Analysis of RNA-Seq Data with R/Bioconductor

... 

Thomas Girke

December 6, 2014
Overview

RNA-Seq Analysis
  Quality Report
  Aligning Short Reads
  Counting Reads per Feature
  DEG Analysis
  GO Analysis
  View Results in IGV & ggbio
  Differential Exon Usage

References
Outline

Overview

RNA-Seq Analysis
  Quality Report
  Aligning Short Reads
  Counting Reads per Feature
  DEG Analysis
  GO Analysis
  View Results in IGV & ggbio
  Differential Exon Usage

References
RNA-Seq Technology

1. mRNA Isolation
2. Illumina Sequencing
3. Align Sequences against Genome
4. Generate Sequence Counts for all Genes in Genome

Gene A
Gene B
Sample 1
Sample 2

Gene A:
Gene B:

Sample 1
Sample 2

Gene A: $\frac{30}{10} = 3$ fold change
Gene B: $\frac{10}{5} = 2$ fold change
Analysis Workflow of RNA-Seq Gene Expression Data

1. Alignment of RNA reads to reference
   - Reference can be genome or transcriptome.

2. Count reads overlapping with annotation features of interest
   - Most common: counts for exonic gene regions, but many viable alternatives exist here: counts per exons, genes, introns, etc.

3. Normalization
   - Main adjustment for sequencing depth and compositional bias.

4. Identification of Differentially Expressed Genes (DEGs)
   - Identification of genes with significant expression differences.
   - Identification of expressed genes possible for strongly expressed ones.

5. Specialty applications
   - Splice variant discovery (semi-quantitative), gene discovery, antisense expressions, etc.

6. Cluster Analysis
   - Identification of genes with similar expression profiles across many samples.

7. Enrichment Analysis of Functional Annotations
   - Gene ontology analysis of obtained gene sets from steps 5-6.
Important Aspects in RNA-Seq Analysis

- Alignment reference
  - Genome
  - Transcript models
  - Both
- How to quantify expression?
  - Read count per range
  - Coverage statistics per range
- What features?
  - Genes, transcript models, exons
- Alternative splicing
  - Often restricted to splice junction analysis
  - Objective: discovery vs. quantification
Important Considerations for NGS Alignments

- In NGS we usually want to find the **origin of reads** (NG sequences) in a reference genome or transcriptome. Thus, we are mostly interested in finding the best scoring or multiple best scoring locations for each read, but not lower scoring alternative solutions as in paralog/ortholog search applications.

- **Ambiguous mappings** should be removed, because there is no evidence for their origin. However, for certain applications one needs to include them, e.g. when mapping RNA-Seq reads against transcript sequences instead of genome.
Short Read Aligner for RNA-Seq

No special requirements for alignments with low number of variants

- ChIP-Seq
- RNA-Seq (if mapping against transcriptome or intron-less genome)
- Bis-Seq (with injected reference)
- ...

Variant tolerant aligners to account for mismatches and indels

- VAR-Seq
- Bis-Seq (without injected reference)
- ...

Splice tolerant aligner to account for introns

- RNA-Seq (if mapping against genome with introns)
Sequence Alignment/Map (SAM/BAM) Format

SAM is a tab-delimited alignment format consisting of a header section (lines starting with @) and an alignment section with 12 columns. BAM is the compressed, indexed and binary version of this format.

The below sample alignment contains the following features: (1) bases in lower cases are clipped from the alignment; (2) read r001/1 and r001/2 constitute a read pair; (3) r003 is a chimeric read; (4) r004 represents a split alignment.

```
Coor 12345678901234 5678901234567890123456789012345678901234
ref AGCATGGTAGATAAGAGATAGCTGTGCTAGTAGGCAGTCAGCGCCAT
+r001/1 TTAGATAAAGGATACCTG
+r002 aaaaGATAGGAAGGATA
+r003 gcctAGCTAAA
+r004 ATAGCT..............TCAGC
-r003 ATAGCT..............TCAGC
-r001/2 ttagctTAGGC
```

SAM Format

```
r001 163 ref 7 30 8M2I4M1D3M = 37 39 TTAGATAAAGGATACCTG *
r002 0 ref 9 30 3S6M1P1I4M * 0 0 AAAAAAGATAAGGATA *
r003 0 ref 9 30 5S6M * 0 0 GCCTAAGCTAA * SA:Z:ref,9,+,5S6M,30,1;
r004 0 ref 16 30 6M14N5M * 0 0 ATAGCTTCAGC *
-r003 2064 ref 29 17 6H5M * 0 0 TAGGC * SA:Z:ref,29,+,6H5M,17,0;
r001 83 ref 37 30 9M = 7 -39 CAGCGGCAT *
```

For details see the SAM Format Specification [Link]
Normalization Required

Log ratio distributions (a and b) and MA plot (c) for two tissue samples (from Robinson and Oshlack, 2010).
Be Careful with RPKM/FPKM Values

RPKM Concept (FPKM is paired-end version of it)

- RPKM (FPKM): reads (fragments) per kp per million mapped reads
- The more we sequence, the more reads we expect from each gene. This is the most relevant correction of this method.
- Longer transcript are expected to generate more reads. The latter is only relevant for comparisons among different genes which we rarely perform!
- RPKM/FPKM are not suitable for statistical testing. Why? Consider the following example: in two libraries, each with one million reads, gene X may have 10 reads for treatment A and 5 reads for treatment B, while it is 100x as many after sequencing 100 millions reads from each library. In the latter case we can be much more confident that there is a true difference between the two treatments than in the first one. However, the RPKM values would be the same for both scenarios.
- Thus, RPKM/FPKM are useful for reporting expression values, but not for statistical testing!
Trimmed Mean of M Values (TMM) by Robinson and Oshlack (2010)

- Many normalization RNA-Seq normalization methods perform poorly on samples with extreme composition bias. For instance, in one sample a large number of reads comes from rRNAs while in another they have been removed more efficiently. Most scaling based methods, including RPKM and CPM, will underestimate the expression of weaker expressed genes in the presence of extremely abundant mRNAs (less sequencing real estate available for them). The TMM methods tries to correct this bias.

- Method implemented in edgeR library (Robinson et al., 2010).
Analysis of Differentially Expressed Genes (DEGs)

- Data is discrete, positively skewed
  ⇒ no (log-)normal model
- Small numbers of replicates
  ⇒ no rank based or permutation methods
- Sequencing depth (coverage) varies among samples
  ⇒ normalization
Requirements

- One would like to perform a t-test or something similar for each gene.
- t-test assumes normal distribution and no mean-variance dependence. Both are not appropriate assumptions for RNA-Seq data.
- Variance estimation and rank-order statistics is difficult on small sample numbers.

Statistical Testing

- Poisson distribution (initially used but not very common anymore)
- Most statistical methods for RNA-Seq DEG analysis use negative binomial distribution along with modified statistical tests based on that.
- The multiple testing issue is very similar as in microarray data analysis. Thus, most tools provide False Discovery Rates (FDRs), which are derived from p-values corrected for multiple testing using the Benjamini-Hochberg method.
- For variance estimation most methods borrow information across genes
Software for RNA-Seq DEG Analysis

- edgeR (Robinson et al., 2010)
- DESeq/DESeq2 (Anders and Huber, 2010)
- DEXSeq (Anders et al., 2012)
- limmaVoom
- Cuffdiff/Cuffdiff2 (Trapnell et al., 2013)
- PoissonSeq
- baySeq
- ...
Packages for RNA-Seq Analysis in R

- **GenomicRanges**: high-level infrastructure for range data
- **Rsamtools**: BAM support
- **rtracklayer**: Import/export of range and annotation data, interface to online genome browsers, etc.
- **DESeq**: RNA-Seq DEG analysis
- **DESeq2**: RNA-Seq DEG analysis
- **edgeR**: RNA-Seq DEG analysis
- **DEXSeq**: RNA-Seq Exon analysis
- **QuasR**: RNA-Seq workflows
- **systemPipeR**: NGS workflows and reports
Outline

Overview

RNA-Seq Analysis
  Quality Report
  Aligning Short Reads
  Counting Reads per Feature
  DEG Analysis
  GO Analysis
  View Results in IGV & ggbio
  Differential Exon Usage

References
Data Sets and Experimental Variables

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

- Download and unpack the sample data for this practical.
- Direct your R session to the resulting Rrnaseq directory. It contains 18 slimmed down FASTQ files (SRP010938) from *A. thaliana* (Howard et al., 2013). To minimize processing time, each FASTQ file has been subsetted to 90,000-100,000 randomly sampled reads that map to the first 100,000 nucleotides of each chromosome. The corresponding reference genome sequence (FASTA) and its GFF annotation files have been truncated accordingly.
- Start the analysis by opening in your R session the Rrnaseq.R script 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:

```r
> # download.file("http://biocluster.ucr.edu/~tgirke/HTML_Presentations/Manuals/Workshop_Dec_5_8_2014/Rrnaseq.zip", "Rrnaseq.zip")
> # unzip("Rrnaseq.zip")

> library(systemPipeR)
> args <- systemArgs(sysma="tophat.param", mytargets="targets.txt")
> targets <- read.delim("targets.txt", comment.char = ")
> targets[1:3,]

<table>
<thead>
<tr>
<th>FileName</th>
<th>SampleName</th>
<th>Factor</th>
<th>SampleLong</th>
<th>Experiment</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>./data/SRR446027_1.fastq</td>
<td>M1A</td>
<td>M1</td>
<td>Mock.1h.A</td>
<td>1</td>
<td>23-Mar-2012</td>
</tr>
<tr>
<td>./data/SRR446028_1.fastq</td>
<td>M1B</td>
<td>M1</td>
<td>Mock.1h.B</td>
<td>1</td>
<td>23-Mar-2012</td>
</tr>
<tr>
<td>./data/SRR446029_1.fastq</td>
<td>A1A</td>
<td>A1</td>
<td>Avr.1h.A</td>
<td>1</td>
<td>23-Mar-2012</td>
</tr>
</tbody>
</table>
```
Outline

Overview

RNA-Seq Analysis
  Quality Report
  Aligning Short Reads
  Counting Reads per Feature
  DEG Analysis
  GO Analysis
  View Results in IGV & ggbio
  Differential Exon Usage

References
The following shows how to create read quality reports with the seeFastq function from systemPipeR.

```r
> fplist <- seeFastq(fastq=filepath1(args), batchsize=10000, klength=8)
> pdf("./results/fastqReport.pdf", height=18, width=4*length(fplist))
> seeFastqPlot(fplist); dev.off()
```

Figure: QC report for 18 FASTQ files.
Overview

RNA-Seq Analysis
  Quality Report
  Aligning Short Reads
  Counting Reads per Feature
  DEG Analysis
  GO Analysis
  View Results in IGV & ggbio
  Differential Exon Usage

References
QuasR is an extremely versatile NGS mapping and postprocessing pipeline for RNA-Seq and many other application areas, such as BS-Seq, allele-specific RNA-Seq, etc. It uses Rbowtie for ungapped alignments and SpliceMap for spliced alignments.

1) Environment settings

```r
> library(QuasR)
> QuasR_samples <- targets[,1:2]; QuasR_samples[,1] <- gsub(".*", ",", QuasR_samples[,1])
> write.table(QuasR_samples, "data/QuasR_samples.txt", row.names=FALSE, quote=FALSE, sep=\"t\")
> sampleFile <- "./data/QuasR_samples.txt"
> genomeFile <- "./data/tair10.fasta"
> results <- "./results" # defines location where to write results
> cl <- makeCluster(1) # defines number of CPU cores to use
```

2) Single command to index reference, align all samples and generate BAM files.

```r
> proj <- qAlign(sampleFile, genome=genomeFile, maxHits=1, splicedAlignment=FALSE, alignmentsDir=results,
+                 clObj=cl, cacheDir=results)
> # Note: splicedAlignment should be set to TRUE when the reads are >=50nt long
> (alignstats <- alignmentStats(proj)[1:4,]) # Alignment summary report
```
**Align Reads Option 2: Rsubread**

*Rsubread* is an R/Bioc package that implements an extremely fast aligner for RNA-Seq data. It is currently only available for OS X and Linux, but not for Windows.

1. Index reference genome

```r
> library(Rsubread); library(systemPipeR)
> args <- systemArgs(sysma="rsubread.param", mytargets="targets.txt")
> buildindex(basename=reference(args), reference=reference(args)) # Build indexed reference genome
```

2. Align all FASTQ files with *Rsubread* in loop. Includes generation of indexed BAM files.

```r
> align(index=reference(args), readfile1=infile1(args), input_format="FASTQ",
+ output_file=outfile1(args), output_format="SAM", nthreads=8, indels=1, TH1=2
> for(i in seq(along=outfile1(args))) asBam(file=outfile1(args)[i], destination=gsub(".sam", ".bam", outfile1(args)[i]), overwrite=TRUE, indexDestination=TRUE)
> unlink(outfile1(args)); unlink(paste0(outfile1(args), ".indel"))
```
systemPipeR NGS workflow and report generation environment that can run command-line software on local computers and compute clusters. Note: this step requires the command-line tools tophat2/bowtie2.

(1) Index reference genome

```r
library(systemPipeR)
args <- systemArgs(sysma="tophat.param", mytargets="targets.txt")
moduleload(modules(args)) # Skip if a module system is not available
system("bowtie2-build ./data/tair10.fasta ./data/tair10.fasta")
```

(2) Align all FASTQ files with Bowtie2/Tophat2 on a single computer. Includes generation of indexed BAM files.

```r
bampaths <- runCommandline(args=args)
```

Missing alignment results (bam files): 0
Existing alignment results (bam files): 18

```r
bampaths

M1A M1B A1A A1B V1A V1B M6A M6B A6A A6B V6A V6B M12A M12B A12A A12B V12A V12B
"TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE" "TRUE"
```

(3) Alternatively, align all FASTQ files with Bowtie2/Tophat2 on many compute nodes in parallel. The following submits to Torque scheduler 18 processes each with 4 CPU cores.

```r
resources <- list(walltime="20:00:00", nodes=paste0("1:ppn=", cores(args)), memory="10gb")
reg <- clusterRun(args, conffile=".BatchJobs.R", template="torque.tmpl", Njobs=18, runid="01", resourceList=resources)
showStatus(reg)
```
Alignment Summary

The following provides an overview of the number of reads in each sample and how many of them aligned to the reference.

```r
> (read_statsDF <- alignStats(args=args))[1:8,]

<table>
<thead>
<tr>
<th>FileName</th>
<th>Nreads</th>
<th>Nalign</th>
<th>Perc_Aligned</th>
<th>Nalign_Primary</th>
<th>Perc_Aligned_Primary</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1A</td>
<td>M1A</td>
<td>96459</td>
<td>92.65698</td>
<td>89376</td>
<td>92.65698</td>
</tr>
<tr>
<td>M1B</td>
<td>M1B</td>
<td>98742</td>
<td>87.10984</td>
<td>86014</td>
<td>87.10984</td>
</tr>
<tr>
<td>A1A</td>
<td>A1A</td>
<td>94935</td>
<td>93.07421</td>
<td>88360</td>
<td>93.07421</td>
</tr>
<tr>
<td>A1B</td>
<td>A1B</td>
<td>94427</td>
<td>88.08074</td>
<td>83172</td>
<td>88.08074</td>
</tr>
<tr>
<td>V1A</td>
<td>V1A</td>
<td>99366</td>
<td>81.38498</td>
<td>80869</td>
<td>81.38498</td>
</tr>
<tr>
<td>V1B</td>
<td>V1B</td>
<td>97771</td>
<td>95.77175</td>
<td>93637</td>
<td>95.77175</td>
</tr>
<tr>
<td>M6A</td>
<td>M6A</td>
<td>98617</td>
<td>94.22006</td>
<td>92917</td>
<td>94.22006</td>
</tr>
<tr>
<td>M6B</td>
<td>M6B</td>
<td>90452</td>
<td>88.52651</td>
<td>80074</td>
<td>88.52651</td>
</tr>
</tbody>
</table>
```

> write.table(read_statsDF, "results/alignStats.xls", row.names=FALSE, quote=FALSE,

Analysis of RNA-Seq Data with R/Bioconductor
Outline

Overview

RNA-Seq Analysis

Quality Report
Aligning Short Reads
Counting Reads per Feature
DEG Analysis
GO Analysis
View Results in IGV & ggbio
Differential Exon Usage

References
Import Annotation Data from GFF

Annotation data from GFF

```r
> library(rtracklayer); library(GenomicRanges); library(Rsamtools)
> gff <- import.gff("./data/tair10.gff", asRangedData=FALSE)
> seqlengths(gff) <- end(ranges(gff[which(elementMetadata(gff)[,"type"] == "chromosome"),]))
> subgene_index <- which(elementMetadata(gff)[,"type"] == "exon")
> gffsub <- gff[subgene_index,] # Returns only gene ranges
> gffsub[1:4, c(2,5)]
```

GRanges object with 4 ranges and 2 metadata columns:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>type</th>
<th>group</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Rle&gt;</td>
<td>&lt;IRanges&gt;</td>
<td>&lt;Rle&gt;</td>
<td>&lt;factor&gt;</td>
<td>&lt;factor&gt;</td>
</tr>
<tr>
<td>[1]</td>
<td>Chr1 [3631, 3913]</td>
<td>+</td>
<td>exon</td>
<td>Parent=AT1G01010.1</td>
</tr>
<tr>
<td>[2]</td>
<td>Chr1 [3996, 4276]</td>
<td>+</td>
<td>exon</td>
<td>Parent=AT1G01010.1</td>
</tr>
<tr>
<td>[3]</td>
<td>Chr1 [4486, 4605]</td>
<td>+</td>
<td>exon</td>
<td>Parent=AT1G01010.1</td>
</tr>
<tr>
<td>[4]</td>
<td>Chr1 [4706, 5095]</td>
<td>+</td>
<td>exon</td>
<td>Parent=AT1G01010.1</td>
</tr>
</tbody>
</table>

seqinfo: 7 sequences from an unspecified genome

```r
> ids <- gsub("Parent=\|\|..*", ",", elementMetadata(gffsub)$group)
> gffsub <- split(gffsub, ids) # Coerce to GRangesList
```
Storing annotation ranges in TranscriptDb databases makes many operations more robust and convenient.

```r
> library(GenomicFeatures)
> txdb <- makeTranscriptDbFromGFF(file="data/tair10.gff",
+    format="gff3",
+    dataSource="TAIR",
+    species="Arabidopsis thaliana")
> saveDb(txdb, file="./data/tair10.sqlite")
> txdb <- loadDb("./data/tair10.sqlite")
> eByg <- exonsBy(txdb, by="gene")
```
Old Read Counting with countOverlaps

Number of reads overlapping gene ranges

```r
> args <- systemArgs(sysma="tophat.param", mytargets="targets.txt")
> countDF <- data.frame(row.names=names(eByg))
> for(i in outpaths(args)) {
+     aligns <- readGAlignmentsFromBam(i) # Substitute next two lines with this
+     counts <- countOverlaps(eByg, aligns, ignore.strand=TRUE)
+     countDF <- cbind(countDF, counts)
+ }
> colnames(countDF) <- names(outpaths(args))
> countDF[1:4,]

M1A  M1B  A1A  A1B  V1A  V1B  M6A  M6B  A6A  A6B  V6A  V6B  M12A  M12B  A12A  A12B  V12A
AT1G01010  28  128  99  87  183  118  22  19  77  23  148  201  60  68  66  33  113
AT1G01020  12  47  35  50  49  41  9  11  9  3  42  71  21  37  16  9  20
AT1G01030  19  51  36  33  47  78  5  8  4  4  18  73  16  15  7  5  34
AT1G01040  98  354  259  345  298  350  82  81  128  34  359  530  171  298  183  41  101
```
New Read Counting with `summarizeOverlaps`

The `summarizeOverlaps` function from the GenomicRanges package is easier to use, it provides more options and it is much more memory efficient. See here for details.

```r
> library("GenomicFeatures"); library(BiocParallel)
> txdb <- loadDb("./data/tair10.sqlite")
> eByg <- exonsBy(txdb, by=c("gene"))
> bfl <- BamFileList(outpaths(args), yieldSize=50000, index=character())
> multicoreParam <- MulticoreParam(workers=2); register(multicoreParam)
> counteByg <- bplapply(bfl, function(x) summarizeOverlaps(eByg, x, mode="Union", ignore.strand=TRUE, inter.feature=FALSE, singleEnd=TRUE))
> countDFeByg <- sapply(seq(along=counteByg), function(x) assays(counteByg[[x]])$counts)
> rownames(countDFeByg) <- names(rowData(counteByg[[1]])); colnames(countDFeByg) <- names(bfl)
> countDFeByg[1:4,1:12]

<table>
<thead>
<tr>
<th></th>
<th>M1A</th>
<th>M1B</th>
<th>A1A</th>
<th>A1B</th>
<th>V1A</th>
<th>V1B</th>
<th>M6A</th>
<th>M6B</th>
<th>A6A</th>
<th>A6B</th>
<th>V6A</th>
<th>V6B</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT1G01010</td>
<td>28</td>
<td>128</td>
<td>99</td>
<td>87</td>
<td>183</td>
<td>118</td>
<td>22</td>
<td>19</td>
<td>77</td>
<td>23</td>
<td>148</td>
<td>201</td>
</tr>
<tr>
<td>AT1G01020</td>
<td>12</td>
<td>47</td>
<td>35</td>
<td>50</td>
<td>49</td>
<td>41</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>3</td>
<td>42</td>
<td>71</td>
</tr>
<tr>
<td>AT1G01030</td>
<td>19</td>
<td>51</td>
<td>36</td>
<td>33</td>
<td>47</td>
<td>78</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>18</td>
<td>73</td>
</tr>
<tr>
<td>AT1G01040</td>
<td>98</td>
<td>354</td>
<td>259</td>
<td>345</td>
<td>298</td>
<td>350</td>
<td>82</td>
<td>81</td>
<td>128</td>
<td>34</td>
<td>359</td>
<td>530</td>
</tr>
</tbody>
</table>

> write.table(countDFeByg, "results/countDFeByg.xls", col.names=NA, quote=FALSE, sep="\t")
```
**Simple RPKM Normalization**

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

```r
> rpkmDFeByg <- apply(countDFeByg, 2, function(x) returnRPKM(counts=x, ranges=eByg))
> write.table(rpkmDFeByg, "results/rpkmDFeByg.xls", col.names=NA, quote=FALSE, sep="
> rpkmDFeByg[1:4,1:7]
```

<table>
<thead>
<tr>
<th></th>
<th>M1A</th>
<th>M1B</th>
<th>A1A</th>
<th>A1B</th>
<th>V1A</th>
<th>V1B</th>
<th>M6A</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT1G01010</td>
<td>2424.745</td>
<td>6201.798</td>
<td>5431.999</td>
<td>4225.653</td>
<td>11608.558</td>
<td>5482.332</td>
<td>1315.552</td>
</tr>
<tr>
<td>AT1G01020</td>
<td>988.799</td>
<td>2166.827</td>
<td>1827.306</td>
<td>2310.805</td>
<td>2957.618</td>
<td>1812.533</td>
<td>512.0905</td>
</tr>
<tr>
<td>AT1G01030</td>
<td>1457.938</td>
<td>2189.552</td>
<td>1750.267</td>
<td>1420.254</td>
<td>2641.816</td>
<td>3211.111</td>
<td>264.9311</td>
</tr>
<tr>
<td>AT1G01040</td>
<td>2290.597</td>
<td>4629.408</td>
<td>3835.648</td>
<td>4522.808</td>
<td>5102.206</td>
<td>4389.003</td>
<td>1323.4691</td>
</tr>
</tbody>
</table>
Read Counting with qCount from QuasR

QuasR does everything in one command.

```r
> countDF3 <- qCount(proj, txdb, reportLevel="gene", orientation="any")
> countDF3[1:4,1:12]
> write.table(countDF3, "results/countDFgene.xls", col.names=NA, quote=FALSE, sep="\\t")
```

RPKM: for QuasR results

```r
> rpkmDFgene <- t(t(countDF3[,,-1]/countDF3[,1] * 1000)/colSums(countDF3[,,-1]) *1e6)
```
Reproducibility Check by Sample-Wise Clustering

QC check of the sample reproducibility by computing a correlating matrix and plotting it as a tree. Note: the plotMDS function from edgeR is a more robust method for this task.

```r
> library(ape)
> d <- cor(rpkmDFeByg, method="spearman")
> hc <- hclust(dist(1-d))
> plot.phylo(as.phylo(hc), type="p", edge.col=4, edge.width=3, show.node.label=TRUE, no.margin=TRUE)
```
Sample-Wise Clustering with \texttt{rlog} Values

```r
> library(DESeq2)
> countDF <- as.matrix(read.table("./results/countDFeByg.xls"))
> colData <- data.frame(row.names=targets$SampleName, condition=targets$Factor)
> dds <- DESeqDataSetFromMatrix(countData = countDF, colData = colData, design = ~ condition)
> d <- cor(assay(rlog(dds)), method="spearman")
> hc <- hclust(dist(1-d))
> plot.phylo(as.phylo(hc), type="p", edge.col=4, edge.width=3, show.node.label=TRUE, no.margin=TRUE)
```
Exercise 1: *QuasR* with Antisense Read Counting

**Task 1** Align reads from all 4 samples.

**Task 2** Count reads in sense and antisense. Discuss differences. Why is this analysis meaningless for the provided non-strand-specific RNA-Seq samples?

**Task 3** Identify all genes where the antisense counts are $\geq$3-fold higher than the sense counts in at least 2 out of the 4 samples.

**Task 4** Plot the result of the most pronounced antisense expression case with *ggbio*.
Outline

Overview

RNA-Seq Analysis
  Quality Report
  Aligning Short Reads
  Counting Reads per Feature

DEG Analysis
  GO Analysis
  View Results in IGV & ggbio
  Differential Exon Usage

References
Compute mean values for replicates

```r
source("http://faculty.ucr.edu/~tgirke/Documents/R_BioCond/My_R_Scripts/colAg.R")
rpkmDFeByg <- read.table("./results/rpkmDFeByg.xls")[, M12_A12]
countDFrpkm_mean <- colAg(myMA=rpkmDFeByg, group=c(1,1,2,2), myfct=mean)
countDFrpkm_mean[1:4,]

<table>
<thead>
<tr>
<th></th>
<th>AT1G01010</th>
<th>AT1G01020</th>
<th>AT1G01030</th>
<th>AT1G01040</th>
</tr>
</thead>
<tbody>
<tr>
<td>M12A_M12B</td>
<td>3557.1959</td>
<td>1520.5605</td>
<td>766.2811</td>
<td>3488.4589</td>
</tr>
<tr>
<td>A12A_A12B</td>
<td>4279.9292</td>
<td>1054.1279</td>
<td>495.6289</td>
<td>2245.1415</td>
</tr>
</tbody>
</table>
```

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,]

<table>
<thead>
<tr>
<th></th>
<th>AT1G01030</th>
<th>AT1G01050</th>
<th>AT1G01070</th>
<th>AT1G01080</th>
<th>AT1G01090</th>
</tr>
</thead>
<tbody>
<tr>
<td>M12A_M12B</td>
<td>766.2811</td>
<td>25903.2856</td>
<td>723.8589</td>
<td>15247.6545</td>
<td>35151.8743</td>
</tr>
<tr>
<td>A12A_A12B</td>
<td>495.6289</td>
<td>7901.1976</td>
<td>138.3461</td>
<td>2608.0398</td>
<td>8586.1872</td>
</tr>
<tr>
<td>log2ratio</td>
<td>1.712992</td>
<td>1.712992</td>
<td>2.387427</td>
<td>2.547550</td>
<td>2.033512</td>
</tr>
</tbody>
</table>
```

```r
write.table(degs2fold, ".results/degs2fold.xls", quote=FALSE, sep="\t", col.names = NA)
degs2fold <- read.table("./results/degs2fold.xls")
```
Identify DEGs with \texttt{DESeq2} Library

Raw count data are expected here!

\begin{verbatim}
> library(DESeq2)
> countDF <- read.table("./results/countDFeByg.xls")[, M12_A12]
> countData <- as.matrix(countDF)
> dds <- DESeqDataSetFromMatrix(countData = countData, # Create DESeqDataSet object
+     colData = colData, design = ~ condition)
> dds <- DESeq(dds) # (i) estimation of size factors, (ii) dispersion,
>     # (iii) negative binomial GLM fitting and (iv) Wald statistics
> res <- results(dds) # Extracts DEG results from DESeqDataSet object
> res[is.na(res[,"padj"]], "padj"] <- 1
> res <- na.omit(res)
> res2fold <- res[res$log2FoldChange >= 1 | res$log2FoldChange <= -1,]
> res2foldpadj <- res2fold[res2fold$padj <= 0.2, ]
> res2foldpadj[1:4,]
\end{verbatim}

log2 fold change (MAP): condition M12 vs A12
Wald test p-value: condition M12 vs A12

Data Frame with 4 rows and 6 columns:

\begin{verbatim}
   baseMean log2FoldChange  lfcSE   stat   pvalue   padj
    <numeric>      <numeric> <numeric>  <numeric> <numeric> <numeric>
 AT1G01080   79.53825     1.385699 0.5954125 2.327293 0.019949670 0.07979868
 AT1G01090   252.22561    1.011465 0.5285502 1.913660 0.055663666 0.15289865
 AT2G01021   3466.65548   -1.373892 0.5714111 -2.404384 0.016199752 0.07979868
 ATCG00020   972.84314    -1.899092 0.6606360 -2.874642 0.004044864 0.02696576
\end{verbatim}
Identify DEGs with *edgeR*'s Exact Method

DEG analysis with classical *edgeR* approach. Note: raw read count data are expected by all methods!

```r
> library(edgeR)
> countDF <- read.table("./results/countDFeByg.xls")[, M12_A12]
> conds <- colData$condition
> 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("M12", "A12")) # Computes exact test for the negative binomial distribution.
> topTags(et, n=4)

Comparison of groups: A12-M12

<table>
<thead>
<tr>
<th>logFC</th>
<th>logCPM</th>
<th>PValue</th>
<th>FDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT1G01080</td>
<td>-2.532293</td>
<td>13.56691</td>
<td>0.0008697014</td>
</tr>
<tr>
<td>AT3G01060</td>
<td>-2.088054</td>
<td>12.06507</td>
<td>0.0023504132</td>
</tr>
<tr>
<td>AT1G01090</td>
<td>-2.032593</td>
<td>15.13123</td>
<td>0.0064141533</td>
</tr>
<tr>
<td>AT5G01020</td>
<td>-1.747026</td>
<td>14.73618</td>
<td>0.0124241193</td>
</tr>
</tbody>
</table>

> edge <- as.data.frame(topTags(et, n=50000))
> edge2fold <- edge[edge$logFC >= 1 | edge$logFC <= -1,]
> edge2foldpadj <- edge2fold[edge2fold$FDR <= 0.2, ]
```
Identify DEGs with *edgeR*’s GLM Approach

**DEG analysis with *edgeR* using generalized linear models (glms)**

```r
> library(edgeR)
> countDF <- read.table("./results/countDFeByg.xls", , M12_A12)
> y <- DGEList(counts=countDF, group=conds) # Constructs DGEList object
> ## Filtering and normalization
> keep <- rowSums(cpm(y)>1) >= 2; y <- y[keep, ]
> y <- calcNormFactors(y)
> design <- model.matrix(~0+group, data=y$samples); colnames(design) <- levels(y$samples$group) # Design matrix
> ## Estimate dispersion
> y <- estimateGLMCommonDisp(y, design, verbose=TRUE) # Estimates common dispersions
> Disp = 0.1936 , BCV = 0.44
> y <- estimateGLMTrendedDisp(y, design) # Estimates trended dispersions
> y <- estimateGLMTagwiseDisp(y, design) # Estimates tagwise dispersions
> ## Fit the negative binomial GLM for each tag
> fit <- glmFit(y, design) # Returns an object of class DGEGLM
> contrasts <- makeContrasts(contrasts="M12-A12", levels=design) # Contrast matrix is optional
> lrt <- glmLRT(fit, contrast=contrasts[,1]) # Takes DGEGLM object and carries out the likelihood ratio test.
> edgeglm <- as.data.frame(topTags(lrt, n=length(rownames(y))))
> ## Filter on fold change and FDR
> edgeglm2fold <- edgeglm[edgeglm$logFC >= 1 | edgeglm$logFC <= -1, ]
> edgeglm2foldpadj <- edgeglm2fold[edgeglm2fold$FDR <= 0.2, ]
```
Comparison Among DEG Results

```r
> source("http://faculty.ucr.edu/~tgirke/Documents/R_BioCond/My_R_Scripts/overLapper.R")
> setlist <- list(edgeRexact=rownames(edge2foldpadj), edgeRglm=rownames(edgeglm2foldpadj), DESeq2=rownames(res2foldpadj), RPKM=rownames(degs2fold))
> OLlist <- overLapper(setlist=setlist, sep="_", type="vennsets")
> counts <- sapply(OLlist$Venn_List, length)
> vennPlot(counts=counts, mymain="DEG Comparison")
```

DEG Comparison

Unique objects: All = 52; S1 = 2; S2 = 9; S3 = 8; S4 = 45
Heatmap of Top Ranking DEGs

Note: gene-wise clustering is not possible with a single sample pair. The following shows the scaled expression values (here RPKMs) in form of a heatmap.

```r
> library(lattice); library(gplots)
> rpkmDFeByg <- read.table("./results/rpkmDFeByg.xls")
> y <- rpkmDFeByg[rownames(edgeglm2foldpadj),]
> colnames(y) <- targets$Factor
> y <- t(scale(t(as.matrix(y))))
> y <- y[order(y[,1]),]
> levelplot(t(y), height=0.2, col.regions=colorpanel(40, "darkblue", "yellow", "white"), main="Expression Values (DEG Filter: FDR 1%, FC > 2)")
```
DEGs for all Comparisons Defined in targets File

```r
> countDF <- read.table("./results/countDFeByg.xls")
> cmp <- readComp(file="targets.txt", format="matrix", delim="-")
> edgeDF <- run_edgeR(countDF=countDF, targets=targets, cmp=cmp[[1]], independent=FALSE, mdsplot="")

Disp = 0.17287 , BCV = 0.4158

> DEG_list <- filterDEGs(degDF=edgeDF, filter=c(Fold=2, FDR=20))
```

![Bar chart showing DEG Counts (Fold: 2 & FDR: 20%) with comparisons A1-V1, A12-V12, A6-V6, M1-A1, M1-V1, M12-A12, M12-V12, M6-A6, M6-V6. The chart displays the number of downregulated and upregulated genes for each comparison.]

Analysis of RNA-Seq Data with R/Bioconductor
Overview

RNA-Seq Analysis
- Quality Report
- Aligning Short Reads
- Counting Reads per Feature
- DEG Analysis

GO Analysis
- View Results in IGV & ggbio
- Differential Exon Usage

References
The following performs GO term enrichment analysis of one of the identified DEG sets using the GOstats package. Another package, among many others, to consider here is the goseq that considers gene length bias in RNA-Seq data.

```r
> library(GOstats); library(GO.db); library(ath1121501.db)
> geneUniverse <- rownames(countDF)
> geneSample <- rownames(res2foldpadj)
> params <- new("GOHyperGParams", geneIds = geneSample, universeGeneIds = geneUniverse,
+   annotation="ath1121501", ontology = "MF", pvalueCutoff = 0.5,
+   conditional = FALSE, testDirection = "over")
> hgOver <- hyperGTest(params)
> summary(hgOver)[c(1,2,4,5),]

<table>
<thead>
<tr>
<th>GOMFID</th>
<th>Pvalue</th>
<th>OddsRatio</th>
<th>ExpCount</th>
<th>Count</th>
<th>Size</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>GO:0016168</td>
<td>0.0001127453</td>
<td>94.66667</td>
<td>0.4430380</td>
<td>4</td>
<td>5</td>
<td>chlorophyll binding</td>
</tr>
<tr>
<td>GO:0046906</td>
<td>0.0003273065</td>
<td>46.66667</td>
<td>0.5316456</td>
<td>4</td>
<td>6</td>
<td>tetrapyrrole binding</td>
</tr>
<tr>
<td>GO:0097159</td>
<td>0.0589172699</td>
<td>5.00000</td>
<td>2.5696203</td>
<td>5</td>
<td>29</td>
<td>organic cyclic compound binding</td>
</tr>
<tr>
<td>GO:1901363</td>
<td>0.0589172699</td>
<td>5.00000</td>
<td>2.5696203</td>
<td>5</td>
<td>29</td>
<td>heterocyclic compound binding</td>
</tr>
</tbody>
</table>

> htmlReport(hgOver, file = "results/MyhyperGresult.html")
```
The following shows how to obtain gene-to-GO mappings from biomaRt. This is relatively slow, but it needs to be done only once.

```r
> library("biomaRt")
> listMarts() # To choose BioMart database
> m <- useMart("ENSEMBL_MART_PLANT"); listDatasets(m)
> m <- useMart("ENSEMBL_MART_PLANT", dataset="athaliana_eg_gene")
> listAttributes(m) # Choose data types you want to download
> go <- getBM(attributes=c("go_accession", "tair_locus", "go_namespace_1003"), mart=m)
> go <- go[go[,3]!="",]; go[,3] <- as.character(go[,3])
> dir.create("./data/GO")
> write.table(go, "data/GO/GOannotationsBiomart_mod.txt", quote=FALSE, row.names=FALSE, col.names=FALSE, sep="\t")
> catdb <- makeCATdb(myfile="data/GO/GOannotationsBiomart_mod.txt", lib=NULL, org="", colno=c(1,2,3), idconv=NULL)
> save(catdb, file="data/GO/catdb.RData")
```
The Batch enrichment analysis of many gene sets is performed with the `GOCluster_Report` function. When `method="all"`, it returns all GO terms passing the p-value cutoff specified under the `cutoff` arguments. When `method="slim"`, it returns only the GO terms specified under the `myslimv` argument. The given example shows how one can obtain such a GO slim vector from BioMart for a specific organism.

```r
> load("data/GO/catdb.RData")
> DEG_list <- filterDEGs(degDF=edgeDF, filter=c(Fold=2, FDR=50), plot=FALSE)
> up_down <- DEG_list$UporDown; names(up_down) <- paste(names(up_down), "_up_down", sep="")
> up <- DEG_list$Up; names(up) <- paste(names(up), "_up", sep="")
> down <- DEG_list$Down; names(down) <- paste(names(down), "_down", sep="")
> DEGlist <- c(up_down, up, down)
> DEGlist <- DEGlist[sapply(DEGlist, length) > 0]
> BatchResult <- GOCluster_Report(catdb=catdb, setlist=DEGlist, method="all", id_type="gene", CLSZ=2, cutoff=0.9, gocats=c("MF", "BP", "CC"), recordSpecGO=NULL)
> library("biomaRt"); m <- useMart("ENSEMBL_MART_PLANT", dataset="athaliana_eg_gene")
> goslimvec <- as.character(getBM(attributes=c("goslim_goa_accession"), mart=m)[,1])
> BatchResultslim <- GOCluster_Report(catdb=catdb, setlist=DEGlist, method="slim", id_type="gene", myslimv=goslimvec)
```
Batch GO Term Enrichment Analysis (Part III)

Plot batch GO term results

```r
> goBarplot(BatchResultslim, gocat="MF")
```

Figure: GO Slim Barplot for MF Ontology.
Outline

Overview

RNA-Seq Analysis
- Quality Report
- Aligning Short Reads
- Counting Reads per Feature
- DEG Analysis
- GO Analysis

View Results in IGV & ggbio

Differential Exon Usage

References
Inspect Results in IGV

View results 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 File...
  
  
  `./data/SRR446039_1.fastq.tophat/accepted_hits.bam`
  
  `./data/SRR446040_1.fastq.tophat/accepted_hits.bam`
  
  `./data/SRR446041_1.fastq.tophat/accepted_hits.bam`
  
  `./data/SRR446042_1.fastq.tophat/accepted_hits.bam`

- To view area of interest, enter its coordinates Chr1:45,296-47,019 in position menu on top.

![Genome browser screenshot](image.png)
Controlling IGV from R

Create previous IGV session with required tracks automatically, and direct it to a specific position, here Chr1:45,296–47,019.

```r
> library(SRAdb)
> startIGV("lm")
> sock <- IGVsocket()
> session <- IGVsession(files=outpaths(args)[M12_A12],
+        sessionFile="session.xml",
+        genome="A. thaliana (TAIR10)")
> IGVload(sock, session)
> IGVgoto(sock, 'Chr1:45296-47019')
```
Generate Similar View with **ggbio** Programmatically

```r
> library(ggbio)
> M12A <- readGAlignmentsFromBam(outpaths(args)["M12A"], use.names=TRUE, param=ScanBamParam(which=GRanges("Chr1", IRanges(45296, 47019))))
> A12A <- readGAlignmentsFromBam(outpaths(args)["A12A"], use.names=TRUE, param=ScanBamParam(which=GRanges("Chr1", IRanges(45296, 47019))))
> p1 <- autoplot(M12A, geom = "rect", aes(color = strand, fill = strand))
> p2 <- autoplot(A12A, geom = "rect", aes(color = strand, fill = strand))
> p3 <- autoplot(txdb, which=GRanges("Chr1", IRanges(45296, 47019)), names.expr = "gene_id")
> tracks(M12A=p1, A12A=p2, Transcripts=p3, heights = c(0.3, 0.3, 0.4)) + ylab("")
```
Exercise 2: Venn Diagram for Up/Down DEGs

Task 1  Store the identifiers of the upregulated genes from each of the four DEG methods in separate components of a list. Note: the definition of up and down is arbitrary and one needs to check how it is defined by the different DEG methods!

Task 2  Do the same for the downregulated genes.

Task 3  Compare the overlaps among the different up/down sets in a single 4-way venn diagram.
Outline

Overview

RNA-Seq Analysis
  Quality Report
  Aligning Short Reads
  Counting Reads per Feature
  DEG Analysis
  GO Analysis
  View Results in IGV & ggbio

Differential Exon Usage

References
Number of reads overlapping gene ranges

```r
> library(DEXSeq)
> exonicParts <- disjointExons(txdb, aggregateGenes=FALSE)
> bamlst <- BamFileList(outpaths(args)[M12_A12], index=character(), yieldSize=100000, obeyQname=TRUE)
> SE <- summarizeOverlaps(exonicParts, bamlst, mode="Union", singleEnd=TRUE, ignore.strand=TRUE, inter.feature=FALSE)
> colData(SE)$condition <- colData$condition
> dxd <- DEXSeqDataSetFromSE(SE, design= ~ sample + exon + condition:exon)
> featureCounts(dxd)[1:2,] # Counts for individual exons

M12A M12B A12A A12B
AT1G01010:E001  9  12  14  6
AT1G01010:E002 12  15  12 11

> assays(dxd)$counts[1:2,] # Counts for individual exons plus for all remaining exons of a gene

AT1G01010:E001  9  12  14  6  59  71  67  37
AT1G01010:E002 12  15  12 11  56  68  69  32

> # rowData(dxd)[1:4,] # Exon ranges
> # colData(dxd) # Sample data
> write.table(featureCounts(dxd), ".results/countDFdex", quote=FALSE, sep="\t", col.names = NA)
```
## Analysis of Differential Exon Usage with DEXSeq

### Identify genes with differential exon usage

```r
> ## Performs normalization
> dxd <- estimateSizeFactors(dxd)
> ## Evaluate variance of the data by estimating dispersion using Cox-Reid (CR) likelihood estimation
> dxd <- estimateDispersions(dxd)
> ## Performs Chi-squared test on each exon and Benjmini-Hochberg p-value adjustment for multiple testing
> dxd <- testForDEU(dxd)
> ## Estimates fold changes of exons
> dxd <- estimateExonFoldChanges(dxd, fitExpToVar="condition")
> ## Obtain results as DataFrame
> dxr1 <- DEXSeqResults(dxd)
> ## Column descriptions
> col_descr <- elementMetadata(dxr1)$description
> ## Count number of genes with differential exon usage
> dxr1[is.na(dxr1$padj), "padj"] <- 1
> table(tapply(dxr1$padj < 0.2, dxr1$groupID, any))

<table>
<thead>
<tr>
<th></th>
<th>FALSE</th>
<th>TRUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>115</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

> ## DEU sample
> dxr1[dxr1$groupID=="AT4G00050",][1:4, c(1:2,7:10)]

### DataFrame with 4 rows and 6 columns

<table>
<thead>
<tr>
<th>groupID</th>
<th>featureID</th>
<th>padj</th>
<th>A12</th>
<th>M12</th>
<th>log2fold_A12_M12</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT4G00050:E001</td>
<td>AT4G00050</td>
<td>0.00000000</td>
<td>6.275199</td>
<td>6.688707</td>
<td>-0.09206611</td>
</tr>
<tr>
<td>AT4G00050:E002</td>
<td>AT4G00050</td>
<td>0.08536468</td>
<td>12.137741</td>
<td>10.860877</td>
<td>0.16035925</td>
</tr>
<tr>
<td>AT4G00050:E003</td>
<td>AT4G00050</td>
<td>1.00000000</td>
<td>4.998105</td>
<td>5.463383</td>
<td>-0.12841322</td>
</tr>
<tr>
<td>AT4G00050:E004</td>
<td>AT4G00050</td>
<td>1.00000000</td>
<td>4.793227</td>
<td>5.288418</td>
<td>-0.14183906</td>
</tr>
</tbody>
</table>
```
DEXSeq Plots

Sample plot showing fitted expression of exons

```r
> plotDEXSeq(dxr1, "AT4G00050", displayTranscripts=TRUE, legend=TRUE, cex.axis=1.2, cex=1.3, lwd=2 )
```

Generate many plots and write them to results directory

```r
> mygeneIDs <- unique(as.character(na.omit(dxr1[dxr1$groupID %in% unique(dxr1$groupID),] [,"groupID"])))
> DEXSeqHTML(dxr1, genes=mygeneIDs[1:10], path="results", file="DEU.html")
```
systemPipeR: Run Entire RNA-Seq Workflow and Generate Report

- *systemPipeR* is useful for building end-to-end analysis pipelines with automated report generation for NGS applications such as RNA-Seq and many others.
- It provides support for running command-line software, such as NGS aligners, on both single machines or compute clusters. This includes both interactive job submissions or batch submissions to queuing systems of clusters.
- To generate the report for the data sets and analysis steps demonstrated in this tutorial, open the file *systemPipeR.Rnw* in RStudio’s code editor and then click the Compile PDF button. This will run the entire analysis and generate the corresponding RNA-Seq analysis report (PDF format) along with a bibliography of citations included in the text.
- Alternatively, one can achieve the same result by running the following commands from the command-line:

  ```
  echo 'Sweave("systemPipeR.Rnw")' | R --slave # Runs R code
  echo 'Stangle("systemPipeR.Rnw")' | R --slave # Extracts R code
  pdflatex systemPipeR.tex; bibtex systemPipeR; pdflatex systemPipeR.tex # Compiles PDF
  ```

- Note: for time reasons, not all code chunks are evaluated (change `eval=FALSE` to `eval=TRUE`) when the report is generated.

- A sample report can be viewed here [systemPipeR.pdf](#).
- To efficiently customize these reports, users want to learn how to use *LaTeX/Sweave and/or knitr*. 

---

**Analysis of RNA-Seq Data with R/Bioconductor**

**RNA-Seq Analysis**

**Differential Exon Usage**

Slide 58/61
Session Information

> sessionInfo()

R version 3.1.2 (2014-10-31)
Platform: x86_64-unknown-linux-gnu (64-bit)
locale:
[1] C

attached base packages:
[1] parallel stats4 stats graphics utils datasets grDevices methods base

other attached packages:
[1] DEXSeq_1.12.1 ggbio_1.14.0 ggplot2_1.0.0 xtable_1.7-4 ath1121501.db_3.0.0 org.At.tair.db_3.0.0 GOstats_2.32.0 graph_1.44.0 Category_2.32.0
[10] GO.db_3.0.0 Matrix_1.1-4 gplots_2.14.2 lattice_0.20-29 edgeR_3.8.2
[19] ape_3.1-4 GenomicFeatures_1.18.2 rtracklayer_1.26.2 systemPipeR_1.0.0 AnnotationDbi_1.28.1 Biobase_2.26.0 RSQLite_1.0.0 DBI_0.3.1 ShortRead_1.24.0
[28] GenomicAlignments_1.2.1 BiocParallel_1.0.0 Rsamtools_1.18.2 Bistrings_2.34.0 XVector_0.6.0
[37] BiocGenerics_0.12.1

loaded via a namespace (and not attached):
[1] AnnotationForge_1.8.1 BBmisc_1.8 BSgenome_1.34.0 BatchJobs_1.5 Formul[...
[9] KernSmooth_2.23-13 MASS_7.3-35 OrganismDbi_1.8.0 RBGL_1.42.0 RColorB...
Outline

Overview

RNA-Seq Analysis
  Quality Report
  Aligning Short Reads
  Counting Reads per Feature
  DEG Analysis
  GO Analysis
  View Results in IGV & ggbio
  Differential Exon Usage

References
URL http://www.hubmed.org/display.cgi?uids=20979621

URL http://www.hubmed.org/display.cgi?uids=22722343

URL http://dx.doi.org/10.1371/journal.pone.0074183

URL http://www.hubmed.org/display.cgi?uids=19910308

URL http://www.hubmed.org/display.cgi?uids=20196867

URL http://www.hubmed.org/display.cgi?uids=23222703