Analysis of RNA-Seq Data with R/Bioconductor

...

Thomas Girke

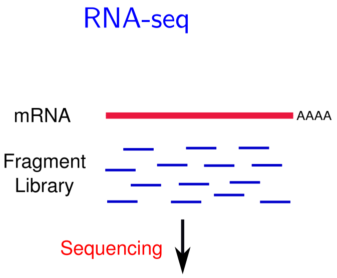
December 8, 2012

Outline

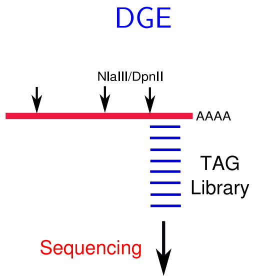
Packages for RNA-Seq Analysis in R

- 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 DEG analysis
- edgeR Link: RNA-Seq DEG analysis
- DEXSeq Link: RNA-Seq Exon analysis

RNA-Seq versus DGE



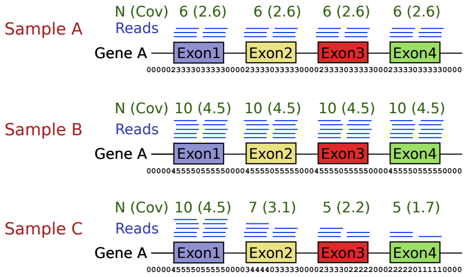
1. Alternative splicing
2. Limited expression profiling
3. SNP detection
4. Many other applications



1. Expression profiling

$\rightarrow$ more appropriate for many biosamples

Identification of Differentially Expressed Genes



	N Raw	RPKM*	Cov*
Sample A	24	24.0	2.6
Sample B	40	40.0	4.5
Sample C	27	27.0	1.1

RPKM*: for 1kb gene and library with 1Mio reads; Cov*: average coverage

Normalization often by library size.

RNA-Seq Analysis Workflow

- Read mapping
- Counting reads overlapping with genes
- Analysis of differentially expressed genes (DEGs)
- Clustering of co-expressed genes
- Gene set/GO term enrichment analysis

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

  Samples Factor           Fastq
1 AP3_fl4    AP3 SRR064154.fastq
2 AP3_fl4    AP3 SRR064155.fastq
3  Tl_fl4    TRL SRR064166.fastq
4  Tl_fl4    TRL SRR064167.fastq
```

Outline

Align Reads and Output Indexed Bam Files

Note: this steps requires the command-line tool `bowtie2` Link. If it is not available on a system then one can skip this mapping step and download the pre-generated Bam files from here: Link

```
> library(Rsamtools)
> dir.create("results") # Note: all output data will be written to directory 'results'
> system("bowtie2-build ./data/tair10chr.fasta ./data/tair10chr.fasta") # Build indexed reference genome
> targets <- read.delim("./data/targets.txt") # Import experiment design information
> targets
> input <- paste("./data/", targets[,3], sep="")
> output <- paste("./results/", targets[,3], ".sam", sep="")
> reference <- "./results/tair10chr.fasta"
> for(i in seq(along=targets[,3])) {
+         command <- paste("bowtie2 -x ./data/tair10chr.fasta -U", input[i], "-S", output[i])
+         system(command)
+         asBam(file=output[i], destination=gsub(".sam", "", output[i]), overwrite=TRUE, indexDestination=T
+         unlink(output[i])
+ }
```

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 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
   100000 100000 100000 100000 100000 100000 100000

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

Read Counting per Annotation Range - Old

Number of reads overlapping gene ranges

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

          SRR064154.fastq SRR064155.fastq SRR064166.fastq SRR064167.fastq
AT1G01010              52              26              60              75
AT1G01020             149              79              84              66
AT1G01030               5               1              13              14
AT1G01040             492             355             303             365

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

Read Counting per Annotation Range - New

The summarizeOverlaps function from the GenomicRanges is easier to use and provides more options. See here Link for details.

```
> bfl <- BamFileList(samplespath, index=character())
> countDF2 <- summarizeOverlaps(gffsub, bfl, mode="Union", ignore.strand=TRUE)
> countDF2 <- assays(countDF2)$counts
> rownames(countDF2) <- gsub(".*=", "", rownames(countDF2))
> countDF2[1:4,]

          SRR064154.fastq SRR064155.fastq SRR064166.fastq SRR064167.fastq
AT1G01010              52              26              60              75
AT1G01020             149              79              84              66
AT1G01030               5               1              13              14
AT1G01040             480             346             282             336
```

Simple RPKM Normalization

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

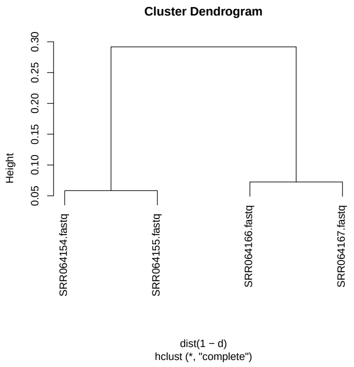
```
> returnRPKM <- function(counts, gffsub) {
+         geneLengthsInKB <- sum(width(gffsub))/1000 # Number of bases per exonRanges element in kbp
+         millionsMapped <- sum(counts)/1e+06 # Factor for converting to million of mapped reads.
+         rpm <- counts/millionsMapped # RPK: reads per 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.fastq SRR064155.fastq SRR064166.fastq SRR064167.fastq
AT1G01010       38.961967      18.3057160        287.8127       271.20512
AT1G01020       90.147145      44.9126457        325.3615       192.71199
AT1G01030        4.114449       0.7732458         68.4867        55.59924
AT1G01040      103.494741      70.1709530        408.0534       370.54857
```

QC Check

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

```
> d <- cor(countDF, method="spearman")
> 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.fastq_SRR064155.fastq SRR064166.fastq_SRR064167.fastq
AT1G01010                       28.633841                       279.50891
AT1G01020                       67.529895                       259.03674
AT1G01030                        2.443848                        62.04297
AT1G01040                       86.832847                       389.30097
```

Log2 fold changes

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

          SRR064154.fastq_SRR064155.fastq SRR064166.fastq_SRR064167.fastq
AT1G01010                       28.633841                       279.50891
AT1G01020                       67.529895                       259.03674
AT1G01030                        2.443848                        62.04297
AT1G01040                       86.832847                       389.30097
          log2ratio
AT1G01010  3.287101
AT1G01020  1.939559
AT1G01030  4.666042
AT1G01040  2.164573

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

Identify DEGs with DESeq Library

Raw count data are expected here!

```
> 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.

          SRR064154.fastq SRR064155.fastq SRR064166.fastq SRR064167.fastq
AT1G01010              52              26              60              75
AT1G01020             149              79              84              66
AT1G01030               5               1              13              14
AT1G01040             492             355             303             365

> cds <- estimateSizeFactors(cds) # Estimates library size factors from count data. Alternatively, one can
> cds <- estimateDispersions(cds) # Estimates the variance within replicates
> res <- nbinomTest(cds, "AP3", "TRL") # Calls DEGs with nbinomTest
> res <- na.omit(res)
> res2fold <- res[res$log2FoldChange >= 1 | res$log2FoldChange <= -1,]
> res2foldpadj <- res2fold[res2fold$padj <= 0.05, ]
> res2foldpadj[1:4,1:8]

          id   baseMean  baseMeanA baseMeanB foldChange log2FoldChange
6  AT1G01050  602.73819  270.09411  935.3823   3.463172       1.792094
7  AT1G01060  302.65623  168.64177  436.6707   2.589339       1.372584
8  AT1G01070   30.06197    5.62154   54.5024   9.695281       3.277283
12 AT1G01100 6353.10140 4226.29888 8479.9039   2.006461       1.004653
           pval         padj
6  5.840333e-13 6.891592e-12
7  2.433388e-06 1.511262e-05
8  4.413539e-05 2.003068e-04
12 4.243839e-07 2.782072e-06
```

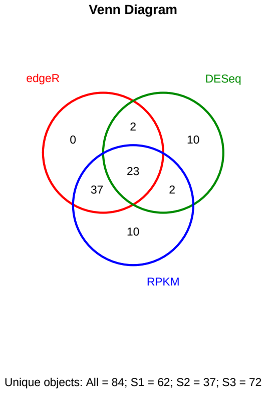
Identify DEGs with edgeR Library

Raw count data are expected here!

```
> 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("AP3", "TRL")) # Computes exact test for the negative binomial distribution.
> topTags(et, n=4)

Comparison of groups:  TRL-AP3
              logFC   logCPM       PValue          FDR
AT4G00050  5.433513 10.35929 7.946221e-49 1.144256e-46
AT1G01050  4.203520 12.08703 1.013791e-45 7.299293e-44
AT3G01120  3.865095 14.07493 5.930030e-41 2.846415e-39
ATMG00030 -3.745470 10.89011 2.594715e-34 9.340974e-33

> 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

```
> 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] 0.9990604

> 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:0008324 0.004436386      15.6 2.303797     6    7
2 GO:0015075 0.004436386      15.6 2.303797     6    7
3 GO:0015077 0.004436386      15.6 2.303797     6    7
4 GO:0015078 0.004436386      15.6 2.303797     6    7
                                                   Term
1                      cation transmembrane transporter activity
2                         ion transmembrane transporter activity
3 monovalent inorganic cation transmembrane transporter activity
4                hydrogen ion transmembrane transporter activity
```

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

Outline

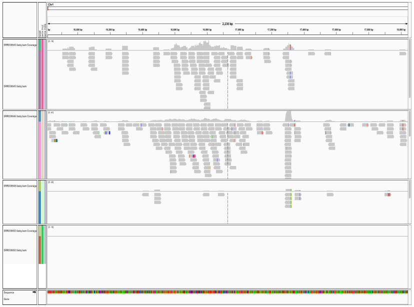
Inspect Results in IGV

View results 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/Workshop_Dec_6_10_2012/Rrnaseq/results/SRR064154.fast
http://faculty.ucr.edu/~tgirke/HTML_Presentations/Manuals/Workshop_Dec_6_10_2012/Rrnaseq/results/SRR064155.fast
http://faculty.ucr.edu/~tgirke/HTML_Presentations/Manuals/Workshop_Dec_6_10_2012/Rrnaseq/results/SRR064166.fast
http://faculty.ucr.edu/~tgirke/HTML_Presentations/Manuals/Workshop_Dec_6_10_2012/Rrnaseq/results/SRR064167.fast

- To view area of interest, enter its coordinates `Chr1:49,457-51,457` in position menu on top.



Analysis of Differential Exon Usage with DEXSeq

Number of reads overlapping gene ranges

```
> source("data/Fct/gffexonDEXSeq.R")
> gffexonDEXSeq <- exons2DEXSeq(gff=gff)
> ids <- as.character(elementMetadata(gffexonDEXSeq)[, "ids"])
> countDFdex <- data.frame(row.names=ids)
> for(i in samplespath) {
+         aligns <- readBamGappedAlignments(i) # Substitute next two lines with this one.
+         counts <- countOverlaps(gffexonDEXSeq, aligns)
+         countDFdex <- cbind(countDFdex, counts)
+ }
> colnames(countDFdex) <- samples
> countDFdex[1:4,1:2]

                                                      SRR064154.fastq
Parent=AT1G01010:E001__Chr1_3631_3913_+_Parent=AT1G01010.1               2
Parent=AT1G01010:E002__Chr1_3996_4276_+_Parent=AT1G01010.1               2
Parent=AT1G01010:E003__Chr1_4486_4605_+_Parent=AT1G01010.1               3
Parent=AT1G01010:E004__Chr1_4706_5095_+_Parent=AT1G01010.1               6
                                                      SRR064155.fastq
Parent=AT1G01010:E001__Chr1_3631_3913_+_Parent=AT1G01010.1               4
Parent=AT1G01010:E002__Chr1_3996_4276_+_Parent=AT1G01010.1               1
Parent=AT1G01010:E003__Chr1_4486_4605_+_Parent=AT1G01010.1               3
Parent=AT1G01010:E004__Chr1_4706_5095_+_Parent=AT1G01010.1               1

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

Analysis of Differential Exon Usage with DEXSeq

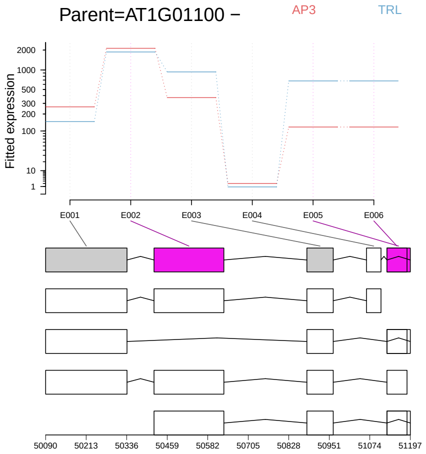
Identify genes with differential exon usage

```
> library(DEXSeq)
> samples <- as.character(targets$Factor); names(samples) <- targets$Fastq
> countDFdex[is.na(countDFdex)] <- 0
> ## Construct ExonCountSet from scratch
> exset <- newExonCountSet2(countDF=countDFdex) # fData(exset)[1:4,]
> ## Performs normalization
> exset <- estimateSizeFactors(exset)
> ## Evaluate variance of the data by estimating dispersion using Cox-Reid (CR) likelihood estimation
> exset <- estimateDispersions(exset)
> ## Fits dispersion-mean relation to the individual CR dispersion values
> exset <- fitDispersionFunction(exset)
> ## Performs Chi-squared test on each exon and Benjmini-Hochberg p-value adjustment for mutliple testing
> exset <- testForDEU(exset)
> ## Estimates fold changes of exons
> exset <- estimatelog2FoldChanges(exset)
> ## Obtain results in data frame
> deuDF <- DEUresultTable(exset)
> ## Count number of genes with differential exon usage
> table(tapply(deuDF$padjust < 0.01, geneIDs(exset), any))

FALSE  TRUE
   20     1
```

DEXSeq Plots

Sample plot showing fitted expression of exons

```
> plotDEXSeq(exset, "Parent=AT1G01100", displayTranscripts=TRUE, expression=TRUE, legend=TRUE)
> ## Generate many plots and write them to results directory
> mygeneIDs <- unique(as.character(na.omit(deuDF[deuDF$geneID %in% unique(deuDF$geneID),])[,"geneID"]))
> DEXSeqHTML(exset, geneIDs=mygeneIDs, path="results", file="DEU.html")
```



Session Information

```
> sessionInfo()

R version 2.15.2 (2012-10-26)
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)
locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

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

other attached packages:
 [1] DEXSeq_1.4.0         xtable_1.7-0         ath1121501.db_2.8.0
 [4] org.At.tair.db_2.8.0 GO.db_2.8.0          GOstats_2.24.0
 [7] RSQLite_0.11.2       DBI_0.2-5            graph_1.36.1
[10] Category_2.24.0      AnnotationDbi_1.20.3 edgeR_3.0.4
[13] limma_3.14.3         DESeq_1.10.1         lattice_0.20-10
[16] locfit_1.5-8         Biobase_2.18.0       Rsamtools_1.10.2
[19] Biostrings_2.26.2    rtracklayer_1.18.1   GenomicRanges_1.10.5
[22] IRanges_1.16.4       BiocGenerics_0.4.0

loaded via a namespace (and not attached):
 [1] annotate_1.36.0       AnnotationForge_1.0.3 biomaRt_2.14.0
 [4] bitops_1.0-4.2        BSgenome_1.26.1       genefilter_1.40.0
 [7] geneplotter_1.36.0    grid_2.15.2           GSEABase_1.20.1
[10] hwriter_1.3           parallel_2.15.2       plyr_1.7.1
[13] RBGL_1.34.0           RColorBrewer_1.0-5    RCurl_1.95-3
[16] splines_2.15.2        statmod_1.4.16        stats4_2.15.2
[19] stringr_0.6.1         survival_2.36-14      tools_2.15.2
[22] XML_3.95-0.1          zlibbioc_1.4.0
```