Single Cell scale RNA-seq Analysis Protocol to analyze Smart-3SEQ data from RAGP neurons of pig heart

Lakshmi Kuttippurathu (lakshmi.kuttippurathu@jefferson.edu)
Thomas Jefferson University https://orcid.org/0000-0001-6612-9040
Alison Moss
Thomas Jefferson university https://orcid.org/0000-0002-7907-8796
Rajanikanth Vadigepalli (Rajanikanth.Vadigepalli@jefferson.edu)
Thomas Jefferson University https://orcid.org/0000-0002-8405-1037

Method Article

Keywords: SMART-3seq, scRNA-Seq, Data Analysis, LCM, RAGP, Mapping

Posted Date: June 8th, 2021

DOI: https://doi.org/10.21203/rs.3.pex-962/v1

Abstract

The present protocol describes transcriptome mapping, data normalization and analysis pipeline with detailed steps for each of these aspects for single cell/ low input RNASeq data from Right Atrial Ganglionated Plexus (RAGP) of pig heart. The protocol with minor modifications can be adapted for low input samples with short reads or samples with low quality input RNA. Single cell samples acquired using Laser Capture Microdissection (LCM) were processed for RNA-Seq library preparation using Smart-3SEQ technique (Foley et al 2019). The data analysis workflow consists of (a) pre-processing- data trimming, read alignment and feature count and (b) downstream analysis- annotation, batch correction, filtering and normalization. The entire protocol is performed using freely available packages. Most of them are available within the R framework.

Introduction

The present protocol was developed as part of the National Institutes of Health SPARC project efforts to construct a comprehensive anatomical, molecular and functional map of the peripheral nervous system at the visceral organs. A key goal is to develop an anatomical framework onto which various data sets can be mapped, e.g., spatial location and distribution of hundreds to thousands of single neurons within a tissue context, molecular profiles of spatially-tracked single neurons, etc. The method integrates precise 3D location of pig RAGP neurons using imaging technologies with high throughput single cell gene expression data.

The experimental method is conceptually similar to the prototype RNA-seq method, but with a streamlined protocol and optimizations for LCM and FFPE material (Foley et al , 2019). Primarily designed to analyze the gene expression of single cells captured using LCM from fresh frozen tissue, it can be adapted to other low input samples such as cell culture samples (1-1000 cells), and extracted RNA (10pg-1ng). The workflow to analyze the single cell scale expression data consists of two main steps – pre-processing and downstream analysis. In the pre-processing stage, the raw sequencing data (Illumina sequencer's base call files (BCLs) was converted to Fastq files using the Illumina bcl2fastq program (version 1.8.4). As a next step, the fastq reads were trimmed to remove the first 5 nts (to append as UMI), and the next 3 nts as G-overhang. The trimmed reads were aligned to the reference genome using STAR-2.7.2a (Dobin et al. 2013).

We did not do UMI based deduplication at this stage because with 3SEQ protocol, duplicates are expected since we are sampling sequences from narrower windows upstream of the expressed genes and the step itself might introduce additional noise. We observed that the read alignments tend to be at sites other than the 3' ends of annotated transcripts. It is expected of this protocol, hence we did not limit our analysis to 3' ends. We used the Feature count algorithm, Subread R (Liao et al 2013) to count the genes/exons per transcripts from the merged reads. The resulting gene count matrix was used for downstream analysis.

The annotated gene count matrices from different experimental batches were combined using a batch correction algorithm ComBat-seq (Zhang et al 2020). ComBat-seq uses Bayesian linear regression to remove batch effects and is ideal for smaller data sets with larger number of overdispersed counts. The outlier samples were removed and low expressed samples and genes were filtered out based on the "zero inflation". DESeq2 (Love et al. 2014) was used to do regularized log transformation. To account for the system level dependence of gene expression variation on sequencing depth and to estimate scaling factors within groups, the data was normalized using SCnorm (Bacher et al. 2017). SCnorm is known to be effective for datasets that have large numbers of zero counts.

Reagents

Equipment

Procedure

Procedure

1. <u>Pre-processing</u>

-

Conversion: Raw sequencing data (Illumina sequencer's base call files (BCLs) was converted to fastq files using the Illumina *bcl2fastq* program

Trimming: The reads from Fastq files were trimmed to remove poly A tail and G overhang, primer dimers and the 5 base umi was extracted. umi_homopolymer.py is downloaded from https://github.com/jwfoley/3SEQtools

Aligning: The trimmed reads were aligned, using STAR-2.7.2a (Dobin et al. 2013) to the Sus scrofa reference genome sequence version Sus_scrofa.Sscrofa11.1.fasta, available in the Ensembl database. Sus_scrofa.Sscrofa11.1.95.gtf was used as a reference transcriptome. The STAR parameters we used specific to our dataset were --outFilterMismatchNmax 999 (for mismatch) , –outFilterMultimapNmax 1 (to take care of the multi-mapping) and clip3pAdapterSeq AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA (clipping the adapter reads at 3` end). Below are the steps.

- Download the latest version of STAR aligner software
- Download the reference fasta file from Ensembl database
- Download the reference annotation file from Ensembl database
- Create the genome index file using the command

STAR –runThreadN 4 –runMode genomeGenerate –genomeDir path -
- genomeFastaFiles Sus_scrofa.Sscrofa11.1.fasta –sjdbGTFfile Sus_scrofa.Sscrofa11.1.95.gtf –sjdbOverhang 100

Merging sam files: Aligned files from reads from 4 lanes are merged using MergeSamfiles from picard (https://github.com/broadinstitute/picard/releases/download/2.22.3/picard.jar)

Below is the shell script for executing these steps starting from trimming

#Note: Variable “name” indicates the folder names (PR1643-xxx) corresponding to each of the samples. Each sample has 4 lanes (L001,L002,L003,L004) giving rise to 4 fastq files. This script is executed from the main folder that has all the library sub-folders. Within the main folder is a folder for each sample with the folder names (PR1643-xxx). Within each of these sample folders are four folders, one for each lane, labeled L001-4. Prior to running the following shell script, it is expected that one unzipped fastq file corresponding to the appropriate file and lane resides in each folder. The following script will write the output of umi_homopolyer.py and STAR into the respective lane folders. The output of the MergeSamFiles will write the combined sam file to top level of the sample folder.

```
#!/bin/bash

for name in PR1643*; do

echo $name
```

#1. run python code to trim polyA tail and g-overhang

```
python umi_homopolymer.py ./$name/L001/*.fastq ./$name/L001/$name\_L001.fastq

python umi_homopolymer.py ./$name/L002/*.fastq ./$name/L002/$name\_L002.fastq

python umi_homopolymer.py ./$name/L003/*.fastq ./$name/L003/$name\_L003.fastq

python umi_homopolymer.py ./$name/L004/*.fastq ./$name/L004/$name\_L004.fastq
```

#2. Run STAR to align the reads to reference genome

```
STAR –runThreadN 60 –quantMode GeneCounts –genomeDir Pig_genome/star_index –
readFilesIn ./$name/L001/${name}_L001.fastq  –outFileNamePrefix ./$name/L001/${name}_L001_star –
outFilterMultimapNmax 1 –outFilterScoreMinOverLread 0.1 –outFilterMatchNminOverLread 0.1 –outFilterMatchNmin
0 –outFilterMismatchNmax 999 -clip3pAdapterMMP 0.2 -clip3pAdapterSeq
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA –twopassMode Basic


STAR –runThreadN 60 –quantMode GeneCounts –genomeDir Pig_genome/star_index –
readFilesIn ./$name/L002/${name}_L002.fastq  –outFileNamePrefix ./$name/L002/${name}_L002_star –
outFilterMultimapNmax 1 –outFilterScoreMinOverLread 0.1 –outFilterMatchNminOverLread 0.1 –outFilterMatchNmin
0 –outFilterMismatchNmax 999 -clip3pAdapterMMP 0.2 -clip3pAdapterSeq
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA –twopassMode Basic
```

```
STAR --runThreadN 60 --quantMode GeneCounts --genomeDir Pig_genome/star_index --
readFilesIn ./$name/L003/${name}_L003.fastq  --outFileNamePrefix ./$name/L003/${name}_L003_star --
outFilterMultimapNmax 1 --outFilterScoreMinOverLread 0.1 --outFilterMatchNminOverLread 0.1 --outFilterMatchNmin
0 --outFilterMismatchNmax 999 -clip3pAdapterMMP 0.2 -clip3pAdapterSeq
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA --twopassMode Basic


STAR --runThreadN 60 --quantMode GeneCounts --genomeDir Pig_genome/star_index --
readFilesIn ./$name/L004/${name}_L004.fastq  --outFileNamePrefix ./$name/L004/${name}_L004_star --
outFilterMultimapNmax 1 --outFilterScoreMinOverLread 0.1 --outFilterMatchNminOverLread 0.1 --outFilterMatchNmin
0 --outFilterMismatchNmax 999 -clip3pAdapterMMP 0.2 -clip3pAdapterSeq
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA --twopassMode Basic
```

#3. Merge sam files from 4 lanes into one file

```
java -jar ~/Downloads/picard-2.18.14.jar MergeSamFiles

I=./$name/L001/$name\_L001_star-Aligned.out.sam

I=./$name/L002/$name\_L002_star-Aligned.out.sam

I=./$name/L003/$name\_L003_star-Aligned.out.sam

I=./$name/L004/$name\_L004_star-Aligned.out.sam

O=PR1643/STAR/$name\.sam


done
```

Feature Count

Feature count algorithm, Subread R (featureCounts) package (Liao et al 2013) was used to count genomic features genes and exons per transcript. A digital gene expression matrix was created from this data.

- Download the R package SUBREAD, for Read summarization
- Use the same reference annotation file used for read aligning

Below is the R script for counting the reads to their genomic features - genes**.** The folder contains the merged sam files.

<u>**#1. Load the necessary package**</u>

library(Rsubread)

#Import the sam files to a folder and Set the RStudio working directory to the location of the folder

extension <- "sam"

fileNames <- Sys.glob(paste(".", extension, sep=""))*

savefolder <- "PR1643/FeatureCount"

<u>**#2. Retrieve the genomic features - gene and exon**</u>

for (fileName in fileNames)

{

##for genes

gene.result <- featureCounts(file=sprintf(fileName),annot.ext=

"Pig_genome/Sus_scrofa.Sscrofa11.1.95_clean.gtf",isGTFAnnotationFile=TRUE, GTF.featureType = "gene",readExtension3=200, ignoreDup = TRUE, isPairedEnd = FALSE,allowMultiOverlap = TRUE, nthreads = 20)

##for exons

exon.result <- featureCounts(file=sprintf(fileName),annot.ext= “Pig_genome/Sus_scrofa.Sscrofa11.1.95_clean.gtf",isGTFAnnotationFile=TRUE, GTF.featureType = "exon",readExtension3=200, ignoreDup = TRUE, isPairedEnd = FALSE,allowMultiOverlap = TRUE, nthreads = 20)

<u>**#3. Export the gene and exon count matrices**</u>

```
write.table(x=data.frame(gene.result$annotation,gene.result$counts), file = sprintf("%s/%s-gene-counts.txt",savefolder,fileName), quote=FALSE, sep="\t")

write.table(x=data.frame(exon.result$annotation,exon.result$counts), file = sprintf("%s/%s-exon-counts.txt",savefolder,fileName), quote=FALSE, sep="\t")

}
```

2. Downstream Analysis

The gene matrix is used for downstream analysis.

1. Annotation: We annotated the genes with gene symbols. The missing gene symbols were substituted by ensemble/wiki gene names.

2. Batch Correction: Multiple batch correction algorithm ComBat-seq (Zhang et al 2020) was used to account for technical variability arising from batch effect.

3. Filtering: Quality of the final matrix is ensured by filtering samples based on the number of dropouts and abundance threshold. Out of 142 samples, 52 Samples with non-zero gene counts <6000 were considered as outliers. Additionally, 10800 genes that are present in very low quantities (<30 non-zero gene counts) were filtered out.

4. Rlog transformation: A regularized log transformation was carried out using DESeq2 (Love et al. 2014).

5. Normalization: We normalized the data using a quantile regression method SCnorm (Bacher et al. 2017).

#1. Load necessary packages:

```
library(tidyr)

library(dplyr)

library(reshape2)

library(biomaRt)

library(DESeq2)

library(Rtsne)
```

library(ggplot2)

library(cluster)

library(gtools)

library(pheatmap)

library(pcaMethods)

library(rgl)

library(limma)

library(sva)

library(edgeR)

library(SCnorm)

library(sva)

#2. Download functions from https://github.com/zhangyuqing/ComBat-seq

source ("help_combat_seq.R")

source ("ComBat_seq.R")

#Import the gene counts file and Set the RStudio working directory to the location of the #gene count matrix.

-

#3. Conversion of gene IDs and annotation

raw <- read.table("PR1643-genes-counts.txt",sep="\t", header = T, row.names = 1)

sums <- rowSums(raw)

raw <- raw[-which(sums==0),]

#Using ensembl database, retrieve gene names, description, wiki gene names and

```
#description

genes <- rownames(raw)

ensembl <- useMart("ensembl")

ensembl <- useMart("ensembl",dataset="sscrofa_gene_ensembl", host="uswest.ensembl.org")

G_list <- getBM(filters= "ensembl_gene_id", attributes=
c("ensembl_gene_id","external_gene_name","description","wikigene_description","wikigene_name"),values=genes,mart=
ensembl)

#Remove duplicate gene rows and substitute the absent gene names (NAs) with #corresponding ensemble IDs/wiki
names

raw2 <- raw; raw2$ensembl_gene_id <- rownames(raw2)

new.list <- G_list[-which(duplicated(G_list$ensembl_gene_id)),]

full.list <- right_join(new.list,raw2)

gene.names <- make.names(full.list$ensembl_gene_id, unique = T)

##gene names where the external gene name and wikigene name are the same

gene.names[which(full.list$external_gene_name != "" & full.list$wikigene_name != "" & full.list$external_gene_name
== full.list$wikigene_name)] <- full.list$external_gene_name[which(full.list$external_gene_name != "" &
full.list$wikigene_name != "" & full.list$external_gene_name == full.list$wikigene_name)]

##gene names where no external gene name was available but wikigene name is, set #to wikigene name

gene.names[which(full.list$external_gene_name == "" & full.list$wikigene_name != "")] <-
full.list$wikigene_name[which(full.list$external_gene_name == "" & full.list$wikigene_name != "")]

##gene names where external gene name was available but wikigene name is not, set #to external gene name
```

```
gene.names[which(full.list$external_gene_name != "" & full.list$wikigene_name == "")] <-
full.list$external_gene_name[which(full.list$external_gene_name != "" & full.list$wikigene_name == "")]


##when external gene name and wikigene name are both available, but not the same, #combine both, separated by
a “_” for easier downstream reference

gene.names[which(full.list$external_gene_name != "" & full.list$wikigene_name != "" & full.list$external_gene_name !=
full.list$wikigene_name)] <- paste0(full.list$external_gene_name[

 which(full.list$external_gene_name != "" & full.list$wikigene_name != "" & full.list$external_gene_name !=
full.list$wikigene_name)], "_" ,

full.list$wikigene_name[which(full.list$external_gene_name != "" & full.list$wikigene_name != "" &
full.list$external_gene_name != full.list$wikigene_name)])


full.list$gene_name <- make.names(gene.names, unique=T)


##change order

full.list <- full.list[,c(1,ncol(full.list), 2:(ncol(full.list)-1))]


##set blanks to NA

full.list2 <- full.list; full.list2[which(full.list=="", arr.ind = T)] <- NA


##substitute commas for a period in the description to enable properly writing output as csv file

full.list2$description <- gsub(",","\\.", full.list2$description); full.list2$wikigene_description <- gsub(",","\\.",
full.list2$wikigene_description)


#Save the corresponding gene annotation csv file


write.csv(full.list2[,1:6], "PR1643_RNAseq_IDs_Genes_Description.csv", row.names=F, quote=F)


full_dataset <- as.matrix(full.list2[,7:ncol(full.list2)]); rownames(full_dataset) <- full.list2$ensembl_gene_id
```

```
rownames(full_dataset) <- full.list2$gene_name



#Save the gene count matrix with the gene IDs as row names

 write.table(full_dataset,"PR1643_raw_RNAseqData_gene_names.txt",sep="\t",quote=F, col.names = NA)
```

#4. Batch correction for the two sets of runs

```
#Remove genes with total expression count = NULL


full <- full_dataset[which(rowSums(full_dataset)>0),]


#Separate the batches of 47 and 95 samples

 batch47 <- full[,1:47]

batch95 <- full[,48:142]


col.data <- data.frame(batch=c(rep("Batch47",47), rep("Batch95",95)));

rownames(col.data) <- colnames(full)

annot_samp <- col.data

annot_cols <- NA


counts <- as.matrix(full)

batch <- as.factor(c(rep("b47", 47), rep("b95", 95)))


#Make sure that the parameters for the function ComBat_seq are individually specified

 group <- NULL
```

```
full_mod <- FALSE

covar_mod <- NULL

shrink <- FALSE

shrink.disp <- NULL

gene.subset.n <- NULL


adjusted<- ComBat_seq(counts, batch=batch)


#Save the batch adjusted data into a new file


write.table(adjusted,"FullSet_142_adjusted.txt", sep="\t", quote=F)
```

# 5. Filtering Data

```
PR1643.142.adjusted <- adjusted


#Assess genes – samples with non-zero counts


samples.with.counts <- apply(PR1643.142.adjusted, 1, function(x)(sum(x!=0)))

tot.counts <- apply(PR1643.142.adjusted,1, function(x)(sum(x)))

average <- apply(PR1643.142.adjusted,1, function(x)(mean(x)))


stats <- data.frame(samples.with.counts,tot.counts,average)

ggplot(stats, aes(samples.with.counts)) + geom_histogram() + scale_x_log10() + scale_y_log10()

ggplot(stats, aes(1:142,samples.with.counts)) + geom_bar() + scale_x_log10() + scale_y_log10()


#Assess samples
```

```
samples.with.counts <- apply(PR1643.142.adjusted, 2, function(x)(sum(x!=0)))
tot.counts <- apply(PR1643.142.adjusted,2, function(x)(sum(x)))
average <- apply(PR1643.142.adjusted,2, function(x)(mean(x)))

stats <- data.frame(samples.with.counts,tot.counts,average)
ggplot(stats, aes(samples.with.counts)) + geom_histogram() #+ scale_x_log10() + scale_y_log10()
ggplot(stats, aes(tot.counts)) + geom_histogram() + scale_x_log10() + scale_y_log10()
ggplot(stats, aes(average)) + geom_histogram() + scale_x_log10() + scale_y_log10()

#First level filtering

PR1643.142.adjusted.filt <- PR1643.142.adjusted[,-which(samples.with.counts < 3000 & tot.counts < 100000)]

#Save the file –after first filtering -132 samples

write.table(PR1643.142.adjusted.filt, "PR1643-132-adjusted.txt", sep="\t", quote=F)
 raw <- PR1643.142.adjusted.filt


#Remove genes with max rld value of 0 or less

raw2 <- t(apply(raw+1,1,function(x)(log2(x))))
max.genes <- apply(raw2,1,max)
max.samples <- apply(raw2,2,max)

samples.with.counts.filt <- apply(raw, 2, function(x)(sum(x!=0)))
```

```
genes.with.counts.filt <- apply(raw, 1, function(x)(sum(x!=0)))


mat.ordered2 <- raw2[rev(order(genes.with.counts.filt)), rev(order(samples.with.counts.filt))]

mat.ordered3 <- mat.ordered2; mat.ordered3[which(mat.ordered2==0)] <- NA        ##still log2

raw.ordered2 <- raw[rev(order(genes.with.counts.filt)), rev(order(samples.with.counts.filt))]


#Filtering samples with non-zero gene counts <6000 and genes <30


filtered <- mat.ordered2[1:15000,1:90]

filtered.raw <- raw.ordered2[1:15000,1:90]

write.table(filtered.raw, "PR1643_adjusted_raw_90samps_15kgenes.txt", sep="\t", quote=F)
```

#6. Regularized log normalization - DESeq

```
ragp.raw <- filtered.raw


#Regularized log transformation : rlog takes long to run

ragp.rlog <- rlog(ragp.raw, fitType = "local", blind = T);

 rownames(ragp.rlog) <- rownames(ragp.raw)


 #Median centering

ragp.rlog.med <- t(apply(ragp.rlog,1,function(x)(x-median(x))))


#Extract high expression genes for visualization

count.means <- apply(ragp.raw,1,mean)
```

```
highgenes <- rownames(ragp.raw)[which(count.means>10)]


#Using total counts

top200 <- rev(sort(rowSums(ragp.raw)))[1:200]

top2000 <- rev(sort(rowSums(ragp.raw)))[1:2000]
```

#7. Normalization- SCnorm

```
#Input is DESeq rlog matrix (90 samples, 15000 genes)

mySCData <- SingleCellExperiment::SingleCellExperiment(assays = list('counts' = ragp.rlog))

Conditions <- as.factor(annot_samp$batch)


countDeptEst <- plotCountDepth(Data = mySCData, Conditions = Conditions,FilterCellProportion = .1, NCores=4)

str(countDeptEst)

head(countDeptEst[[1]])


mySCData = SingleCellExperiment::counts(mySCData)


#SCnormalization

DataNorm <- SCnorm(Data = mySCData,Conditions = Conditions,PrintProgressPlots = TRUE,FilterCellNum = 10,
NCores=4, reportSF = TRUE)


 DataNorm

 NormalizedData.rlog <- SingleCellExperiment::normcounts(DataNorm)

 row.names(NormalizedData.rlog) <- row.names(ragp.rlog)
```

NormalizedData.rlog[1:5,1:5]

#Save the normalized count data into a new file

write.table(NormalizedData.rlog,"PR1643-normalized_90samples_15kgenes.txt", sep="\t", quote=F, col.names = NA, row.names = T)

Troubleshooting

Problem: SCnorm can not converge.

Solution: Define the value of parameter K=18, based on the evaluation plots.

set.seed(xxx)

DataNorm <- SCnorm(Data = mySCData,Conditions = Conditions,PrintProgressPlots = TRUE,FilterCellNum = 10, K=18, NCores=4, reportSF = TRUE)

Time Taken

Anticipated Results

References

Foley, JW, Zhu, C, Jolivet, P, Zhu, SX, Lu, P, Meaney, MJ *et al.*. Gene expression profiling of single cells from archival tissue with laser-capture microdissection and Smart-3SEQ. Genome Res. **29** (11):1816-1825 (2019)

A. Dobin, C.A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, T.R. Gingeras. STAR: ultrafast universal RNA-seq aligner. Bioinformatics, **29,** 15-21 (2013).

Yang Liao, Gordon K Smyth and Wei Shi The R package Rsubread is easier, faster, cheaper and better for alignment and quantification of RNA sequencing reads. *Nucleic Acids Research*, **47**(8):e47 (2019).

Zhang, Y., Parmigiani, G., and Johnson, W.E. (2020). ComBat-Seq: batch effect adjustment for RNA-Seq count data. BioRxiv.

Love, M. I., Huber, W. & Anders, S. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol.* **15**, 550 (2014).

Bacher R, Chu LF, Leng N, Gasch AP, Thomson JA, Stewart RM, Newton M, Kendziorski C. SCnorm: robust normalization of single-cell RNA-seq data. Nature Methods. **14**(6):584-6 (2017).

Acknowledgements

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- scRNAseqdataanalysisprotocol.pdf