ChIP-Seq workflow template: Some Descriptive Title

Project ID: ChIPseq_Pi_Name_Organism_Jun2015
Project PI: First Last (first.last@inst.edu)
Author of Report: First Last (first.last@inst.edu)

October 21, 2016

Contents

1 Introduction .................................................. 2
  1.1 Background and objectives ................................. 2
  1.2 Experimental design ....................................... 2

2 Load workflow environment ..................................... 2
  2.1 Load packages and sample data ............................ 2
  2.2 Experiment definition provided by targets file ............ 2

3 Read preprocessing .............................................. 3
  3.1 Read quality filtering and trimming ....................... 3
  3.2 FASTQ quality report ..................................... 3

4 Alignments ....................................................... 3
  4.1 Read mapping with Bowtie2 ............................... 4
  4.2 Read and alignment stats .................................. 4
  4.3 Create symbolic links for viewing BAM files in IGV .... 4

5 Peak calling with MACS2 ....................................... 4
  5.1 Merge BAM files of replicates prior to peak calling .... 4
  5.2 Peak calling without input/reference sample ............ 5
  5.3 Peak calling with input/reference sample ................ 5

6 Annotate peaks with genomic context .......................... 5
  6.1 Annotation with ChIPpeakAnno package ................... 5
  6.2 Annotation with ChIPseeker package ....................... 6

7 Count reads overlapping peak regions ........................ 6

8 Differential binding analysis of peaks ........................ 6

9 GO term enrichment analysis .................................. 6

10 Motif analysis .................................................. 7
  10.1 Parse DNA sequences of peak regions from genome ...... 7
  10.2 Motif discovery with BCRANK ............................. 7

11 Version Information .......................................... 8

12 Funding ......................................................... 9
1 Introduction

1.1 Background and objectives

This report describes the analysis of several ChIP-Seq experiments studying the DNA binding patterns of the transcription factors ... from organism ....

1.2 Experimental design

Typically, users want to specify here all information relevant for the analysis of their NGS study. This includes detailed descriptions of FASTQ files, experimental design, reference genome, gene annotations, etc.

2 Load workflow environment

2.1 Load packages and sample data

The systemPipeR package needs to be loaded to perform the analysis steps shown in this report (Girke, 2014).

library(systemPipeR)

Load workflow environment with sample data into your current working directory. The sample data are described here.

library(systemPipeRdata)
genWorkenvir(workflow="chipseq")
setwd("chipseq")

In the workflow environments generated by genWorkenvir all data inputs are stored in a data/ directory and all analysis results will be written to a separate results/ directory, while the systemPipeChIPseq.Rnw script and the targets file are expected to be located in the parent directory. The R session is expected to run from this parent directory. Additional parameter files are stored under param/.

To work with real data, users want to organize their own data similarly and substitute all test data for their own data. To rerun an established workflow on new data, the initial targets file along with the corresponding FASTQ files are usually the only inputs the user needs to provide.

If applicable users can load custom functions not provided by systemPipeR. Skip this step if this is not the case.

source("systemPipeChIPseq_Fct.R")

2.2 Experiment definition provided by targets file

The targets file defines all FASTQ files and sample comparisons of the analysis workflow.

targetspath <- system.file("extdata", "targets_chip.txt", package="systemPipeR")
targets <- read.delim(targetspath, comment.char = ")"
targets[1:4,-c(5,6)]

<table>
<thead>
<tr>
<th>FileName</th>
<th>SampleName</th>
<th>Factor</th>
<th>SampleLong</th>
<th>SampleReference</th>
</tr>
</thead>
<tbody>
<tr>
<td>./data/SRR446027_1.fastq</td>
<td>M1A</td>
<td>M1</td>
<td>Mock.1h.A</td>
<td></td>
</tr>
<tr>
<td>./data/SRR446028_1.fastq</td>
<td>M1B</td>
<td>M1</td>
<td>Mock.1h.B</td>
<td></td>
</tr>
</tbody>
</table>
3 Read preprocessing

3.1 Read quality filtering and trimming

The following example shows how one can design a custom read preprocessing function using utilities provided by the ShortRead package, and then apply it with preprocessReads in batch mode to all FASTQ samples referenced in the corresponding SYSargs instance (args object below). More detailed information on read preprocessing is provided in systemPipeR’s main vignette.

```r
args <- systemArgs(sysma="param/trim.param", mytargets="targets_chip.txt")
filterFct <- function(fq, cutoff=20, Nexceptions=0) {
  qcount <- rowSums(as(quality(fq), "matrix") <= cutoff)
  fq[qcount <= Nexceptions] # Retains reads where Phred scores are >= cutoff with N exceptions
}
preprocessReads(args=args, Fct="filterFct(fq, cutoff=20, Nexceptions=0)", batchsize=100000)
writeTargetsOut(x=args, file="targets_chip_trim.txt", overwrite=TRUE)
```

3.2 FASTQ quality report

The following seeFastq and seeFastqPlot functions generate and plot a series of useful quality statistics for a set of FASTQ files including per cycle quality box plots, base proportions, base-level quality trends, relative k-mer diversity, length and occurrence distribution of reads, number of reads above quality cutoffs and mean quality distribution. The results are written to a PDF file named fastqReport.pdf.

```r
args <- systemArgs(sysma="param/bowtieSE.param", mytargets="targets_chip_trim.txt")
fqlist <- seeFastq(fastq=infile1(args), batchsize=100000, klength=8)
pdf("./results/fastqReport.pdf", height=18, width=4*length(fqlist))
seeFastqPlot(fqlist)
dev.off()
```

Figure 1: QC report for 18 FASTQ files.

4 Alignments
4.1 Read mapping with Bowtie2

The NGS reads of this project will be aligned with Bowtie2 against the reference genome sequence (Langmead and Salzberg, 2012). The parameter settings of the aligner are defined in the bowtieSE.param file. In ChIP-Seq experiments it is usually more appropriate to eliminate reads mapping to multiple locations. To achieve this, users want to remove the argument setting '-k 50 --non-deterministic' in the bowtieSE.param file.

```r
args <- systemArgs(sysma="param/bowtieSE.param", mytargets="targets_chip_trim.txt")
sysargs(args)[1] # Command-line parameters for first FASTQ file
moduleload(modules(args)) # Skip if a module system is not used
system("bowtie2-build ./data/tair10.fasta ./data/tair10.fasta") # Indexes reference genome
runCommandline(args)
writeTargetsout(x=args, file="targets_bam.txt", overwrite=TRUE)
```

Check whether all BAM files have been created

```r
file.exists(outpaths(args))
```

4.2 Read and alignment stats

The following provides an overview of the number of reads in each sample and how many of them aligned to the reference.

```r
read_statsDF <- alignStats(args=args)
write.table(read_statsDF, "results/alignStats.xls", row.names=FALSE, quote=FALSE, sep="\t")
read.delim("results/alignStats.xls")
```

4.3 Create symbolic links for viewing BAM files in IGV

The symLink2bam function creates symbolic links to view the BAM alignment files in a genome browser such as IGV without moving these large files to a local system. The corresponding URLs are written to a file with a path specified under urlfile, here IGVurl.txt.

```r
symLink2bam(sysargs=args, htmldir=c("~/html/", "somedir/"),
            urlbase="http://biocluster.ucr.edu/~tgirke/",
            urlfile="./results/IGVurl.txt")
```

5 Peak calling with MACS2

5.1 Merge BAM files of replicates prior to peak calling

Merging BAM files of technical and/or biological replicates can improve the sensitivity of the peak calling by increasing the depth of read coverage. The mergeBamByFactor function merges BAM files based on grouping information specified by a factor, here the Factor column of the imported targets file. It also returns an updated SYSargs object containing the paths to the merged BAM files as well as to any unmerged files without replicates. This step can be skipped if merging of BAM files is not desired.

```r
args <- systemArgs(sysma=NULL, mytargets="targets_bam.txt")
args_merge <- mergeBamByFactor(args, overwrite=TRUE)
writeTargetsout(x=args_merge, file="targets_mergeBamByFactor.txt", overwrite=TRUE)
```
5.2 Peak calling without input/reference sample

MACS2 can perform peak calling on ChIP-Seq data with and without input samples (Zhang et al., 2008). The following performs peak calling without input on all samples specified in the corresponding `args` object. Note, due to the small size of the sample data, MACS2 needs to be run here with the `--nomodel` setting. For real data sets, users want to remove this parameter in the corresponding `*.param` file(s).

```r
args <- systemArgs(sysma="param/macs2_noinput.param", mytargets="targets_mergeBamByFactor.txt")
sysargs(args)[1]  # Command-line parameters for first FASTQ file
runcmdline(args)
file.exists(outpaths(args))
writeTargetsout(x=args, file="targets_macs.txt", overwrite=TRUE)
```

5.3 Peak calling with input/reference sample

To perform peak calling with input samples, they can be most conveniently specified in the `SampleReference` column of the initial targets file. The function `writeTargetsRef` uses this information to create a targets file intermediate for running MACS2 with the corresponding input samples.

```r
writeTargetsRef(infile="targets_mergeBamByFactor.txt", outfile="targets_bam_ref.txt", silent=FALSE, overwrite=TRUE)
args <- systemArgs(sysma="param/macs2.param", mytargets="targets_bam_ref.txt")
sysargs(args)[1]  # Command-line parameters for first FASTQ file
runcmdline(args)
file.exists(outpaths(args))
writeTargetsout(x=args, file="targets_macs.txt", overwrite=TRUE)
```

The peak calling results from MACS2 are written for each sample to separate files in the `results` directory. They are named after the corresponding `*.bam` files with extensions used by MACS2.

6 Annotate peaks with genomic context

6.1 Annotation with ChIPpeakAnno package

The following annotates the identified peaks with genomic context information using the `ChIPpeakAnno` and `ChIPseeker` packages, respectively (Zhu et al., 2010; Yu et al., 2015).

```r
library(ChIPpeakAnno); library(GenomicFeatures)
args <- systemArgs(sysma="param/annotate_peaks.param", mytargets="targets_macs.txt")
txdb <- loadDb("./data/tair10.sqlite")
ge <- genes(txdb, columns=c("tx_name", "gene_id", "tx_type"))
for(i in seq(along=args)) {
  peaksGR <- as(read.delim(infile1(args)[i], comment="#"), "GRanges")
  annotatedPeak <- annotatePeakInBatch(peaksGR, AnnotationData=genes(txdb))
  df <- data.frame(as.data.frame(annotatedPeak), as.data.frame(values(ge[values(annotatedPeak)$feature,])))
  write.table(df, outpaths(args[i]), quote=FALSE, row.names=FALSE, sep="\t")
}
writeTargetsout(x=args, file="targets_peakanno.txt", overwrite=TRUE)
```

The peak annotation results are written for each peak set to separate files in the `results` directory. They are named after the corresponding peak files with extensions specified in the `annotate_peaks.param` file, here `*.peaks.annotated.xls`.

---

**systemPipeR ChIP-Seq Workflow**

6. Annotate peaks with genomic context
6.2 Annotation with ChIPseeker package

Same as in previous step but using the ChIPseeker package for annotating the peaks.

```r
library(ChIPseeker)

library(ChIPseeker)

txdb <- loadDb("./data/tair10.sqlite")

for(i in seq_along(args)) {
  peakAnno <- annotatePeak(infile1(args)[i], TxDb=txdb, verbose=FALSE)
  df <- as.data.frame(peakAnno)
  write.table(df, outpaths(args[i]), quote=FALSE, row.names=FALSE, sep="\t")
}

writeTargetsOut(x=args, file="targets_peakanno.txt", overwrite=TRUE)
```

Summary plots provided by the ChIPseeker package. Here applied only to one sample for demonstration purposes.

```r
peak <- readPeakFile(infile1(args)[1])

covplot(peak, weightCol="X.log10.pvalue.")

peakHeatmap(outpaths(args)[1], TxDb=txdb, upstream=1000, downstream=1000, color="red")

plotAvgProf2(outpaths(args)[1], TxDb=txdb, upstream=1000, downstream=1000, xlab="Genomic Region (5'->3')")
```

7 Count reads overlapping peak regions

The countRangeset function is a convenience wrapper to perform read counting iteratively over several range sets, here peak range sets. Internally, the read counting is performed with the summarizeOverlaps function from the GenomicAlignments package. The resulting count tables are directly saved to files, one for each peak set.

```r
library(GenomicRanges)

args <- systemArgs(sysma="param/count_rangesets.param", mytargets="targets_macs.txt")

args_bam <- systemArgs(sysma=NULL, mytargets="targets_bam.txt")

bfl <- BamFileList(outpaths(args_bam), yieldSize=50000, index=character())

countDFnames <- countRangeset(bfl, args, mode="Union", ignore.strand=TRUE)

writeTargetsOut(x=args, file="targets_countDF.txt", overwrite=TRUE)
```

8 Differential binding analysis of peaks

The function runDiff performs differential binding analysis in batch mode for several count tables using edgeR or DESeq2 (Robinson et al., 2010; Love et al., 2014). Internally, it calls the functions run_edgeR and run_DESeq2. It also returns the filtering results and plots from the downstream filterDEGs function using the fold change and FDR cutoffs provided under the dbrfilter argument.

```r
args_diff <- systemArgs(sysma="param/rundiff.param", mytargets="targets_countDF.txt")

cmp <- readComp(file=argls_bam, format="matrix")

dbrlist <- runDiff(args=args_diff, diffFct=run_edgeR, targets=targetsin(args_bam), cmp=cmp[[1]], independent=TRUE, dbrfilter=c(Fold=2, FDR=1))

writeTargetsOut(x=args_diff, file="targets_rundiff.txt", overwrite=TRUE)
```

9 GO term enrichment analysis

The following performs GO term enrichment analysis for each annotated peak set.
systemPipeR ChIP-Seq Workflow

10 Motif analysis

10.1 Parse DNA sequences of peak regions from genome

Enrichment analysis of known DNA binding motifs or de novo discovery of novel motifs requires the DNA sequences of the identified peak regions. To parse the corresponding sequences from the reference genome, the getSeq function from the Biostrings package can be used. The following example parses the sequences for each peak set and saves the results to separate FASTA files, one for each peak set. In addition, the sequences in the FASTA files are ranked (sorted) by increasing p-values as expected by some motif discovery tools, such as BCRANK.

library(Biostrings); library(seqLogo); library(BCRANK)
args <- systemArgs(sysma="param/annotate_peaks.param", mytargets="targets_macs.txt")
rangefiles <- infile1(args)
for(i in seq(along=rangefiles)) {
  df <- read.delim(rangefiles[i], comment="#")
  peaks <- as(df, "GRanges")
  names(peaks) <- paste0(as.character(seqnames(peaks)), "\_", start(peaks), "\-", end(peaks))
  peaks <- peaks[order(values(peaks)$X.log10.pvalue, decreasing=TRUE)]
  pseq <- getSeq(FaFile("./data/tair10.fasta"), peaks)
  names(pseq) <- names(peaks)
  writeXStringSet(pseq, paste0(rangefiles[i], ".fasta"))
}

10.2 Motif discovery with BCRANK

The Bioconductor package BCRANK is one of the many tools available for de novo discovery of DNA binding motifs in peak regions of ChIP-Seq experiments. The given example applies this method on the first peak sample set and plots the sequence logo of the highest ranking motif.

set.seed(0)
BCRANKout <- bcrank(paste0(rangefiles[1], ".fasta"), restarts=25, use.P1=TRUE, use.P2=TRUE)
toptable(BCRANKout)
topMotif <- toptable(BCRANKout, 1)
weightMatrix <- pwm(topMotif, normalize = FALSE)
weightMatrixNormalized <- pwm(topMotif, normalize = TRUE)
pdf("results/seqlogo.pdf")
seqLogo(weightMatrixNormalized)
dev.off()
Figure 2: One of the motifs identified by BCRANK

11 Version Information

\texttt{toLatex(sessionInfo())}

- R version 3.3.1 (2016-06-21), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8, LC_PAPER=en_US.UTF-8
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: Biobase 2.34.0, BiocGenerics 0.20.0, BiocParallel 1.8.0, BiocStyle 2.2.0, Biostrings 2.42.0, DESeq2 1.14.0, GenomInforDb 1.10.0, GenomicAlignments 1.10.0, GenomicRanges 1.26.1, IRanges 2.8.0, Rsamtools 1.26.1, S4Vectors 0.12.0, ShortRead 1.32.0, SummarizedExperiment 1.4.0, XVector 0.14.0, ape 3.5, ggplot2 2.1.0, knitr 1.14, systemPipeR 1.8.1
- Loaded via a namespace (and not attached): AnnotationDbi 1.36.0, AnnotationForge 1.16.0, BBmisc 1.10, BatchJobs 1.6, Category 2.40.0, DBI 0.5-1, Formula 1.2-1, GO.db 3.4.0, GOstats 2.40.0, GSEABase 1.36.0, GenomicFeatures 1.26.0, Hmisc 3.17-4, Matrix 1.2-7.1, RBGL 1.50.0, RColorBrewer 1.1-2, RCurl 1.95-4.8, RSQLite 1.0.0, Rcpp 0.12.7, XML 3.98-1.4, acepack 1.4.0, annotate 1.52.0, assertthat 0.1, backports 1.0.3, base64enc 0.1-3, biomaRt 2.30.0, bitops 1.0-6, brew 1.0-6, checkmate 1.8.1, chron 2.3-47, cluster 2.0.5, codetools 0.2-15, colorspace 1.2-7, data.table 1.9.6, digest 0.6.10, edgeR 3.16.0, evaluate 0.10, fail 1.3, foreign 0.8-67, formatR 1.4, genefilter 1.56.0, geneplotter 1.52.0, grid 3.3.1, gridExtra 2.2.1, gtable 0.2.0, highr 0.6, htmltools 0.3.5, hwriter 1.3.2, labeling 0.3, lattice 0.20-34, latticeExtra 0.6-28, limma 3.30.0, locfit 1.5-9.1, magrittr 1.5, munsell 0.4.3, nlme 3.1-128, nnet 7.3-12, pheatmap 1.0.8, plyr 1.8.4, rjson 0.2.15, rmarkdown 1.1, rpart 4.1-10, rtracklayer 1.34.0, scales 0.4.0, sendmailR 1.2-1, splines 3.3.1, stringi 1.1.2, stringr 1.1.0, survival 2.39-5, tibble 1.2, tools 3.3.1, xtable 1.8-2, yaml 2.1.13, zlibbioc 1.20.0
This project was supported by funds from the National Institutes of Health (NIH) and the National Science Foundation (NSF).

13 References


