

Analysis of ChIP-Seq Data with R/Bioconductor

...

Thomas Girke

December 15, 2013

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

References

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

References

Outline

Introduction

ChIP-Seq Technology

Bioconductor Resources for ChIP-Seq

ChIP-Seq Analysis

Sample Data

Aligning Short Reads

Coverage Data

Peak Calling

Annotating Peaks

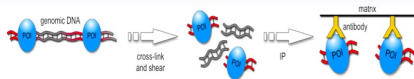
Differential Binding Analysis

View Peaks in Genome Browser

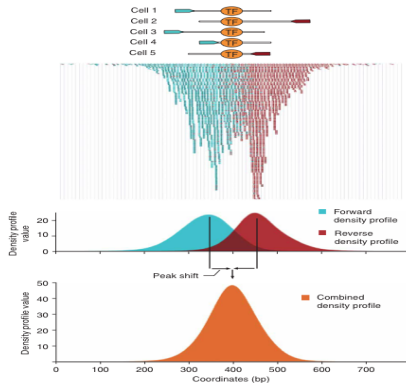
Common Motifs in Peak Sequences

References

ChIP-Seq Technology



ChIP-Seq



ChIP-Seq Workflow

- Read mapping
- Peak calling
- Peak annotation/filtering
- Differential binding 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
- ...

Pepke et al. (2009) Computation for ChIP-seq and RNA-seq studies. Nat Methods 6, 22-32. [Link](#)

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

References

General Purpose Resources for ChIP-Seq Analysis in R

- GenomicRanges [Link](#): high-level infrastructure for range data
- Rsamtools [Link](#): BAM support
- DiffBind [Link](#): Differential binding analysis of ChIP-Seq peak data
- 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
- MotifDb [Link](#): Collection of motif databases
- motifStack [Link](#): Stacked logo plots
- PWMEnrich [Link](#): PWM enrichment analysis
- Bioc Workflow [Link](#): Overview of resources
- ...

Peak Calling in R

- BayesPeak [Link](#): hidden Markov models (HMM) and Bayesian statistics
- PICS [Link](#): probabilistic inference
- MOSAiCS [Link](#): model-based analysis of ChIP-Seq data
- iSeq [Link](#): Hidden Ising Models
- ChIPseqR [Link](#)
- CSAR [Link](#): tests based on Poisson distribution
- ChIP-Seq [Link](#)
- SPP [Link](#)
- ...

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

References

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

References

Data Sets and Experimental Variables

To make the following sample code work, please follow these instructions:

- Download and unpack the sample data [Link](#). Afterwards direct your R session into the new project directory called Rchipseq.
- The sample data contains slimmed down versions of four FASTQ files from the ChIP-Seq experiment published by Kaufmann et al. (2010) (GSE20176 [Link](#)), a shortened GFF3 annotation file [Link](#) and the corresponding reference genome from *Arabidopsis thaliana*.
- Start the analysis by opening in your R session the Rchipseq.R script [Link](#) which contains the code shown in this slide show in pure text format.

The FASTQ files are organized in the provided `targets.txt` file. This is the only file in this analysis workflow that needs to be generated manually, e.g. in a spreadsheet program. To import `targets.txt`, we run the following commands from R:

```
> targets <- read.delim("./data/targets.txt")  
> targets
```

	FileName	SampleName	Factor	Factor_long
1	SRR038845.fastq	AP1IND1	sig1	sig1_long
2	SRR038846.fastq	AP1IND2	sig1	sig1_long
3	SRR038848.fastq	AP1UIND1	bgr1	bgr1_long
4	SRR038850.fastq	AP1UIND2	bgr1	bgr1_long

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

References

Align Reads and Output Indexed Bam Files

Note: Rsubread is for Linux and OS X only. Windows users want to skip the mapping step.

```
> 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")
> input <- paste("./data/", targets$FileName, sep="")
> output <- paste("./results/", targets$FileName, ".sam", sep="")
> reference <- "./results/tair10chr.fasta"
> for(i in seq(along=targets$FileName)) {
+     align(index=reference, readfile1=input[i], output_file=output[i], nthreads=1)
+     asBam(file=output[i], destination=gsub(".sam", "", output[i]), overwrite=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

References

Important Resources for ChIP-Seq Analysis

Coverage and peak slicing

```
> library(rtracklayer); library(GenomicRanges); library(Rsamtools)
> samples <- as.character(targets$FileName)
> samplespath <- paste("./results/", samples, ".bam", sep="")
> aligns <- readGAlignmentsFromBam(samplespath[3])
> cov <- coverage(aligns)
> islands <- slice(cov, lower = 15)
> islands[[1]][1:12,]
```

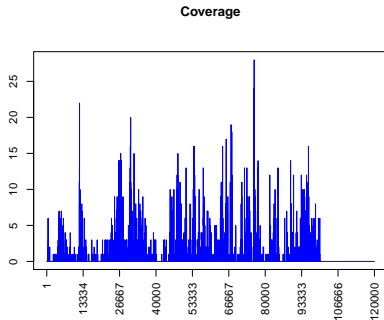
Views on a 30427671-length Rle subject

views:

	start	end	width	
[1]	11998	12003	6	[22 22 22 22 22 22]
[2]	27064	27077	14	[15 15 15 15 15 15 15 15 15 15 15 15 15 15]
[3]	30619	30623	5	[16 16 16 16 16]
[4]	30709	30725	17	[16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 16 20 20]
[5]	31956	31975	20	[15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15]
[6]	48015	48029	15	[15 15 15 15 15 15 15 15 15 15 15 15 15 15 15]
[7]	53839	53844	6	[16 16 16 16 16 16]
[8]	54031	54066	36	[16 16]
[9]	64384	64394	11	[16 16 16 16 16 16 16 16 16 15 15]
[10]	65692	65719	28	[15 15]
[11]	67505	67508	4	[19 19 19 19]
[12]	67743	67770	28	[18 18]

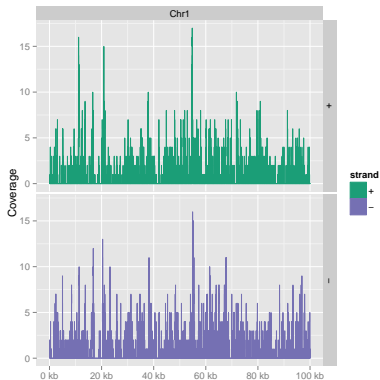
Coverage Plot with Base Graphics

```
> plotCovBase <- 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)  
+ }  
> plotCovBase(mycov=cov, mychr="Chr1", mypos=c(1,120000)) # Remember: read data is truncated to first 100kbp
```



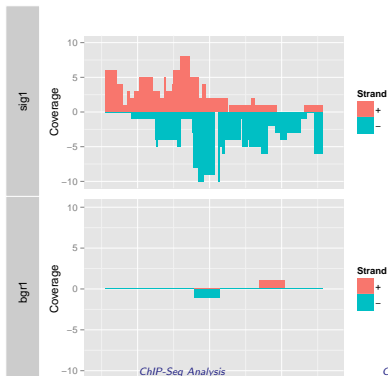
Coverage Plot with *ggbio*

```
> library(ggbio)
> myloc <- c("Chr1", 1, 100000)
> ga <- readGAlignmentsFromBam(samplespath[1], use.names=TRUE, param=ScanBamParam(which=GRanges(myloc[1], IRanges(1, 100000))))
> autoplot(ga, aes(color = strand, fill = strand), facets = strand ~ seqnames, stat = "coverage")
```



Mirror Plot for +/- Coverage of Sample Peak

```
> library(ggplot2)
> plotCov <- function(bampath, mychr, mystart, myend, ymin, ymax) {
+   ga <- readGAlignmentsFromBam(bampath, use.names=TRUE, param=ScanBamParam(which=GRanges(mychr, IRanges(
+   pos <- as.numeric(coverage(ga[strand(ga)=="+"])[[mychr]])[mystart:myend])
+   pos <- data.frame(Chr=rep(mychr, length(pos)), Strand=rep("+", length(pos)), Position=mystart:myend, Coverage=pos)
+   neg <- as.numeric(coverage(ga[strand(ga)=="-"])[[mychr]])[mystart:myend]
+   neg <- data.frame(Chr=rep(mychr, length(neg)), Strand=rep("-", length(neg)), Position=mystart:myend, Coverage=neg)
+   covdf <- rbind(pos, neg)
+   ggplot(covdf, aes(Position, Coverage, fill=Strand)) + geom_bar(stat="identity", position="identity")
+ }
> p1 <- plotCov(bampath=samplespath[1], mychr="Chr1", mystart=16656, myend=16956, ymin=-10, ymax=10)
> p2 <- plotCov(bampath=samplespath[3], mychr="Chr1", mystart=16656, myend=16956, ymin=-10, ymax=10)
> tracks(sig1=p1, bgr1=p2)
```



Import Aligned Read Data

Import aligned reads (bam files) and extend to 200bp mappings. Note: to remove PCR duplicates and secondary alignments, one can flag them by assigning the `flag_pcr` filter to the `param` argument of the `readGAlignmentsFromBam` function. Subsequently, one would filter on the `flag` column of the alignment object.

```
> chip_signal_list <- sapply(samplespath, list)
> for(i in seq(along=samplespath)) {
+   flag_pcr <- ScanBamParam(flag=scanBamFlag(isDuplicate=FALSE, isNotPrimaryRead=FALSE), what=c("qual",
+   # To use flag_pcr, assign it to param argument in next line.
+   aligns <- readGAlignmentsFromBam(samplespath[i], param=NULL)
+   chip_signal_list[[i]] <- as(aligns, "GRanges")
+ }
> chip_signal_list[["./results/SRR038845.fastq.bam"]][1:4,]
```

GRanges with 4 ranges and 0 metadata columns:

	seqnames	ranges	strand
	<Rle>	<IRanges>	<Rle>
[1]	Chr1	[121, 156]	-
[2]	Chr1	[121, 156]	-
[3]	Chr1	[216, 251]	+
[4]	Chr1	[295, 330]	+

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

References

Peak Calling with BayesPeak

Compute coverage and call peaks

```
> library(BayesPeak)
> sig <- readGAlignmentsFromBam(samplespath[1])
> bgr <- readGAlignmentsFromBam(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)

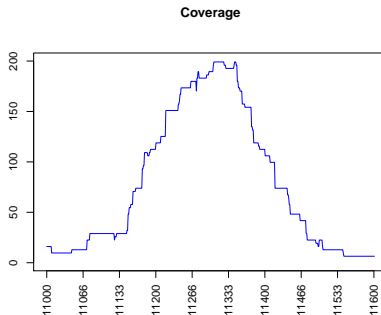
.....

> # 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=
> bgrcovDF <- rangeCoverage(summaryFct=viewMeans, myname="bgr_", peaksIR=bpeaks[,1:3], sig=bgr, readextend=
> bpeaksDF <- cbind(bpeaks, sigcovDF[,-1], bgrcovDF[,-1])
> bpeaksDF[1:4,]
```

	space	start	end	width	PP	sig_cov	sig_cov.pos	sig_cov.neg	bgr_cov	bgr_cov.pos	bgr_cov.neg
1	Chr1	8301	8401	101	0.5072765	0.1907938	0.1271959	0.06359793	0.039605386	0.039605386	0.000000000
2	Chr1	11151	11351	201	0.9998914	0.6391434	0.5912077	0.04793576	0.009950607	0.009950607	0.000000000
3	Chr1	13601	13751	151	0.9998761	0.4041207	0.2339646	0.17015606	0.092718570	0.079473060	0.013245510
4	Chr1	16601	16951	351	0.9999604	0.4941070	0.2379034	0.25620365	0.011396422	0.005698211	0.005698211

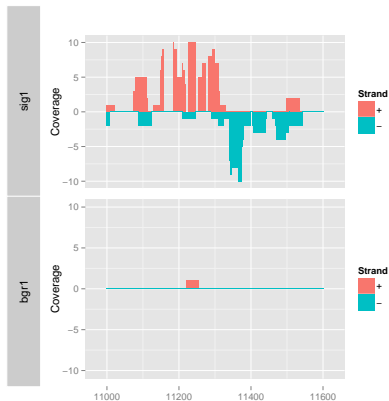
Coverage Plot

```
> plotCovBase(mycov=chip_signal_list[[1]], mychr="Chr1", mypos=c(11000, 11600), ylim=c(0, 200))
```



Coverage Plot with plotCov

```
> p1 <- plotCov(bamopath=samplespath[1], mychr="Chr1", mystart=11000, myend=11600, ym=10)  
> p2 <- plotCov(bamopath=samplespath[3], mychr="Chr1", mystart=11000, myend=11600, ym=10)  
> tracks(sig1=p1, bgr1=p2)
```



Identify Common Peaks Among Two Methods

Compares results from simple cutoff method with BayesPeak results

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

	space	start	end	width	PP	sig_cov	sig_cov.pos	sig_cov.neg	bgr_cov	bgr_cov.pos	bgr_cov.neg
1	Chr1	8301	8401	101	0.5072765	0.1907938	0.1271959	0.06359793	0.039605386	0.039605386	0.000000000
2	Chr1	11151	11351	201	0.9998914	0.6391434	0.5912077	0.04793576	0.009950607	0.009950607	0.000000000
3	Chr1	13601	13751	151	0.9998761	0.4041207	0.2339646	0.17015606	0.092718570	0.079473060	0.013245510
4	Chr1	16601	16951	351	0.9999604	0.4941070	0.2379034	0.25620365	0.011396422	0.005698211	0.005698211

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:

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

	PeakID	space	start	end	score_position	score	length
chr1_3132	chr1_3132	Chr1	3094	3172	3132	4.656515	79
chr1_8365	chr1_8365	Chr1	8222	8425	8365	11.046212	204
chr1_11298	chr1_11298	Chr1	11149	11440	11298	52.109592	292
chr1_13686	chr1_13686	Chr1	13602	13756	13686	5.341907	155

```
> # Import olRanges function, which accepts two GRanges (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

References

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(values(gff)[,"type"]=="chromosome"),]))
> subgene_index <- which(values(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 metadata columns:

	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
[4]	Chr1	[23146, 31227]	*	TAIR10	gene

seqlengths:

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

```
> ids <- values(gffsub)[, "group"]
> gffgene <- gffsub
> gffsub <- split(gffsub, ids) # Coerce to GRangesList
```

Annotate Peaks with ChIPpeakAnno

```
> library(ChIPpeakAnno)
> annoRD <- unlist(gffsub)
> names(annoRD) <- gsub(".*=", "", values(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]
```

	space	start	end	width	names	peak	strand	feature	start_position	end_position	insideFeature
1	Chr1	8301	8401	101	01	AT1G01020	01	+ AT1G01020	5928	8737	inside
2	Chr1	11151	11351	201	02	AT1G01030	02	+ AT1G01030	11649	13714	upstream
3	Chr1	13601	13751	151	03	AT1G01030	03	+ AT1G01030	11649	13714	overlapEnd
4	Chr1	16601	16951	351	04	AT1G01030	04	+ AT1G01030	11649	13714	downstream

```
> bpeaksDF[1:4,]
```

	space	start	end	width	PP	sig_cov	sig_cov.pos	sig_cov.neg	bgr_cov	bgr_cov.pos	bgr_cov.neg
1	Chr1	8301	8401	101	0.5072765	0.1907938	0.1271959	0.06359793	0.039605386	0.039605386	0.000000000
2	Chr1	11151	11351	201	0.9998914	0.6391434	0.5912077	0.04793576	0.009950607	0.009950607	0.000000000
3	Chr1	13601	13751	151	0.9998761	0.4041207	0.2339646	0.17015606	0.092718570	0.079473060	0.013245510
4	Chr1	16601	16951	351	0.9999604	0.4941070	0.2379034	0.25620365	0.011396422	0.005698211	0.005698211

Alternative Peak Annotation Approach

Alternative approach using olRanges function

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

  space Qindex Sindex Qstart  Qend Sstart  Send OLstart  OLeid  OLlength  OLpercQ  OLpercS OLtype
1  Chr1     2      1  5928  8737   8301  8401   8301  8401     101 3.594306 100.00000 inside
2  Chr1     3      3 11649 13714 13601 13751 13601 13714     114 5.517909  75.49669 oldown

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

  space start  end width      names peak strand  feature start_position end_position insideFeature
2  Chr1 11151 11351   201 02 AT1G01030  02    + AT1G01030      11649      13714      upstream
5  Chr1 20901 21101   201 05 AT1G01040  05    + AT1G01040      23146      31227      upstream
```


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

References

Read Counting for Peaks

Store peak ranges in GRanges container

```
> peakranges <- GRanges(seqnames = Rle(bpeaksDF$space), ranges = IRanges(bpeaksDF$st  
+ strand = Rle(strand("*")), peakIDs=paste("peak", seq(along=bpeaksDF,
```

Count reads with summarizeOverlaps function from the GenomicRanges package.

```
> library(GenomicRanges)  
> bfl <- BamFileList(samplespath, yieldSize=50000, index=character())  
> countDF <- summarizeOverlaps(peakranges, bfl, mode="Union", ignore.strand=TRUE)  
> countDF <- assays(countDF)$counts  
> colnames(countDF) <- samples; rownames(countDF) <- values(peakranges)$peakIDs  
> 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	68	87	2	8

```
> write.table(countDF, "./results/countDF.xls", quote=FALSE, sep="\t", col.names = I  
> countDF <- read.table("./results/countDF.xls")
```

Simple RPKM Normalization

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

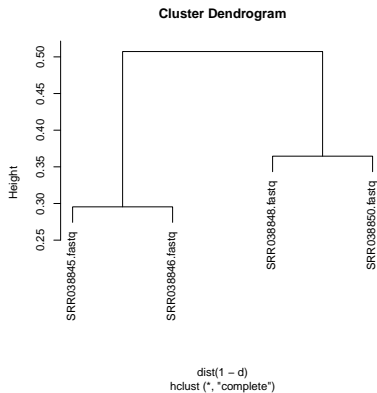
```
> returnRPKM <- function(counts, ranges) {  
+   peakLengthsInKB <- width(ranges)/1000 # Number of bases per sequence range in kbp  
+   millionsMapped <- sum(counts)/1e+06 # Factor for converting to million of mapped reads.  
+   rpm <- counts/millionsMapped # RPK: reads per kilobase of sequence range.  
+   rpkm <- rpm/peakLengthsInKB # 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,]
```

	SRR038845.fastq	SRR038846.fastq	SRR038848.fastq	SRR038850.fastq
peak_1	3860.370	12947.172	623.0761	1968.3877
peak_2	11638.726	12469.436	156.5440	1780.3626
peak_3	8893.905	3127.233	1458.6582	789.9622
peak_4	8392.852	9003.387	179.2897	906.2435

QC Check

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

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



Identify DiffPeaks with Simple Fold Change Method

Compute mean values for replicates

```
> 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	8403.771	1295.7319
peak_2	12054.081	968.4533
peak_3	6010.569	1124.3102
peak_4	8698.119	542.7666

Log2 fold changes

```
> 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	8403.771	1295.7319	2.697270
peak_2	12054.081	968.4533	3.637695
peak_3	6010.569	1124.3102	2.418461
peak_4	8698.119	542.7666	4.002300

```
> write.table(degs2fold, "../results/degs2fold", quote=FALSE, sep="\t", col.names = NA)
> degs2fold <- read.table("../results/degs2fold")
```

Identify DiffPeaks with DESeq Library

Raw count data are expected here!

```
> library(DESeq)
> countDF <- read.table("./results/countDF.xls")
> 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.
```

	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	68	87	2	8

```
> 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]
```

	id	baseMean	baseMeanA	baseMeanB	foldChange	log2FoldChange	pval	padj
2	peak_2	24.58011	7.378231	41.78199	5.662874	2.501534	0.0019269419	0.02055405
4	peak_4	30.15418	7.657406	52.65094	6.875820	2.781532	0.0002409555	0.01542115

Identify DiffPeaks with edgeR Library

Raw count data are expected here!

```
> library(edgeR)
> countDF <- read.table("./results/countDF.xls")
> 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

	logFC	logCPM	PValue	FDR
peak_32	3.829373	10.61003	2.299470e-10	1.494656e-08
peak_4	4.032856	10.70564	6.118597e-08	1.988544e-06
peak_43	3.651178	11.00799	1.227632e-07	2.659870e-06
peak_29	3.200007	11.25382	5.714735e-06	9.286445e-05

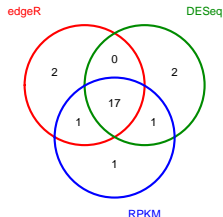
```
> edge <- as.data.frame(topTags(et, n=50000))
> edge2fold <- edge[edge$logFC >= 1 | edge$logFC <= -1,]
> edge2foldpadj <- edge2fold[edge2fold$FDR <= 0.01, ]
```

Merge Results and Compute Overlaps Among Methods

Here overlaps for 20 best ranking peaks of each method!

```
> 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] 0.9906593
> 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$log2ratio),][1:20,]))
> OList <- overLapper(setlist=setlist, sep="_", type="vennsets")
> counts <- sapply(OList$Venn_List, length)
> vennPlot(counts=counts)
```

Venn Diagram



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

References

Inspect Results in IGV

View peak₃ in IGV

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

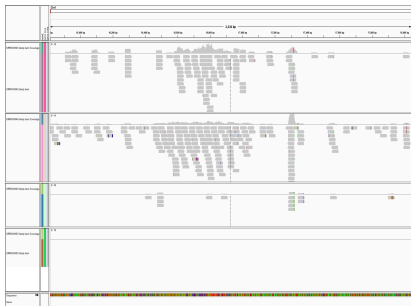
http://faculty.ucr.edu/~tgirke/HTML_Presentations/Manuals/Rngsapps/chipseqBioc2012/results/SRR038845.fastq.bam

http://faculty.ucr.edu/~tgirke/HTML_Presentations/Manuals/Rngsapps/chipseqBioc2012/results/SRR038846.fastq.bam

http://faculty.ucr.edu/~tgirke/HTML_Presentations/Manuals/Rngsapps/chipseqBioc2012/results/SRR038848.fastq.bam

http://faculty.ucr.edu/~tgirke/HTML_Presentations/Manuals/Rngsapps/chipseqBioc2012/results/SRR038850.fastq.bam

- To view peak₃, enter its coordinates Chr1:16656-16956 in position menu on top.



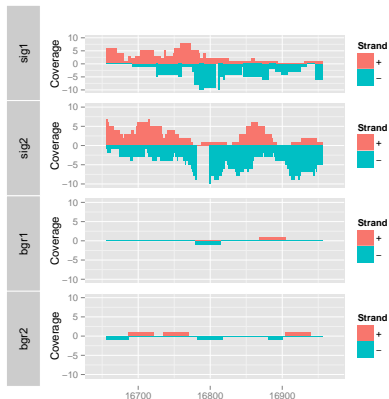
Controlling IGV from R

Create previous IGV session with required tracks automatically, and direct it to the desired position, here Chr1:16656-16956.

```
> library(SRAdb)
> startIGV("lm")
> sock <- IGVsocket()
> session <- IGVsession(files=samplespath,
+                       sessionFile="session.xml",
+                       genome="A. thaliana (TAIR10)")
> IGVload(sock, session)
> IGVgoto(sock, 'Chr1:16656-16956')
```

Plot Peak Programmatically with *ggbio*

```
> p1 <- plotCov(bamPath=samplesPath[1], mychr="Chr1", mystart=16656, myend=16956, ymin=-10, ymax=10)
> p2 <- plotCov(bamPath=samplesPath[2], mychr="Chr1", mystart=16656, myend=16956, ymin=-10, ymax=10)
> p3 <- plotCov(bamPath=samplesPath[3], mychr="Chr1", mystart=16656, myend=16956, ymin=-10, ymax=10)
> p4 <- plotCov(bamPath=samplesPath[4], mychr="Chr1", mystart=16656, myend=16956, ymin=-10, ymax=10)
> tracks(sig1=p1, sig2=p2, bgr1=p3, bgr2=p4)
```



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

References

Sequence Motifs Enriched in Peak Sequences I

Extract peak sequences and predict enriched motifs with BCRANK library

```
> library(Biostrings); library(seqLogo); library(BCRANK)
> pseq <- getSeq(FaFile("./data/tair10chr.fasta"), as(as(bpeaksDF, "RangedData"), "GRanges"))
> names(pseq) <- paste(bpeaksDF$space, bpeaksDF$start, sep="_")
> writeXStringSet(pseq[1:8], "./results/pseq.fasta") # Note: reduced to 8 sequences to run quickly.
> set.seed(0)
> BCRANKout <- bcrank("./results/pseq.fasta", restarts=25, use.P1=TRUE, use.P2=TRUE)

**** Running BCRANK on 8 regions, starting from HGRMHGHVSS ****
Iteration 1 - HGRMHGHVSS: 0
Scanning sequences.....
Computing scores.....

Iteration 2 - AGRMHGHVSS: 7.673544
Scanning sequences.....
Computing scores.....

Iteration 3 - AGAMHGHVSS: 10.23139
Scanning sequences.....
Computing scores.....

Iteration 4 - AGAAHGHVSS: 12.78924
Scanning sequences.....
Computing scores.....

Iteration 5 - AGAATGHVSS: 15.34709
Scanning sequences.....
Computing scores.....

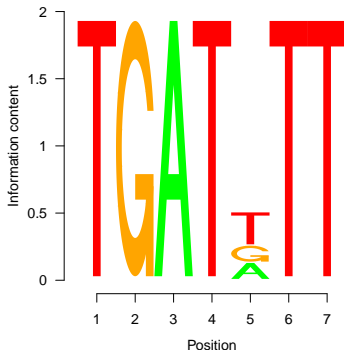
Iteration 6 - AGAATGTVSS: 17.90494
Scanning sequences.....
Computing scores.....

Iteration 7 - AGAATGTASS: 20.46278
```

Sequence Motifs Enriched in Peak Sequences II

Plot BCRANK result

```
> topMotif <- toptable(BCRANKout, 1)
> weightMatrix <- pwm(topMotif, normalize = FALSE)
> weightMatrixNormalized <- pwm(topMotif, normalize = TRUE)
> seqLogo(weightMatrixNormalized)
```



Exercise 2: Motif Enrichment Analysis

- Task 1** Extract from the BCRANK result stored in `weightMatrix` 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.

Session Information

```
> sessionInfo()
```

```
R version 3.0.2 (2013-09-25)
```

```
Platform: x86_64-unknown-linux-gnu (64-bit)
```

```
locale:
```

```
[1] C
```

```
attached base packages:
```

```
[1] grid      parallel  stats      graphics  utils      datasets  grDevices  methods    base
```

```
other attached packages:
```

```
[1] BCRANK_1.24.0                seqLogo_1.28.0                edgeR_3.4.0
 [7] GenomicFeatures_1.14.0      limma_3.18.3                  org.Hs.eg.db_2.10.1
[13] AnnotationDbi_1.24.0       BSgenome.Ecoli.NCBI.20080805_1.3.17 multtest_2.18.0
[19] chipseq_1.12.0             BSgenome_1.30.0              ShortRead_1.20.0
[25] ggplot2_0.9.3.1           Rsamtools_1.14.1            Biostrings_2.30.1
[31] IRanges_1.20.1            BiocGenerics_0.8.0
```

```
loaded via a namespace (and not attached):
```

```
[1] Hmisc_3.12-2                MASS_7.3-29                   RColorBrewer_1.0-5          RCurl_1.95-4.1             Vari
[10] cluster_1.14.4             colorspace_1.2-4             dichromat_2.0-0            digest_0.6.3              gene
[19] labeling_0.2               latticeExtra_0.6-26         munsell_0.4.2              plyr_1.8                   proto
[28] stats4_3.0.2               stringr_0.6.2               survival_2.37-4            tools_3.0.2               xtbl
```

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

References

References I

Kaufmann, K., Wellmer, F., Muiño, J. M., Ferrier, T., Wuest, S. E., Kumar, V., Serrano-Mislata, A., Madueño, F., Krajewski, P., Meyerowitz, E. M., Angenent, G. C., Riechmann, J. L., Apr 2010. Orchestration of floral initiation by APETALA1. *Science* 328 (5974), 85–89.

URL <http://www.hubmed.org/display.cgi?uids=20360106>

Pepke, S., Wold, B., Mortazavi, A., Nov 2009. Computation for ChIP-seq and RNA-seq studies. *Nat Methods* 6 (11 Suppl), 22–32.

URL <http://www.hubmed.org/display.cgi?uids=19844228>