
- `--d [path]`, the path to the read distribution model file.
- `--s [n]`, the size of uniquely mappable genome. It depends on the genome and the read length.
- `--exptX [path]`, the path to the aligned reads file for experiment (IP). `X` is condition name (can be blank).
- `--ctrlX [path]`, the path to the aligned reads file for control. `X` should match the condition name in `--expt`.
- `--f [BED`SAM—BOWTIE]—, read file format: BED/SAM/BOWTIE. The SAM option allows SAM or BAM file input (default = BED).
- `--t [n]`, number of threads (default: number of CPUs).
- `--ex [region file]`, file that contains subset of genome regions to be excluded. Each line contains a region in `"chr:start-end"` format.
- `--out [name]`, output file base name.
- `--genome [path]`, the path to the genome sequence directory, which contains fasta files by chromosomes.
- `--q [value]`, significance level for q-value, specified as -log10(q-value).
- `--k [n]`, the length of k-mer in motif discovery
- `--k_min [n]`, `--k_max [n]`, minimum and maximum values of k
- `--outNP`, output narrowPeak files (default is NO narrowPeak file output)

Commands used to run PeakSeq

The ChIP and Input reads (`chip.bam` and `input.bam`) should be preprocessed before running:

```
mkdir chip

mkdir input

samtools view chip.bam | PeakSeq -preprocess SAM stdin ./chip

samtools view input.bam | PeakSeq -preprocess SAM stdin ./input
```

Then it is necessary to setup the configuration file (`config.dat`). An example configuration file is included with the PeakSeq download. An example configuration file:

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Experiment_id GABP-HELAS3_rep1

Enrichment_mapped_fragment_length 200

target_FDR 0.05

N_Simulations 10

Minimum_interpeak_distance 200

Mappability_map_file Mapability_HG.txt

ChIP_Seq_reads_data_dirs ../wgEncodeHaibTfbsHelas3GabpPcr1xAlnRep1.bam.unique.tagAlign_reads

Input_reads_data_dirs ../wgEncodeHaibTfbsHelas3RxlchPcr1xAlnRep1.bam.unique.tagAlign_reads

max_Qvalue 0.05

Background_model Simulated

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Finally, the peaks are called using the configuration file:

```
PeakSeq -peak_select config.dat
```

For optimal peaks and relaxed peaks, use q-value threshold of 0.05 and 0.10 respectively. This can be set in the configuration file.

Commands used to run MACS2

Optimal peaks calling. Keep almost everything by default:

```
macs2 callpeak --call-summits -n NAME -t TREATMENT_FILENAME -c INPUT_FILENAME
```

Relaxed peaks calling. Keep almost everything by default, but relax the cutoff to use `-p 0.1`.

```
macs2 callpeak --call-summits -n NAME -t TREATMENT_FILENAME -c INPUT_FILENAME -p 0.1
```

Peaks calling for paired-end reads. Keep almost everything by default, but specify format. Note, the paired-end alignments should be stored in BAM format files.

```
macs2 callpeak --call-summits -n NAME -t TREATMENT_FILENAME -c INPUT_FILENAME -f BAMPE
```

Commands used to run MOSAICS

MOSAiCS is a platform-independent R package and available in Bioconductor. To install the MOSAiCS R package, start R and enter:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("mosaics")
```

R ($\geq$ 2.11.1) and the following R packages are required: *MASS*, *splines*, *lattice*, *IRanges*, *Rcpp*. Parallel computing is supported when parallel package is available.

The following steps are common for both optimal and relaxed peaks.

```
> constructBins( infile = "SNYDER_HG19_GM12878_CTCF_Rep0.tagAlign", fileFormat = "bed", PET = FALSE )
> constructBins( infile = "SNYDER_HG19_GM12878_INPUT_Rep0.tagAlign", fileFormat = "bed", PET = FALSE )
> bin <- readBins( type = c("chip","input"),
fileName = c( "SNYDER_HG19_GM12878_CTCF_Rep0.tagAlign_fragL200_bin200.txt",
"SNYDER_HG19_GM12878_INPUT_Rep0.tagAlign_fragL200_bin200.txt" ) )
> fit <- mosaicsFit( bin, bgEst="rMOM" )
```

Optimal peaks (default threshold)

```
> if ( fit@bic2S <= fit@bic1S ) {
peak <- mosaicsPeak( fit, signalModel = "2S", FDR = 1e-20, thres = quantile( bin@tagCount, 0.90 ))
} else {
peak <- mosaicsPeak( fit, signalModel = "1S", FDR = 1e-20, thres = quantile( bin@tagCount, 0.90 ))
}
> export( peak, type = "narrowPeak", filename = "MOSAiCS_GM12878_CTCF_Rep0.narrowPeak" )
```

Relaxed peaks (for use with IDR)

```
> if ( fit@bic2S <= fit@bic1S ) {
peak <- mosaicsPeak( fit, signalModel = "2S", FDR = 0.50, thres = quantile( bin@tagCount, 0.96 ))
} else {
peak <- mosaicsPeak( fit, signalModel = "1S", FDR = 0.50, thres = quantile( bin@tagCount, 0.96 ))
}
> export( peak, type = "narrowPeak", filename = "MOSAiCS_GM12878_CTCF_Rep0.narrowPeak" )
```

Commands used to run DPeak

dPeak is a platform-independent R package. Download the source code package from the dPeak website and type:

```
$ R CMD INSTALL dpeak_0.9.9.tar.gz
```

R ($\geq$2.11.1) and the following R packages are required: *MASS*, *IRanges*, *Rcpp*. Parallel computing is supported when multicore package is available.

dPeak takes MOSAiCS peak lists as an input and MOSAiCS determines whether to generate optimal or relaxed peaks for dPeak. Optimal (relaxed) peaks for dPeak can be generated as follows, when `MOSAiCS_GM12878_CTCF_Rep0.narrowPeak` is assumed to be the file for optimal (relaxed) peaks generated by MOSAiCS:

```
> data <- dpeakRead( peakfile = "MOSAiCS_GM12878_CTCF_Rep0.narrowPeak",
readfile = "SNYDER_HG19_GM12878_CTCF_Rep0.tagAlign", fileFormat="bed",
PET = FALSE, parallel = TRUE, nCore = 8 )
> fit <- dpeakFit( data, nCore = 8 )
> export( fit, type = "narrowPeak", filename = "dPeak_GM12878_CTCF_Rep0.narrowPeak" )
```