BD2K-LINCS
DATA COORDINATION AND
INTEGRATION CENTER

Annotated Example of using RNA-Seq Pipeline for LINCS

Introduction

This pipeline describes a reproducible analysis of RNA-Seq data using R Markdown. We use an example data set from Grifford et al., that contains seven libraries. We took two samples of HCC1419 cells that were untreated and another two treated with Lapatinib for only 9 days before harvesting (drug tolerant persisters, DTPs). The raw data (GSE62074) is downloaded from the Sequence Read Archive (SRA) database.

In this document, all necessary softwares and R packages are already installed in the system and all the FASTQ files are downloaded, unzipped, and saved in the input directory. In this particular example, feature is gene, species is Homo Sapiens and we consider single-end reads. The following steps are used in this pipleine to analyze the raw sequence data.

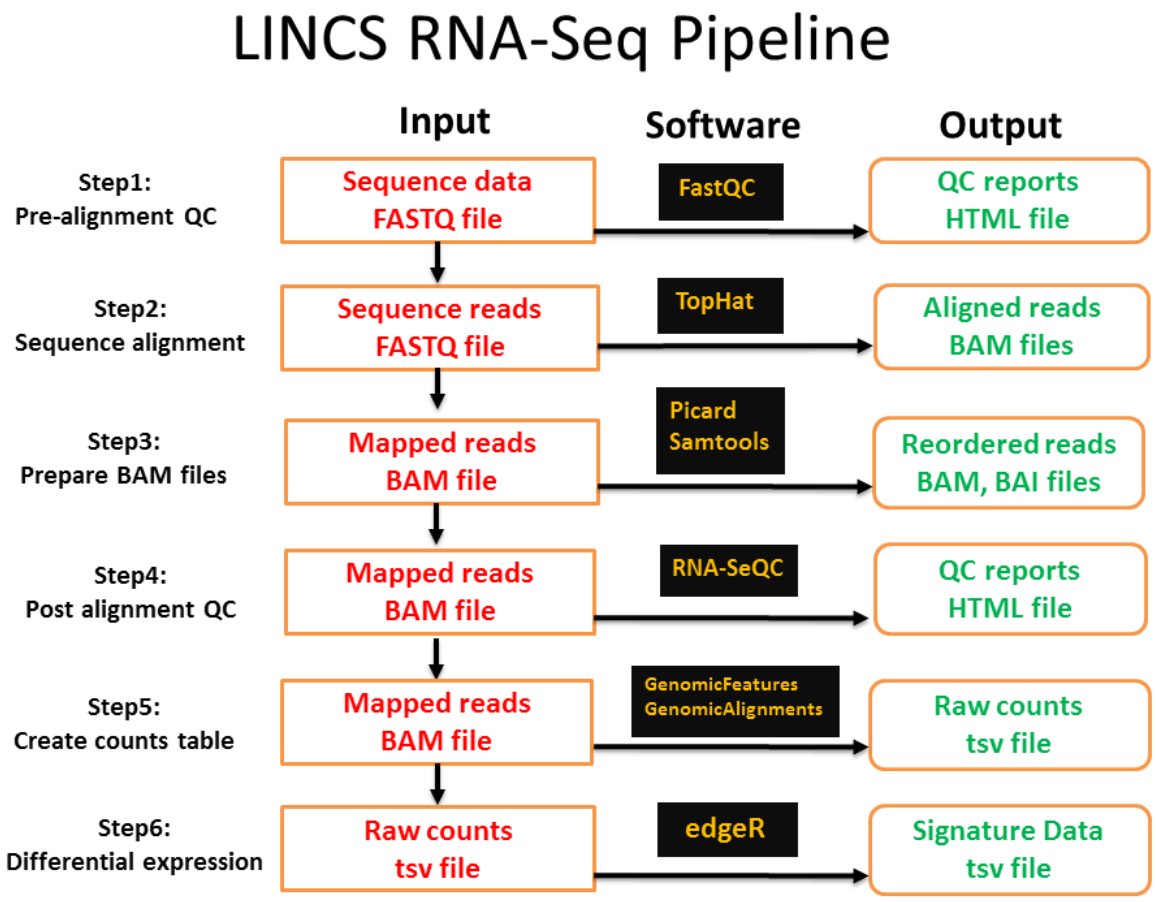

Figure 1: Steps for LINCS RNA-Seq pipeline.

Directories and some parameters

- Directory of the FASTQ files:

```
inputDirectory= "./inputDirectory"
```

- Download FASTQ files to the inputDirectory

```
for (names in c("SRR1600265_1_1m.fastq", "SRR1600266_1_1m.fastq", "SRR1600267_1_1m.fastq",
    "SRR1600268_1_1m.fastq")) {
    download.file(paste0("http://eh3.uc.edu/genomics/lincs/public/rnaseqEx/",
        names, sep = ""), paste0(inputDirectory, "/", names, sep = ""))
}
```

```
dir("inputDirectory/")
```

```
## [1] "SRR1600265_1_1m.fastq" "SRR1600266_1_1m.fastq" "SRR1600267_1_1m.fastq"
## [4] "SRR1600268_1_1m.fastq"
```

- Directory of the results

```
outputDirectory= "./outputDirectory"
```

- Sample metadata file

```
sampleInfo= read.table(file="./sampleInfo.tsv",header=TRUE, sep="\t")
sampleInfo
```

```
##                fastq_id  sample_id cell_line              perturbagen
## 1 SRR1600265_1_1m.fastq SRR1600265   HCC1419                untreated
## 2 SRR1600266_1_1m.fastq SRR1600266   HCC1419                untreated
## 3 SRR1600267_1_1m.fastq SRR1600267   HCC1419 1micron Lapatinib 9 days
## 4 SRR1600268_1_1m.fastq SRR1600268   HCC1419 1micron Lapatinib 9 days
```

- Type of sequencing and species

```
PE <- FALSE
Species= "Hs"
```

- Gene annotation file. This file contains information of the genes and is downloaded from UCSC with the genome version "hg19" (hg19, GRCh37 Genome Reference Consortium Human Reference 37 (GCA_000001405.1)).

```
Hs.genome.gtf <- "/opt/ga4raid/pipeline/opt/iGenomes-2015-08-15/Homo_sapiens/UCSC/hg19/Annotation/Genes/
```

- Genome index file. This index file is built using Bowtie2 in order to align the reads quickly.

```
Hs.genome.index <- "/opt/ga4raid/pipeline/opt/iGenomes-2014-05-28/Homo_sapiens/UCSC/hg19/Sequence/Bowti
```

- Human genome reference file. We consider human reference hg19 and associated FASTA file is downloaded from UCSC along with the gene annotation.

```
Hs.genome.fa <- "/opt/ga4raid/pipeline/opt/iGenomes-2014-05-28/Homo_sapiens/UCSC/hg19/Sequence/WholeGen
```

- Location of Picard which is a set of command line tools for manipulating high-throughput sequencing data and formats such as SAM/BAM.

```
picard.path="/opt/ga4raid/pipeline/opt/picard/trunk/dist/"
```

- Location of FastQC software for post alignment QC

```
fastqc.path="/opt/ga4raid/pipeline/opt/FastQC/fastqc"
```

- Location of RNA-SeQC software for post alignment QC

```
rna_seqc.path="/opt/ga4raid/pipeline/opt/RNA-SeQC_v1.1.7.jar"
```

STEP 1: Pre-alignment QC

We start with pre-alignment quality control for each of the FASTQ files using FastQC (version: v0.9.4).

```
#get all the fastq files
fastq.files = list.files(inputDirectory, pattern=".fastq$", full=TRUE)
lapply(fastq.files,function(x)
{
#run FastQC on each of the files
    cmd=paste(fastqc.path," ",x,sep="")
    system(cmd)
#move the files created by FastQC to the output directroy
    cmd=paste("mv ",sub(".fastq","_fastqc",x),"* ",out.dir=outputDirectory,sep="")
    system(cmd)
})
```

STEP 2: Sequence alignment

Step 2 runs sequence alignment using TopHat (version: v2.0.3) software and creates BAM files. TopHat has dependencies on Bowtie2 (version: 2.2.9) and Samtools (version: 1.3.1). The output includes:

1. accepted_hits.bam: A list of read alignments in SAM format
2. junctions.bed: A UCSC BED track of junctions reported by TopHat.
3. insertions.bed and deletions.bed: UCSC BED tracks of insertions and deletions reported by TopHat.

```
lapply(fastq.files, function(ff) {
    # '-p' indicate number of threads to run which is 1 here. '-G' indicate the
    # gtf file location. Library type is fr-unstranded which means reads from
    # the left-most end of the fragment (in transcript coordinates) map to the
    # transcript strand and the right-most end maps to the opposite strand.
    cmd = paste("nohup ", tophat.path = "tophat", " -p 1 -G ", genome.gtf = Hs.genome.gtf,
        " --library-type fr-unstranded --keep-fasta-order --no-novel-junc -o ",
        outputDirectory, "/tophat_out", sub(".fastq", "", basename(ff)), " ",
        genome.index = Hs.genome.index, " ", ff, sep = "")
    system(cmd)
})
```

STEP 3: Prepare BAM files

Step 3 reorders and add read group information to the bam files using Picard (version: 1.59) and indexes them using samtools (version: 1.3.1) softwares.

```
bam.files = list.files(outputDirectory, "accepted_hits.bam$", full = TRUE, recursive = T)
lapply(bam.files, function(s) {
    # reorder bam files (accepted_hits_sorted.bam) according to reference genome
    # which is needed to run RNA-SeQC in step 4.
    cmd = paste("java -jar ", picard.path, "/ReorderSam.jar ALLOW_INCOMPLETE_DICT_CONCORDANCE=true I=",
        s, " O=", sub(".bam", "_sorted.bam", s), " R=", genome.fa = Hs.genome.fa,
        sep = "")
    system(cmd)

    # The following command add read group information to the bam files. In the
    # simple case where a single library preparation derived from a single
    # biological sample that was run on a single lane of a flowcell, all the
    # reads from that lane run belong to the same read group. When multiplexing
    # is involved, then each subset of reads originating from a separate library
    # run on that lane will constitute a separate read group.
    cmd = paste("java -jar ", picard.path, "/AddOrReplaceReadGroups.jar I=",
        sub(".bam", "_sorted.bam", s), " O=", sub(".bam", "_added.bam", s),
        " SORT_ORDER=coordinate RGID=1 RGLB=1 RGPL=illumina RGPU=barcode RGSM=1",
        sep = "")
    system(cmd)

    # The following command creates an index file (.bai) corresponding to the
    # bam files. This file acts like an external table of contents and allows
    # programs (e.g., IGV) to jump directly to specific parts of the bam file
    # without reading through all of the sequences.
    cmd = paste(samtools.path = "samtools", " index ", sub(".bam", "_added.bam",
        s), sep = "")
    system(cmd)
})
```

STEP 4: Post alignment QC

Step 4 generates post alignment QC reports using RNA-SeQC (version: 1.1.7) which includes read count metrics, correlation analysis, coverage metrics, mean coverage etc.

```
bam.files = list.files(outputDirectory, "accepted_hits_added.bam$", full = TRUE,
    recursive = T)
sampleid = sapply(strsplit(bam.files, "/"), function(x) x[length(x) - 1])
sampleid = sub("tophat_out", "", sampleid)
note = sapply(strsplit(sampleid, "_"), function(x) x[1])
dat = data.frame(`Sample ID` = sampleid, `Bam File` = bam.files, Notes = note,
    stringsAsFactors = F)
write.table(dat, file = paste0(outputDirectory, "/sample.file"), col.name = T,
    row.name = F, quote = F, sep = "\t")

if (PE == TRUE) {
    single.or.paired <- ""
} else {
    single.or.paired <- "-singleEnd"
}
system(paste0("mkdir ", outputDirectory, "/rna_seqc"))
cmd = paste("nohup java -jar ", rna_seqc.path, " -o ", paste0(outputDirectory,
    "/rna_seqc"), " -r ", genome.fa = Hs.genome.fa, " -s ", samplefile = paste0(outputDirectory,
    "/sample.file"), " ", single.or.paired, " -t ", genome.gtf = Hs.genome.gtf,
    sep = "")
system(cmd)
```

STEP 5: Counting reads and create counts table

Step 5 generates raw counts table using R packages GenomicAlignments (version: 1.6.3) and GenomicFeatures (version: 1.22.13).

```
library(GenomicFeatures)
library(GenomicAlignments)

# Download refGene as a TranscriptDb(TxDb) object.
refGene <- makeTxDbFromUCSC(genome = "hg19", tablename = "refGene")

# Download exons by gene
exonRangesList <- exonsBy(refGene, "gene")

bam.files = list.files(outputDirectory, "accepted_hits_added.bam$", full = TRUE,
    recursive = T)
countsTable <- NULL
for (i in bam.files) {
    if (PE == TRUE) {
        params <- ScanBamParam(flag = scanBamFlag(isUnmappedQuery = FALSE),
            tag = "NH")
        aligns <- readGAlignmentPairs(i, format = "BAM", param = params)
        # read genomic alignments for paired-end
    } else {
        params <- ScanBamParam(flag = scanBamFlag(isUnmappedQuery = FALSE),
            tag = "NH")
```

```
        aligns <- readGAlignments(i, index = i, param = params)
        # read genomic alignments for single-end
    }
    strand(aligns) = "*"
    unique_hits <- aligns[mcols(aligns)$NH == 1]  #remove multimapping reads

    # Counting reads using summarizeOverlaps with mode='Union'. In this mode,
    # reads that overlap any portion of exactly one feature are counted. Reads
    # that overlap multiple features are discarded. Also, inter.feautre is a
    # logical indicating if the counting mode should be aware of overlapping
    # features. When TRUE, reads mapping to multiple features are dropped.
    counts <- assays(summarizeOverlaps(exonRangesList, unique_hits, mode = "Union",
        inter.feature = TRUE, SingleEnd = (!PE)))$counts
    countsTable <- cbind(countsTable, counts)
}
colnames(countsTable) <- as.vector(sampleInfo[, 2])
countsTable <- data.matrix(data.frame(entrez_geneID = rownames(countsTable),
    countsTable, stringsAsFactors = FALSE))
rownames(countsTable) <- NULL

write.table(countsTable, file = paste0(outputDirectory, "/countsTable.tsv"),
    quote = FALSE, sep = "\t", row.names = FALSE)

head(countsTable)
```

```
##      entrez_geneID SRR1600265 SRR1600266 SRR1600267 SRR1600268
## [1,]             1          0          5          4          1
## [2,]            10          0          0          0          0
## [3,]           100          0          0          0          0
## [4,]          1000          0          0          0          0
## [5,]         10000          0          1          1          0
## [6,]     100008586          0          0          0          0
```

STEP 6: Differential expression analysis for two group comparison

Step 6 analyze the raw count data cretaed in step 5 using R package edgeR (version:3.12.1), locfit (version: 1.5.9.1), and "org.Hs.eg.db" (version: 3.2.3).

```
library(edgeR)
library(locfit)
library(paste0("org.", Species, ".eg.db"), character.only = TRUE)

# Create DGEList
y <- DGEList(counts = countsTable[, -1], genes = countsTable[, 1])

# Get the levels of the two groups
sampleGroup = as.factor(ifelse(colnames(countsTable)[-1] %in% c("SRR1600267",
    "SRR1600268"), "drugTreatment", "Control"))
Group <- relevel(sampleGroup, ref = "Control")
y$samples$group <- Group

# Genes with a count per million (CPM) value greater than 1 in more samples
```

```
# than the smaller sample size between two groups are retained for the
# subsequent analyses and other genes are filtered out.
o <- order(rowSums(y$counts))
y <- y[o, ]
keep <- rowSums(cpm(y) > 1) >= min(summary(factor(Group)))
y <- y[keep, ]
dim(y)
```

```
## [1] 12835     4
```

```
# Recompute library sizes
y$samples$lib.size <- colSums(y$counts)
y$samples
```

```
##                     group lib.size norm.factors
## SRR1600265       Control   333860            1
## SRR1600266       Control   314452            1
## SRR1600267 drugTreatment   295435            1
## SRR1600268 drugTreatment   297628            1
```

```
# Normalization: to get effective library size; default method: trimmed mean
# of M-values(TMM)
y <- calcNormFactors(y)

# estimate dispersion and test for DE
y <- estimateDisp(y)
```

```
## Design matrix not provided. Switch to the classic mode.
```

```
fit <- exactTest(y, pair = levels(Group))

# sort the result table by the p-values.
top_degs <- topTags(fit, n = nrow(fit$table))
signaturesData <- top_degs$table
rownames(signaturesData) <- NULL

# Get gene symbol
signaturesData$Gene <- sapply(mget(as.character(signaturesData$genes), get(paste0("org.",
    Species, ".egSYMBOL")), ifnotfound = NA), function(x) x[1])

# signaturesData table
signaturesData <- signaturesData[, c(6, 2, 3, 4)]
colnames(signaturesData) <- c("Gene", "Sig1_LogFoldChange", "Sig1_LogCountPerMillion",
    "Sig1_PValue")

# SignaturesMetaData table
SignaturesMetaData <- data.frame(SignatureID = "Sig1", TreatmentSamples = paste(sampleInfo[-which(sampl
    "untreated"), 2], collapse = ", "), ControlSamples = paste(sampleInfo[which(sampleInfo$perturbagen =
    "untreated"), 2], collapse = ", "), CellLine = unique(sampleInfo$cell_line))

# save the results
```

```
write.table(signaturesData, file = paste0(outputDirectory, "/signaturesData.tsv"),
    quote = FALSE, sep = "\t", row.names = FALSE)
write.table(SignaturesMetaData, file = paste0(outputDirectory, "/SignaturesMetaData.tsv"),
    quote = FALSE, sep = "\t", row.names = FALSE)

head(signaturesData)
```

```
##      Gene Sig1_LogFoldChange Sig1_LogCountPerMillion  Sig1_PValue
## 1   MUC19           6.006471               11.151981 3.495780e-54
## 2  IGFBP5           4.466220                9.801600 2.300321e-28
## 3    PSCA           4.013605               10.568195 4.928881e-23
## 4  SYNPO2           3.405706                8.163808 1.539999e-19
## 5 SLC7A11          -5.614975                6.646949 1.893714e-18
## 6     SCD          -3.402830               11.777374 2.741182e-16
```

```
head(SignaturesMetaData)
```

```
##   SignatureID       TreatmentSamples         ControlSamples CellLine
## 1        Sig1 SRR1600267, SRR1600268 SRR1600265, SRR1600266  HCC1419
```

```
sessionInfo()
```

```
## R version 3.3.0 (2016-05-03)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.4 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] org.Hs.eg.db_3.2.3   RSQLite_1.0.0        DBI_0.4-1
##  [4] AnnotationDbi_1.32.3 IRanges_2.4.8        S4Vectors_0.8.11
##  [7] Biobase_2.30.0       BiocGenerics_0.16.1  locfit_1.5-9.1
## [10] edgeR_3.12.1         limma_3.26.8
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.5     lattice_0.20-33 digest_0.6.9    grid_3.3.0
##  [5] formatR_1.4     magrittr_1.5    evaluate_0.9    stringi_1.1.1
##  [9] rmarkdown_1.0   splines_3.3.0   tools_3.3.0     stringr_1.0.0
## [13] yaml_2.1.13     htmltools_0.3.5 knitr_1.13
```