RNA-Seq and ChIP-Seq Analysis with R and Bioconductor

Overview

Thomas Girke

December 11, 2011

Overview

RNA-Seq Analysis

ChIP-Seq Analysis

Outline

Packages for RNA-Seq and 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
- BayesPeak Link: ChIP-Seq peak caller

Outline

Data Sets and Experimental Variables

- To make the following sample code work, please download and unpack the sample data Link in the directory of your current R session.
- It contains four simplified alignment files from RNA-Seq experiment SRA023501 Link and a shortened GFF Link to allow fast analysis on a laptop.
- The alignments were created by aligning the reads with Bowtie against the Arabidopsis reference genome.
- Note: usually, the aligned reads would be stored in BAM format and then imported into R with the `readBamGappedAlignments` function (see below)!

This information could be imported from an external targets file

```
> targets <- data.frame(Samples=c("AP3 domain, flower stage 4", "AP3 domain, flower stage 4",
+                       "Translatome, flower stage 4", "Translatome, flower stage 4"),
+                       Factor=c("AP3", "AP3", "TRL", "TRL"),
+                       Fastq=c("SRR064154", "SRR064155", "SRR064166", "SRR064167"))
> targets
```

```
                      Samples Factor     Fastq
1  AP3 domain, flower stage 4    AP3 SRR064154
2  AP3 domain, flower stage 4    AP3 SRR064155
3 Translatome, flower stage 4    TRL SRR064166
4 Translatome, flower stage 4    TRL SRR064167
```

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 values:
      seqnames         ranges strand |     type   source
         <Rle>      <IRanges>  <Rle> | <factor> <factor>
  [1]        1 [ 3631,  5899]      * |     gene   TAIR10
  [2]        1 [ 5928,  8737]      * |     gene   TAIR10
  [3]        1 [11649, 13714]      * |     gene   TAIR10
  [4]        1 [23146, 31227]      * |     gene   TAIR10
  ---
  seqlengths:
             1           2           3           4           5  chloroplast mitochondria
      30427671    19698289    23459830    18585056    26975502      154478      366924

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

Read Counting per Annotation Range

Number of reads overlapping gene ranges

```
> samples <- as.character(targets$Fastq)
> samplespath <- paste("./data/", samples, sep="")
> countDF <- data.frame(row.names=ids)
> for(i in samplespath) {
+         # aligns <- readBamGappedAlignments(i) # Substitute next two lines with this one.
+         aligns <- read.table(i, header=TRUE)
+         aligns <- GRanges(seqnames=aligns$seqnames, IRanges(aligns$start, aligns$end), strand=aligns$stra
+         counts <- countOverlaps(gffsub, aligns)
+         countDF <- cbind(countDF, counts)
+ }
> colnames(countDF) <- samples
> rownames(countDF) <- gsub(".*=", "", rownames(countDF))
> countDF[1:4,]
```

	SRR064154	SRR064155	SRR064166	SRR064167
AT1G01010	50	24	64	76
AT1G01020	132	79	89	59
AT1G01030	5	0	16	15
AT1G01040	491	347	330	374

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

Simple RPKM Normalization

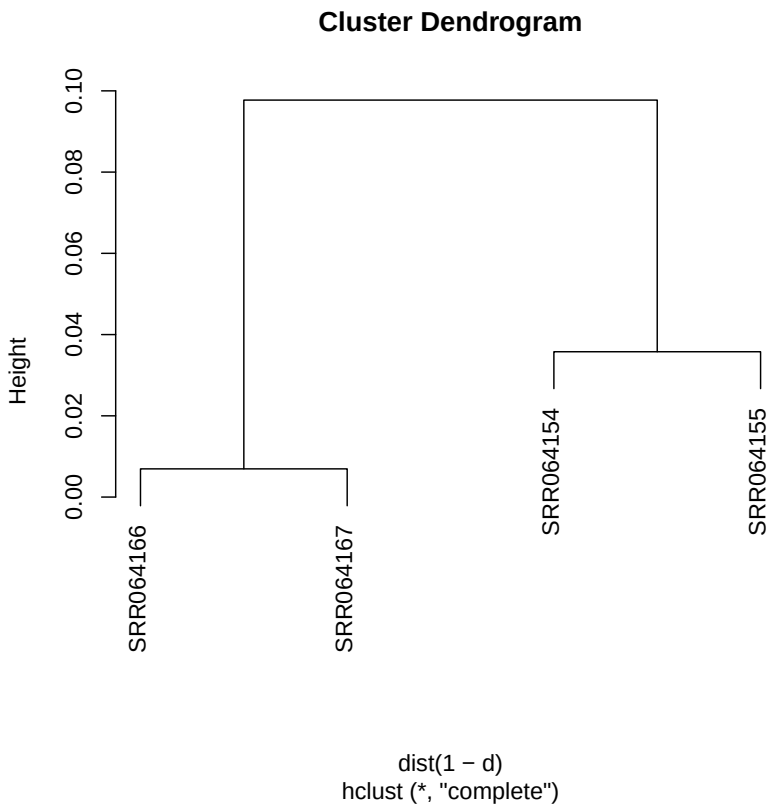
RPKM: reads per kilobase of exon model per million mapped reads

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

           SRR064154 SRR064155 SRR064166 SRR064167
AT1G01010  41.094413  18.76055 345.02646 347.28848
AT1G01020  87.602174  49.86428 387.42765 217.69927
AT1G01030   4.513225   0.00000  94.73198  75.27872
AT1G01040 113.294785  76.15166 499.46150 479.80419
```

QC Check

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

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



Identify DEGs with Simple Fold Change Method

Compute mean values for replicates

```
> source("http://faculty.ucr.edu/~tgirke/Documents/R_BioCond/My_R_Scripts/colAg.R")
> countDFrpkm_mean <- colAg(myMA=countDFrpkm, group=c(1,1,2,2), myfct=mean)
> countDFrpkm_mean[1:4,]
```

```
          SRR064154_SRR064155 SRR064166_SRR064167
AT1G01010           29.927480           346.15747
AT1G01020           68.733226           302.56346
AT1G01030            2.256612            85.00535
AT1G01040           94.723224           489.63284
```

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,]
```

```
          SRR064154_SRR064155 SRR064166_SRR064167 log2ratio
AT1G01010           29.927480           346.15747 -3.531886
AT1G01020           68.733226           302.56346 -2.138158
AT1G01030            2.256612            85.00535 -5.235323
AT1G01040           94.723224           489.63284 -2.369910
```

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

Identify DEGs with DESeq Library

Raw count data are expected here!

```
> library(DESeq)

locfit 1.5-6          2010-01-20

> countDF <- read.table("./data/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.

          SRR064154 SRR064155 SRR064166 SRR064167
AT1G01010        50        24        64        76
AT1G01020       132        79        89        59
AT1G01030         5         0        16        15
AT1G01040       491       347       330       374

> cds <- estimateSizeFactors(cds) # Estimates library size factors from count data. Alternatively, one can
> cds <- estimateDispersions(cds) # Estimates the variance within replicates
> res <- nbinomTest(cds, "TRL", "AP3") # Calls DEGs with nbinomTest
> res <- na.omit(res)
> res2fold <- res[res$log2FoldChange >= 1 | res$log2FoldChange <= -1,]
> res2foldpadj <- res2fold[res2fold$padj <= 0.01, ]
> res2foldpadj[1:4,1:8]

          id     baseMean   baseMeanA    baseMeanB foldChange log2FoldChange         pval         padj
6  AT1G01050    565.38535   858.46503 2.723057e+02 0.31720066      -1.656532 1.106761e-09 1.291221e-08
7  AT1G01060    299.47779   423.36877 1.755868e+02 0.41473725      -1.269730 4.613776e-05 2.202029e-04
8  AT1G01070     29.30875    53.97968 4.637819e+00 0.08591786      -3.540898 2.499200e-05 1.353315e-04
16 AT2G01010 223420.41841 59874.64881 3.869662e+05 6.46293875       2.692190 4.142756e-06 2.718684e-05
```

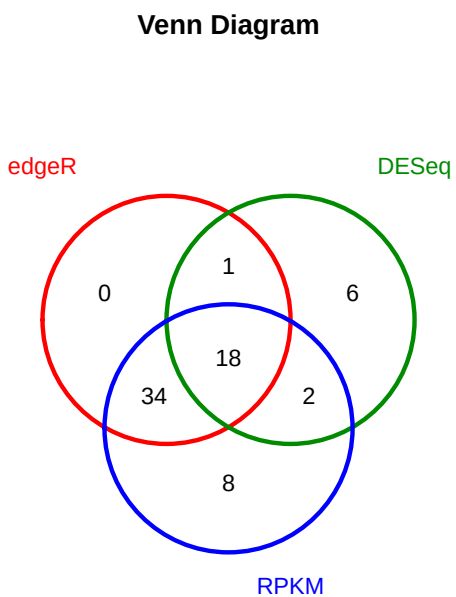
Identify DEGs with edgeR Library

Raw count data are expected here!

```
> library(edgeR)
> countDF <- read.table("./data/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("TRL", "AP3")) # Computes exact test for the negative binomial distribution.

Comparison of groups:  AP3 - TRL

> topTags(et, n=4)

Comparison of groups:  AP3-TRL
             logConc     logFC      P.Value          FDR
AT4G00050 -11.014277 -5.730672 2.094199e-60 3.015647e-58
AT1G01050  -8.825097 -4.324321 3.166161e-54 2.279636e-52
AT3G01120  -6.711507 -4.068848 5.305841e-52 2.546804e-50
AT1G01060  -9.630091 -3.885589 1.245939e-37 4.485379e-36

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

Merge Results and Compute Overlaps Among Methods

```
> 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] 0.9989686

> source("http://faculty.ucr.edu/~tgirke/Documents/R_BioCond/My_R_Scripts/overLapper.R")
> setlist <- list(edgeR=rownames(edge2foldpadj), DESeq=as.character(res2foldpadj[,1]), RPKM=rownames(degs2f
> OLlist <- overLapper(setlist=setlist, sep="_", type="vennsets")
> counts <- sapply(OLlist$Venn_List, length)
> vennPlot(counts=counts)
```



Enrichment of GO Terms in DEG Sets

GO Term Enrichment Analysis

```
> library(GOstats); library(GO.db); library(ath1121501.db)
> geneUniverse <- rownames(countDF)
> geneSample <- res2foldpadj[,1]
> params <- new("GOHyperGParams", geneIds = geneSample, universeGeneIds = geneUniverse,
+         annotation="ath1121501", ontology = "MF", pvalueCutoff = 0.5,
+         conditional = FALSE, testDirection = "over")
> hgOver <- hyperGTest(params)
> summary(hgOver)[1:4,]

      GOMFID      Pvalue OddsRatio ExpCount Count Size
1 GO:0016168 0.009102704 16.857143 1.153846     4    5                                                  chloroph
2 GO:0046906 0.009102704 16.857143 1.153846     4    5                                                 tetrapyr
3 GO:0015077 0.023160627  8.285714 1.384615     4    6 monovalent inorganic cation transmembrane transport
4 GO:0015078 0.023160627  8.285714 1.384615     4    6                 hydrogen ion transmembrane transport

> htmlReport(hgOver, file = "data/MyhyperGresult.html")
```

Coverage Data for Identifying New Genes

Very similar to peak calling in ChIP-Seq section!

Compute coverage and call peaks

```
> cov <- coverage(aligns)
> cov[1:2]

SimpleRleList of length 2
$`1`
'integer' Rle of length 49278 with 3530 runs
  Lengths: 1055    2    3    3   30    2    3    3 2559    3   11   11    2    2    4    2    3    3    4
  Values :    0    1    2    3    4    3    2    1    0    2    3    4    5    6    7    8    9    7    6
$`2`
'integer' Rle of length 48814 with 5429 runs
  Lengths:  1217    38   656    38   293    76   343    38   534    38    20    24    14     7     2     1
  Values :     0     1     0     1     0     1     0     1     0     1     0     1     2     1     2     3
```

Outline

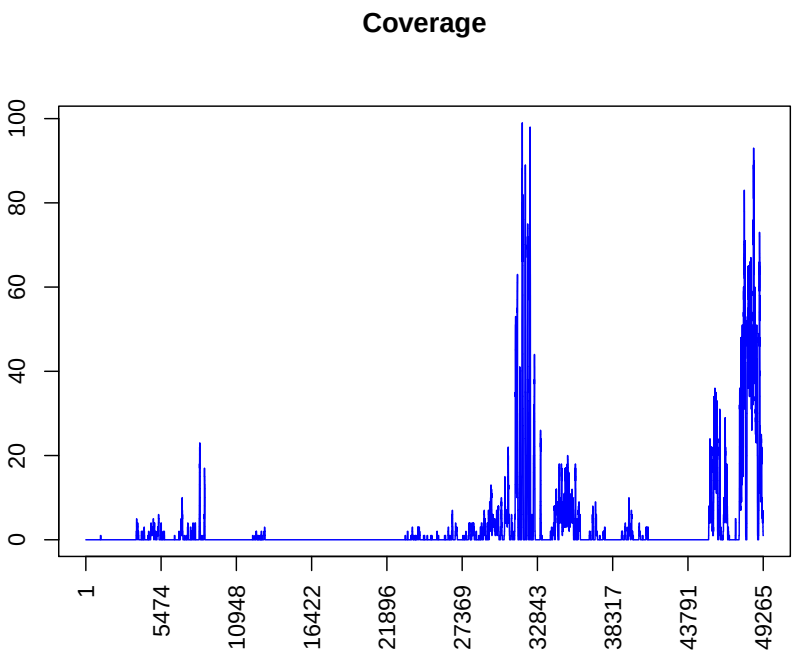
Important Resources for ChIP-Seq Analysis

Coverage and peak slicing

```
> aligns <- read.table(samplespath[3], header=TRUE)
> aligns <- GRanges(seqnames=aligns$seqnames, IRanges(aligns$start, aligns$end), strand=aligns$strand)
> cov <- coverage(aligns)
> islands <- slice(cov, lower = 50)
> islands[[1]]

Views on a 49265-length Rle subject
views:
     start   end width
 [1] 31244 31259    16 [50 51 51 51 51 51 51 51 51 51 51 51 51 52 53 53]
 [2] 31366 31367     2 [50 51]
 [3] 31370 31395    26 [50 52 52 52 52 53 54 54 55 54 57 57 57 59 59 59 60 61 61 61 62 63 63 63 63 63]
 [4] 31716 31796    81 [52 53 54 55 57 63 64 66 67 68 72 73 73 74 74 75 75 75 81 88 88 82 83 84 83 93 99 9
 [5] 31946 31979    34 [50 51 63 65 67 68 68 70 70 71 74 75 76 76 76 78 78 81 82 88 89 87 87 84 80 78 76 6
 [6] 32114 32154    41 [50 51 52 53 62 64 65 70 67 71 72 72 69 70 72 74 74 75 75 75 75 75 69 71 63 57 59 5
 [7] 32295 32331    37 [51 58 59 68 71 73 73 77 80 80 81 82 83 92 93 94 95 95 95 98 95 95 94 90 86 86 71 7
 [8] 47741 47743     3 [51 51 51]
 [9] 47824 47824     1 [50]
 ...   ...   ...   ... ...
[28] 48468 48468     1 [50]
[29] 48528 48528     1 [51]
[30] 48530 48613    84 [50 53 54 52 52 52 55 59 58 58 60 63 68 69 70 71 76 75 72 73 73 70 71 71 71 70 89 8
[31] 48648 48666    19 [51 55 53 53 52 52 56 58 57 59 60 59 59 55 54 53 53 53 50]
[32] 48668 48669     2 [50 50]
[33] 48671 48675     5 [51 53 51 51 52]
[34] 48679 48679     1 [50]
[35] 48793 48793     1 [51]
[36] 48978 49017    40 [52 54 54 58 58 60 60 60 65 66 65 63 64 62 59 65 64 73 72 72 68 68 67 67 64 63 62 5
```

Coverage Plot

```
> 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="1", mypos=c(1,length(cov[["1"]])))
```



Process All ChIP-Seq Data

Import and read extension to 200bp

```
> chip_signal_list <- sapply(samples, list)
> for(i in seq(along=samplespath)) {
+         aligns <- read.table(samplespath[i], header=TRUE)
+         aligns <- GRanges(seqnames=aligns$seqnames, IRanges(aligns$start, aligns$end), strand=aligns$stra
+         chip_signal_list[[i]] <- aligns
+ }
> chip_signal_list[["SRR064154"]][1:4,]

GRanges with 4 ranges and 0 elementMetadata values:
      seqnames       ranges strand
         <Rle>    <IRanges>  <Rle>
  [1]        2 [8047, 8084]      -
  [2]        2 [6365, 6402]      -
  [3]        2 [6699, 6736]      -
  [4]        2 [6535, 6572]      +
  ---
  seqlengths:
             1            2            3            4            5  chloroplast mitochondria
            NA           NA           NA           NA           NA           NA           NA

> chip_signal_list <- sapply(names(chip_signal_list), function(x) resize(chip_signal_list[[x]], width = 200
> for(i in names(chip_signal_list)) start(chip_signal_list[[i]])[start(chip_signal_list[[i]]) < 0 ] <- 1
```

Peak Calling by Coverage Value

Compute coverage and call peaks

```
> 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]]) * normfac
> chip_signal_list[["SRR064154"]][1:2,]

SimpleRleList of length 2
$`1`
'numeric' Rle of length 49456 with 1579 runs
  Lengths:               889               176                 7                17               176
  Values :                 0 0.649478436341696  1.29895687268339  1.94843530902509  1.29895687268339 0.649
$`2`
'numeric' Rle of length 41441 with 6072 runs
  Lengths:              1395               200               166                 4                40
  Values :                 0 0.649478436341696                 0  1.94843530902509  2.59791374536678  4.54

> chip_peak_list <- sapply(names(chip_signal_list), function(x) slice(chip_signal_list[[x]], lower=5))
> chip_peak_list[[1]][[1]][1:3]

Views on a 49456-length Rle subject
views:
    start  end width
[1]  3639 3777   139 [5.845306 5.845306 5.845306 5.845306 5.845306 5.845306 5.845306 5.845306 5.845306 5.8
[2]  5113 5312   200 [5.195827 5.195827 5.195827 5.195827 5.195827 5.195827 5.195827 5.195827 5.195827 5.1
[3]  5480 5640   161 [7.793741 7.793741 7.793741 7.793741 7.793741 7.793741 7.793741 7.793741 7.793741 7.7
```

Peak Calling with BayesPeak

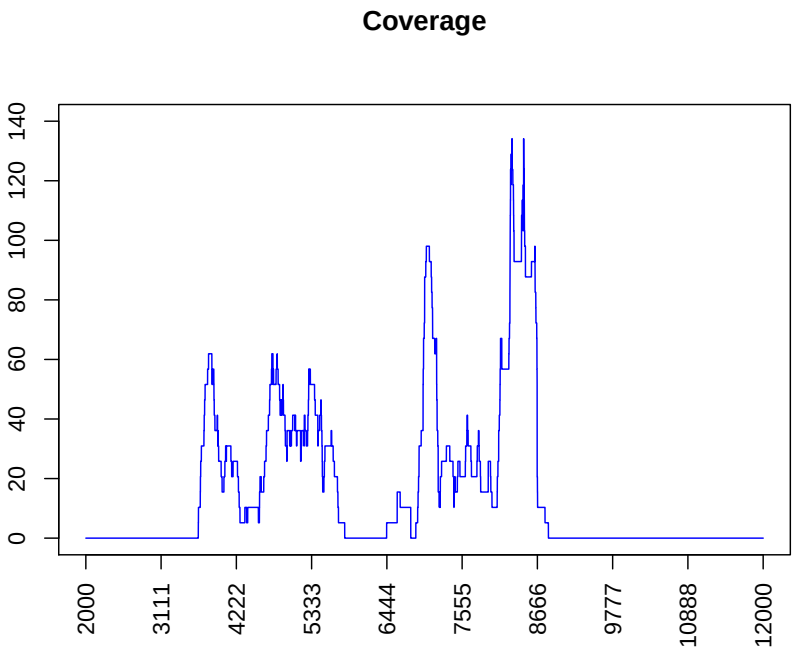
Compute coverage and call peaks

```
> library(BayesPeak)
> sig <- read.table(samplespath[1], header=TRUE)
> bgr <- read.table(samplespath[3], header=TRUE)
> sig <- RangedData(space=sig[,1], IRanges(start=sig[,2], end=sig[,3]), strand=sig[,5])
> bgr <- RangedData(space=bgr[,1], IRanges(start=bgr[,2], end=bgr[,3]), strand=bgr[,5])
> raw.output <- bayespeak(treatment=sig, control=bgr)

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

> # 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.jo
> bpeaks <- as.data.frame(summarise.peaks(raw.output, method = "lowerbound"))
> bpeaksIR <- IRanges(start=bpeaks[, 2], end=bpeaks[, 3])
> bpeaksIR <- lapply(unique(bpeaks[,1]), function(x) IRanges(start=bpeaks[bpeaks[,1]==x,2], end=bpeaks[bpea
> names(bpeaksIR) <- unique(names(sig))
> source("http://faculty.ucr.edu/~tgirke/Documents/R_BioCond/My_R_Scripts/chipseqFct.R") # Imports the rang
> sigcovDF <- rangeCoverage(summaryFct=viewMaxs, myname="sig_", peaksIR=bpeaksIR, sig=sig, readextend=1, no
> bgrcovDF <- rangeCoverage(summaryFct=viewMaxs, myname="bgr_", peaksIR=bpeaksIR, sig=bgr, readextend=1, no
> 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	1	6966	7066	101	0.7362081	3.247392	3.247392	1.298957	15.477400	15.477400	0.000000
11	1	8566	8716	151	0.9996578	5.845306	2.597914	5.845306	30.954801	5.159133	30.954801
12	1	29216	29566	351	0.9999153	5.195827	5.195827	1.948435	20.636534	20.636534	10.318267
13	1	29616	29716	101	0.9850553	3.247392	3.247392	1.298957	5.159133	5.159133	5.159133

Coverage Plot

```
> plotCov(mycov=chip_signal_list[[3]], mychr="1", mypos=c(2000,12000), ylim=c(0,140))
```



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"))
>    # 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,]
```

	space	start	end	width	PP	sig_cov	sig_cov.pos	sig_cov.neg	bgr_cov	bgr_cov.pos	bgr_cov.neg
1	1	6966	7066	101	0.7362081	3.247392	3.247392	1.298957	15.477400	15.477400	0.000000
11	1	8566	8716	151	0.9996578	5.845306	2.597914	5.845306	30.954801	5.159133	30.954801
12	1	29216	29566	351	0.9999153	5.195827	5.195827	1.948435	20.636534	20.636534	10.318267
13	1	29616	29716	101	0.9850553	3.247392	3.247392	1.298957	5.159133	5.159133	5.159133

Annotate Peaks with ChIPpeakAnno

```
> 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]
```

	space	start	end	width	names	peak	strand	feature	start_position	end_position	insideFeature
1	1	6966	7066	101	001 AT1G01020	001	+	AT1G01020	5928	8737	inside
2	1	8566	8716	151	002 AT1G01020	002	+	AT1G01020	5928	8737	inside
3	1	29216	29566	351	003 AT1G01046	003	+	AT1G01046	28500	28706	downstream
4	1	29616	29716	101	004 AT1G01046	004	+	AT1G01046	28500	28706	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	1	6966	7066	101	0.7362081	3.247392	3.247392	1.298957	15.477400	15.477400	0.000000
11	1	8566	8716	151	0.9996578	5.845306	2.597914	5.845306	30.954801	5.159133	30.954801
12	1	29216	29566	351	0.9999153	5.195827	5.195827	1.948435	20.636534	20.636534	10.318267
13	1	29616	29716	101	0.9850553	3.247392	3.247392	1.298957	5.159133	5.159133	5.159133

Session Information

```
> sessionInfo()

R version 2.14.0 (2011-10-31)
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] ChIPpeakAnno_2.2.0                gplots_2.10.1                     KernSmooth_2.23-6
 [7] limma_3.10.0                      org.Hs.eg.db_2.6.4                BSgenome.Ecoli.NCBI.20080805_
[13] BSgenome_1.22.0                   ShortRead_1.12.0                  latticeExtra_0.6-19
[19] ath1121501.db_2.6.3               org.At.tair.db_2.6.4              GO.db_2.6.1
[25] graph_1.32.0                      Category_2.20.0                   AnnotationDbi_1.16.0
[31] lattice_0.20-0                    akima_0.5-4                       Biobase_2.14.0
[37] IRanges_1.12.1                    rtracklayer_1.14.1                RCurl_1.7-0

loaded via a namespace (and not attached):
 [1] GSEABase_1.16.0    MASS_7.3-16        RBGL_1.30.0        XML_3.4-3          annotate_1.32.0    genefi
[12] zlibbioc_1.0.0
```