Data Analysis Pipeline for RNA-seq Experiments: From Differential Expression to Cryptic Splicing

Hari Krishna Yalamanchili,[1,2,5] Ying-Wooi Wan,[1,2,5] and Zhandong Liu[2,3,4]

[1]Department of Molecular and Human Genetics, Baylor College of Medicine, Houston, Texas
[2]Bioinformatics Core, Jan and Dan Duncan Neurological Research Institute at Texas Children's Hospital, Houston, Texas
[3]Computational and Integrative Biomedical Research Center, Baylor College of Medicine, Houston, Texas
[4]Department of Pediatrics-Neurology, Baylor College of Medicine, Houston, Texas
[5]These authors contributed equally to the manuscript

RNA sequencing (RNA-seq) is a high-throughput technology that provides unique insights into the transcriptome. It has a wide variety of applications in quantifying genes/isoforms and in detecting non-coding RNA, alternative splicing, and splice junctions. It is extremely important to comprehend the entire transcriptome for a thorough understanding of the cellular system. Several RNA-seq analysis pipelines have been proposed to date. However, no single analysis pipeline can capture dynamics of the entire transcriptome. Here, we compile and present a robust and commonly used analytical pipeline covering the entire spectrum of transcriptome analysis, including quality checks, alignment of reads, differential gene/transcript expression analysis, discovery of cryptic splicing events, and visualization. Challenges, critical parameters, and possible downstream functional analysis pipelines associated with each step are highlighted and discussed. This unit provides a comprehensive understanding of state-of-the-art RNA-seq analysis pipeline and a greater understanding of the transcriptome.

Keywords: RNA-seq • differential gene expression • differential isoform usage • alternative splicing • cryptic splicing

How to cite this article:
Yalamanchili, H. K., Wan, Y.-W., & Liu, Z. (2017). Data analysis pipeline for RNA-seq experiments: From differential expression to cryptic splicing. *Current Protocols in Bioinformatics*, *59*, 11.15.1–11.15.21. doi: 10.1002/cpbi.33

INTRODUCTION

Due to advancement in technology and drop in cost for high-throughput profiling of molecular assays and next-generation sequencing, RNA sequencing (RNA-seq) has becoming a common tool for scientists studying the transcriptomic phenomena observed in biological samples. RNA-seq data allows one to study system-wide transcriptional changes from a variety of aspects, ranging from expression changes in gene or isoform levels to complex analyses like discovery of novel, alternative, or cryptic splicing sites, RNA-editing sites, fusion genes, or single nucleotide variations (Conesa et al., 2016). These analyses have been applied in studies ranging from those aimed at understanding complex neurological diseases (Sztainberg et al., 2015) to those investigating basic molecular mechanisms (Lo et al., 2017).

Current Protocols in Bioinformatics 11.15.1–11.15.21, September 2017
Published online September 2017 in Wiley Online Library (wileyonlinelibrary.com).
doi: 10.1002/cpbi.33

In this article, we describe three protocols for RNA-seq data analysis. These protocols are applicable to RNA-seq data sets from the most common Illumina sequencing platform with an available reference genome. Basic Protocol 1 covers the standard pipeline of RNA-seq data that interrogates the transcriptome at the gene level, which is usually referred as differentially expressed gene (DEG) analysis. This pipeline starts from the raw sequence reads, and ends with a set of differentially expressed genes. Basic Protocol 2 goes beyond gene-level analysis into isoforms, focusing primarily on differential expression (DE) and differential usage (DU) of isoforms. Lastly, Basic Protocol 3 pinpoints specific splice junctions and deciphers cryptic splicing events. A detailed motivational rationale for each protocol is given in the respective sections below. Together, the three protocols help to thoroughly comprehend a transcriptome of interest from genes to junctions.

STRATEGIC PLANNING

Before starting the protocols, users should install the software listed in the software section of the protocol Necessary Resources from the appropriate download URL listed in Table 11.15.1. By following the downloading instructions, programs will be installed so that they can be executed from any directory. Lastly, users should download required libraries, reference files such as reference genome sequences, and index files necessary for the experiments. Respective Web resources are listed in Table 11.15.1.

Table 11.15.1 Required Tools and Respective Web Resources

Tool and resource	Download URL
Data resources	
Sample Fastq files	*https://bcm.box.com/s/c9q7otvxoog7cmn1b4p9mp8c7thu7q32*
Bowtie2 index	*https://ccb.jhu.edu/software/tophat/igenomes.shtml*
Basic Protocol 1	
FastQC	*http://www.bioinformatics.babraham.ac.uk/projects/fastqc/*
Tophat2 (Kim et al., 2013)	*https://ccb.jhu.edu/software/tophat/index.shtml*
Samtools (Li et al., 2009)	*http://www.htslib.org*
HTSeq (Anders, Pyl, & Huber, 2015)	*https://pypi.python.org/pypi/HTSeq*
Rstudio	*https://www.rstudio.com/*
DESeq2 (Love et al., 2014)	*http://bioconductor.org/packages/release/bioc/html/DESeq2.html*
Pheatmap	*https://cran.r-project.org/web/packages/pheatmap/*
Basic Protocol 2	
Kallisto (Bray et al., 2016)	*https://pachterlab.github.io/kallisto/*
Sleuth	*http://pachterlab.github.io/sleuth/about*
IUTA (Niu et al., 2014)	*http://www.niehs.nih.gov/research/resources/software/biostatistics/iuta/index.cfm*
Genome files/GTF files	*http://www.ensembl.org/info/data/ftp/index.html*
Basic Protocol 3	
CrypSplice (Tan et al., 2016)	*http://www.liuzlab.org/CrypSplice/*
IGV (Thorvaldsdottir, Robinson, & Mesirov, 2013)	*https://software.broadinstitute.org/software/igv/download*
Bedtools (Quinlan & Hall, 2010)	*http://bedtools.readthedocs.io/en/latest/*
R: Ibb (Pham, Piersma, Warmoes, & Jimenez, 2010)	*http://www.oncoproteomics.nl/software/BetaBinomial.html*
R: MASS	*https://cran.r-project.org/web/packages/MASS/index.html*

DIFFERENTIAL GENE EXPRESSION ANALYSIS OF RNA-seq

The immediate question that one may ask from an experiment with RNA-seq is what genes are dysregulated due to the designed perturbation, treatment, etc. Thus, this first protocol consists of the basic pipeline for analyzing raw sequence reads of RNA data to reveal the set of significantly dysregulated genes. Specifically, this pipeline consists of five main steps, where each step corresponds to one phase of the analysis that achieves a certain milestone. In the following protocol, we will use example data from six samples, three wild type (WT) and three mutant (MT) samples from GSE72790, and provide commands based on this example. To reduce the amount of execution time in testing the protocol, we will focus on data from chromosome 19 only.

Necessary Resources

Hardware

A computer or server with access to Unix command environment

Software

FastQC, Tophat2, samtools, HTSeq, Rstudio, DESeq2. Web resources of respective tools are listed in Table 11.15.1.

Input files

Raw sequence reads in `fastq` format. Data files (two files each for six samples) can be downloaded from the sample fastq files URL listed in Table 11.15.1. Each of these files contains about five million reads of 90 base-pair (bp) which originate from the chromosome 19 of the mouse genome.

1. Quality check on the raw reads.

   Create a directory named `FastQC` *to store the results. Then, call* `FastQC` *to obtain quality check metrics to inspect the quality of raw sequence reads (stored in the* `Fastq` *directory), and output the metrics to the* `FastQC` *directory:*

   ```
   $ mkdir FastQC
   $ fastqc Fastq/*.fastq -o FastQC
   ```

   FastQC provides a quick view of the quality of the raw sequence reads from multiple analyses, ranging from the sequence quality to GC content to library complexity. The command above will produce a report in HTML format which can be viewed in a Web browser. In the report, each metric evaluated will be annotated with a green check, red cross, or yellow exclamation mark to indicate pass, fail, or caution, respectively. Usually, the quality score (Fig. 11.15.1A) and A,C,G,T content (Fig. 11.15.1B) across bases can be used to decide how the reads should be groomed prior to mapping.

2. Groom raw reads.

   Remove sequences with low quality to get better alignment in the later steps. Based on the FastQC reports from the above commands for this example, the quality of the reads are fine except for random distribution of sequence content at the 5′ end of reads. Thus, we trim 10 bp from the beginning of each read:

   ```
   $ awk -v s=10 -v e=0 ' {if (NR%2 == 0) print
     substr($0, s+1, length($0)-s-e); else print $0;} '
     Fastq/MT1_1.fastq > Fastq/Trim_MT1_1.fastq
   $ awk -v s=10 -v e=0 ' {if (NR%2 == 0) print
     substr($0, s+1, length($0)-s-e); else print $0;} '
     Fastq/MT1_2.fastq > Fastq/Trim_MT1_2.fastq
   ```

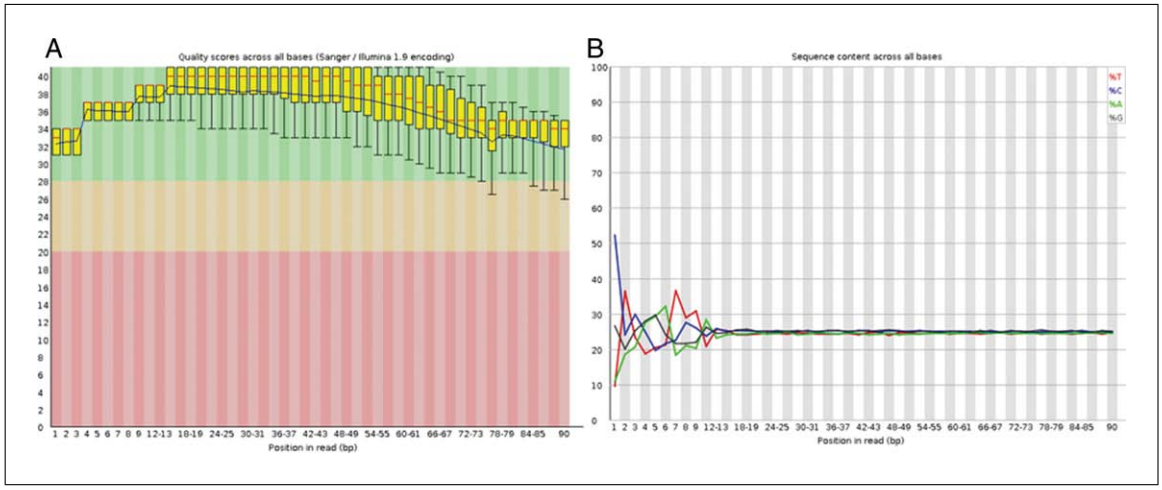

Figure 11.15.1 Quality check metrics on raw sequence reads from FastQC. Bar plot of quality score (Phred score) for each base in the reads (**A**). Line plot showing the distribution of each nucleotide (A, C, G, T) in the sequence reads on each bases (**B**).

```
$ awk -v s=10 -v e=0 ' {if (NR%2 == 0) print
  substr($0, s+1, length($0)-s-e); else print $0;} '
  Fastq/MT2_1.fastq > Fastq/Trim_MT2_1.fastq
$ awk -v s=10 -v e=0 ' {if (NR%2 == 0) print
  substr($0, s+1, length($0)-s-e); else print $0;} '
  Fastq/MT2_2.fastq > Fastq/Trim_MT2_2.fastq
$ awk -v s=10 -v e=0 ' {if (NR%2 == 0) print
  substr($0, s+1, length($0)-s-e); else print $0;} '
  Fastq/MT3_1.fastq > Fastq/Trim_MT3_1.fastq
$ awk -v s=10 -v e=0 ' {if (NR%2 == 0) print
  substr($0, s+1, length($0)-s-e); else print $0;} '
  Fastq/MT3_2.fastq > Fastq/Trim_MT3_2.fastq
```

We used the `awk` *command native to the Unix system here. The* `s = 10` *indicates that the first 10 bp (the 5′ end) will be trimmed, whereas* `e = 0` *indicates that none will be trimmed from the end (3′ end). Users should change these according to the FastQC reports. The file name for input fastq reads is specified before the > sign, whereas the file name for the trimmed reads comes after the > sign.*

Note that although these are paired-end (PE) reads, the forward and reverse reads are trimmed separately. Also, users should repeat these steps for the other six fastq files for WT samples.

Usually, quality of the sequence reads at both ends of the reads is low, which can be seen from the drop in Phred score at the 3′ end and random A,C,G,T concentration at the 5′ end. Although some would suggest removing the adapter sequences as well, due to the nature of next-generation sequencing, more than 99% of the adapter sequences are cleaved from the RNA fragments before sequencing and should not be present in the sequence reads. Thus, we did not include that step. However, if it happened that some adapters were not successfully cleaved and were being sequenced, they should be removed prior to mapping to avoid poor mapping quality.

3. Align raw reads to reference genome.

Map the trimmed sequence reads to reference genome using tophat2. Specifically, we first create a directory with the sample name to store the output, then call tophat2 and output results to the directory's subdirectory named `Tophat_Out`*. Since this is a mouse sample, UCSC mm10 is used as the reference genome, and the reference transcriptome file and bowtie index files are stored under the* `Indexes` *directory in the home directory. Bowtie2 index files contain the genome sequences to be aligned to in bowtie2 format. Tophat2 uses bowtie2 as the base sequence aligner.*

```
$ mkdir MT1
$ tophat2 -r 200 -p 8 -o MT1/Tophat_Out -G
  ~/Indexes/Mus_musculus/UCSC/mm10/Genes/genes.gtf
  ~/Indexes/Mus_musculus/UCSC/mm10/
  Sequence/Bowtie2Index/genome Fastq/Trim_MT1_1.fastq
  Fastq/Trim_MT1_2.fastq
```

Users should repeat these commands for the other five samples, MT2, MT3, WT1, WT2, WT3, and change the directory and file names accordingly.

Note that with PE reads, the order of the input file names for R1 (forward reads) and R2 (reverse reads) should not be mixed up. Default parameters for tophat2 are used here. The insert size of 200 is used because the average fragments for the data in this example are ~400 bp and common to data sequenced from Illumina HiSeq 2000. This step will take approximately 30 min on each sample for this example. The execution time for this step would depend on the number of reads and number of processing CPU (specified with `-p` *parameter of tophat2) used.*

4. Assemble gene expression from aligned reads.

   Use HTSeq to quantify the number of reads mapped to each gene:

```
$ samtools view
  MT1/Tophat_Out/accepted_hits.sorted.bam | python -m
  HTSeq.scripts.count -q -s no -
  ~/Indexes/Mus_musculus/UCSC/mm10/Genes/genes.gtf >
  MT1/MT1.count.txt
```

   Repeat this command on the other five samples and change the input and output file names accordingly as in the last step. Upon successful execution of this step, six text files suffixed with `count.txt` *will be generated, each prefixed with the sample ID and stored under the subdirectory of the sample, for example* `MT1/MT1.count.txt`, `MT2/MT2.count.txt`, *and so so on.*

   Note that we need to sort the alignment file (BAM file) prior to calling HTSeq, as HTSeq requires the reads ordered by reads identifier. The output of this step will be a tab-delimited text file with two columns and about 24,000 rows, where each row represents a gene, the first column is the gene identifier, and the second column is the number of reads mapped to the gene.

5. Differential gene expression analysis from assembled gene expression.

   Follow the steps as suggested by the DESeq2 manual to carry out the analysis. These standard steps are listed in the following five substeps:

   a. Launch RStudio and load necessary library.

```
> library("DESeq2")
```

   b. Create necessary data object.

```
> sample.names <- sort(paste(c("MT", "WT"), rep(1:3,
  each=2), sep=""))
> file.names <- paste("../", sample.names, "/",
  sample.names, ".count.txt", sep="")
> conditions <- factor(c(rep("MT", 3), rep("WT", 3)))
> sampleTable <- data.frame(sampleName=sample.names,
  fileName=file.names, condition=conditions)
> # read in the HTSeq count data
> ddsHTSeq<-DESeqDataSetFromHTSeqCount
  (sampleTable=sampleTable, directory=".", design=~
  condition)
```

In this substep, we specify the sample identifiers, names of files with gene counts for each sample, and experimental condition(s) for each sample, then pass this information to the `DESeqDataSetFromHTSeqCount` *function to make a* `DESeqDataSet` *object for following analysis.*

c. Run differential gene analysis.

```
> ddsHTSeq <- ddsHTSeq[rowSums(counts(ddsHTSeq)) >
  10,]
> dds <-DESeq(ddsHTSeq)
```

Prior to differential gene analysis, filter out genes with low counts, which are analogous to genes that are not expressed. Then, use the DESeq function for the differential gene analysis based on negative binomial distribution. Specifically, the DESeq function is a wrapper function with multiple default analyses, including estimating the size factors and dispersions, fitting the negative binomial generalized linear model, and performing Wald tests for differential gene analysis (Love, Huber, & Anders, 2014)

d. Quality checks on the samples.

```
> rld <- rlogTransformation(dds, blind=FALSE)
> # Plot PCA plot
> plotPCA(rld, intgroup="condition",
  ntop=nrow(counts(ddsHTSeq)))

> # Plot correlation heatmap
> cU <-cor( as.matrix(assay(rld)))
> cols <- c("dodgerblue3", "firebrick3")[condition]
> heatmap.2(cU, symm=TRUE, col= color
            RampPalette(c("darkblue","white""))(100),
            labCol=colnames(cU), labRow=colnames(cU),
            distfun=function(c) as.dist(1 - c),
              trace="none",
            Colv=TRUE, cexRow=0.9, cexCol=0.9, key=F,
              font=2,
            RowSideColors=cols, ColSideColors=cols)
```

Draw a PCA plot (Fig. 11.15.2A) and correlation heatmap (Fig. 11.15.2B) to visualize whether the samples cluster per their conditions. In this example, samples are clustered into two groups, which are their genotypes. In cases where samples do cluster in groups but the grouping is not according to the experimental conditions or genotypes, this indicates that the samples are clustered by other factors. These factors could be latent biological subtypes or technical factors such as the batch effect. For example, the samples in Figure S1 (can be downloaded from https://bcm.box.com/s/c9q7otvxoog7cmn1b4p9mp8c7thu7q32) are prepared in two different batches and PCA analysis shows two clusters corresponding to the batches, instead of four designed phenotype groups.

e. Output differential gene analysis results.

```
> res <- results(dds, contrast=c("condition", "MT",
  "WT"))
> grp.mean <- sapply(levels(dds$condition),
  function(lvl)
   rowMeans(counts(dds,normalized=TRUE)[,dds$condition
     == lvl]))
> norm.counts <- counts(dds, normalized=TRUE)
> all <- data.frame(res, grp.mean, norm.counts)
> write.table(all, file="DESeq2_all_rm.txt",
  sep="\t")
```

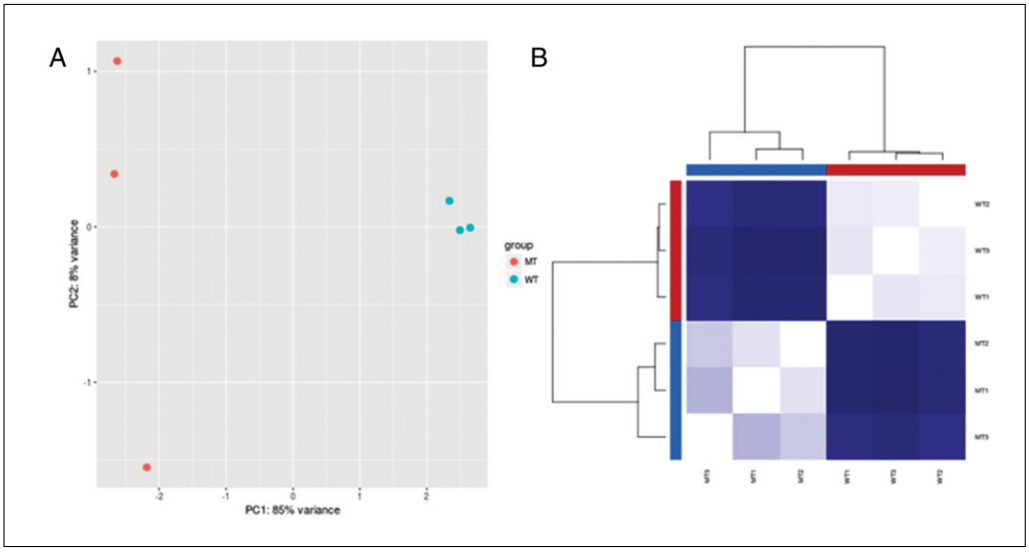

Figure 11.15.2 Visualization inspection of sample clustering. PCA plot (**A**) and heatmap on correlation coefficient between samples (**B**) based on gene expression profiles of the six samples.

Call the results function from DESeq2 package to extract the results from the differential gene analysis. These results include base means across samples, log2 fold changes, standard errors, test statistics, p-values, and adjusted p-values. Then, combine the results obtained with the average expressions for each genotype (`grp.mean`*) and normalized read counts (*`norm.counts`*) for each sample into a data table. Finally, save the data table as a tab-delimited file. Group means and normalized read counts are useful when users want to inspect how a gene is expressed in the experiment. Furthermore, users could get these values and apply to other software for more analysis.*

f. Generate figures on significantly differentiated genes.

```
> plotMA(res, ylim=c(-5,5), alpha = 0.01)

> topGene <- rownames(res)[res$padj <= sort(res$
  padj)[5] &!is.na(res$padj)]
> with(res[topGene, ], {
  points(baseMean, log2FoldChange, col="dodgerblue",
    cex=1.5, lwd=2)
  text(baseMean, log2FoldChange, topGene, pos=2,
    col="dodgerblue")
  })

> library(pheatmap)

> sig.dat <- assay(rld)[res$padj < 0.01 &
    !is.na(res$padj), ]
> annC <- data.frame(condition=conditions)
> rownames(annC) <- colnames(sig.dat)

> pheatmap(sig.dat, scale="row", fontsize_row=9,
           annotation_col = annC)
```

Two common plots to visualize findings from a differential gene analysis are MA plot and gene expression heatmap. The MA plot (Fig. 11.15.3A) shows how gene expressed between two genotypes (log fold-change on y-axis) with respect to overall expression on all samples (on x-axis) highlights the significantly differentially expressed genes (DEGs, adjusted P-value < 0.01) in red, and annotates the five most statistical significant DEGs.

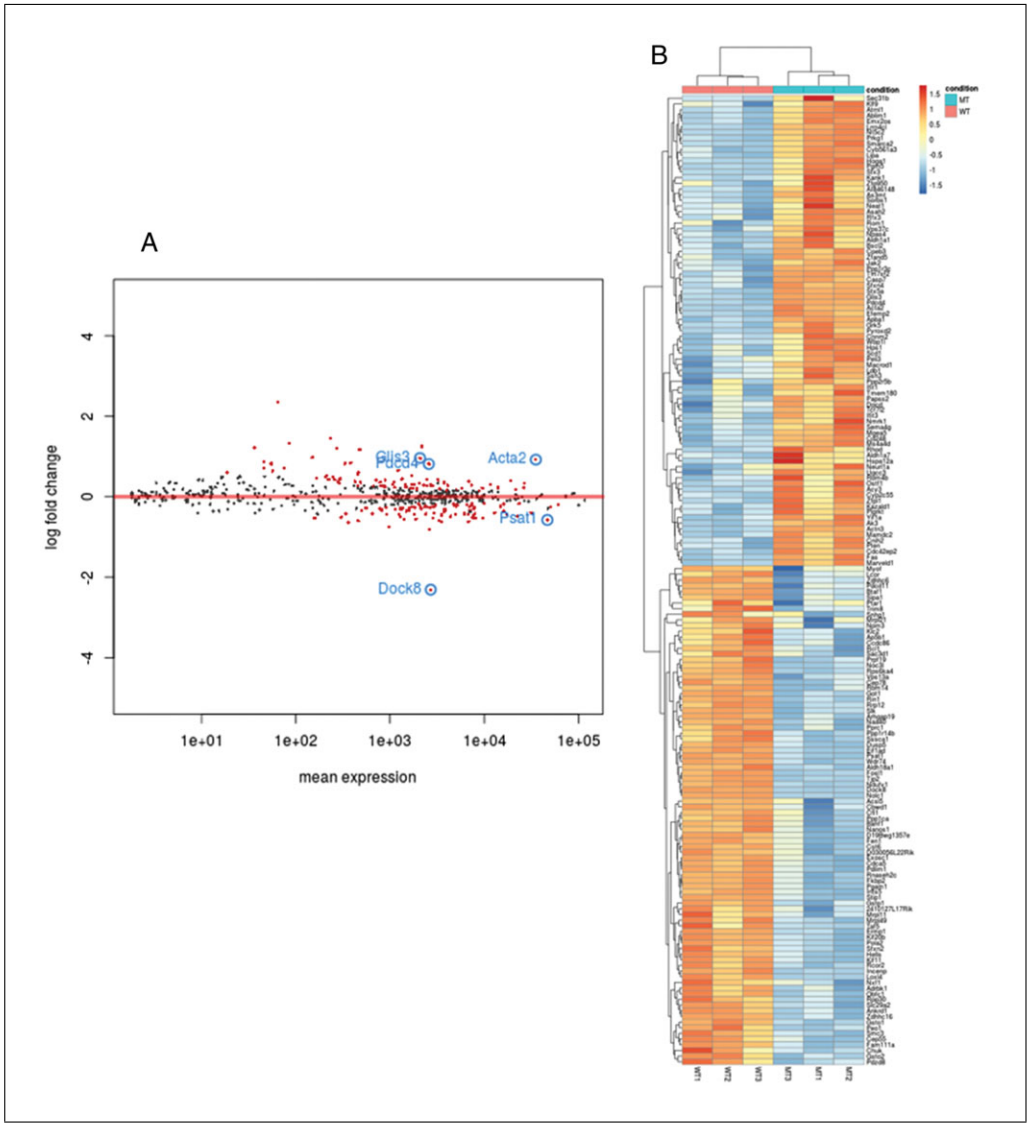

Figure 11.15.3 Results of differential gene expression analysis. MA plot (**A**) and expression heatmap on the DEGs (adjusted $P < 0.01$) (**B**).

In Figure 11.15.3B, a heatmap is used to show the expression pattern of DEGs for all the samples and annotate the samples with their genotypes.

Background information

Since RNA-seq was first presented in studies published in 2008 (Lister et al., 2008; Mortazavi, Williams, McCue, Schaeffer, & Wold, 2008; Nagalakshmi et al., 2008), it has become the default tool to use for whole-transcriptome experiments in less than a decade. Various algorithms, tools, and pipelines were developed to analyze RNA-seq data throughout these years, from bwa, bowtie, tophat, and Cufflinks to the new alignment-free methods such as Kallisto (Conesa et al., 2016). Among different analyses proposed, differential gene analysis, isoform analysis, and splicing events are the three most common analyses carried out in studies with RNA-seq data, with differential gene analysis as the first and default analysis to go with.

Critical parameters and troubleshooting

The first potential problem for this pipeline can be related to the input data: the raw sequence reads. We can detect this risk from FastQC reports in the first step of the protocol. Although we cannot transform the problematic sequence reads into high-quality

sequence reads, we could apply a more stringent threshold in grooming the reads to retain only good-quality reads, in order to prevent the noisy data from being propagated into downstream analysis.

The insertion size for tophat2 (see step 3, above) is another important factor in this pipeline. As discussed above, a wrong insertion size will lead to poor mapping. Subsequently, this will affect the accuracy of quantified expressions and splice-junction detection.

Lastly, reference genome index and gene annotation used are critical. An inaccurate path to these annotations will cause failure during mapping. Wrong annotation with the correct path set would allow the program to execute without error, but the mapping would be poor. This could be detected from low mappability.

Advanced parameters

Standard Illumina RNA-seq data consists of stranded paired-end reads. Note that standard RNA-seq is not strand-specific, and reads are aligned to both the forward or reverse strand regardless if the read is an upstream read (read 1) or downstream read (read 2). In non-strand-specific RNA-seq, antisense genes/transcripts will not be able to be differentiated from the overlapping sense genes/transcripts. A significant number of antisense transcripts are observed from the sense strand, resulting in complementary non-coding RNAs; such non-coding RNAs are often linked to important regulatory functions (Levin et al., 2010). Therefore, other sequencing technologies with library preparation methods such as dUTP and NSR were developed to effectively retain the strand information in sequencing reads.

If the library is prepared such that sequence reads are strand-specific RNA-seq reads, two steps should be modified as follows:

1. In step 3, above, specify `--library-type` to be `fr-firstrand` or `fr-secondstrand` according to the library preparation protocol. In the protocol above, we did not specify this parameter, as tophat2 assumes `--library-type` to be `fr-unstranded` by default.
2. In step 4, above, set `-s` to be 'yes'. We set '`-s no`' above, as we have to specify that it is not strand-specific since the `HTSeq-count` by default is a strand-specific assay.

Suggestions for further analysis

Users will obtain a list of differentially expression genes (DEGs) upon successful completion of this first protocol. The list of DEGs obtained could be used as the input for downstream analysis and functional analysis such as the enrichment of Gene Ontology (GO) terms (Gene Ontology, 2015), gene set enrichment analysis to known pathways, network analysis to study the interactions between the DEGs (Yalamanchili et al., 2014), or even RNA-editing discovery.

Furthermore, users could integrate the transcriptomic findings from RNA-seq with genomics, ChIP-seq, or DNA-methylation to get the systems view of regulatory mechanisms.

BEYOND DEGs: DIFFERENTIAL EXPRESSION AND USAGE OF ISOFORMS

BASIC PROTOCOL 2

A single gene can code multiple proteins with different functions by re-arranging constituent exons (Yalamanchili, Xiao, & Wang, 2012). This process of exon rearrangement (inclusion/skipping) is called splicing, and different forms of the same gene are called isoforms. Standard DEG analysis presented in Basic Protocol 1 is based on overall gene expression and could underutilize significant biological details such as the

isoform-specific expression. Different isoforms vary in expression under different conditions, making them primary targets to explain biological anomalies that cannot be explained by gene-level changes (Garcia-Blanco, Baraniak, & Lasda, 2004). Isoform-specific expression is found to be associated with different diseases, for example the human epidermal growth factor receptor (HER-2) in breast cancer (Menon & Omenn, 2010). It is very crucial to explore expression differences of isoforms to decipher disease mechanisms and therapeutic targets (Yalamanchili et al., 2014). This protocol guides the reader through two analyses to dissect the biological phenomenon at the isoform level: (1) absolute difference in expression for each isoform, and (2) change of relative abundance of isoforms of a gene across conditions.

Necessary Resources

Hardware

A computer or server with access to Unix command environment

Software

Kallisto, Sleuth, Isoform Usage Two-step Analysis: IUTA, samtools, Rstudio. Web resources of respective tools are listed in Table 11.15.1.

Input files

Raw `fastq` reads for differentially expressed isoforms (DEIs) and aligned `bam` files for differential usage of isoforms. Alignment `bam` files must be sorted and indexed (can use the files from step 3 of Basic Protocol 1.

Model reference genome in `fasta` format, gene annotation file in GTF format, and model transcriptome (cDNA sequences).

Web resources of respective tools and files are listed in Table 11.15.1

1. Quantification of isoforms from raw reads.

Typically, in most of the analysis pipelines for RNA-seq, raw reads are first mapped to the reference genome and then quantified, as presented in Basic Protocol 1, above. This is highly time consuming, especially when the number of samples is large. Recent computational approaches have made quantification two orders of magnitude faster by pseudo-aligning reads to a reference, i.e., producing a list of target transcripts to each read while avoiding alignment of individual bases. This step demonstrates the use of Kallisto, an alignment-free, near-optimal isoform-quantification tool.

Kallisto pseudo-alignment is a two-step process: creating a "transcriptome index" and quantifying the transcripts. To begin, first create a directory Index and copy the reference genome fasta files into it. Next, build the index using the Kallisto index command. After building the index, the Kallisto quant command is used to quantify reads with respect to the reference transcriptome. See the following commands:

```
# Create a directory Kallisto_Analysis and copy all
  fastq files to it #
$ cd Kallisto_Analysis
# Creating Index directory #
$ mkdir Index
$ cp ~/genome.fa Index/Reference_Genome.fa
# Building Index #
$ kallisto index -i Index/transcripts.idx
  Index/Reference_Genome.fa
# Running Kallisto #
$ kallisto quant -i Index/transcripts.idx -o
  Kallisto_output/MT1/Quants
-b 100 MT1_1.fastq MT2_2.fastq
```

Parameters: `-i` *is the index built,* `-o` *is the output directory, and* `-b` *is the number of bootstraps required. Isoform quantifications are output to a tab-separated file (*`.tsv`*)* `abundance.tsv`*, and respective bootstrapping results in a file called* `abundance.h5`*. These two files will be stored in the specified output directory (*`Kallisto_output/MT1/Quants` *in this example).*

Repeat the last step (`kallisto quant`*) for the other five samples, MT2, MT3, WT1, WT2, and WT3 by changing the directory and file names accordingly.*

2. Differential expression of isoforms (DEI) analysis.

   Sleuth is an R package for analyzing transcript abundances quantified by Kallisto. It implements statistical algorithms for differential analysis and provides an exploratory data analysis interface through the Shiny framework in RStudio. The following is the commonly used Kallisto-Sleuth pipeline:

   a. Launch RStudio and load necessary library.
   ```
   > library("sleuth")
   ```
   b. Specify the Kallisto results directory where each subdirectory corresponds to a sample. Here `Kallisto_output` has six subdirectories (`WT1`, `WT2`, `WT3`, `MT1`, `MT2`, and `MT3`), one per sample.
   ```
   > base_dir <- "Kallisto_Analysis"
   ```
   c. Getting sample ID information from the Kallisto results directory.
   ```
   > sample_id <- dir(file.path(base_dir,"Kallisto_
      output"))
   ```
   d. The result can be displayed by typing:
   ```
   > sample_id
   ## [1] "MT1" "MT2" "MT3" "WT1" "WT2" "WT3"
   ```
   Lines beginning with `##` show the output of the command.
   e. Assign file paths to sample IDs:
   ```
   > kal_dirs <- sapply(sample_id, function(id)
      file.path(base_dir, "Kallisto_output", id,
      "Quants"))

   > kal_dirs
   ## MT1
   ## "Kallisto_Analysis/Kallisto_output/MT1/Quants"
   ## MT2
   ## "Kallisto_Analysis/Kallisto_output/MT1/Quants"
   ......... , and so on for rest of the samples.
   ```
   f. Reading experimental design from a file (sample names in column 1 and condition in column2) and adding file paths.
   ```
   ### samples_info.txt #
   sample_name    condition
   WT1    WT
   WT2    WT
   MT1    MT  ... so on
   > s2c <- read.table(file.path(base_dir, "samples_
      info.txt"), header = TRUE, stringsAsFactors=
      FALSE)
   > s2c <- dplyr::select(s2c, sample = sample_name,
      condition)
   > s2c <- dplyr::mutate(s2c, path = kal_dirs)
   # Check output
   > print(s2c)
   # prints experimental design and path to respective
      sample quantification files.
   ```

g. *Creating Sleuth object:* This triggers reading in of Kallisto results, normalizing est_counts, normalizing tpm values, merging in metadata, normalizing bootstrap samples, and summarizing bootstraps.

```
# Creating object
> so <- sleuth_prep(s2c, ~ condition)
```

h. Fitting full models and testing using likelihood ratio test.

```
# Fit full model
> so <- sleuth_fit(so)
# Fit reduced model. In this case, the reduced model
   is the intercept-only model:
> so <- sleuth_fit(so, ~1, 'reduced')
# Test #
> so <- sleuth_lrt(so, 'reduced', 'full')
```

i. Writing results to a file and exploratory data analysis and visualization in the Sleuth Shiny app.

```
# Write results to a file #
> results_table <- sleuth_results(so,
   'reduced:full', test_type = 'lrt')
> write.table(results_table,file="Out.txt",sep="\t")
# Visualization in Shiny#
> sleuth_live(so)
```

Sleuth outputs a tab-delimited text file. Each row corresponds to a transcript. The user can screen for statistically significant differentially expressed transcripts by applying a q-value cutoff, column 4 in the example output (provided as supplementary file; can be downloaded from https://bcm.box.com/s/c9q7otvxoog7cmn1b4p9mp8c7thu7q32). Sleuth also provides an intuitive exploratory data analysis and visualization interface. As described in Basic Protocol 1, several analytical plots like PCA, heatmap, MA, scatter plots, density, and fragment distribution can be plotted using the Sleuth Shiny app. Representative plots are shown in Figure 11.15.4. PCA and heatmaps plot are discussed in Basic Protocol 1 above. Scatter plots and density plots are useful in comparing transcriptome expressions of any two samples (Drăghici, 2012).

3. Differential usage of isoforms analysis.

Cells often express different isoforms of a gene at different concentrations. This is one of the key mechanisms controlling cell fate and tissue differentiation. Any imbalance in expression proportions could lead to adverse effects in several diseases, including cancer. For example, imbalance between the long and short isoforms of B cell lymphoma affects lung cancer prognosis (Dou et al., 2010). Standard analysis of individual isoforms cannot completely decipher the shift in isoform proportions. It is very possible to have expression level differences with no change in relative abundance. The following steps demonstrate how to identify differential isoform usage using Isoform Usage Two-step Analysis (IUTA; Niu, Huang, Umbach, & Li, 2014).

a. First, create a directory named DUI and copy all the alignment (bam) files by renaming them to sample names. Bam files from step 3 of Basic Protocol 1 can be used here. Use samtools to sort and index all bam files.

```
$ mkdir DUI
$ cp MT1/Tophat_Out/accepted_hits.sorted.bam.bam
   DUI/MT1.bam
$ samtools sort MT1.bam MT1.sorted
$ samtools index MT1.sorted.bam
```

Repeat this step for the other five samples `MT2`, `MT3`, `WT1`, `WT2`, *and* `WT3`, *and change the directory and file names accordingly.*

b. Launch RStudio and load IUTA library.

```
> library("IUTA")
```

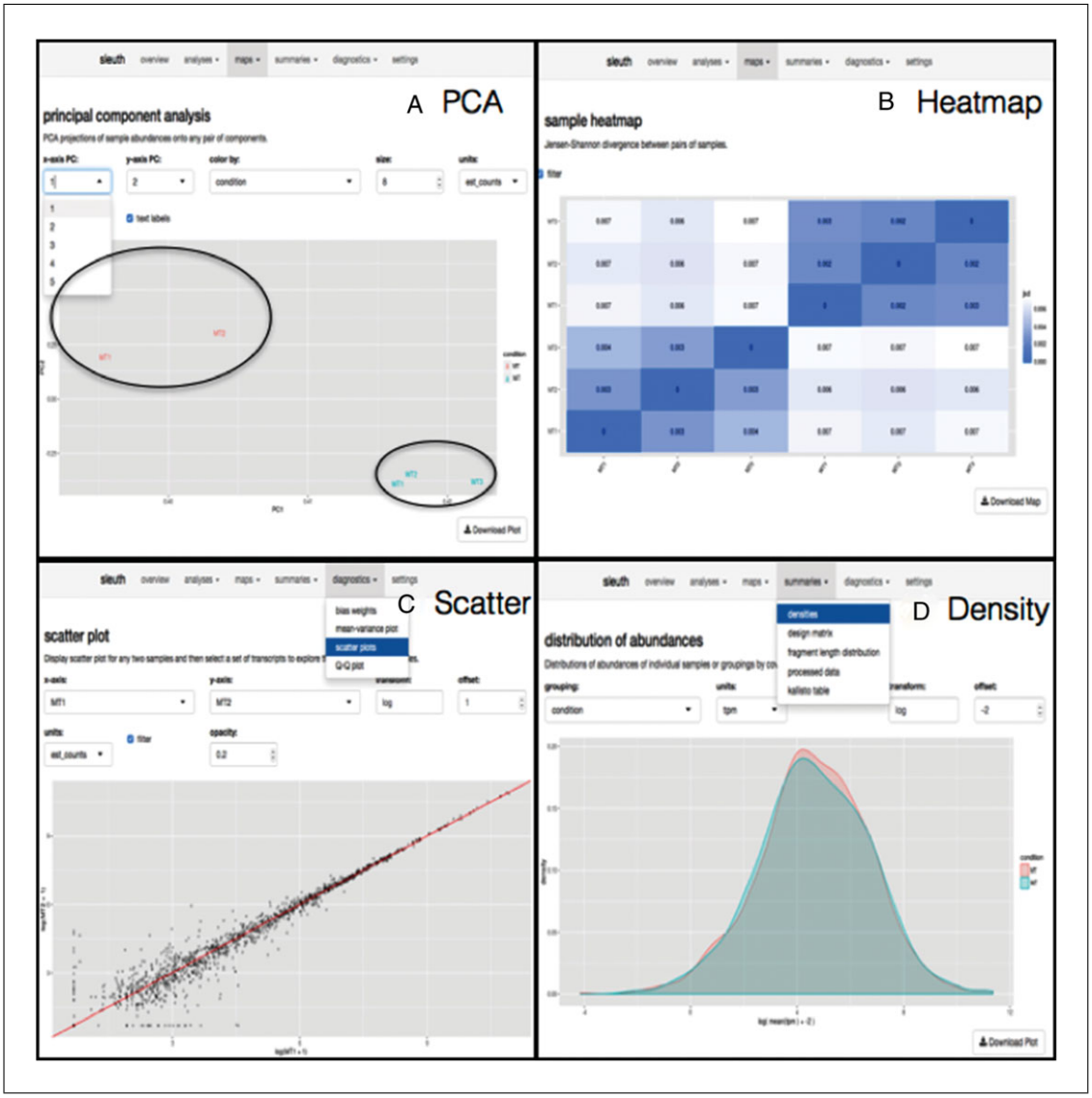

Figure 11.15.4 Representative plots from Sleuth analysis shiny app. (**A**) PCA, (**B**) heatmap, (**C**) scatterplot, and (**D**) density plots.

c. Set input, output and other parameters.

```
# Output directory #
> outdir <- "IUTA_OUT"

# number of core to use #
> ncores <- 6

# Control files #
> bam.list.1<-c('WT1.sorted.bam','WT2.sorted.bam',
  'WT3.sorted.bam')
> rep.info.1 = rep(1,length(bam.list.1))

# Treatment files #
> bam.list.2<-c('MT1.sorted.bam','MT2.sorted.bam',
  'MT3.sorted.bam')
> rep.info.2 = rep(1,length(bam.list.2))
```

d. Provide reference transcriptome annotations in gtf format.

```
> transcript.info <- '/Projectdirectory/genes.gtf'
```

e. Run IUTA command with desired parameters.

```
> IUTA(bam.list.1, bam.list.2, transcript.info,rep.
  info.1,rep.info.2,output.dir =outdir,
        output.na = FALSE,
        genes.interested = "all",
        strand.specific = rep("1.5",length
          (rep.info.1)+length(rep.info.2)),
        gene.filter.chr = c("_", "M", "Un"),
        mapq.cutoff = NA, alignment.per.kb.cutoff =
          10,
        IU.for.NA.estimate = "even",
        sample.FLD = FALSE, FLD = "empirical",
        mean.FL.normal = NA, sd.FL.normal = NA,
        number.samples.EFLD = 1e+06,
        isoform.weight.cutoff = 1e-4,
        adjust.weight = 1e-4, epsilon = 1e-05,
        test.type = "SKK", log.p = FALSE, fwer =
          1e-2,
        mc.cores.user = ncores)
```

A few important parameters for IUTA are: `alignment.per.kb.cutoff` *to specify the minimum expression value (in number of reads) to be considered,* `mapq.cutoff` *for read quality,* `strand.specific` *to indicate the sample library strand type (1, 2 and 1.5 for sense, anti-sense and un-stranded libraries), and* `FLD` *to specify what fragment length distribution to be used. Two options are available for* `FLD`*: normal or empirical; empirical is recommended when we are not sure about the distribution and wish to let the program estimate from the data. For most analyses "SKK" is recommend for* `test.type`*.*

In general, IUTA first estimates transcript abundance and then tests for differential usage of isoforms. Upon successful execution, IUTA will output two files: `estimates.txt` *with sample-wise transcript estimates and* `p_values.txt` *with results indicating the significance of differential isoform usage.*

f. Visualize results for a gene with differential isoform usage with the intuitive pie chart plot. Below is an example on inspecting the differential usage of gene *Gfra1*.

```
> gene<-"Gfra1"
> pie_compare(gene, 3, estimates.file =
          "estimates.txt",
          geometry = "Euclidean", adjust.weight =
              1e-300,
          output.file = paste("Pieplot_",gene,
              ".pdf", sep = ""),
          group.name = c("WT", "MT"),
          output.screen=FALSE)
```

A pie chart is plotted based on isoform abundance estimates from the `estimates.txt` *output from the IUTA function in the last step. The pie charts from the command above are shown in Figure 11.15.5.*

Background information

A typical transcript-level analytical pipeline will first align sequence reads to a reference and then estimate the abundance. This approach is highly time consuming and trails far behind the accelerating next-generation sequencing rate. For instance, widely used programs like TopHat2 and Cufflinks require 20 hr to map and 14 hr to quantify 20 samples, each with 30 million reads on a 20-core computer (Bray, Pimentel, Melsted, & Pachter, 2016). Despite several streamlined quantification algorithms proposed to speed up the pipeline, alignment overhead cannot be avoided. To address these challenges,

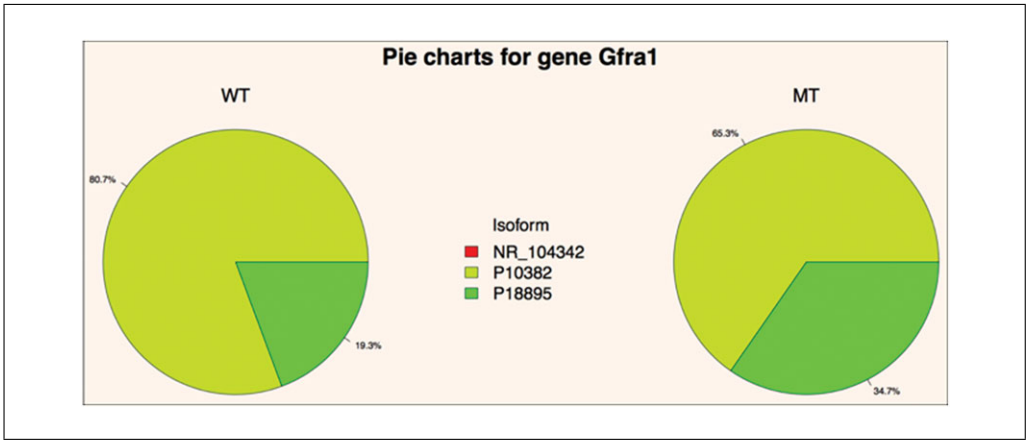

Figure 11.15.5 Pie chart showing differential isoform usage for gene *Gfra1* between wild type (WT) and mutant (MT) samples. The usage of isoform P10382 is decreased from 80.7% in WT to 65.3% in MT samples. On the other hand, the usage of P18895 increased in MT samples.

ultra-fast lightweight alignment-free quantification methods including Kallisto and Salmon have been proposed. Kallisto pseudo-aligns reads to a reference transcriptome rather than to a reference genome, and is faster than previous methods by two orders of magnitude without compromising on accuracy (Bray et al., 2016; Jin, Wan, & Liu, 2017). Though transcriptome analysis can drive new insights, it is limited in terms of understanding non-coding genome, transcriptional regulation, and other non-coding variations (Conesa et al., 2016).

Critical parameters

Kallisto is an expectation maximization (EM)–based, alignment-free quantification method. As in any EM-based algorithm, sample size is the main factor affecting the estimation. In this scenario, the number of reads available for quantification is the sample size. Thus, to avoid any quantification bias, it is important to design experiments such that all the samples have comparable sequencing depth.

To build the Kallisto index, the authors of Kallisto recommended a default "kmer" value of 31 (maximum kmer allowed). However, the kmer value should be adjusted according to the sequenced read length—a smaller kmer is used for shorter reads. It is recommended to try out different kmer values and do a saturation check.

Performance of EM-based methods will drop with the increase in number of isoform for a gene. Both IUTA and Kallisto suffer from this issue. Any significant hits with high number of isoforms should be interpreted cautiously.

Fragment length distribution (`FLD`) used in IUTA is a critical parameter. Since different samples may have different fragment lengths, it is always recommended to set `FLD` to "empirical" for the distribution to be computed for each sample based on the sample input itself.

Troubleshooting

Processing data from public repositories like GEO (gene expression omnibus) may cause read-pair mismatch issues with IUTA. This can be fixed by renaming paired-read IDs to be the same.

Advanced parameters

Similar to advanced parameters section discussed in Basic Protocol 1, the strand-specific RNA-seq library should be handled slightly differently. Specifically, the quantification step in step 1 of this protocol should be modified as follows:

```
$ kallisto quant [--single/--fr-stranded/--rf-stranded]
  --i index fastq files
```

Use `--single` for single-end data, `--fr-stranded` for first read forward, and `--rf-stranded` for first read reverse.

BASIC PROTOCOL 3

DEEP INTO RNA-seq: CRYPTIC SPLICING

With recent advances in deep sequencing technologies, a significant number of novel splice junctions are observed in several eukaryotic datasets (Kapustin et al., 2011). Such un-annotated cryptic splice cites (CSS) are usually inactive or recognized only at very low levels (Green, 1986). Because of their dormant nature, cryptic splice sites are often ignored as noise. However, it has only recently been shown that cryptic sites can be activated if a nearby canonical splice is mutated (Padgett, Grabowski, Konarska, Seiler, & Sharp, 1986). Cryptic site activation is also linked to a wide range of genetic diseases (Wang & Cooper, 2007). Almost 50% of disease-causing mutations disrupt alternative splicing either by mutating canonical splice sites or by activating CSS. Several recent studies have established the role of cryptic site activation in various human diseases, from cancers to neurological diseases like amyotrophic lateral sclerosis (ALS) and frontotemporal dementia (FTD) (Ling, Pletnikova, Troncoso, & Wong, 2015). Owing to their potential roles in disease, it is extremely important to thoroughly understand the mechanism of CSS activation. Identification of target cryptic sites is the first step to decode any CSS-dependent disease mechanism. The following protocol guides the reader through CrypSplice (Tan et al., 2016), a novel computational approach to identify cryptic splice sites from RNA-seq data.

Necessary Resources

Hardware

A computer or server with access to Unix command environment

Software

CrypSplice, Bedtools version 2.17 or higher, R libraries ibb and MASS, Integrative Genomics Viewer (IGV), and samtools. Web resources of respective tools are listed in Table 11.15.1.

Input files

Aligned *bam* files. Files must be sorted and indexed (one can use the files from step 3 of Basic Protocol 1)

Junction bed files, also from step 3 of Basic Protocol 1

Genome model files: alternative splicing events, gene model, and junction database files (can be downloaded directly from the CrypSplice link; see Table 11.15.5).

1. Preparing alignment and junction files.

   CrypSplice needs alignment files in bam format and junction files in bed format. Refer to steps 1 to 3 in Basic Protocol 1 to get bam and bed files from raw fastq reads using tophat2. Create a directory named `CrypSpliceAnalysis` *and copy all the required input files mentioned above into it. By default, alignment files and junction files (from step 3 of Basic Protocol 1) are saved as* `accepted_hits.sorted.bam` *and* `junctions.bed` *respectively. Rename them according to sample names while copying. Then, sort and index all the bam files using samtools. The sort command will output sorted alignment files based on chromosomal location. The samtools index command will create an index file for the sorted bam file. This index is used by other tools like bedtools to quickly extract the information.*

   ```
   $ mkdir CrypSpliceAnalysis
   ```

```
$ cp MT1/Tophat_Out/accepted_hits.sorted.bam
  CrypSpliceAnalysis/MT1.bam
$ cp MT1/Tophat_Out/junctions.bed
  CrypSpliceAnalysis/MT1.bed
$ samtools sort CrypSpliceAnalysis/MT1.bam
  CrypSpliceAnalysis/MT1.sorted
$ samtools index CrypSpliceAnalysis/MT1.sorted.bam
```

Repeat these commands for other five samples, MT2, MT3, WT1, WT2, and WT3 by changing the directory and file names accordingly.

2. Running CrypSplice.

Next, run CrypSplice on sorted and indexed bam and junction files with user-desired parameters.

```
# Download genome annotation files from
  http://www.liuzlab.org/CrypSplice/ to the #
  directory CrypSpliceAnalysi and uzip the downloaded
  file.
$ cd CrypSpliceAnalysis
$ python CrypSplice.py -C WT1.bed,WT2.bed,WT3.bed -T
  MT1.bed,MT2.bed,MT3.bed -G MM10 -F 10 -M 0.95 -P 6
```

Input junction files should be indicated in a comma-separated list followed by parameters `-C` *and* `-T` *for control and treatment samples, respectively. Known alternative splicing events, gene models, and known junction database files (files for model organisms available from http://www.liuzlab.org/CrypSplice/) should be stored in the genome directory specified by* `-G`*. The noise filter* `-F` *is used to indicate that any junction less than this threshold is considered noise and will be discarded from analysis. The junction match* `-M` *followed by a floating value ranging from* `0` *to* `1` *specifies the minimum fractional overlap to filter out known junctions. Parallel processes,* `-P`*, specifies the number of processing cores used to run CrypSplice (one core per sample is recommended). More details on how to choose appropriate parameters for CrypSplice will be discussed in the critical parameters section below. In brief, CrypSplice first filters out noisy junctions and all annotated junctions provided in genome model directory from the junction files provided. Then, 5′ exon coverage is computed for every junction. Junction counts and respective exon counts are used to test for statistical significance using a beta binomial model. Finally, every cryptic junction is adjusted for multiple testing and classified into three categories: junction gains, junction losses, and differential junctions.*

3. Filtering and visualizing results.

CrypSplice outputs results to three tab-delimited text files as follows.

Junction gains: Cryptic junctions observed only in treatment samples.

Junction losses: Cryptic junctions observed only in control samples.

Differential junctions: Cryptic junctions observed in both control and treatment samples but with different expression strengths.

Statistically significant results can be filtered using an adjusted p-value cutoff. Finally, junctions can be visualized in Integrative Genomics Viewer (IGV) with the following steps: (i) Choose appropriate genome, mm10 for this example. (ii) Load bam files to IGV using the "File" load option. (iii) Paste junction coordinates of interest to visualize junction reads (Fig. 11.15.6A). (iv) Right click on any track to open a menu and choose sashimi plot. Sashimi plot is an intuitive junction visualization graph showing both exon coverage and junction connections. Example sashimi plots from the demo data are shown in Figure 11.15.6B.

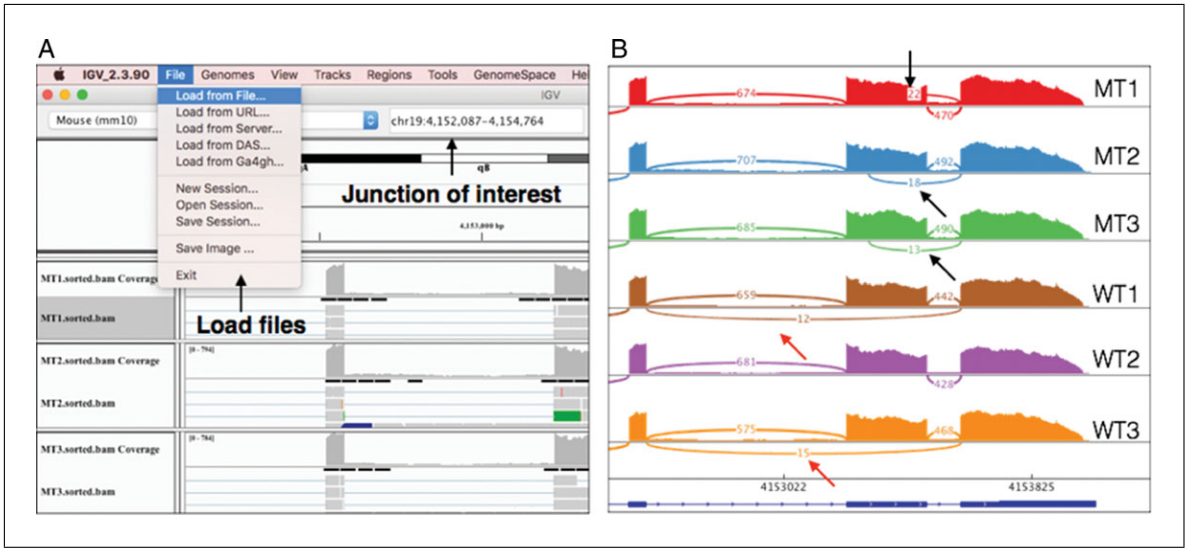

Figure 11.15.6 Visualizing splicing events in IGV. (**A**) Screen shot showing exon coverage and (b) Sashimi plots with black arrows pointing to junction gains and red arrows pointing to junction losses.

Background information

Recently, more studies have shown the important connections of cryptic splice sites to various human diseases, ranging from cancer to neurological disease such as amyotrophic lateral sclerosis. Despite huge interest from the transcriptomic communities over the past decade, we continue to lack a robust computational approach to identify cryptic splice sites from RNA-seq data. CrypSplice is the first computational tool focusing exclusively on identifying cryptic splice sites from RNA-seq data. It uses a beta-binomial model to effectively handle both inter- and intra-sample variance. Its merit is apparent from the large number of TDP-43-dependent splicing defects identified that were not previously known (Tan et al., 2016). Supporting experimental validations also confirm the practical usability of the method.

Critical parameters

Noise threshold (`-F`*):* Noisy alignments are unavoidable in any high-throughput sequencing data set. However, these noisy alignments should not be recognized as cryptic events. CrypSplice uses a user-defined noise threshold to filter out these noisy alignments. By default, a cutoff of 10 reads is used to filter out noisy junctions. However, it is recommended to choose this cutoff as a function of sequencing depth, with a large cutoff for higher coverage and vice versa.

Junction match (`-M`*):* CrypSplice filters out all known junctions from those that are observed, in order to detect cryptic sites. Any minor mapping errors could result in putative cryptic sites. To account for such mapping shifts, CrypSplice allows a junction match threshold. Any two junctions falling within the cutoff are collapsed into one. However, if the threshold is weak, the potential novel junction can be collapsed. It is recommended to choose this cutoff as a function of anchor length used to map junction reads (0.95 is used in this protocol). One should use a higher cutoff for longer anchor length.

Troubleshooting

CrypSplice is mainly dependent on the bedtools and ibb packages. Any changes or upgrades in these packages can potentially cause problems. In case of any issues, downgrading to the original reported versions of bedtools (v2.17) and ibb package (13.06) can help. Alternatively, one can edit the bedtools `intersect` and `coverage` commands accordingly in the file `CrypSplice.py`.

GUIDELINES FOR UNDERSTANDING RESULTS

Upon successful execution of the steps described in the protocols above, one obtains a list of candidates that are dysregulated at the gene, isoform, and junction levels. With the candidate list, users can further investigate the regulatory mechanisms that are disturbed due to the experiment, for example, from a functional analysis on the candidate genes, gene-set enrichment analysis, etc. (Pathan et al., 2015).

Although the pipeline was successfully executed, problems with the analysis may still exist. A few common red flags can be checked while interpreting the results:

- Poor quality of raw sequencing reads can be detected from the FastQC step (step 1 of Basic Protocol 1). If FastQC reports failures on many metrics, the quality of the library being sequenced might have been compromised.
- Poor overall mappability of the samples. Overall mappability of typical paired-end RNA-seq data is 80% or higher. Quality of the sample and library will lead to variation in mappability. For example, degraded samples such as frozen tissue, degraded RNA, or a library with high GC content will pose a challenge in mapping and lead to lower mappability.
- Discordant mappability of 10% or higher. This hints that a large number of the paired-end reads were not mapped according to the expected pair-reads assumptions, such as the insertion size and pairing of the input reads. In this case, users should make sure that the input files are in the correct order and the reads in the files are in pairs in the same order. In addition, users can tune the insertion size parameter (`-r` parameter of tophat2) to obtain better mappability.
- Outlier sample(s) observed from PCA plot and correlation heatmap. If a sample was not clustered with the other samples of the same genotype from both plots, it could be considered as an outlier. Users can remove the sample(s), re-normalize the counts, and plot the PCA and correlation heatmaps again to assess the sample clusters. Due to limited number of samples in most experiments (less than five samples per group), outlier detection with PCA and heatmap should be interpreted cautiously. In addition, users should assess sample clustering on other factors such as batch, library preparation, gender of mice, types of tissues, and others to check if the samples cluster according to some other factor instead of the designed treatment.
- Differentially expressed genes and isoforms can be filtered and ranked based on adjusted *P*-value and fold change filters, respectively. Adjusted *P*-value ≤ 0.05 and fold change ≥ 1.5 (at least 50% change) should suffice for a majority of studies. Fold-change cutoff can be adjusted according to the impact of treatment, with smaller values to capture subtle changes and vice versa.
- Huge differences in isoform usage are attractive. However, it is strongly recommended to cross check the estimated isoform proportions with actual mapping using IGV. The EM (expectation maximization) procedure of isoform quantification may introduce some bias.
- Cryptic junction changes are classified into junction gains, junction losses, and differential junctions, as described in step 3 of Basic Protocol 3. Two consecutive junction gains between any two consecutive exons indicate cryptic exon inclusion. On the other hand, a loss of junction suggests intron retention. A single junction gain corresponds to an alternative start/stop according to the strand orientation. Though junctions can be ranked based on relative strength (junction score), the absolute baseline should always also be considered in selecting top hits. High relative junction strength (junction score) does not guarantee high impact/cryptic activity.

ACKNOWLEDGMENTS

This work is partially supported by NSF [DMS 1263932], CPRIT [RP170387], NIH [R01 GM120033], Houston Endowment, Huffington Foundation, and Belfer Foundation. Y.W is also supported by NIH [5R01HD079442-02, and 1RF1AG054111-01].

LITERATURE CITED

Anders, S., Pyl, P. T., & Huber, W. (2015). HTSeq: A Python framework to work with high-throughput sequencing data. *Bioinformatics*, *31*(2), 166–169. doi: 10.1093/bioinformatics/btu638.

Bray, N. L., Pimentel, H., Melsted, P., & Pachter, L. (2016). Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, *34*(5), 525–527. doi: 10.1038/nbt.3519.

Conesa, A., Madrigal, P., Tarazona, S., Gomez-Cabrero, D., Cervera, A., McPherson, A., ... Mortazavi, A. (2016). A survey of best practices for RNA-seq data analysis. *Genome Biology*, *17*, 13. doi: 10.1186/s13059-016-0881-8.

Dou, T., Xu, J., Gao, Y., Gu, J., Ji, C., Xie, Y., & Zhou, Y. (2010). Evolution of peroxisome proliferator-activated receptor gamma alternative splicing. *Frontiers in Bioscience (Elite Edition)*, *2*, 1334–1343.

Drăghici, S. (2012). *Statistics and data analysis for microarrays using R and Bioconductor*. Boca Raton, FL: CRC Press.

Garcia-Blanco, M. A., Baraniak, A. P., & Lasda, E. L. (2004). Alternative splicing in disease and therapy. *Nature Biotechnology*, *22*(5), 535–546. doi: 10.1038/nbt964.

Gene Ontology Consortium. (2015). Gene ontology consortium: Going forward. *Nucleic Acids Research*, *43*(Database issue), D1049–1056. doi: 10.1093/nar/gku1179.

Green, M. R. (1986). Pre-mRNA splicing. *Annual Review of Genetics*, *20*, 671–708. doi: 10.1146/annurev.ge.20.120186.003323.

Jin, H., Wan, Y. W., & Liu, Z. (2017). Comprehensive evaluation of RNA-seq quantification methods for linearity. *BMC Bioinformatics*, *18*(Suppl 4), 117. doi: 10.1186/s12859-017-1526-y.

Kapustin, Y., Chan, E., Sarkar, R., Wong, F., Vorechovsky, I., Winston, R. M., ... Dibb, N. J. (2011). Cryptic splice sites and split genes. *Nucleic Acids Research*, *39*(14), 5837–5844. doi: 10.1093/nar/gkr203.

Kim, D., Pertea, G., Trapnell, C., Pimentel, H., Kelley, R., & Salzberg, S. L. (2013). TopHat2: Accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biology*, *14*(4), R36. doi: 10.1186/gb-2013-14-4-r36.

Levin, J. Z., Yassour, M., Adiconis, X., Nusbaum, C., Thompson, D. A., Friedman, N., ... Regev, A. (2010). Comprehensive comparative analysis of strand-specific RNA sequencing methods. *Nature Methods*, *7*(9), 709–715. doi: 10.1038/nmeth.1491.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., ... 1000 Genome Project Data Processing Subgroup (2009). The sequence alignment/map format and SAMtools. *Bioinformatics*, *25*(16), 2078–2079. doi: 10.1093/bioinformatics/btp352.

Ling, J. P., Pletnikova, O., Troncoso, J. C., & Wong, P. C. (2015). TDP-43 repression of nonconserved cryptic exons is compromised in ALS-FTD. *Science*, *349*(6248), 650–655. doi: 10.1126/science.aab0983.

Lister, R., O'Malley, R. C., Tonti-Filippini, J., Gregory, B. D., Berry, C. C., Millar, A. H., & Ecker, J. R. (2008). Highly integrated single-base resolution maps of the epigenome in Arabidopsis. *Cell*, *133*(3), 523–536. doi: 10.1016/j.cell.2008.03.029.

Lo, Y. H., Chung, E., Li, Z., Wan, Y. W., Mahe, M. M., Chen, M. S., ... Shroyer, N. F. (2017). Transcriptional regulation by ATOH1 and its target SPDEF in the intestine. *Cellular and Molecular Gastroenterology and Hepatology*, *3*(1), 51–71. doi: 10.1016/j.jcmgh.2016.10.001.

Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, *15*(12), 550. doi: 10.1186/s13059-014-0550-8.

Menon, R., & Omenn, G. S. (2010). Proteomic characterization of novel alternative splice variant proteins in human epidermal growth factor receptor 2/neu-induced breast cancers. *Cancer Research*, *70*(9), 3440–3449. doi: 10.1158/0008-5472.CAN-09-2631.

Mortazavi, A., Williams, B. A., McCue, K., Schaeffer, L., & Wold, B. (2008). Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature Methods*, *5*(7), 621–628. doi: 10.1038/nmeth.1226.

Nagalakshmi, U., Wang, Z., Waern, K., Shou, C., Raha, D., Gerstein, M., & Snyder, M. (2008). The transcriptional landscape of the yeast genome defined by RNA sequencing. *Science*, *320*(5881), 1344–1349. doi: 10.1126/science.1158441.

Niu, L., Huang, W., Umbach, D. M., & Li, L. (2014). IUTA: A tool for effectively detecting differential isoform usage from RNA-Seq data. *BMC Genomics [Electronic Resource]*, *15*, 862. doi: 10.1186/1471-2164-15-862.

Padgett, R. A., Grabowski, P. J., Konarska, M. M., Seiler, S., & Sharp, P. A. (1986). Splicing of messenger RNA precursors. *Annual Review of Biochemistry*, *55*, 1119–1150. doi: 10.1146/annurev.bi.55.070186.005351.

Pathan, M., Keerthikumar, S., Ang, C. S., Gangoda, L., Quek, C. Y., Williamson, N. A., ... Mathivanan, S. (2015). FunRich: An open access standalone functional enrichment and interaction network analysis tool. *Proteomics*, *15*(15), 2597–2601. doi: 10.1002/pmic.201400515.

Pham, T. V., Piersma, S. R., Warmoes, M., & Jimenez, C. R. (2010). On the beta-binomial model for analysis of spectral count data in label-free tandem mass spectrometry-based proteomics. *Bioinformatics*, *26*(3), 363–369. doi: 10.1093/bioinformatics/btp677.

Quinlan, A. R., & Hall, I. M. (2010). BEDTools: A flexible suite of utilities for comparing genomic features. *Bioinformatics*, *26*(6), 841–842. doi: 10.1093/bioinformatics/btq033.

Sztainberg, Y., Chen, H. M., Swann, J. W., Hao, S., Tang, B., Wu, Z., ... Zoghbi, H. Y. (2015). Reversal of phenotypes in MECP2 duplication mice using genetic rescue or antisense oligonucleotides. *Nature*, *528*(7580), 123–126. doi: 10.1038/nature16159.

Tan, Q., Yalamanchili, H. K., Park, J., De Maio, A., Lu, H. C., Wan, Y. W., ... Zoghbi, H. Y. (2016). Extensive cryptic splicing upon loss of RBM17 and TDP43 in neurodegeneration models. *Human Molecular Genetics*, *25*(23), 5083–5093. doi: 10.1093/hmg/ddw337.

Thorvaldsdottir, H., Robinson, J. T., & Mesirov, J. P. (2013). Integrative Genomics Viewer (IGV): High-performance genomics data visualization and exploration. *Briefings in Bioinformatics*, *14*(2), 178–192. doi: 10.1093/bib/bbs017.

Wang, G. S., & Cooper, T. A. (2007). Splicing in disease: Disruption of the splicing code and the decoding machinery. *Nature Reviews Genetics*, *8*(10), 749–761. doi: 10.1038/nrg2164.

Yalamanchili, H. K., Li, Z., Wang, P., Wong, M. P., Yao, J., & Wang, J. (2014). SpliceNet: Recovering splicing isoform-specific differential gene networks from RNA-Seq data of normal and diseased samples. *Nucleic Acids Research*, *42*(15), e121. doi: 10.1093/nar/gku577.

Yalamanchili, H. K., Xiao, Q. W., & Wang, J. (2012). A novel neural response algorithm for protein function prediction. *BMC Systems Biology*, *6*(Suppl 1), S19. doi: 10.1186/1752-0509-6-S1-S19.

Yalamanchili, H. K., Yan, B., Li, M. J., Qin, J., Zhao, Z., Chin, F. Y., & Wang, J. (2014). DDGni: Dynamic delay gene-network inference from high-temporal data using gapped local alignment. *Bioinformatics*, *30*(3), 377–383. doi: 10.1093/bioinformatics/btt692.