Aplicações da Linguagem R em Análises de Vegetação

Écio Souza Diniz
Pedro Manuel Villa
(Organizadores)

Atena Editora
Ano 2020

Aplicações da Linguagem R em Análises de Vegetação

Écio Souza Diniz
Pedro Manuel Villa
(Organizadores)

Atena Editora
Ano 2020

2020 *by Atena Editora*

Editora Chefe: Profª Drª Antonella Carvalho de Oliveira
Diagramação: Geraldo Alves
Edição de Arte: Lorena Prestes
Revisão: Os Autores

Atena Editora
Ano 2020

Atena Editora
Ano 2020

Dados Internacionais de Catalogação na Publicação (CIP)
(eDOC BRASIL, Belo Horizonte/MG)

A642 Aplicações da linguagem R em análises de vegetação [recurso eletrônico] / Organizadores Écio Souza Diniz, Pedro Manuel Villa. – Ponta Grossa, PR: Atena, 2020.

Formato: PDF
Requisitos de sistema: Adobe Acrobat Reader
Modo de acesso: World Wide Web
Inclui bibliografia
ISBN 978-65-86002-35-5
DOI 10.22533/at.ed.355200903

1. Desenvolvimento sustentável. 2. R (Linguagem de programação de computador). 3. Recursos vegetais – Manejo. I.Diniz, Écio Souza. II. Villa, Pedro Manuel.

CDD 333.7511

Elaborado por Maurício Amormino Júnior – CRB6/2422

Atena Editora
Ponta Grossa – Paraná - Brasil
www.atenaeditora.com.br
contato@atenaeditora.com.br

APRESENTAÇÃO

Os diferentes tipos de vegetação ao redor do globo, principalmente as florestas tropicais, se destacam por fornecer importantes bens e serviços ecossistêmicos para a humanidade como, por exemplo, regulação climática, provisão de alimentos e diversas fontes de energia. Contudo, as crescentes e rápidas mudanças no meio ambiente causadas por sua intensa exploração no século 21 têm promovido reduções drásticas de importantes vegetações distribuídas em distintos Biomas. O Brasil como um país de dimensão continental e rico em recursos vem atravessando profundas transformações em seus Biomas, o que é destacadamente devido aos usos intensos da terra sem técnicas adequadas de manejo para a sua exploração.

Diante desse panorama de significativas transformações do meio natural, se faz necessário e urgente o estudo de diferentes tipos de comunidades vegetais e ecossistêmicas para definir estratégias de manejo e conservação, assim como pesquisas que visem a otimização de produções agrícolas de forma sustentável. A união de compreensão ecológica precisa e adequadas técnicas de manejo permitem uma exploração sustentável a longo-prazo dos recursos vegetais, assegurando manutenção de diversidade e provisões para o futuro.

A execução de estudos robustos para alcançar essa interface entre conservação e exploração demanda o uso de eficientes ferramentas analíticas. Dentre essas ferramentas, as linguagens de programação têm se sido importantes aliadas para obtenções de predições e resultados estatísticos confiáveis e informativos. A linguagem contida no software R é a mais amplamente utilizada para processamento de dados e análises de vegetação. O R engloba diversos pacotes importantes para análises de dados de plantas em diversos contextos ecológicos e agrários. Com seus diversos pacotes, o R permite a busca mais apurada pela compreensão de padrões e processos ecológicos, avaliação de impactos antrópicos sobre vegetação, monitoramentos e previsões de condições do solo para plantios e predições de efeitos de mudanças climáticas em florestas. Essa gama de possibilidades analíticas amplifica o acerto em tomadas de decisão com relação ao uso dos nossos recursos naturais de forma geral.

Este livro tem como objetivo trazer uma compilação de algumas potencialidades do software R para análise de vegetação, contribuindo para o aumento da capacidade técnica de diversos profissionais das áreas de Ciências da Terra ou Naturais no uso dessa poderosa ferramenta analítica. Para tal, os capítulos aqui presentes discorrem de forma aplicada sob temas em contextos ecológicos e agrários. Todos os capítulos possuem links de compartilhamento livre de dados e scripts com códigos para execução das análises que eles abordam no R. Assim, desejamos que o conteúdo aqui presente auxilie você leitor (a) em sua tarefa analítica, amplificando a obtenção de resultados informativos e potenciais de aplicação prática.

Écio Souza Diniz

Pedro Manuel Villa

SUMÁRIO

CAPÍTULO 11

PIPELINE DE EXPRESSÃO DIFERENCIAL EM R APLICADO À *Arabidopsis thaliana*

Data de aceite: 12/02/2020

Sheila Tiemi Nagamatsu
Laboratório de Genômica e BioEnergia, Departamento de Genética e Evolução, Microbiologia e Imunologia, UNICAMP, Campinas, São Paulo, 13083-970, Brasil

Lucas Miguel de Carvalho
Laboratório de Genômica e BioEnergia, Departamento de Genética e Evolução, Microbiologia e Imunologia, UNICAMP, Campinas, São Paulo, 13083-970, Brasil

Luciana Souto Mofatto
Laboratório de Genômica e BioEnergia, Departamento de Genética e Evolução, Microbiologia e Imunologia, UNICAMP, Campinas, São Paulo, 13083-970, Brasil

Nicholas Vinícius Silva
Laboratório de Genômica e BioEnergia, Departamento de Genética e Evolução, Microbiologia e Imunologia, UNICAMP, Campinas, São Paulo, 13083-970, Brasil

Marcelo Falsarella Carazzolle
Laboratório de Genômica e BioEnergia, Departamento de Genética e Evolução, Microbiologia e Imunologia, UNICAMP, Campinas, São Paulo, 13083-970, Brasil

Gonçalo Amarante Guimarães Pereira
Laboratório de Genômica e BioEnergia, Departamento de Genética e Evolução, Microbiologia e Imunologia, UNICAMP, Campinas, São Paulo, 13083-970, Brasil

RESUMO: Devido a imensa complexidade genômica das plantas, a análise de transcriptômica assume um papel relevante para entendimento do mecanismo biológico, visto que além de dar indícios de como o organismo responde em determinada situação, o sequenciamento de RNA também pode ser utilizado para auxiliar a resolução de problemas de ordem genômica. A análise de expressão diferencial pode ser dividida em quatro etapas: i) análise de qualidade e trimagem dos dados; ii) alinhamento dos fragmentos à referência; iii) quantificação dos transcritos; e iv) análise de expressão diferencial. Em geral, apenas a etapa de análise de expressão diferencial (iv) é realizada em linguagem R, porém, nesse capítulo apresentamos uma versão diferenciada, em que os softwares externos também foram executados por comandos no R. Como estudo de caso foi utilizado um experimento de *Arabidopsis thaliana*, um organismo modelo que apresenta um ciclo de vida curto, é facilmente manipulado em laboratório e tem seu genoma bem estudado.

PALAVRAS-CHAVE: Transcriptômica de plantas, expressão diferencial, *Arabidopsis thaliana*, RNA-seq

ABSTRACT: Due to the immense complexity of plant genomes, transcriptomic analysis plays a relevant role in understanding the biological

mechanism, since in addition to providing evidence of how the organism responds in a given situation, RNA sequencing can also be used to aid genomic information. The analysis can be divided into four steps: i) quality analysis and data trimming; ii) alignment of fragments to the reference; iii) quantification of transcripts; and iv) differential expression analysis. In general, only the differential expression analysis step (iv) is performed in R language, but in this chapter we present a different version, in which external software was executed by commands in R. As an example was used an experiment of *Arabidopsis thaliana*, a model organism with a short life cycle, a well studied genome and easy to manipulate in laboratory.
KEYWORDS: Plant transcriptomics, differential expression, *Arabidopsis thaliana*, RNA-seq

1 I INTRODUÇÃO

Com o avanço da biologia molecular e das técnicas de sequenciamento foram desenvolvidas as ômicas; por exemplo, genômica, transcriptômica e proteômica. Cada uma dessas ômicas têm suas particularidades e nos auxiliam a responder perguntas específicas sobre o metabolismo celular. A genômica é utilizada para caracterizar um organismo a nível de DNA, molécula que é transmitida aos descendentes e que contém todo o potencial de genes que podem ser expressos em uma célula. A transcriptômica, por sua vez, surge para estudar o RNA das células, a parte do genoma que realmente é produzida em uma condição específica, sendo também utilizado para montagem dos transcritos (transcriptoma). A proteômica, por sua vez, visa estudar o conjunto de proteínas encontradas na célula em uma determinada situação.

Nesse contexto, apesar da transcriptômica ser utilizada principalmente para identificar expressão de genes diferenciais em condições específicas, ela também pode ser empregada para suprir demandas de análises genômicas, como, por exemplo, no auxilio da montagem e anotação de genomas em plantas complexas (Nascimento et al., 2019); busca de variantes genômicas (polimorfismos de nucleotídeos únicos, deleções e inserções de bases) (Chopra et al., 2015); e identificação de pequenas regiões repetitivas no genoma (microssatélites) em regiões codantes, utilizadas como marcadores genéticos (Bazzo, de Carvalho, Carazzolle, Pereira, & Colombo, 2018).

Dentre as metodologias *high-throughput* o RNA-seq recebe grande destaque. Esta metodologia apresenta como grande vantagem a possibilidade de identificação de novos genes, além de ser um método de maior acurácia quanto ao nível de quantificação gênica (Edwards & Batley, 2004; Hrdlickova, Toloue, & Tian, 2017). A primeira etapa para esse sequenciamento é a preparação das bibliotecas de cDNA, que deve ser específica para o tipo de RNA que será analisado. Ela é composta,

principalmente, pelas etapas: i) seleção de RNA de interesse na amostra celular; ii) depleção de RNA ribossomal em experimentos que não o tenham como objeto de estudo; iii) fragmentação do RNA; e iv) ligação de adaptador e identificador da biblioteca na extremidade dos fragmentos (Hrdlickova, Toloue, & Tian, 2017). Nessa etapa é necessário um cuidado para que as bibliotecas reflitam a proporção contida na amostra original (Rani & Sharma, 2017, Hrdlickova, Toloue, & Tian, 2017), e que haja uma preocupação com o número de réplicas a serem sequenciadas (Schurch et al., 2016).

Dessa forma, as informações obtidas durante a etapa de sequenciamento são definidas de acordo com os protocolos utilizados para construção das bibliotecas. Os protocolos mais aplicados são: *single-end* e *paired-end*, que diferem basicamente se será sequenciada uma extremidade dos fragmentos gerados na etapa (iii) ou ambas as extremidades, respectivamente. Os resultados gerados do sequenciador (dados brutos) em formato *.fastq* é utilizado como dados de entrada para início do nosso roteiro (pipeline) de identificação de genes diferenciais. O pipeline é composto pelas etapas: i) análise de qualidade e trimagem dos dados; ii) escolha de uma referência; iii) alinhamento dos fragmentos sequenciados (reads) à referência; iv) quantificação dos transcritos; e v) análise de expressão diferencial. É importante ressaltar que durante a etapa de alinhamento deve-se considerar informações sobre o protocolo utilizado na preparação das bibliotecas.

Dessa forma, esse capítulo tem como objetivo apresentar um pipeline de análise de expressão gênica pareada desenvolvido em linguagem R e aplicado a duas plantas submetidas duas condições de temperatura, 23ºC (controle) e 42ºC (estresse). O pipeline descrito engloba trimagem dos dados utilizando o software Trimmomatic (Bolger, Lohse, & Usadel, 2014); alinhamento e quantificação dos reads com o Kallisto (Bray, Pimentel, Melsted, & Pachter, 2016); análises de PCA e dendograma a partir da matriz de contagem; análises de expressão diferencial com os pacotes: Sleuth (Pimentel, Bray, Puente, Melsted, & Pachter, 2017) 2017, DESeq2 (Love, Huber, & Anders, 2014) e EdgeR (Robinson, McCarthy, & Smyth, 2010), sendo apresentado de forma detalhada para introduzir os comandos básicos utilizados na análise.

2 I METODOLOGIA

2.1 Estudo de caso

Como estudo de caso para apresentação de um pipeline de análise de expressão diferencial foi escolhido o sistema modelo *A. thaliana.* Essa espécie foi a primeira planta a ter seu genoma sequenciado no ano de 2000, com cinco cromossomos que compõem aproximadamente 135Mb (Balfagon et al., 2019;)(“TAIR,”). A versão

atual, TAIR10, contém 33.602 genes, incluindo 27.416 genes codantes, 1.359 ncRNA e 4.827 pseudogenes e transposons (“TAIR,”). Além da vantagem de ter um genoma pequeno, bem montado e estudado, a *Arabidopsis thaliana* ainda apresenta uma facilidade em aplicação de técnicas de transformação e tem um ciclo de vida curto (aproximadamente 6 semanas) com rápido crescimento (Agrawal, 2018). Foi estabelecido um experimento de *Arabidopsis thaliana* submetida a duas condições de temperatura: 23ºC na situação controle (C) e 42ºC em estresse de temperatura (HS). Inicialmente, as plantas foram submetidas a 50 µmol m^{-2} s^{-1} de iluminação e 23ºC de temperatura por 30 dias, seguido de 7h nas condições de controle e estresse citados anteriormente. Posteriormente foi retomada a condição inicial para recuperação da planta (Balfagon, et al., 2019)1. A partir desse experimento foram sequenciados tecidos de folhas em triplicata biológica (Balfagon, et al., 2019). Para isso, foi utilizada a plataforma Next-Seq 500 (Illumina) sendo gerados reads single-end (Balfagon, et al., 2019)hading 2 Accent 5. Os dados brutos foram baixados no banco de dados NCBI e podem ser encontrados no link https://www.ncbi.nlm.nih.gov/bioproject/555093.

O primeiro passo da análise é baixar o arquivo SraRunInfo.tsv. Ele contém todos os links do SRA e as informações das amostras. Para isso, devemos selecionar as amostras de alta temperatura (HS) (SRR9696660, SRR9696664, SRR9696668) e Controle (C) (SRR9696658, SRR9696662, SRR9696666). Posteriormente, clicar em ‘Send to’ -> File -> RunInfo. O arquivo SraRunInfo.tsv é utilizado para realização do download e conversão do arquivo SRA em fastq, através do comando fastqdump, disponível no pacote SRA Toolkit (“SRA Toolkit,”), como mostra o comando abaixo:

```
> setwd(".")
> base_dir <- getwd()
> dados <-read.csv("SraRunInfo.csv", stringsAsFactors=FALSE)
> arquivos <- basename(dados$download_path)
> for(i in 1:length(arquivos)){
   download.file(dados$download_path[i], arquivos[i])
  }

> for(a in arquivos) {
   cmd = paste("fastq-dump --split-3", a)
   system(cmd)
  }
```

2.2 Pré-tratamento dos dados brutos

Os dados brutos foram analisados com o software FastQC (versão 0.11.8) e trimados com o software trimmomatic (versão 0.39). Salientamos que todos os programas externos ao R devem estar previamente instalados e disponibilizados no /usr/bin/. Para o download automático dos programas devem-se seguir os comandos:

```
> cmd = paste("wget
http://www.usadellab.org/cms/uploads/supplementary/Trimmomatic/Trimmomatic-0.39.zip")
> system(cmd)
> cmd = paste("unzip Trimmomatic-0.39.zip")
> system(cmd)
>cmd = paste("wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.8.zip")
> cmd = paste("unzip fastqc_v0.11.8.zip")
> system(cmd)
> cmd = paste("chmod 755 ",base_dir,"FastQC/fastqc")
> system(cmd)
```

Para análise dos dados e trimagem:

```
>for(a in arquivos){
  cmd = paste(base_dir,"FastQC/fastqc ",a,".fastq ",sep = "")
  system(cmd)
  }

>for(a in arquivos){
     cmd = paste("java -jar ",base_dir,"/Trimmomatic-0.39/trimmomatic-0.39.jar SE -threads 10 -trimlog
  ",a,".trimlog -  summary  ",a,".summary  ",a,".fastq  ",a,".trim.fastq  ILLUMINACLIP:Trimmomatic-
  0.39/adapters/TruSeq2-SE.fa:2:30:10  LEADING:3  TRAILING:3  SLIDINGWINDOW:4:15
  MINLEN:36",sep = "")
  system(cmd)
  }
```

2.3 Obtenção da matriz de contagem para análise de expressão diferencial

Após a etapa de preparação dos dados foi realizado um alinhamento utilizando o software Kallisto (versão 0.44) utilizando como referência TAIR10. A partir desse alinhamento os transcritos/genes foram quantificados e a matriz de counts foi transferida ao R. Salientamos que o download do software e o índice do genoma devem ser realizados pelos comandos:

```
> cmd = paste("wget https://github.com/pachterlab/kallisto/releases/download/v0.44.0/kallisto_linux-
v0.44.0.tar.gz")
> cmd = paste("tar -xzvf kallisto_linux-v0.44.0.tar.gz")
> system(cmd)
> cmd = paste(base_dir,"/kallisto_linux-v0.44.0/kallisto index -i arabidopsis_index
<caminho_transcriptoma.fa>",sep="")
> system(cmd)
```

O comando abaixo reflete a execução da quantificação sobre o transcriptoma da *Arabidopsis thaliana* utilizando o index 'arabidopsis_thaliana' sobre a lista 'arquivos', que contém os IDs das bibliotecas do SRA.

```
>for(a in arquivos){
  cmd = paste(base_dir,"/kallisto_linux-v0.44.0/kallisto quant -i arabidopsis_index -o ",a,"_kallisto -b
  100 -t 10 --single -l 100 -s 0.001 ",a,".trim.fastq",sep="")
  system(cmd)
  }
```

Após quantificar todos os arquivos é gerada uma cópia de cada arquivo fora do diretório para ser utilizado na montagem da matriz, como mostra o comando abaixo:

```
>cmd = paste("for file in ls -l -d SRR*;do cp $file/abundance.tsv $file.tsv;done")
>print(cmd)
>system(cmd)
```

Por último é gerada a matriz de contagem dos genes do TAIR10 para ser utilizada na expressão diferencial. Para isso, utilizamos o script '*abundance_estimates_to_matrix.pl*' presente no software Trinity (versão 2.8.6). Os comandos abaixo mostram o download automático da versão mais recente do software, sua descompactação e a geração da matriz, utilizando os arquivos *.tsv* de quantificação gerados pelo kallisto.

```
>cmd = ("wget https://github.com/trinityrnaseq/trinityrnaseq/releases/download/v2.8.6/trinityrnaseq-
v2.8.6.FULL.tar.gz")
>system (cmd)
>cmd = (" tar –xzvf trinityrnaseq-v2.8.6.FULL.tar.gz")
>system(cmd)

>lista = paste0(arquivos,".tsv",collapse = " ")
>cmd = paste("perl ",base_dir,"/trinityrnaseq-2.8.6/util/abundance_estimates_to_matrix.pl --
est_method kallisto --gene_trans_map none ",lista,sep="")
>print (cmd)

>head(kallisto.isoform.counts.matrix)

          SRR9696658.tsv SRR9696662.tsv SRR9696666.tsv SRR9696660.tsv SRR9696664.tsv SRR9696668.tsv
AT1G74180.1      67.4293       134.851    1.01137e-05        337.132        202.121        67.4288
AT5G44980.1         15.5       12.5937        7.87109         50.375        23.6133           16.5
AT2G33680.5          129       174.532    1.9115e-06        134.225    1.43061e-07             38
AT4G27300.1          649          1109            678            567            383            296
AT4G03220.1            0             3              3              3              1              6
AT1G74810.3      18.3047       34.9407    0.00015031        0.80261        7.23279              0
AT2G41720.2      285.198       300.788        291.572        446.138        283.682        298.439
AT4G01220.1      250.564       294.753            298         290.35        113.997        147.923
AT5G23220.1            0             0              0              1              0              2
```

2.4 Análise de componentes principais (PCA) e dendograma

A análise de componentes principais (PCA) é um método de redução de dimensionalidade através de transformações ortogonais das variáveis em componentes principais. Nessa análise as variáveis são as amostras, e a variância anotada para o teste é dos transcritos em relação às amostras. O código abaixo consiste na execução do PCA sobre a matriz de contagem. Os comandos utilizados serão descritos na Seção 2.5.2, que realiza a análise de expressão diferencial pelo DESeq2. O último comando *varianceStabilizingTransformation()* gera o objeto 'vsd' através da normalização da tabela de contagem pelos fatores de dispersão e o tamanho da biblioteca. A variância normalizada de cada variável é gerada e então, é construído o gráfico do PCA (Figura 1 - A) pelo comando *plotPCA()*.

```
>library( "DESeq" )
>countsTable <- read.delim("kallisto.isoform.counts.matrix", header=TRUE, stringsAsFactors=TRUE)
>rownames(countsTable) <- countsTable[,1]
>countsTable <- countsTable[,2:23]
>conds <- factor(c(names(countsTable)))

>countsTable_novo <- apply(countsTable,2,as.integer)
>countsTable_novo[is.na(countsTable_novo)] <- 0

>cds<-newCountDataSet(countsTable_novo,conds)
>cds<-estimateSizeFactors(cds)
>sizeFactors(cds)
>cds <- estimateDispersions(cds,method='blind')
>vsd <- varianceStabilizingTransformation(cds)

>plotPCA(vsd)
```

A análise de dendograma gera um diagrama de árvore que mostra o agrupamento das variáveis de interesse. No exemplo utilizado das amostras o agrupamento é dado por similaridade. Esse algoritmo é derivado de algoritmos de agrupamentos (clusterização).

O código abaixo reflete a Figura 1 - B. Ele consiste na geração do dendograma sobre os dados de contagem através do pacote *ggdendro*, que necessita do pacote *ggplot2*. Utilizando-se da matriz de contagem, encontramos a distancia euclidiana sobre os dados transpostos através do comando *dist()* e, posteriormente, é gerada a árvore através do comando *hclust()*. Por fim, o comando *ggdendogram()* nos mostra o gráfico rotacionado.

```
> #install.packages("ggdendro")
> #install.packages("ggplot2")
> library("ggplot2")
> library("ggdendro")

> countsTable <- read.delim("kallisto.isoform.counts.matrix", header=TRUE,
stringsAsFactors=TRUE,row.names = 1)
> dd <- dist(t(scale(countsTable)), method = "euclidean")
> hc <- hclust(dd, method = "ward.D2")
> ggdendrogram(hc, rotate = TRUE, theme_dendro = FALSE,
size  = 1) + xlab("Amostras") +ylab("Altura")
```

2.5 Análise de expressão diferencial utilizando pacotes do R

A partir da matriz de contagem foram realizadas análises de análise de componentes principais (PCA) e dendograma para identificar similaridade entre as amostras. Em seguida foram explorados três pacotes para seleção de genes diferenciais, DESeq2 (versão 1.26.0), Sleuth (versão 0.27.3) e edgeR (versão 3.28.0). Os códigos do DESeq2 e edgeR foram baseados no programa Trinity.

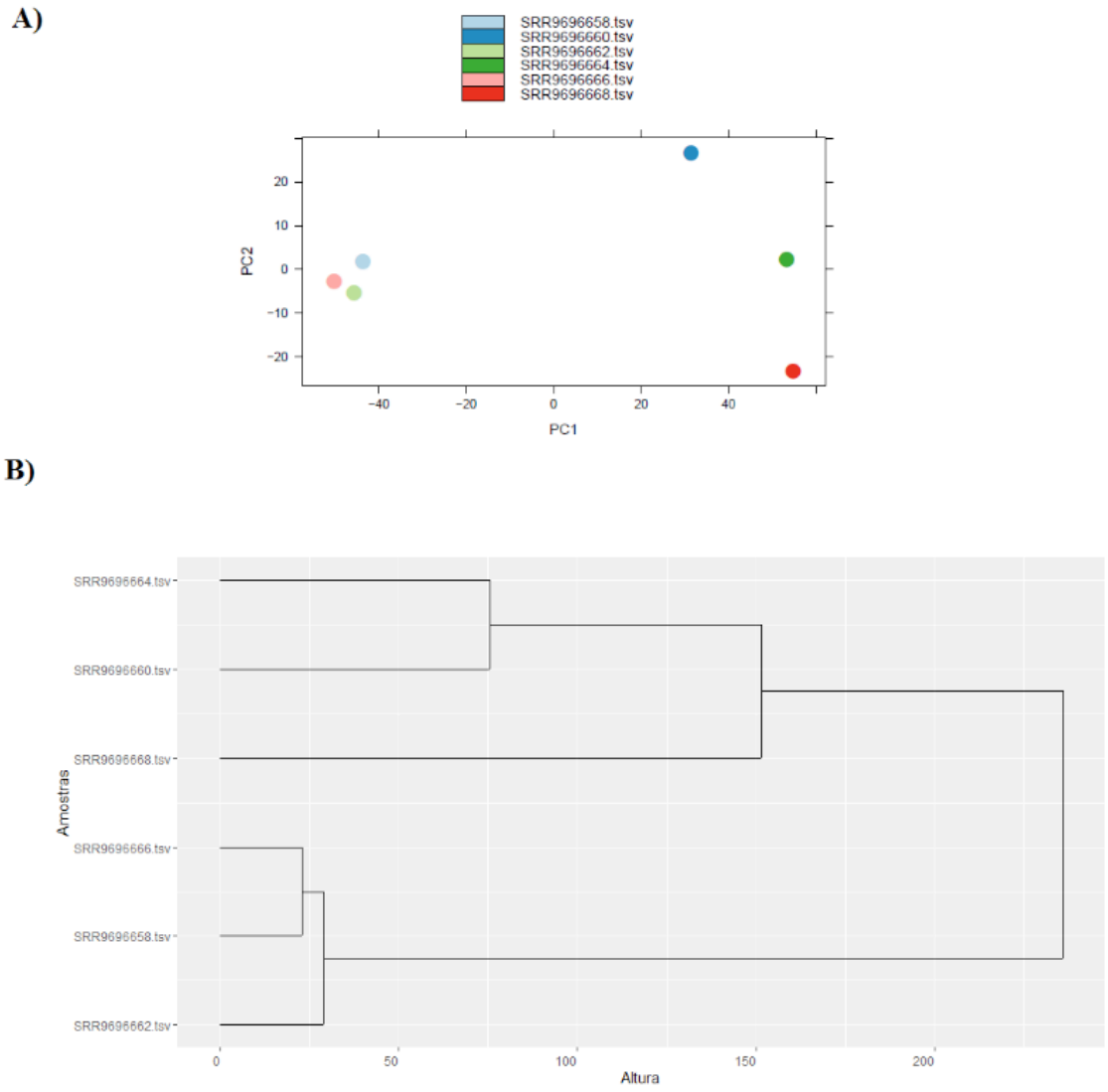

Figura 1. **Resultado PCA e dendograma.** (A) Gráfico da análise de componentes principais (PCA) mostra que as amostras controle (SRR9696658, SRR969662, SRR9696666) e tratado em alta temperatura (SRR9696660, SRR9696664, SRR9696668) estão agrupados, ou seja, possuem variância semelhante em relação a quantificação dos genes. (B) Dendograma das amostras controle e tratado, mostrando um agrupamento respeitando a condição biológica.

2.5.1 EdgeR

Primeiramente é verificado a existência do pacote edgeR, caso não haja ele instalado, ocorre a instalação e a sua chamada.

```
>if (! require(edgeR)) {
  source("https://bioconductor.org/biocLite.R")
  biocLite("edgeR")
  library(edgeR)
  }
```

Posteriormente é lida a matriz de quantificação através do comando '*read. table*'. Também ocorre a aproximação inteira dos *counts* através do comando '*round*' e a verificação de no mínimo dois reads counts por linha, através do comando '*rowSums(cpm(rnaseqMatrix) > 1) >= 2*'.

```
>data = read.table("kallisto.isoform.counts.matrix", header=T, row.names=1, com='')
>col_ordering = c(1,2,3,4,5,6)
>rnaseqMatrix = data[,col_ordering]
>rnaseqMatrix = round(rnaseqMatrix)
>rnaseqMatrix = rnaseqMatrix[rowSums(cpm(rnaseqMatrix) > 1) >= 2,]
```

A parte do teste diferencial começa criando a variável *conditions*, que agrupa os dados presentes no arquivo '*kallisto.isoform.counts.matrix*'. A função *DGEList* gera um objeto que contém as contagens inteiras dos genes, um dataframe contendo informações sobre as amostras ou bibliotecas, e um dataframe adicional contendo informações adicionais dos genes. A função *calcNormFactors* normaliza a quantificação do RNA por um conjunto de fatores de escala para os tamanhos da biblioteca que minimizam as alterações do logFC entre as amostras para a maioria dos genes. As dispersões do modelo são calculadas pelo comando *estimateDisp*. Por fim, é executado o teste comparativo pelo comando *exactTest*.

```
>conditions = factor(c(rep("Controle", 3), rep("Tratado", 3)))

>exp_study = DGEList(counts=rnaseqMatrix, group=conditions)
>exp_study = calcNormFactors(exp_study)
>exp_study = estimateDisp(exp_study)
>et = exactTest(exp_study, pair=c("Controle", "Tratado"))
>tTags = topTags(et,n=NULL)
```

A matriz de resultados é extraída e convertido o valor do logFC.

```
>result_table = tTags$table
>result_table = data.frame(sampleA="Controle", sampleB="Tratado", result_table)
>result_table$logFC = -1 * result_table$logFC
```

Os resultados e a matriz são guardados em seus respectivos arquivos através do comando *write.table*.

```
>write.table(result_table, file='edgeR.resultados, sep='     ', quote=F, row.names=T)
>write.table(rnaseqMatrix, file='edgeR.count_matrix', sep='   ', quote=F, row.names=T)
```

Um dos primeiros resultados é a análise do *Volcano plot* (Figura 2), um gráfico que mostra a dispersão dos dados e verifica quais os genes que realmente passaram pelo teste estatístico. O comando abaixo utiliza os dados de logFC (eixo x) e o valor negativo do resultado do teste de expressão ajustado pelo teste de *False Discovery Ratio (FDR)*. Além disso, a adição de cores depende do valor do FDR, sendo que valores inferiores a 0.05 são pintados de vermelho (red), e representam um limite máximo de 5% de falsos positivos no teste, caso contrário de preto (black).

```
>pdf("edgeR.Volcano.pdf")
>plot(result_table$logFC,  -1*log10(result_table$FDR),  col=ifelse(result_table$FDR<=0.05,  "red",
"black"),xlab="logCounts", ylab="logFC", title="Volcano plot", pch=20)
>dev.off()
```

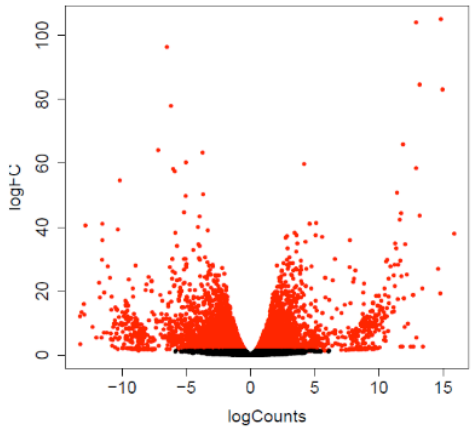

Figura 2. **Volcano plot gerado a partir do resultado do edgeR.** O gráfico mostra a dispersão dos dados e indica através de cores os genes que realmente passaram pelo teste estatístico. Em vermelho estão os genes que passaram pelo teste com valores inferiores a 0.05; e em preto os genes que apresentaram um FDR maior que 0.05.

Os genes significativos do teste são selecionados utilizando o pacote *dplyr()* sobre o campo FDR da tabela *result_table*. Por fim, geramos uma lista do nome dos genes que passaram sobre o teste através de um loop sobre a matriz, gerando a lista '*edgeR_list*'.

```
>edger_significant <- dplyr::filter(result_table, FDR <= 0.05)
>edgeR_list <- NULL
>for(i in 1:length(result_table[,1])){
   if(result_table[i,]$FDR <= 0.05){
     edgeR_list[i] <- row.names(result_table)[i]
     }
   }

>head(edgeR_list)

[1] "AT2G29630.4" " AT2G48060.3" " AT5G12110.1" " AT2G18960.2" " AT2G18960.3" " AT1G12900.4"
```

2.5.2 DESeq2

Primeiramente é verificado a existência do pacote DESeq2, caso não haja ele instalado, ocorre a instalação e a sua chamada.

```
>if (! require(DESeq2)) {
  source("https://bioconductor.org/biocLite.R")
  biocLite("DESeq2")
  library(DESeq2)
  }
```

Posteriormente são executados os mesmos filtros que a seção 2.5.1.

```
>data = read.table("kallisto.isoform.counts.matrix", header=T, row.names=1, com='')
>col_ordering = c(1,2,3,4,5,6)
>rnaseqMatrix = data[,col_ordering]
>rnaseqMatrix = round(rnaseqMatrix)
>rnaseqMatrix = rnaseqMatrix[rowSums(cpm(rnaseqMatrix) > 1) >= 2,]
>conditions = data.frame(conditions=factor(c(rep("Controle", 3), rep("Tratado", 3))))
>rownames(conditions) = colnames(rnaseqMatrix)
```

A função DESeq realiza a comparação entre as condições, estimando os fatores de escala (*estimateSizeFactors()*), as dispersões (*estimateDispersions()*) e realizando o teste estatístico através de um modelo binomial negativa como hipótese.

```
>ddsFullCountTable <- DESeqDataSetFromMatrix(
  countData = rnaseqMatrix,
  colData = conditions,
  design = ~ conditions)

>dds = DESeq(ddsFullCountTable)
>contrast=c("conditions","Controle","Tratado")
>res = results(dds, contrast)
```

A variável *baseMeanA* e *baseMeanB* guardam o valor médio da expressão de cada gene na condição 'Controle' e 'Tratado', respecrivamente. Elas são adicionadas ao dataframe '*res*' através do comando *cbind()*. O p-value ajustado dos genes com resultado NA são modificados para 1. Por último é gerado o arquivo de resultados através do comando *write.table()*.

```
>baseMeanA <- rowMeans(counts(dds, normalized=TRUE)[,colData(dds)$conditions == "Controle"])
>baseMeanB <- rowMeans(counts(dds, normalized=TRUE)[,colData(dds)$conditions == "Tratado"])
>res = cbind(baseMeanA, baseMeanB, as.data.frame(res))
>res = cbind(sampleA="Controle", sampleB="Tratado", as.data.frame(res))
>res$padj[is.na(res$padj)]  <- 1
>res = as.data.frame(res[order(res$pvalue),])
>write.table(res, file='DESeq2.DE_results', sep='     ', quote=FALSE)
```

A geração do *Volcano plot* (Figura 3) e a lista de genes também são geradas da mesma maneira que na Seção 2.5.1.

```
>pdf("DESeq2_Volcano.pdf")
>plot(res$log2FoldChange, -1*log10(res$padj), col=ifelse(res$padj<=0.05, "red",
"black"),xlab="logCounts", ylab="logFC", title="Volcano plot", pch=20)
>dev.off()

>Deseq2_list <- NULL
>for(i in 1:length(res[,1])){
  if(res[i,]$padj <= 0.05){
    Deseq2_list[i] <- row.names(res)[i]
    }
  }

>head(Deseq2_list)

[1] "AT5G12110.1" "AT1G64780.1" "AT1G77510.1" "AT2G29500.1" "AT5G61590.1" "AT1G09750.1"
```

2.5.3 Sleuth

Primeiramente é verificada a existência do pacote Sleuth. Caso não haja ele instalado, ocorre a instalação e a sua chamada.

```
if (! require(DESeq2)) {
  devtools::install_github("pachterlab/sleuth")

  library("sleuth")}
```

Primeiramente criamos uma lista das amostras que serão testadas e os caminhos para os diretórios gerados pelo kallisto, já que o sleuth depende dos seus resultados para sua execução.

```
>sample_id <- list('SRR9696658','SRR9696662','SRR9696666',
'SRR9696660','SRR9696664','SRR9696668')

>paths <- list(paste(base_dir,"/SRR9696658",sep=""),
        paste(base_dir,"/SRR9696662",sep=""),
        paste(base_dir,"/SRR9696666",sep=""),
        paste(base_dir,"/SRR9696660",sep=""),
        paste(base_dir,"/SRR9696664",sep=""),
        paste(base_dir,"/sleuth/SRR9696668",sep=""))

>names(paths) <- sample_id
```

Posteriormente é criado um dataframe que depende de um arquivo chamado 'amostras.txt', composto pela descrição das amostras e suas réplicas (mostrado abaixo). Também são selecionados campos específicos das amostras pelo comando 'dplyr::select' e então anexados a variável 'paths' ao dataframe s2c através do comando 'dplyr::mutate'.

```
> am <- read.table(file="amostras.txt",sep="\t",header=TRUE)
> head(am)

        sample      condition       reps
1       SRR9696658  Controle    1
2       SRR9696662  Controle    1
3       SRR9696666  Controle    1
4       SRR9696660  Tratado 2
5       SRR9696664  Tratado 2
6       SRR9696668  Tratado 2

>s2c <- read.table(file.path(base_dir, "amostras.txt"), header = TRUE, stringsAsFactors=FALSE)
>s2c <- dplyr::select(s2c, sample = sample, condition, reps)
>s2c <- dplyr::mutate(s2c, path = paths)
>print(s2c)
>s2c <- data.frame(lapply(s2c, as.character), stringsAsFactors=FALSE)
```

A preparação do teste de expressão diferencial é realizado pelo *sleuth_prep()*:

```
>#transcrito
>so <- sleuth_prep(s2c, ~condition, extra_bootstrap_summary = TRUE)
>so <- sleuth_fit(so)
>so <- sleuth_wt(so, "conditionTratado")
>models(so)
```

A matriz de resultados é gerada, juntamente com os transcritos significativos, que são guardados na lista '*sleuth_list*'. Os resultados são guardados no arquivo *diferenciais_sleuth.txt*. O comando *sleuth_live()* é uma visualização interativa dos dados com o Shiny.

```
> results_table <- sleuth_results(so, test='conditionTratado', test_type = 'wald')
> head(results_table)

   target_id          pval          qval         b       se_b mean_obs   var_obs     tech_var      sigma_sq
smooth_sigma_sq
1  AT5G12110.1 5.137705e-229 1.730584e-224  4.501009 0.1393090 7.058186  6.099318 0.0044212051
0.022568366     0.02468928
2  AT2G29630.4 4.253194e-226 7.163229e-222 -8.871293 0.2763547 3.742500 23.633993 0.0037819679
0.026265685     0.11077587
3  AT1G12900.3 3.413529e-209 2.874533e-205 -8.927552 0.2892355 3.779464 23.956643 0.0192204939
0.038636202     0.10626525
4  AT1G12900.4 3.413529e-209 2.874533e-205 -8.927552 0.2892355 3.779464 23.956643 0.0192204939
0.038636202     0.10626525
5  AT1G20020.2 1.776434e-117 1.196748e-113 -9.461283 0.4106134 4.037494 27.057087 0.0074449443
0.245460135     0.07994148
6  AT5G61590.1 2.968696e-102  1.666626e-98 -2.905480 0.1353273 6.528624  2.546685 0.0031903136
0.014485792     0.02427990
7  AT1G75750.1  5.986875e-83  2.880884e-79  2.735799 0.1417925 7.718472  2.269505 0.0009833116
0.029174350     0.02667726

> sleuth_significant <- dplyr::filter(results_table, qval <= 0.05)
> sleuth_list <- sleuth_significant[,1]
> head(sleuth_list)

[1] “AT5G12110.1” “AT2G29630.4” “AT1G12900.3” “AT1G12900.4” “AT1G20020.2” “AT5G61590.1”

>write.table(sleuth_significant,file="diferenciais_sleuth.txt")
>sleuth_live(so)

>pdf("Sleuth_Volcano.pdf")
>plot(results_table$b,  -1*log10(results_table$qval),  col=ifelse(results_table$qval<=0.05,  "red",
"black"),xlab="log(qval)", ylab="beta", title="Volcano plot", pch=20)
>dev.off()
```

2.6 Comparação entre os pacotes para expressão diferencial

Nesta seção serão apresentados os resultados as análises de expressão diferenciais realizadas na seção anterior. A Figura 3 mostra o diagrama de venn dos transcritos diferencialmente expressos entre os três testes diferenciais (sleuth, DESeq2 e edgeR). O digrama foi gerado através do pacote ‘*VennDiagram*’, no qual o código está disponibilizado no script.R no repositório github no link: https://github.com/lmigueel/DEG_Athaliana.

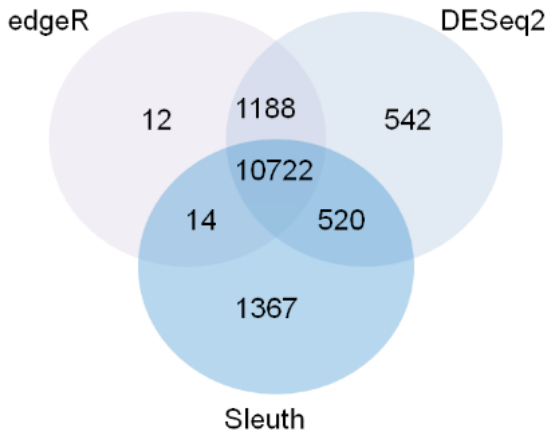

Figura 3. **Comparação entre os genes diferenciais obtidos pelos pacotes utilizados** Na comparação entre os pacotes foram identificados 10722 genes diferenciais em comum (core), sendo que o edgeR identificou 12 genes não detectados nos outros pacotes, DESeq2, 542 e Seuth 1367 genes diferenciais.

A análise revela um grande core de genes em comum entre os três pacotes analisados (10.722 transcritos). Comparado com o artigo original dos dados, que utilizou o pacote DESeq2 para análise de expressão diferencial, o nosso resultado se aproxima, havendo pequenas diferenças causadas pelo algoritmo de quantificação dos transcritos (kallisto *versus* mapeamento no transcriptoma).

A partir desses resultados, podem ser realizadas análises específicas como enriquecimento do core de genes entre os métodos testados, visto que, teoricamente ele se conserva independe do método estatístico utilizado. Um possível pacote para a análise de enriquecimento de genes, utilizando a linguagem R, é o pacote *topGO()*.

3 I ADAPTAÇÃO DO PIPELINE EM DIFERENTES CONTEXTOS

Ao planejar um experimento de RNA-seq para análise de expressão diferencial é importante considerar a quantidade de réplicas que serão sequenciadas, considerando que, em geral, para que todos os genes encontrados sejam relevantes, pede-se um mínimo de 12 réplicas (Schurch, et al., 2016).

Na análise de bioinformática, a primeira tomada de decisão é a escolha de como será realizado o pré-processamento dos dados. No pipeline disponibilizado foi utilizado o software FastQC (Andrews, 2010) para análise de qualidade e o Trimmomatic (Bolger, et al., 2014) para trimagem dos reads. Porém, também pode ser realizada uma análise para descarte de RNA ribossomal com o software SortMeRNA (Carvalho et al., 2019; Kopylova, Noe, & Touzet, 2012).

A segunda etapa importante é escolher o genoma/transcriptoma que possa ser utilizado como referência. Nessa etapa podem ser utilizados os bancos de dados como EnsemblPlants (“EnsemblPlants,”), NCBI (*NCBI Genome*), Phytozome (“Phytozome,”) e alguns específicos para uma espécie como o TAIR (“TAIR,”), desenvolvido para *Arabidopsis thaliana*. Para os casos em que não existe genoma público, é possível realizar a montagem do transcriptoma com os dados sequenciados utilizando, por exemplo, o software Trinity (Haas et al., 2013). Após a escolha da referência é necessário o mapeamento dos reads e a quantificação da expressão, que pode ser realizada a nível de transcrito e/ou de genes. Com esses dados têm-se início a análise de expressão diferencial com os pacotes citados.

É importante considerar que para cada etapa apresentada podem ser utilizados diversos softwares além dos citados (Carvalho, et al., 2019). Dessa forma, o pipeline apresentado pode ser adaptado de acordo com as necessidades do projeto, adequação aos dados e preferência do usuário, sendo de extrema relevância para estudos com plantas.

3.1 Disponibilidade de dados e script

O material pode ser baixado em: doi.org/10.13140/RG.2.2.20790.86086

AGRADECIMENTOS

Agradecimento ao Conselho Nacional de Desenvolvimento Científico e Tecnológico pelo auxílio financeiro à pesquisa (Processos: 150977/2019-0; 140869/2016-6) e também ao Centro de Computação em Engenharia e Ciências (Processo FAPESP/Cepid: 2013/08293-7).

REFERÊNCIAS

Agrawal, S. (2018). Arabidopsis thaliana as a Model Organism to Study Plant-Pathogen Interactions. Springer.

Andrews, S. (2010). FastQC: A quality control tool for high throughput sequence data Retrieved 04/02/2019, from https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Balfagon, D., Sengupta, S., Gomez-Cadenas, A., Fritschi, F. B., Azad, R. K., Mittler, R., & Zandalinas, S. I. (2019). Jasmonic Acid Is Required for Plant Acclimation to a Combination of High Light and Heat Stress. Plant Physiol, 181(4), 1668-1682. doi: 10.1104/pp.19.00956

Bazzo, B. R., de Carvalho, L. M., Carazzolle, M. F., Pereira, G. A. G., & Colombo, C. A. (2018). Development of novel EST-SSR markers in the macauba palm (Acrocomia aculeata) using transcriptome sequencing and cross-species transferability in Arecaceae species. BMC Plant Biol, 18(1), 276. doi: 10.1186/s12870-018-1509-9

Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics, 30(15), 2114-2120. doi: 10.1093/bioinformatics/btu170

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

Carvalho, L. M. d., Borelli, G., Camargo, A. P., Assis, M. A. d., Ferraz, S. M. F. d., Fiamenghi, M. B., Carazzolle, M. F. (2019). Bioinformatics applied to biotechnology: A review towards bioenergy research. Biomass and Bioenergy, 123, 195-224. doi: 10.1016/j.biombioe.2019.02.016

Chopra, R., Burow, G., Farmer, A., Mudge, J., Simpson, C. E., Wilkins, T. A., . . . Burow, M. D. (2015). Next-generation transcriptome sequencing, SNP discovery and validation in four market classes of peanut, Arachis hypogaea L. Mol Genet Genomics, 290(3), 1169-1180. doi: 10.1007/s00438-014-0976-4

Edwards, D., & Batley, J. (2004). Plant bioinformatics: from genome to phenome. Trends Biotechnol, 22(5), 232-237. doi: 10.1016/j.tibtech.2004.03.002 S0167-7799(04)00076-9 [pii] EnsemblPlants. from https://plants.ensembl.org/index.html

Haas, B., Papanicolaou, A., Yassour, M., Grabherr, M., Blood, P., Bowden, J., Regev, A. (2013). De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. Nat Protoc, 8(8). doi: 10.1038/nprot.2013.084

Hrdlickova, R., Toloue, M., & Tian, B. (2017). RNA-Seq methods for transcriptome analysis. Wiley Interdiscip Rev RNA, 8(1). doi: 10.1002/wrna.1364

Kopylova, E., Noe, L., & Touzet, H. (2012). SortMeRNA: fast and accurate filtering of ribosomal RNAs in metatranscriptomic data. Bioinformatics, 28(24), 3211-3217. doi: 10.1093/bioinformatics/bts611

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

Nascimento, L. C., Yanagui, K., Jose, J., Camargo, E. L. O., Grassi, M. C. B., Cunha, C. P., . . . Carazzolle, M. F. (2019). Unraveling the complex genome of Saccharum spontaneum using Polyploid Gene Assembler. DNA Res, 26(3), 205-216. doi: 10.1093/dnares/dsz001

NCBI Genome. Retrieved from https://www.ncbi.nlm.nih.gov/genome/

Pimentel, H., Bray, N. L., Puente, S., Melsted, P., & Pachter, L. (2017). Differential analysis of RNA-seq incorporating quantification uncertainty. Nat Methods, 14(7), 687-690. doi: 10.1038/nmeth.4324

Rani, B., & Sharma, V. K. (2017). Transcriptome profiling: methods and applications- A review. Agricultural Reviews, 38(4), 271-281. doi: 10.18805/ag.R-1549

Robinson, M. D., McCarthy, D. J., & Smyth, G. K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics, 26(1), 139-140. doi: 10.1093/bioinformatics/btp616

Schurch, N. J., Schofield, P., Gierlinsk, M., Cole, C., Sherstnev, A., Singh, V., Barton, G. J. (2016). How many biological replicates are needed in an RNA-seq experiment and which differential expression tool should you use? RNA, 22, 839–851. doi: 10.1261/rna.053959.115

SRA Toolkit. from https://www.ncbi.nlm.nih.gov/sra/docs/toolkitsoft/

Phytozome. from https://phytozome.jgi.doe.gov/pz/portal.html

TAIR. from https://www.arabidopsis.org/

ÍNDICE REMISSIVO

T

V

2020