ChIP-Seq Analysis with R and Bioconductor

Advanced R/Bioconductor Workshop on High-Throughput Genetic Analysis

- Fred Hutchinson Cancer Research Center - Seattle, WA

Thomas Girke

February 27-28, 2012
Introduction
ChIP-Seq Technology
Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis
Sample Data
Aligning Short Reads
Coverage Data
Peak Calling
Annotating Peaks
Differential Binding Analysis
View Peaks in Genome Browser
Common Motifs in Peak Sequences
Introduction

ChIP-Seq Technology
Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis
Sample Data
Aligning Short Reads
Coverage Data
Peak Calling
Annotating Peaks
Differential Binding Analysis
View Peaks in Genome Browser
Common Motifs in Peak Sequences
Introduction

ChIP-Seq Technology
Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis
Sample Data
Aligning Short Reads
Coverage Data
Peak Calling
Annotating Peaks
Differential Binding Analysis
View Peaks in Genome Browser
Common Motifs in Peak Sequences
ChIP-Seq Technology
ChIP-Seq Workflow

- Read mapping
- Peak calling
- Peak annotation/filtering
- Differential peak analysis
- Motif enrichment analysis in sequences under peaks
Peak Callers (Command-line Tools)

- CisGenome
- ERANGE
- FindPeaks
- F-Seq
- GLITR
- MACS
- PeakSeq
- QuEST
- SICER
- SiSSRs
- spp
- USeq
- ...

Outline

Introduction

ChIP-Seq Technology

Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis

Sample Data
Aligning Short Reads
Coverage Data
Peak Calling
Annotating Peaks
Differential Binding Analysis
View Peaks in Genome Browser
Common Motifs in Peak Sequences
General Purpose Resources for ChIP-Seq Analysis in R

- GenomicRanges [Link]: high-level infrastructure for range data
- Rsamtools [Link]: BAM support
- rtracklayer [Link]: Annotation imports, interface to online genome browsers
- DESeq [Link]: RNA-Seq analysis
- edgeR [Link]: RNA-Seq analysis
- chipseq [Link]: Utilities for ChIP-Seq analysis
- ChIPpeakAnno [Link]: Annotating peaks with genome context information

...
Peak Calling in R

- BayesPeak: hidden Markov models (HMM) and Bayesian statistics
- PICS: probabilistic inference
- DiffBind: Differential binding analysis of ChIP-Seq peak data
- MOSAiCS: model-based analysis of ChIP-Seq data
- iSeq: Hidden Ising Models
- ChIPseqR
- CSAR: tests based on Poisson distribution
- ChIP-Seq
- SPP
Outline

Introduction
  - ChIP-Seq Technology
  - Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis
  - Sample Data
  - Aligning Short Reads
  - Coverage Data
  - Peak Calling
  - Annotating Peaks
  - Differential Binding Analysis
  - View Peaks in Genome Browser
  - Common Motifs in Peak Sequences
Introduction

ChIP-Seq Technology
Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis

Sample Data
Aligning Short Reads
Coverage Data
Peak Calling
Annotating Peaks
Differential Binding Analysis
View Peaks in Genome Browser
Common Motifs in Peak Sequences
Data Sets and Experimental Variables

- To make the following sample code work, users can download the sample data into the directory of their current R session as shown below.

- It contains slimmed down versions of four FASTQ files from the ChIP-Seq experiment published by Kaufman et al (2010, GSE20176), a shortened GFF3 annotation file and the corresponding reference genome from *Arabidopsis thaliana*.

For RStudio/AMI setup of workshop: create a symbolic link to data sets located under "/R-pkgs/"

```r
> system("ln -s /R-pkgs/data data")
```

Users working on local Unix/Linux or OS X systems want to download the sample data from the Internet like this:

```r
> system("wget http://biocluster.ucr.edu/~tgirke/HTML_Presentations/Manuals/Rngsapps/chipseqBioc2012/data.zip")
> system("unzip data.zip")
```

Windows users can download the same files manually from here: Slides, R code, Data Sets.

Outline

Introduction
- ChIP-Seq Technology
- Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis
- Sample Data
- Aligning Short Reads
- Coverage Data
- Peak Calling
- Annotating Peaks
- Differential Binding Analysis
- View Peaks in Genome Browser
- Common Motifs in Peak Sequences
Align Reads and Output Indexed Bam Files

Note: Rsubread is Linux only. OS X/Windows users want to skip the mapping step and download the Bam files from here:

```r
> library(Rsubread); library(Rsamtools)
> dir.create("results") # Note: all output data will be written to directory 'results'
> buildindex(basename="./results/tair10chr.fasta", reference="./data/tair10chr.fasta") # Build indexed reference genome
> targets <- read.delim("./data/targets.txt") # Import experiment design information
> targets
     Samples  Factor Fastq
   1     AP1IND1     sig1 SRR038845.fastq
   2     AP1IND2     sig1 SRR038846.fastq
   3     AP1UIND1      bgr1 SRR038848.fastq
   4     AP1UIND2      bgr1 SRR038850.fastq

> input <- paste("./data/", targets[,3], sep="")
> output <- paste("./results/", targets[,3], ".sam", sep="")
> reference <- "./results/tair10chr.fasta"
> for(i in seq(along=targets[,3])) {
+   align(index=reference, readfile1=input[i], output_file=output[i], nthreads=8, indels=1, TH1=2)
+   asBam(file=output[i], destination=gsub(".sam", ",", output[i]), overwrite=TRUE, indexDestination=TRUE)
+   unlink(output[i])
+ }
```
Outline

Introduction
  ChIP-Seq Technology
  Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis
  Sample Data
  Aligning Short Reads
  Coverage Data
  Peak Calling
  Annotating Peaks
  Differential Binding Analysis
  View Peaks in Genome Browser
  Common Motifs in Peak Sequences
**Important Resources for ChIP-Seq Analysis**

**Coverage and peak slicing**

```r
> library(rtracklayer); library(GenomicRanges)
> samples <- as.character(targets$Fastq)
> samplespath <- paste("./results/", samples, ".bam", sep="")
> aligns <- readBamGappedAlignments(samplespath[3])
> cov <- coverage(aligns)
> islands <- slice(cov, lower = 15)
> islands[[1]]
```

Views on a 30427671-length Rle subject

<table>
<thead>
<tr>
<th>views:</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
</tr>
<tr>
<td>[1] 11998</td>
</tr>
<tr>
<td>[2] 27064</td>
</tr>
<tr>
<td>[3] 30619</td>
</tr>
<tr>
<td>[4] 30709</td>
</tr>
<tr>
<td>[5] 31956</td>
</tr>
<tr>
<td>[6] 48015</td>
</tr>
<tr>
<td>[7] 53839</td>
</tr>
<tr>
<td>[8] 54031</td>
</tr>
<tr>
<td>[9] 64384</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>[43] 13972892</td>
</tr>
<tr>
<td>[44] 13973159</td>
</tr>
<tr>
<td>[45] 13973859</td>
</tr>
<tr>
<td>[46] 15562796</td>
</tr>
<tr>
<td>[47] 15563233</td>
</tr>
<tr>
<td>[48] 15564861</td>
</tr>
<tr>
<td>[49] 15871554</td>
</tr>
<tr>
<td>[50] 20477102</td>
</tr>
<tr>
<td>[51] 20558301</td>
</tr>
</tbody>
</table>
```
Coverage Plot

```r
> plotCov <- function(mycov=cov, mychr=1, mypos=c(1,1000), mymain="Coverage", ...) {
+   op <- par(mar=c(8,3,6,1))
+   plot(as.numeric(mycov[[mychr]][mypos[1]:mypos[2]]), type="l",
+     lwd=1, col="blue", ylab="", main=mymain, xlab="", xaxt="n", ...)
+   axis(1, las = 2, at=seq(1,mypos[2]-mypos[1], length.out= 10),
+     labels=as.integer(seq(mypos[1], mypos[2], length.out= 10)))
+   par(op)
+ }
> plotCov(mycov=cov, mychr="Chr1", mypos=c(1,120000)) # Remember: read data is truncated to first 100kbp
```

![Coverage Plot](coverage_plot.png)
Import aligned reads (bam files) and extend to 200bp

```r
> chip_signal_list <- sapply(samplespath, list)
> for(i in seq(along=samplespath)) {
+   aligns <- readBamGappedAlignments(samplespath[i])
+   chip_signal_list[[i]] <- as(aligns, "GRanges")
+ }
> chip_signal_list["./results/SRR038845.fastq.bam"][1:4,]

GRanges with 4 ranges and 0 elementMetadata cols:

    seqnames   ranges   strand
   <Rle>  <IRanges> <Rle>
[1] Chr1 [121, 156]   
[2] Chr1 [121, 156]   
[3] Chr1 [216, 251]   
---

seqlengths:

    Chr1  Chr2  Chr3  Chr4  Chr5  ChrC  ChrM
30427671 19698289 23459830 18585056 26975502 154478 366924

> chip_signal_list <- sapply(names(chip_signal_list), function(x) resize(chip_signal_list[[x]], width = 200)
```
Outline

Introduction
ChIP-Seq Technology
Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis
Sample Data
Aligning Short Reads
Coverage Data
Peak Calling
Annotating Peaks
Differential Binding Analysis
View Peaks in Genome Browser
Common Motifs in Peak Sequences
Naive Peak Calling by Coverage Value

Compute coverage and call peaks

```r
> Nreads <- sapply(names(chip_signal_list), function(x) length(chip_signal_list[[x]]))
> normfactor <- 10^6/Nreads
> chip_signal_list <- sapply(names(chip_signal_list), function(x) coverage(chip_signal_list[[x]]) * normfactor)
> chip_signal_list[['./results/SRR038845.fastq.bam']][1:2,]

SimpleRleList of length 2
$Chr1
'numeric' Rle of length 30427671 with 4709 runs
  Lengths: 139 17 59 10 69 ... 138 62 138 85 200 1248
  Values : 6.42287571414349 12.845751428287 6.42287571414349 9.63431357121524 22.4800649995022 35.3258164277892 ...

$Chr2
'numeric' Rle of length 19698289 with 21336 runs
  Lengths: 821 2 12 1 1 ... 1 3 1 135158 200 265402
  Values : 0 6.42287571414349 9.63431357121524 16.0571892853587 19.2686271424305 22.4800649995022 ...
```

Views on a 30427671-length Rle subject
views:

<table>
<thead>
<tr>
<th>start</th>
<th>end</th>
<th>width</th>
</tr>
</thead>
</table>
Compute coverage and call peaks

```r
> library(BayesPeak)
> sig <- readBamGappedAlignments(samplespath[1])
> bgr <- readBamGappedAlignments(samplespath[3])
> sig <- as(as(sig, "GRanges"), "RangedData")
> bgr <- as(as(bgr, "GRanges"), "RangedData")
> raw.output <- bayespeak(treatment=sig, control=bgr, start = 1, end = 100000)

..............

> # unreliable.jobs <- log(raw.output$QC$lambda1) < 1.5 # Removal of false positives due to overfitting.
> # bpeaks <- as.data.frame(summarise.peaks(raw.output, method = "lowerbound", exclude.jobs = unreliable.jobs))
> bpeaks <- as.data.frame(summarise.peaks(raw.output, method = "lowerbound"))
> source("./data/Fct/chipseqFct.R") # Imports the rangeCoverage function.
> sigcovDF <- rangeCoverage(summaryFct=viewMeans, myname="sig_", peaksIR=bpeaks[,1:3], sig=sig, readextend=1, normfactor=10^6/length(start(sig)))
> bgrcovDF <- rangeCoverage(summaryFct=viewMeans, myname="bgr_", peaksIR=bpeaks[,1:3], sig=bgr, readextend=1, normfactor=10^6/length(start(bgr)))
> bpeaksDF <- cbind(bpeaks, sigcovDF[-1], bgrcovDF[-1])
> bpeaksDF[1:4,]
```

<table>
<thead>
<tr>
<th>space</th>
<th>start</th>
<th>end</th>
<th>width</th>
<th>PP</th>
<th>sig_cov</th>
<th>sig_cov.pos</th>
<th>sig_cov.neg</th>
<th>bgr_cov</th>
<th>bgr_cov.pos</th>
<th>bgr_cov.neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chr1</td>
<td>8301</td>
<td>8401</td>
<td>101</td>
<td>0.5717232</td>
<td>0.1907785</td>
<td>0.1271857</td>
<td>0.06359283</td>
<td>0.039603881</td>
<td>0.039603881</td>
<td>0.000000000</td>
</tr>
<tr>
<td>Chr1</td>
<td>11151</td>
<td>11351</td>
<td>201</td>
<td>0.9998908</td>
<td>0.6390921</td>
<td>0.5911602</td>
<td>0.04793191</td>
<td>0.009950229</td>
<td>0.009950229</td>
<td>0.000000000</td>
</tr>
<tr>
<td>Chr1</td>
<td>13601</td>
<td>13751</td>
<td>151</td>
<td>0.9998793</td>
<td>0.4040882</td>
<td>0.2339458</td>
<td>0.17014240</td>
<td>0.092715046</td>
<td>0.079470040</td>
<td>0.013245007</td>
</tr>
<tr>
<td>Chr1</td>
<td>16601</td>
<td>17051</td>
<td>451</td>
<td>0.9999922</td>
<td>0.4699665</td>
<td>0.1922590</td>
<td>0.27770749</td>
<td>0.008869162</td>
<td>0.004434581</td>
<td>0.004434581</td>
</tr>
</tbody>
</table>
> plotCov(mycov=chip_signal_list[[1]], mychr="Chr1", mypos=c(11000, 11600), ylim=c(0,200))
**Identify Common Peaks Among Two Methods**

Compares results from simple cutoff method with BayesPeak results

```r
> simple_peak <- as.data.frame(as(chip_peak_list[[1]], "IRangesList"))
> # simple_peak <- as.data.frame(chip_peak_list[[1]])
> commonpeaks <- subsetByOverlaps(as(bpeaks, "RangedData"), as(simple_peak, "RangedData"), minoverlap=100)
> bpeaksDF[bpeaksDF$start %in% start(commonpeaks),][1:4,]
```

<table>
<thead>
<tr>
<th>space</th>
<th>start</th>
<th>end</th>
<th>width</th>
<th>PP</th>
<th>sig_cov</th>
<th>sig_cov.pos</th>
<th>sig_cov.neg</th>
<th>bgr_cov</th>
<th>bgr_cov.pos</th>
<th>bgr_cov.neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chr1</td>
<td>8301</td>
<td>8401</td>
<td>101</td>
<td>0.5717</td>
<td>0.1908</td>
<td>0.1272</td>
<td>0.0636</td>
<td>0.0396</td>
<td>0.0396</td>
<td>0.0000</td>
</tr>
<tr>
<td>Chr1</td>
<td>11151</td>
<td>11351</td>
<td>201</td>
<td>0.9999</td>
<td>0.6400</td>
<td>0.5912</td>
<td>0.0479</td>
<td>0.0099</td>
<td>0.0099</td>
<td>0.0000</td>
</tr>
<tr>
<td>Chr1</td>
<td>13601</td>
<td>13751</td>
<td>151</td>
<td>0.9998</td>
<td>0.4040</td>
<td>0.2339</td>
<td>0.1701</td>
<td>0.0927</td>
<td>0.0794</td>
<td>0.0132</td>
</tr>
<tr>
<td>Chr1</td>
<td>16601</td>
<td>17051</td>
<td>451</td>
<td>0.9999</td>
<td>0.4699</td>
<td>0.1923</td>
<td>0.2777</td>
<td>0.0089</td>
<td>0.0044</td>
<td>0.0044</td>
</tr>
</tbody>
</table>
Exercise 1: Compare Results with Published Peaks

**Task 1** Import peaks predicted by Kaufmann et al (2010).

**Task 2** Determine how many of the published peaks have at least a 50% length overlap with the results from the BayesPeak and the naive peak calling methods.

**Required information:**

```r
> pubpeaks <- read.delim("./data/Kaufmann_peaks100k.txt") # Published peaks for first 100kbp on chromosomes.
> pubpeaks <- pubpeaks[order(pubpeaks$space, pubpeaks$start),]
> pubpeaks[1:4,]

<table>
<thead>
<tr>
<th>PeakID</th>
<th>space</th>
<th>start</th>
<th>end</th>
<th>score_position</th>
<th>score_position</th>
<th>length</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1_3132</td>
<td>chr1_3132</td>
<td>Chr1</td>
<td>3094</td>
<td>3172</td>
<td>3132</td>
<td>4.656515</td>
</tr>
<tr>
<td>chr1_8365</td>
<td>chr1_8365</td>
<td>Chr1</td>
<td>8222</td>
<td>8425</td>
<td>8365</td>
<td>11.046212</td>
</tr>
<tr>
<td>chr1_11298</td>
<td>chr1_11298</td>
<td>Chr1</td>
<td>11149</td>
<td>11440</td>
<td>11298</td>
<td>52.109592</td>
</tr>
<tr>
<td>chr1_13686</td>
<td>chr1_13686</td>
<td>Chr1</td>
<td>13602</td>
<td>13756</td>
<td>13686</td>
<td>5.341907</td>
</tr>
</tbody>
</table>
```

> # Import olRanges function, which accepts two GRranges (IRanges) objects
> source("./data/Fct/rangeoverlapper.R")

Outline

Introduction
  ChIP-Seq Technology
  Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis
  Sample Data
  Aligning Short Reads
  Coverage Data
  Peak Calling

Annotating Peaks
  Differential Binding Analysis
  View Peaks in Genome Browser
  Common Motifs in Peak Sequences
Import Annotation Data from GFF

Annotation data from GFF

> library(rtracklayer); library(GenomicRanges); library(Rsamtools)
> gff <- import.gff("./data/TAIR10_GFF3_trunc.gff", asRangedData=FALSE)
> seqlengths(gff) <- end(ranges(gff[which(elementMetadata(gff)[,"type"]=='"chromosome"'),]))
> subgene_index <- which(elementMetadata(gff)[,"type"] == "gene")
> gffsub <- gff[subgene_index,] # Returns only gene ranges
> strand(gffsub) <- "*" # For strand insensitive analysis
> gffsub[1:4,1:2]

GRanges with 4 ranges and 2 elementMetadata cols:

seqnames  ranges   strand | source  type
    <Rle> <IRanges>  <Rle> | <factor> <factor>
[1]   Chr1  [ 3631,  5899] * |  TAIR10 gene
[2]   Chr1  [ 5928,  8737] * |  TAIR10 gene
[3]   Chr1  [11649, 13714] * |  TAIR10 gene
---

seqlengths:

    Chr1  Chr2  Chr3  Chr4  Chr5  ChrC  ChrM
  30427671 19698289 23459830 18585056 26975502  154478  366924

> ids <- elementMetadata(gffsub)[, "group"]
> gffgene <- gffsub
> gffsub <- split(gffsub) # Coerce to GRangesList
Annotate Peaks with ChIPpeakAnno

```r
> library(ChIPpeakAnno)
> annoRD <- unlist(gffsub)
> names(annoRD) <- gsub(".*="", "", elementMetadata(annoRD)[, "group"])
> annoRD <- as(annoRD, "RangedData")
> peaksRD <- RangedData(space=bpeaksDF$space, IRanges(bpeaksDF$start, bpeaksDF$end))
> annotatedPeak <- annotatePeakInBatch(peaksRD, AnnotationData = annoRD)
> as.data.frame(annotatedPeak)[1:4,1:11]

<table>
<thead>
<tr>
<th>space</th>
<th>start</th>
<th>end</th>
<th>width</th>
<th>names</th>
<th>peak</th>
<th>strand</th>
<th>feature</th>
<th>start_position</th>
<th>end_position</th>
<th>insideFeature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chr1</td>
<td>8301</td>
<td>8401</td>
<td>101</td>
<td>AT1G01020</td>
<td>01</td>
<td>+</td>
<td>AT1G01020</td>
<td>5928</td>
<td>8737</td>
<td>inside</td>
</tr>
<tr>
<td>Chr1</td>
<td>11151</td>
<td>11351</td>
<td>201</td>
<td>AT1G01030</td>
<td>02</td>
<td>+</td>
<td>AT1G01030</td>
<td>11649</td>
<td>13714</td>
<td>upstream</td>
</tr>
<tr>
<td>Chr1</td>
<td>13601</td>
<td>13751</td>
<td>151</td>
<td>AT1G01030</td>
<td>03</td>
<td>+</td>
<td>AT1G01030</td>
<td>11649</td>
<td>13714</td>
<td>overlapEnd</td>
</tr>
<tr>
<td>Chr1</td>
<td>16601</td>
<td>17051</td>
<td>451</td>
<td>AT1G01030</td>
<td>04</td>
<td>+</td>
<td>AT1G01030</td>
<td>11649</td>
<td>13714</td>
<td>downstream</td>
</tr>
</tbody>
</table>

> bpeaksDF[1:4,]

<table>
<thead>
<tr>
<th>space</th>
<th>start</th>
<th>end</th>
<th>width</th>
<th>PP</th>
<th>sig_c cov</th>
<th>sig_c cov.pos</th>
<th>sig_c cov.neg</th>
<th>bgr_c cov</th>
<th>bgr_c cov.pos</th>
<th>bgr_c cov.neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chr1</td>
<td>8301</td>
<td>8401</td>
<td>101</td>
<td>0.5717232</td>
<td>0.1907785</td>
<td>0.06359283</td>
<td>0.039603881</td>
<td>0.00000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chr1</td>
<td>11151</td>
<td>11351</td>
<td>201</td>
<td>0.9998908</td>
<td>0.6390921</td>
<td>0.04793191</td>
<td>0.009950229</td>
<td>0.00000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chr1</td>
<td>13601</td>
<td>13751</td>
<td>151</td>
<td>0.9998793</td>
<td>0.4040882</td>
<td>0.2339458</td>
<td>0.092715046</td>
<td>0.079470040</td>
<td>0.013245007</td>
<td></td>
</tr>
<tr>
<td>Chr1</td>
<td>16601</td>
<td>17051</td>
<td>451</td>
<td>0.9999922</td>
<td>0.4699665</td>
<td>0.1922590</td>
<td>0.07770749</td>
<td>0.08869162</td>
<td>0.004434581</td>
<td></td>
</tr>
</tbody>
</table>
```
Alternative Peak Annotation Approach

Alternative approach using `olRanges` function

```r
> source("./data/Fct/rangeoverlapper.R")
> olRanges(query=gffgene, subject=as(as(bpeaks, "RangedData"), "GRanges"), output="df")[1:2,]
```

<table>
<thead>
<tr>
<th></th>
<th>space</th>
<th>Qindex</th>
<th>Sindex</th>
<th>Qstart</th>
<th>Qend</th>
<th>Sstart</th>
<th>Send</th>
<th>OLstart</th>
<th>OLend</th>
<th>OLlength</th>
<th>OLpercQ</th>
<th>OLpercS</th>
<th>OLtype</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chr1</td>
<td>2</td>
<td>1</td>
<td>5928</td>
<td>8737</td>
<td>8301</td>
<td>8401</td>
<td>8301</td>
<td>8401</td>
<td>101</td>
<td>3.594306</td>
<td>100.0000</td>
<td>inside</td>
</tr>
<tr>
<td>2</td>
<td>Chr1</td>
<td>3</td>
<td>3</td>
<td>11649</td>
<td>13714</td>
<td>13601</td>
<td>13751</td>
<td>13601</td>
<td>13714</td>
<td>114</td>
<td>5.517909</td>
<td>75.49669</td>
<td>oldown</td>
</tr>
</tbody>
</table>

```r
> as.data.frame(annotatedPeak)[c(2,5),1:11] # Corresponding result from ChIPpeakAnno
```

<table>
<thead>
<tr>
<th></th>
<th>space</th>
<th>start</th>
<th>end</th>
<th>width</th>
<th>names</th>
<th>peak</th>
<th>strand</th>
<th>feature</th>
<th>start_position</th>
<th>end_position</th>
<th>insideFeature</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Chr1</td>
<td>11151</td>
<td>11351</td>
<td>201</td>
<td>02 AT1G01030</td>
<td>02 + AT1G01030</td>
<td>11649</td>
<td>13714</td>
<td>upstream</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Chr1</td>
<td>20901</td>
<td>21151</td>
<td>251</td>
<td>05 AT1G01040</td>
<td>05 + AT1G01040</td>
<td>23146</td>
<td>31227</td>
<td>upstream</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

Introduction
  ChIP-Seq Technology
  Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis
  Sample Data
  Aligning Short Reads
  Coverage Data
  Peak Calling
  Annotating Peaks

Differential Binding Analysis
  View Peaks in Genome Browser
  Common Motifs in Peak Sequences
Annotation data from GFF

> peakranges <- GRanges(seqnames = Rle(bpeaksDF$space), ranges = IRanges(bpeaksDF$start, bpeaksDF$end),
+   strand = Rle(strand("*")), peakIDs=paste("peak", seq(along=bpeaksDF[,1]), sep="_"))
> countDF <- data.frame(row.names=elementMetadata(peakranges)[,"peakIDs"])
> peakranges <- split(peakranges) # Coerce to GRangesList
> for(i in samplespath) {
+   aligns <- readBamGappedAlignments(i) # Substitute next two lines with this one.
+   counts <- countOverlaps(peakranges, aligns)
+   countDF <- cbind(countDF, counts)
+ }
> colnames(countDF) <- samples
> rownames(countDF) <- gsub(".*=", ",", rownames(countDF))
> countDF[1:4,]

```
SRR038845.fastq  SRR038846.fastq  SRR038848.fastq  SRR038850.fastq
peak_1           9           36           2           5
peak_2           54          69           1           9
peak_3           31          13           7           3
peak_4           75          101          2          13
```

> write.table(countDF, ".results/countDF", quote=FALSE, sep="\t", col.names = NA)
> countDF <- read.table("./results/countDF")
**Simple RPKM Normalization**

RPKM: here defined as reads per kilobase of sequence range per million mapped reads

```r
> returnRPKM <- function(counts, ranges) {
+   geneLengthsInKB <- sum(width(ranges))/1000 # Number of bases per sequence range in kbp
+   millionsMapped <- sum(counts)/1e+06 # Factor for converting to million of mapped reads.
+   rpm <- counts/millionsMapped # RPK: reads per kilobase of sequence range.
+   rpkm <- rpm/geneLengthsInKB # RPKM: reads per kilobase of sequence range per million mapped reads.
+   return(rpkm)
+ }
> countDFrpkm <- apply(countDF, 2, function(x) returnRPKM(counts=x, ranges=peakranges))
> countDFrpkm[1:4,]

<table>
<thead>
<tr>
<th></th>
<th>SRR038845.fastq</th>
<th>SRR038846.fastq</th>
<th>SRR038848.fastq</th>
<th>SRR038850.fastq</th>
</tr>
</thead>
<tbody>
<tr>
<td>peak_1</td>
<td>3761.298</td>
<td>12458.864</td>
<td>600.4239</td>
<td>1881.3160</td>
</tr>
<tr>
<td>peak_2</td>
<td>11340.033</td>
<td>11999.146</td>
<td>150.8528</td>
<td>1701.6082</td>
</tr>
<tr>
<td>peak_3</td>
<td>8665.654</td>
<td>3009.288</td>
<td>1405.6281</td>
<td>755.0182</td>
</tr>
<tr>
<td>peak_4</td>
<td>7019.422</td>
<td>7827.844</td>
<td>134.4630</td>
<td>1095.4182</td>
</tr>
</tbody>
</table>
```
QC Check

QC check by computing a sample correlating matrix and plotting it as a tree

```r
> d <- cor(countDFRpmk, method="spearman")
> plot(hclust(dist(1-d))) # Sample tree
```

Cluster Dendrogram

```
```

ChIP-Seq Analysis with R and Bioconductor

ChIP-Seq Analysis

Differential Binding Analysis
Identify DiffPeaks with Simple Fold Change Method

Compute mean values for replicates

```r
> source("./data/Fct/colAg.R")
> countDFrpkm_mean <- colAg(myMA=countDFrpkm, group=c(1,1,2,2), myfct=mean)
> countDFrpkm_mean[1:4,]

SRR038845.fastq SRR038846.fastq SRR038848.fastq SRR038850.fastq
peak_1 8110.081 1240.8700
peak_2 11669.590 926.2305
peak_3 5837.471 1080.3232
peak_4 7423.633 614.9406
```

Log2 fold changes

```r
> countDFrpkm_mean <- cbind(countDFrpkm_mean, log2ratio=log2(countDFrpkm_mean[,1]/countDFrpkm_mean[,2]))
> countDFrpkm_mean <- countDFrpkm_mean[is.finite(countDFrpkm_mean[,3]), ]
> degs2fold <- countDFrpkm_mean[countDFrpkm_mean[,3] >= 1 | countDFrpkm_mean[,3] <= -1,]
> degs2fold[1:4,]

SRR038845.fastq SRR038846.fastq SRR038848.fastq SRR038850.fastq log2ratio
peak_1 8110.081 1240.8700 2.708364
peak_2 11669.590 926.2305 3.655239
peak_3 5837.471 1080.3232 2.433881
peak_4 7423.633 614.9406 3.593606
```

```r
> write.table(degs2fold, "/results/degs2fold.txt", quote=FALSE, sep="\t", col.names = NA)
> degs2fold <- read.table("./results/degs2fold.txt")
```
Identify DiffPeaks with DESeq Library

Raw count data are expected here!

```r
> library(DESeq)
> countDF <- read.table("./results/countDF")
> conds <- targets$Factor
> cds <- newCountDataSet(countDF, conds) # Creates object of class CountDataSet derived from eSet class
> counts(cds)[1:4, ] # CountDataSet has similar accessor methods as eSet class.
```

<table>
<thead>
<tr>
<th></th>
<th>SRR038845.fastq</th>
<th>SRR038846.fastq</th>
<th>SRR038848.fastq</th>
<th>SRR038850.fastq</th>
</tr>
</thead>
<tbody>
<tr>
<td>peak_1</td>
<td>9</td>
<td>36</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>peak_2</td>
<td>54</td>
<td>69</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>peak_3</td>
<td>31</td>
<td>13</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>peak_4</td>
<td>75</td>
<td>101</td>
<td>2</td>
<td>13</td>
</tr>
</tbody>
</table>

```r
> cds <- estimateSizeFactors(cds) # Estimates library size factors from count data. Alternatively, one can
> cds <- estimateDispersions(cds, fitType="local") # Estimates the variance within replicates
> res <- nbinomTest(cds, "bgr1", "sig1") # Calls DEGs with nbinomTest
> res <- na.omit(res)
> res2fold <- res[res$log2FoldChange >= 1 | res$log2FoldChange <= -1,]
> res2foldpadj <- res2fold[res2fold$padj <= 0.2, ] # Here padj set very high for demo purpose
> res2foldpadj[1:2,1:8]
```

<table>
<thead>
<tr>
<th>id</th>
<th>baseMean</th>
<th>baseMeanA</th>
<th>baseMeanB</th>
<th>foldChange</th>
<th>log2FoldChange</th>
<th>pval</th>
<th>padj</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>peak_2</td>
<td>26.86951</td>
<td>6.748212</td>
<td>46.99082</td>
<td>6.963447</td>
<td>2.799802</td>
<td>0.0007686464</td>
</tr>
<tr>
<td>4</td>
<td>peak_4</td>
<td>38.65216</td>
<td>10.122757</td>
<td>67.18157</td>
<td>6.636687</td>
<td>2.730463</td>
<td>0.0005413709</td>
</tr>
</tbody>
</table>
Identify DiffPeaks with edgeR Library

Raw count data are expected here!

```r
> library(edgeR)
> countDF <- read.table("./results/countDF")
> y <- DGEList(counts=countDF, group=conds) # Constructs DGEList object
> y <- estimateCommonDisp(y) # Estimates common dispersion
> y <- estimateTagwiseDisp(y) # Estimates tagwise dispersion
> et <- exactTest(y, pair=c("bgr1", "sig1")) # Computes exact test for the negative binomial distribution.
> topTags(et, n=4)

Comparison of groups: sig1-bgr1

<table>
<thead>
<tr>
<th>logFC</th>
<th>logCPM</th>
<th>PValue</th>
<th>FDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.869</td>
<td>10.523</td>
<td>6.28462e-08</td>
<td>4.022157e-06</td>
</tr>
<tr>
<td>3.910</td>
<td>10.905</td>
<td>8.97545e-07</td>
<td>2.085477e-05</td>
</tr>
<tr>
<td>3.710</td>
<td>10.816</td>
<td>9.77567e-07</td>
<td>2.085477e-05</td>
</tr>
</tbody>
</table>

> edge <- as.data.frame(topTags(et, n=50000))
> edge2fold <- edge[|edge$logFC >= 1 | edge$logFC <= -1,]
> edge2foldpadj <- edge2fold[edge2fold$FDR <= 0.01, ]
```
Merge Results and Compute Overlaps Among Methods

Here overlaps for 20 best ranking peaks of each method!

```r
> bothDF <- merge(res, countDFrpkm_mean, by.x=1, by.y=0, all=TRUE); bothDF <- na.omit(bothDF)
> cor(bothDF[,"log2FoldChange"], bothDF[,"log2ratio"], method="spearman")
[1] 0.9949634
> source("./data/Fct/overLapper.R")
> setlist <- list(edgeR=rownames(edge[order(edge$FDR),][1:20,]),
+            DESeq=res[order(res$padj),][1:20,"id"],
+            RPKM=rownames(degs2fold[order(-degs2fold$log2ratio),][1:20,]))
> OLlist <- overLapper(setlist=setlist, sep="_", type="vennsets")
> counts <- sapply(OLlist$Venn_List, length)
> vennPlot(counts=counts)
```

**Venn Diagram**

```
1 2
1 1
2 16
1
```

- edgeR
- DESeq
- RPKM

Unique objects: All = 24; S1 = 20; S2 = 20; S3 = 20
Outline

Introduction
  ChIP-Seq Technology
  Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis
  Sample Data
  Aligning Short Reads
  Coverage Data
  Peak Calling
  Annotating Peaks
  Differential Binding Analysis

View Peaks in Genome Browser
  Common Motifs in Peak Sequences
Inspect Results in IGV

View peak_3 in IGV

- Download and open IGV
- Select in menu in top left corner A. thaliana (TAIR10)
- Upload the following indexed/sorted Bam files with File -> Load from URL...


- To view peak_3, enter its coordinates Chr1:16656-16956 in position menu on top.
Outline

Introduction
- ChIP-Seq Technology
- Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis
- Sample Data
- Aligning Short Reads
- Coverage Data
- Peak Calling
- Annotating Peaks
- Differential Binding Analysis
- View Peaks in Genome Browser

Common Motifs in Peak Sequences
Sequence Motifs Enriched in Peak Sequences

Extract peak sequences and predict enriched motifs with COSMO library

```r
library(Biostrings); library(cosmo); library(seqLogo)

Welcome to cosmo version 1.22.0

cosmo is free for research purposes only. For more details, type license.cosmo(). Type citation('cosmo') for details on how to cite cosmo in publications.

> pseq <- getSeq(FaFile('./data/tair10chr.fasta'), as(as(bpeaksDF, "RangedData"), "GRanges"))
> names(pseq) <- paste(bpeaksDF$space, bpeaksDF$start, sep="_")
> write.XrStringSet(pseq[1:8], ".\results\pseq.fasta") # Note: reduced to 8 sequences to run quickly.
> res <- cosmo(seqs="./results/pseq.fasta", silent=TRUE)
> plot(res)
```
Exercise 2: Motif Enrichment Analysis

Task 1  Extract from the cosmo result stored in res the motif occurrence patterns and generate with them a position weight matrix using the PWM function from Biostrings.

Task 2  Enumerate the motif matches in the peak sequences and the entire genome using Biostring’s countPWM function.

Task 3  Determine which sequence type, peak or genome, shows more matches per 1kbp sequence for this motif.

Task 4  Homework: write a function for computing enrichment p-values for motif matches based on the hypergeometric distribution.

<table>
<thead>
<tr>
<th>seq</th>
<th>pos</th>
<th>orient</th>
<th>motif</th>
<th>prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chr1_8301</td>
<td>67</td>
<td>-1</td>
<td>AAAGAAATAGGG</td>
<td>1.0000000</td>
</tr>
<tr>
<td>Chr1_20901</td>
<td>63</td>
<td>-1</td>
<td>AAAGGGATTGGT</td>
<td>1.0000000</td>
</tr>
<tr>
<td>Chr1_37951</td>
<td>97</td>
<td>-1</td>
<td>AGTGAGTGTGGT</td>
<td>1.0000000</td>
</tr>
<tr>
<td>Chr1_54651</td>
<td>429</td>
<td>1</td>
<td>ATCACTTTTCACC</td>
<td>1.0000000</td>
</tr>
<tr>
<td>Chr1_61351</td>
<td>62</td>
<td>1</td>
<td>ACCAATCCCATT</td>
<td>1.0000000</td>
</tr>
<tr>
<td>Chr1_11151</td>
<td>134</td>
<td>-1</td>
<td>AGAGAGAGAGAGA</td>
<td>1.0000000</td>
</tr>
<tr>
<td>Chr1_16601</td>
<td>90</td>
<td>1</td>
<td>ACCACTCTCCTA</td>
<td>0.9998101</td>
</tr>
<tr>
<td>Chr1_13601</td>
<td>70</td>
<td>-1</td>
<td>AGCGAGATTGGT</td>
<td>0.9359452</td>
</tr>
</tbody>
</table>
> sessionInfo()

R version 2.15.0 (2012-03-30)
Platform: x86_64-unknown-linux-gnu (64-bit)
locale:
[1] C

attached base packages:
[1] grid      stats      graphics  grDevices utils     datasets  methods  base

other attached packages:
[1] cosmo_1.22.0 seqLogo_1.22.0 edgeR_2.6.1
[7] limma_3.12.0 org.Hs.eg.db_2.7.1 GO.db_2.7.1
[13] BSgenome.Ecoli.NCBI.20080805_1.3.17 multtest_2.12.0 Biobase_2.16.0
[19] caTools_1.12 bitops_1.0-4.1 gdata_2.8.2
[25] ShortRead_1.14.2 latticeExtra_0.6-19 RColorBrewer_1.0-5
[31] Rsamtools_1.8.4 Biostrings_2.24.1 GenomicRanges_1.8.4

loaded via a namespace (and not attached):
[1] MASS_7.3-18 RCurl_1.91-1 XML_3.9-4 annotate_1.34.0 genefilter_1.38.0 geneplotter_1.34.0
[12] xtable_1.7-0 zlibbioc_1.2.0