

Analysis of RNA-Seq Data with R/Bioconductor

—

MCBIOS-2015 Workshop

Thomas Girke

March 12, 2015

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

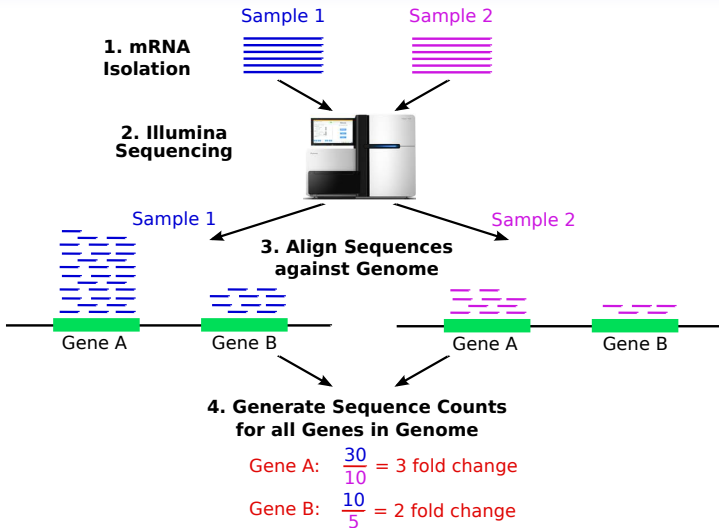
- GO Analysis

- View Results in IGV & ggbio

- Differential Exon Usage

References

RNA-Seq Technology



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.

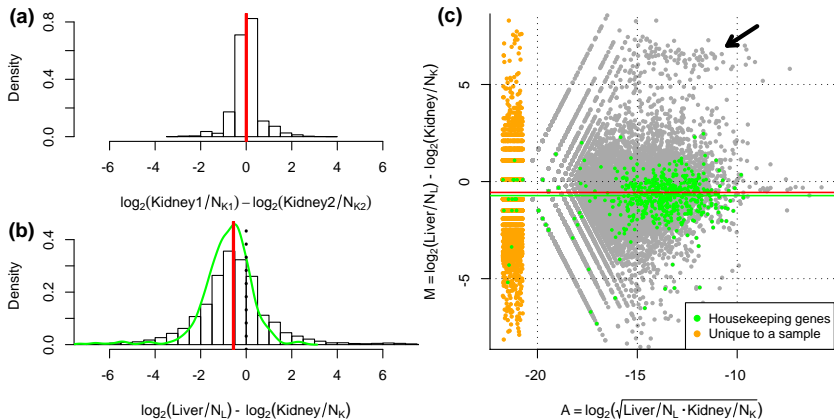
```
Coor      12345678901234 5678901234567890123456789012345
ref       AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT
+r001/1   TTAGATAAAGGATA*CTG
+r002     aaaAGATAA*GGATA
+r003     gcctaAGCTAA
+r004     ATAGCT.....TCAGC
-r003     ttagctTAGGC
-r001/2   CAGCGGCAT
```

⇓ SAM Format

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

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!

TMM Method Corrects for RNA Composition Bias

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

DEG Analysis Methods

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 [Link](#): high-level infrastructure for range data
- Rsamtools [Link](#): BAM support
- rtracklayer [Link](#): Import/export of range and annotation data, interface to online genome browsers, etc.
- DESeq [Link](#): RNA-Seq DEG analysis
- DESeq2 [Link](#): RNA-Seq DEG analysis
- edgeR [Link](#): RNA-Seq DEG analysis
- DEXSeq [Link](#): RNA-Seq Exon analysis
- QuasR [Link](#): RNA-Seq workflows
- systemPipeR [Link](#): 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 [Link](#) for this practical.
- Direct your R session to the resulting Rrnaseq directory. It contains 18 slimmed down FASTQ files (SRP010938 [Link](#)) 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 [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 [Link](#). 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:

```
> # download.file("http://biocluster.ucr.edu/~tgirke/HTML_Presentations/Manuals/MCBIOS2015/Rrnaseq.zip", "Rrnaseq.zip")
> # unzip("Rrnaseq.zip")
> # setwd("Rrnaseq")

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

	FileName	SampleName	Factor	SampleLong	Experiment	Date
1	./data/SRR446027_1.fastq	M1A	M1	Mock.1h.A	1	23-Mar-2012
2	./data/SRR446028_1.fastq	M1B	M1	Mock.1h.B	1	23-Mar-2012
3	./data/SRR446029_1.fastq	A1A	A1	Avr.1h.A	1	23-Mar-2012

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

Quality Reports

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

```
> fqlist <- seeFastq(fastq=infile1(args), batchsize=10000, klength=8)
> pdf("./results/fastqReport.pdf", height=18, width=4*length(fqlist))
> seeFastqPlot(fqlist); dev.off()
```

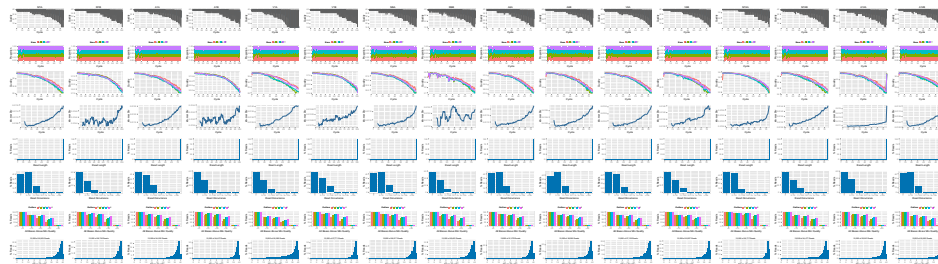


Figure: QC report for 18 FASTQ files.

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

Align Reads with *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

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

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

```
> 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=gsutil)
> unlink(outfile1(args)); unlink(paste0(outfile1(args), ".indel"))
```

Align Reads with *Tophat2*

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* [Link](#).

(1) Index reference genome

```
> 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.

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

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

```
> bampaths
```

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

```
> 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.

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

	FileName	Nreads	Nalign	Perc_Aligned	Nalign_Primary	Perc_Aligned_Primary
M1A	M1A	96459	89376	92.65698	89376	92.65698
M1B	M1B	98742	86014	87.10984	86014	87.10984
A1A	A1A	94935	88360	93.07421	88360	93.07421
A1B	A1B	94427	83172	88.08074	83172	88.08074
V1A	V1A	99366	80869	81.38498	80869	81.38498
V1B	V1B	97771	93637	95.77175	93637	95.77175
M6A	M6A	98617	92917	94.22006	92917	94.22006
M6B	M6B	90452	80074	88.52651	80074	88.52651

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


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

Store GFF/GTF Range Annotations in TranscriptDb

Storing annotation ranges in *TranscriptDb* databases makes many operations more robust and convenient.

```
> 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")
```

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 [Link](#) for details.

```
> txdb <- loadDb("../data/tair10.sqlite")
> eByg <- exonsBy(txdb, by=c("gene"))
> bfl <- BamFileList(outpaths(args), yieldSize=50000, index=character())
> counteByg <- summarizeOverlaps(eByg, bfl, mode="Union", ignore.strand=TRUE, inter=
> countDFeByg <- assays(counteByg)$counts
> countDFeByg[1:4,1:12]
```

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

```
> write.table(countDFeByg, "results/countDFeByg.xls", col.names=NA, quote=FALSE, sep=
```

Simple RPKM Normalization

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

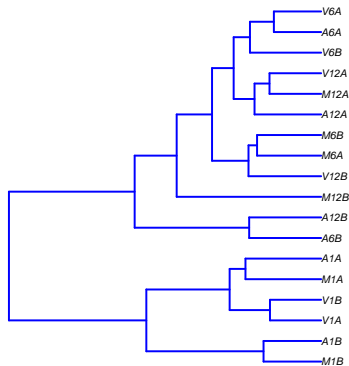
```
> rpkmDFeByg <- apply(countDFeByg, 2, function(x) returnRPKM(counts=x, ranges=eByg))
> write.table(rpkmDFeByg, "results/rpkmDFeByg.xls", col.names=NA, quote=FALSE, sep=" ")
> rpkmDFeByg[1:4,1:7]
```

	M1A	M1B	A1A	A1B	V1A	V1B	M6A
AT1G01010	2424.745	6201.798	5431.999	4225.653	11608.558	5482.332	1315.5522
AT1G01020	988.799	2166.827	1827.306	2310.805	2957.618	1812.533	512.0905
AT1G01030	1457.938	2189.552	1750.267	1420.254	2641.816	3211.111	264.9311
AT1G01040	2290.597	4629.408	3835.648	4522.808	5102.206	4389.003	1323.4691

Reproducibility Check by Sample-Wise Clustering

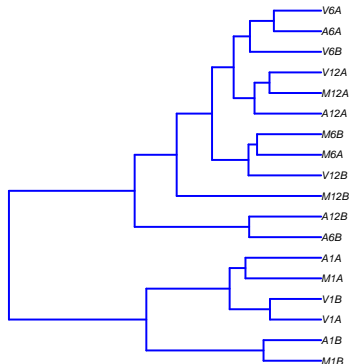
QC check of the sample reproducibility by computing a correlating matrix and plotting it as a tree. Note: the `plotMDS` function from `edgeR` is a more robust method for this task.

```
> 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 `rlog` Values

```
> library(DESeq2)
> countDF <- as.matrix(read.table("./results/countDFeByg.xls"))
> colData <- data.frame(row.names=targetsin(args)$SampleName, condition=targetsin(args)$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: Strand-specific Read Counting

- Task 1** Count reads mapping to both strands and also those mapping only to the sense strand. Discuss differences. Why is this analysis meaningless for the provided non-strand-specific RNA-Seq samples?
- Task 2** Identify all genes where more than 75% of the reads in at least 4 samples map to the sense strand.
- Task 3** Plot the result of the most pronounced strand-selective cases 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

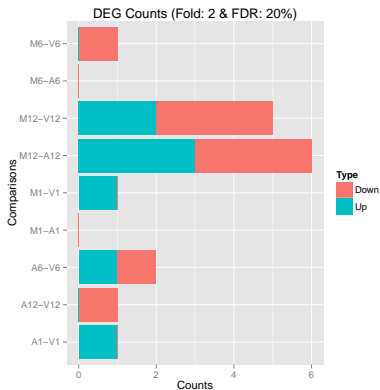
References

DEGs for Comparisons in targets File Using *edgeR*

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

Disp = 0.17287 , BCV = 0.4158

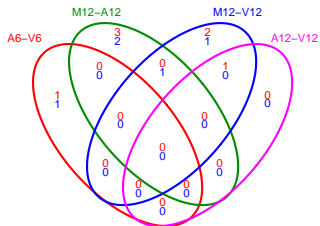
> write.table(edgeDF, "./results/edgeRcomp.xls", quote=FALSE, sep="\t", col.names = NA)
> DEG_list <- filterDEGs(degDF=edgeDF, filter=c(Fold=2, FDR=20))
```



4-Way Venn Diagram of Up and Down DEG Sets

The function `overLapper` can compute Venn intersects for any number of sample sets and `vennPlot` can plot 2-5 way Venn diagrams. A useful feature is the possibility to combine the counts from several Venn comparisons in a single plot.

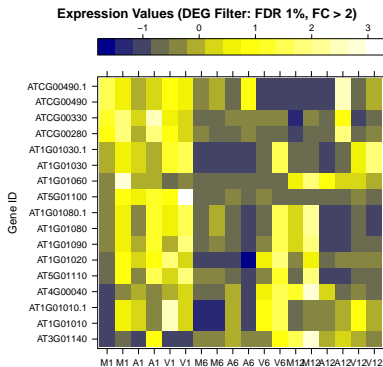
```
> vennsetup <- overLapper(DEG_list$Up[6:9], type="vennsets")
> vennsetdown <- overLapper(DEG_list$Down[6:9], type="vennsets")
> vennPlot(list(vennsetup, vennsetdown), mymain="", mysub="", colmode=2, ccol=c("blue", "red"))
```



Heatmap of DEGs

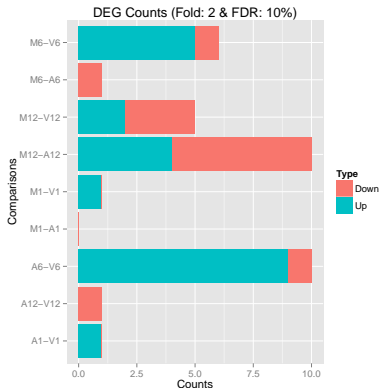
Note: the following plots the scaled expression values (here RPKMs) in form of a heatmap.

```
> library(lattice); library(gplots)
> rpkmDFeByg <- read.table("./results/rpkmDFeByg.xls")
> y <- rpkmDFeByg[unlist(DEG_list$UporDown),]
> colnames(y) <- targetsin(args)$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
```



DEGs for Comparisons in targets File Using DESeq2

```
> countDF <- read.table("./results/countDFeByg.xls")
> cmp <- readComp(args, format="matrix", delim="-")
> degseqDF <- run_DESeq2(countDF=countDF, targets=targetsin(args), cmp=cmp[[1]], independent=FALSE)
> write.table(edgeDF, "./results/DESeq2comp.xls", quote=FALSE, sep="\t", col.names = NA)
> DEG_list2 <- filterDEGs(degDF=degseqDF, filter=c(Fold=2, FDR=10))
```



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

Batch GO Term Enrichment Analysis (Part I)

The following shows how to obtain gene-to-GO mappings from *biomaRt*. This is relatively slow, but it needs to be done only once.

```
> 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")
```

Batch GO Term Enrichment Analysis (Part II)

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.

```
> 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=0.05)
> write.table(BatchResult, "./results/GO_BatchResult.xls", quote=FALSE, sep="\t", row.names = FALSE)
> library("biomaRt"); m <- useMart("ensembl", dataset="mmulatta_gene_ensembl")
> 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)
> write.table(BatchResultslim, "./results/GO_BatchResultslim.xls", quote=FALSE, sep="\t", row.names = FALSE)
```

Batch GO Term Enrichment Analysis (Part III)

Plot batch GO term results

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

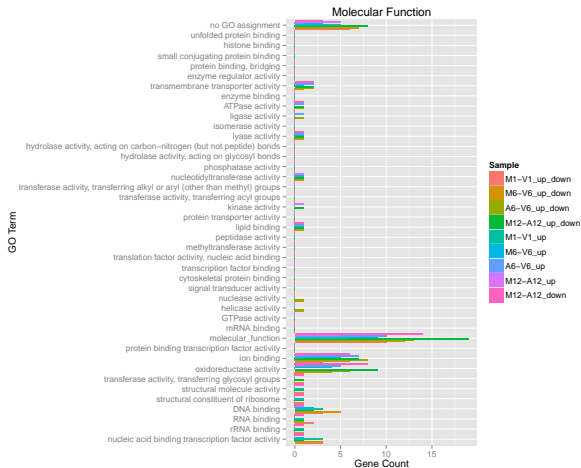


Figure: GO Slim Barplot for MF Ontology.

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

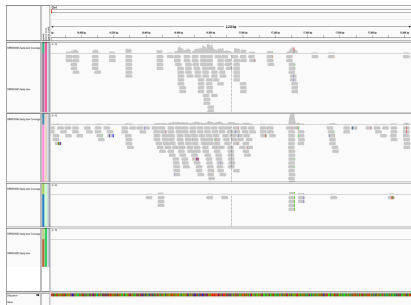
Inspect Results in IGV

View results in IGV

- 1 Download and open IGV [Link](#)
- 2 Select in menu in top left corner *A. thaliana* (TAIR10)
- 3 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
```

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



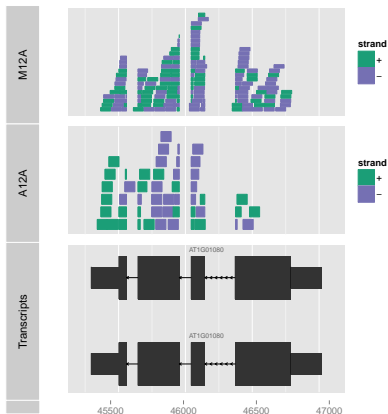
Controlling IGV from R

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

```
> library(SRadb)
> startIGV("lm")
> sock <- IGVsocket()
> session <- IGVsession(files=outpaths(args)[c("M12A", "M12B", "A12A", "A12B")],
+                       sessionFile="session.xml",
+                       genome="A. thaliana (TAIR10)")
> IGVload(sock, session)
> IGVgoto(sock, 'Chr1:45296-47019')
```

Generate Similar View with *ggbio* Programmatically

```
> library(ggbio)
> M12A <- readGAlignmentsFromBam(outpaths(args)["M12A"], use.names=TRUE, param=ScanBamParam(which=GRanges("Chr1
> A12A <- readGAlignmentsFromBam(outpaths(args)["A12A"], use.names=TRUE, param=ScanBamParam(which=GRanges("Chr1
> 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("")
```



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

Analysis of Differential Exon Usage with DEXSeq

Number of reads overlapping gene ranges

```
> library(DEXSeq)
> exonicParts <- disjointExons(txdb, aggregateGenes=FALSE)
> bamlst <- BamFileList(outpaths(args)[c("M12A", "M12B", "A12A", "A12B")], index=character(), yieldSize=100000,
> SE <- summarizeOverlaps(exonicParts, bamlst, mode="Union", singleEnd=TRUE, ignore.strand=TRUE, inter.feature=
> colData <- data.frame(condition=c(M12A="M12", M12B="M12", A12A="A12", A12B="A12"))
> 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
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
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

```
> ## 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 mutliple 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))
```

```
FALSE TRUE
  115     1
```

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

DataFrame with 4 rows and 6 columns

	groupID	featureID	padj	A12	M12	log2fold_A12_M12
	<character>	<character>	<numeric>	<numeric>	<numeric>	<numeric>
AT4G00050:E001	AT4G00050	E001	1.00000000	6.275199	6.688707	-0.09206611
AT4G00050:E002	AT4G00050	E002	0.08536468	12.137741	10.860877	0.16035925
AT4G00050:E003	AT4G00050	E003	1.00000000	4.998105	5.463383	-0.12841322
AT4G00050:E004	AT4G00050	E004	1.00000000	4.793227	5.288418	-0.14183906

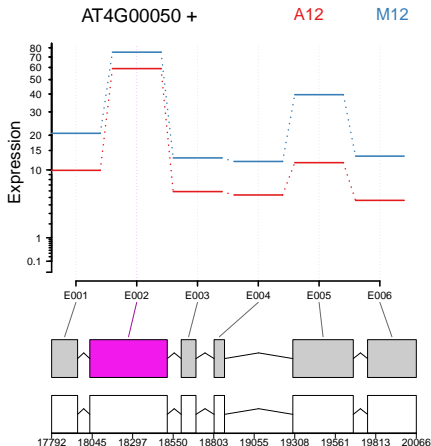
DEXSeq Plots

Sample plot showing fitted expression of exons

```
> 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

```
> 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 Analysis 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*.

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         gplots_2.16.0         1
[9] ape_3.2                GenomicFeatures_1.18.3 systemPipeR_1.0.11    AnnotationDbi_1.28.1  B
[17] GenomicAlignments_1.2.1 BiocParallel_1.0.2    Rsamtools_1.18.2     Biostrings_2.34.1     X
[25] S4Vectors_0.4.0       BiocGenerics_0.12.1
```

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

```
[1] AnnotationForge_1.8.2  BBmisc_1.9           BSgenome_1.34.1      BatchJobs_1.5         Categ
[9] GOstats_2.32.0        GSEABase_1.28.0     Hmisc_3.14-6         KernSmooth_2.23-13    MASS_
[17] RColorBrewer_1.1-2    RCurl_1.95-4.5      VariantAnnotation_1.12.9 XML_3.98-1.1         acepa
[25] biovizBase_1.14.1     bitops_1.0-6        brew_1.0-6           caTools_1.17.1       check
[33] dichromat_2.0-0       digest_0.6.8        edgeR_3.8.5          fail_1.2              forea
[41] geneplotter_1.44.0    graph_1.44.1        grid_3.1.2           gridExtra_0.9.1      gtabl
[49] labeling_0.3          latticeExtra_0.6-26 limma_3.22.4         locfit_1.5-9.1       munse
[57] plyr_1.8.1            proto_0.3-10        reshape_0.8.5        reshape2_1.4.1       rjson
[65] sendmailR_1.2-1       splines_3.1.2       statmod_1.4.20      stringr_0.6.2        survi
```

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

References I

- Anders, S., Huber, W., 2010. Differential expression analysis for sequence count data. *Genome Biol* 11 (10).
URL <http://www.hubmed.org/display.cgi?uids=20979621>
- Anders, S., Reyes, A., Huber, W., Oct 2012. Detecting differential usage of exons from RNA-seq data. *Genome Res* 22 (10), 2008–2017.
URL <http://www.hubmed.org/display.cgi?uids=22722343>
- Howard, B. E., Hu, Q., Babaoglu, A. C., Chandra, M., Borghi, M., Tan, X., He, L., Winter-Sederoff, H., Gassmann, W., Veronese, P., Heber, S., 1 Oct. 2013. High-throughput RNA sequencing of pseudomonas-infected arabidopsis reveals hidden transcriptome complexity and novel splice variants. *PLoS One* 8 (10), e74183.
URL <http://dx.doi.org/10.1371/journal.pone.0074183>
- Robinson, M. D., McCarthy, D. J., Smyth, G. K., Jan 2010. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* 26 (1), 139–140.
URL <http://www.hubmed.org/display.cgi?uids=19910308>
- Robinson, M. D., Oshlack, A., Mar 2010. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biol* 11 (3).
URL <http://www.hubmed.org/display.cgi?uids=20196867>
- Trapnell, C., Hendrickson, D. G., Sauvageau, M., Goff, L., Rinn, J. L., Pachter, L., Jan 2013. Differential analysis of gene regulation at transcript resolution with RNA-seq. *Nat Biotechnol* 31 (1), 46–53.
URL <http://www.hubmed.org/display.cgi?uids=23222703>