ChIP-Seq project report 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)

August 11, 2015

Contents

1 Introduction

This report describes the analysis of an ChIP-Seq project from Dr. First Last's lab which studies the gene expression changes of ... in *organism* .... The experimental design is as follows...

2 Sample definitions and environment settings

2.1 Environment settings and input data

Typically, the user wants to record here the sources and versions of the reference genome sequence along with the corresponding annotations. In the provided sample data set all data inputs are stored in a `data` subdirectory and all 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.

To run this sample report, mini sample FASTQ and reference genome files can be downloaded from here. The chosen data set SRP010938 contains 18 paired-end (PE) read sets from *Arabidposis thaliana* Howard et al. (2013). To minimize processing time during testing, each FASTQ file has been subsetted to 90,000-100,000 randomly sampled PE reads that map to the first 100,000 nucleotides of each chromosome of the *A. thalina* genome. The corresponding reference genome sequence (FASTA) and its GFF annotion files (provided in the same download) have been truncated accordingly. This way the entire test sample data set is less than 200MB in storage space. A PE read set has been chosen for this test data set for flexibility, because it can be used for testing both types of analysis routines requiring either SE (single end) reads or PE reads.

2.2 Required packages and resources

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

```
library(systemPipeR)
```

If applicable load custom functions not provided by *systemPipeR*

```
source("systemPipeChIPseq_Fct.R")
```

2.3 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 = "#")[,1:4]
targets

                   FileName SampleName Factor SampleLong
1  ./data/SRR446027_1.fastq        M1A     M1  Mock.1h.A
2  ./data/SRR446028_1.fastq        M1B     M1  Mock.1h.B
3  ./data/SRR446029_1.fastq        A1A     A1   Avr.1h.A
4  ./data/SRR446030_1.fastq        A1B     A1   Avr.1h.B
5  ./data/SRR446031_1.fastq        V1A     V1   Vir.1h.A
6  ./data/SRR446032_1.fastq        V1B     V1   Vir.1h.B
7  ./data/SRR446033_1.fastq        M6A     M6  Mock.6h.A
8  ./data/SRR446034_1.fastq        M6B     M6  Mock.6h.B
9  ./data/SRR446035_1.fastq        A6A     A6   Avr.6h.A
10 ./data/SRR446036_1.fastq        A6B     A6   Avr.6h.B
```

```
11 ./data/SRR446037_1.fastq        V6A     V6   Vir.6h.A
12 ./data/SRR446038_1.fastq        V6B     V6   Vir.6h.B
13 ./data/SRR446039_1.fastq       M12A    M12 Mock.12h.A
14 ./data/SRR446040_1.fastq       M12B    M12 Mock.12h.B
15 ./data/SRR446041_1.fastq       A12A    A12  Avr.12h.A
16 ./data/SRR446042_1.fastq       A12B    A12  Avr.12h.B
17 ./data/SRR446043_1.fastq       V12A    V12  Vir.12h.A
18 ./data/SRR446044_1.fastq       V12B    V12  Vir.12h.B
```

3 Read preprocessing

3.1 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`.

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

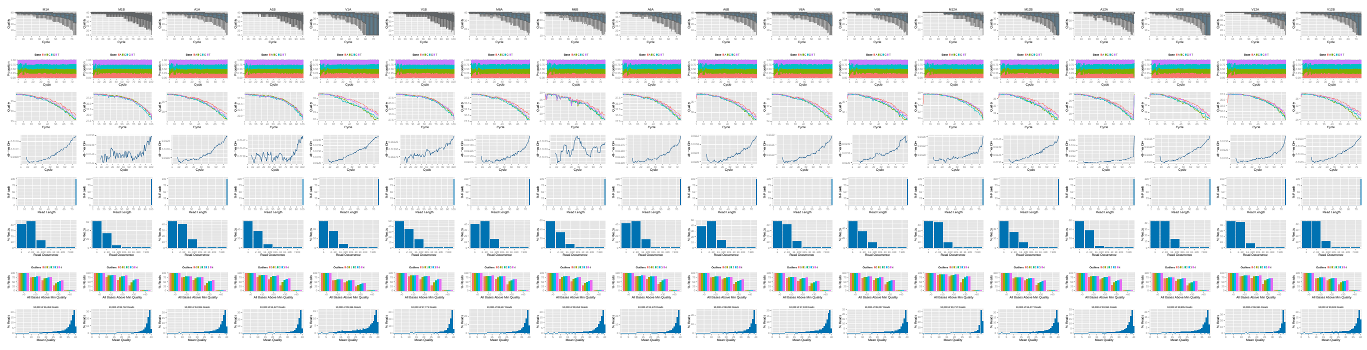

Figure 1: QC report for 18 FASTQ files.

4 Alignments

4.1 Read mapping with `Bowtie2`

The NGS reads of this project will be aligned against the reference genome sequence using `Bowtie2` (Langmead and Salzberg, 2012). The parameter settings of the aligner are defined in the `tophat.param` file.

```
args <- systemArgs(sysma="param/bowtieSE.param", mytargets="targets_chip.txt")
sysargs(args)[1] # Command-line parameters for first FASTQ file
```

Submission of alignment jobs to compute cluster, here using 72 CPU cores (18 `qsub` processes each with 4 CPU cores).

```
moduleload(modules(args))
system("bowtie2-build ./data/tair10.fasta ./data/tair10.fasta")
resources <- list(walltime="20:00:00", nodes=paste0("1:ppn=", cores(args)), memory="10gb")
reg <- clusterRun(args, conffile=".BatchJobs.R", template="torque.tmpl", Njobs=18, runid="01",
                  resourceList=resources)
runCommandline(args)
writeTargetsout(x=args, file="targets_bam.txt", overwrite=TRUE)
```

Check whether all BAM files have been created

```
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.

```
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. The corresponding URLs are written to a file with a path specified under `urlfile`, here IGVurl.txt.

```
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.

```
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

```
args <- systemArgs(sysma="param/macs2_noinput.param", mytargets="targets_mergeBamByFactor.txt")
sysargs(args)[1] # Command-line parameters for first FASTQ file
runCommandline(args)
```

```
file.exists(outpaths(args))
writeTargetsout(x=args, file="targets_macs.txt", overwrite=TRUE)
```

5.3 Peak calling with input/reference sample

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

6 Annotate peaks with genomic context

6.1 Annotation with ChIPpeakAnno package

```
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)
## Perform previous step with full genome annotation from Biomart
# txdb <- makeTxDbFromBiomart(biomart="ENSEMBL_MART_PLANT", dataset="athaliana_eg_gene")
# tx <- transcripts(txdb, columns=c("tx_name", "gene_id", "tx_type"))
# ge <- genes(txdb, columns=c("tx_name", "gene_id", "tx_type")) # works as well
# seqlevels(ge) <- c("Chr1", "Chr2", "Chr3", "Chr4", "Chr5", "ChrC", "ChrM")
# table(mcols(tx)£tx_type)
# tx <- tx[!duplicated(unstrsplit(values(tx)£gene_id, sep=","))] # Keeps only first transcript model for ea
# annotatedPeak <- annotatePeakInBatch(macsOutput, AnnotationData = tx)
```

6.2 Annotation with ChIPseeker package

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

```
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 to only one sample.

```r
peak <- readPeakFile(outpaths(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

```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

To be continued ...

```r
args_diff <- systemArgs(sysma="param/rundiff.param", mytargets="targets_countDF.txt")
cmp <- readComp(file=args_bam, format="matrix")
dbrlist <- runDiff(args=args_diff, diffFct=run_edgeR, targets=targetsin(args_bam), cmp=cmp[[1]], independe
writeTargetsout(x=args_diff, file="targets_rundiff.txt", overwrite=TRUE)
```

9 GO term enrichment analysis

The following performs GO term enrichment analysis for all annotated peak sets disregarding the differential binding results.

```r
args <- systemArgs(sysma="param/macs2.param", mytargets="targets_bam_ref.txt")
args_anno <- systemArgs(sysma="param/annotate_peaks.param", mytargets="targets_chip_anno.txt")
annofiles <- paste0(outpaths(args), ".annotated.xls"); names(annofiles) <- names(outpaths(args))
gene_ids <- sapply(names(annofiles), function(x) unique(as.character(read.delim(annofiles[x])[,"gene_id"])
load("data/GO/catdb.RData")
BatchResult <- GOCluster_Report(catdb=catdb, setlist=gene_ids, method="all", id_type="gene", CLSZ=2, cutof
```

10 Motif analysis

10.1 Parse DNA sequences of peak regions from genome

```r
library(Biostrings); library(seqLogo); library(BCRANK)
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))
    pseq <- getSeq(FaFile("./data/tair10.fasta"), peaks)
    names(pseq) <- names(peaks)
    writeXStringSet(pseq, paste0(rangefiles[i], ".fasta"))
}
```

10.2 Motif discovery with BCRANK

```
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)
seqLogo(weightMatrixNormalized)
```

11 Version Information

```
toLatex(sessionInfo())
```

- R version 3.2.1 (2015-06-18), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, stats4, utils
- Other packages: Biobase 2.29.1, BiocGenerics 0.15.5, BiocParallel 1.3.48, Biostrings 2.37.3, DBI 0.3.1, GenomeInfoDb 1.5.9, GenomicAlignments 1.5.12, GenomicRanges 1.21.17, IRanges 2.3.17, knitr 1.10.5, Rsamtools 1.21.14, RSQLite 1.0.0, S4Vectors 0.7.12, ShortRead 1.27.5, SummarizedExperiment 0.3.2, systemPipeR 1.3.23, XVector 0.9.1
- Loaded via a namespace (and not attached): annotate 1.47.4, AnnotationDbi 1.31.17, AnnotationForge 1.11.15, base64enc 0.1-3, BatchJobs 1.6, BBmisc 1.9, BiocStyle 1.7.6, bitops 1.0-6, brew 1.0-6, Category 2.35.1, checkmate 1.6.2, colorspace 1.2-6, digest 0.6.8, edgeR 3.11.2, evaluate 0.7, fail 1.2, formatR 1.2, futile.logger 1.4.1, futile.options 1.0.0, genefilter 1.51.0, ggplot2 1.0.1, GO.db 3.1.2, GOstats 2.35.1, graph 1.47.2, grid 3.2.1, GSEABase 1.31.3, gtable 0.1.2, highr 0.5, hwriter 1.3.2, lambda.r 1.1.7, lattice 0.20-33, latticeExtra 0.6-26, limma 3.25.15, magrittr 1.5, MASS 7.3-43, Matrix 1.2-2, munsell 0.4.2, pheatmap 1.0.7, plyr 1.8.3, proto 0.3-10, RBGL 1.45.1, RColorBrewer 1.1-2, Rcpp 0.12.0, reshape2 1.4.1, rjson 0.2.15, scales 0.2.5, sendmailR 1.2-1, splines 3.2.1, stringi 0.5-5, stringr 1.0.0, survival 2.38-3, tools 3.2.1, XML 3.98-1.3, xtable 1.7-4, zlibbioc 1.15.0

12 Funding

This project was supported by funds from the National Institutes of Health (NIH) and the National Science Foundation (NSF).

13 References

Thomas Girke. systemPipeR: NGS workflow and report generation environment, 28 June 2014. URL https://github.com/tgirke/systemPipeR.

Brian E Howard, Qiwen Hu, Ahmet Can Babaoglu, Manan Chandra, Monica Borghi, Xiaoping Tan, Luyan He, Heike Winter-Sederoff, Walter Gassmann, Paola Veronese, and Steffen Heber. High-throughput RNA sequencing of pseudomonas-infected arabidopsis reveals hidden transcriptome complexity and novel splice variants. *PLoS One*, 8 (10):e74183, 1 October 2013. ISSN 1932-6203. doi: 10.1371/journal.pone.0074183. URL http://dx.doi.org/10.1371/journal.pone.0074183.

Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nat. Methods*, 9(4):357–359, April 2012. ISSN 1548-7091. doi: 10.1038/nmeth.1923. URL http://dx.doi.org/10.1038/nmeth.1923.