Worksheet 6.1 Introduction to R and featureCounts

Author: Qing Yang (qing.yang@colorado.edu)

Useful Resources:

- Introduction to R
- FeatureCounts Manual (Section 6.2)

Note: As we will be using R throughout today's workshop, to differentiate R commands from bash commands, two different command prompts are used:

```
YOUR_USERNAME>          Corresponds to commands for bash shell
>                       Corresponds to commands for R
```

Section A: Introduction to R

1. Launch R from terminal by entering:

```
YOUR_USERNAME> R
```

2. Once you are in R workspace, you can enter command after the ">" prompt. Try perform a basic calculation in R:

```
> 3+4
```

And once you hit return, you should get the answer:

```
[1] 7
```

3. **Creating new variables**. Just like any other programming languages, variables are user-defined values that can be changed and invoked by the user. In R, user can assign or change the value within the variable using the operator "<-". For example:

```
> a <- 3
```

Assigns integer 3 to the variable designated as "a". In R, user can perform arithmetic operations on numeric variables:

```
> a <- 3
> b <- 4
> a + b
```

Should again return:

```
[1] 7
```

4. **Functions.** Functions in programming are pre-defined procedures that perform operations on user-provided input. Based on the type of the functions, some would return an output (such as 7 when you enter 3+4) but others will not (such as a function for deleting files). R is built as a statistical tool, so there are numerous useful statistical functions even in the base package. The example below will walk you through a few basic functions.

    a. First, we'll use a function simply named "c" to create a list of integers and assign it to a variable.

```
> myList <- c (2, 4, 5, 6, 7, 11, 120, 1, 3)
```

    b. With existing R functions, you can calculate the sum, mean, and standard deviation of the list of integers.

```
> sum(myList)
[1] 159
> mean(myList)
[1] 17.66667
```

```
> sd(myList)
[1] 38.49026
```

As you might have noticed, invoking a R function requires a function name (e.g. sum, mean) followed by user-provided input within the parentheses (e.g. myList). Some functions require more than one input, and multiple inputs are separated by comma. To figure out the required input for a specific function, type in "?" followed by the function name.

Section B: RNA-seq Read counting using featureCounts in R

The RNA-seq data we are analyzing today is generated using human colon cancer cells (HCT116) that are either treated with DMSO or Nutlin. The original publication can be found here. And here's a paragraph from the original paper describing their method for RNA-seq (note that Ion Torrent was used for sequencing, so the resulting data is considered single-end mRNA-seq):

Total RNA was extracted using TRI Reagent, according to the manufacturer's instructions. PolyA+ RNA was purified using the Dynabeads mRNA DIRECT Micro Purification Kit (Life Technologies). Libraries for Ion Torrent sequencing were prepared using the Ion Total RNAseq v2 kit (Life Technologies). Ion Torrent template preparation was carried out using the Ion PI Template OT2 200 kit v2 and sequenced in an Ion Torrent Proton instrument.

The RNA-seq reads have already been trimmed and mapped to human genome (hg38). For read counting, we will need to start with mapped and sorted bam files.

1. Back in terminal, start by creating a new scratch directory for RNA-seq, and go to the directory you just created:
```
YOUR_USERNAME> mkdir /scratch/Users/USERNAME/RNA-seq; cd /scratch/Users/USERNAME/RNA-seq
```
2. Launch R from terminal:
```
YOUR_USERNAME> R
```
3. Before we start, let's define a few variables needed for read counting:
    a. The input mapped bam files from RNA-seq (file needs to be in quotes)
```
> input <- c("/scratch/Workshop/SR2019/6_RNA-
seq/mapped/bams/SRR4098426_chr1.sorted.bam","/scratch/Workshop/SR2019/6_RN
A-seq/mapped/bams/SRR4098427_chr1.sorted.bam",
"/scratch/Workshop/SR2019/6_RNA-
seq/mapped/bams/SRR4098428_chr1.sorted.bam","/scratch/Workshop/SR2019/6_RN
A-seq/mapped/bams/SRR4098429_chr1.sorted.bam")
```
    b. Unique identification for each sample (need to follow the same order as the bams)
```
> sample_ID <- c("hct116_DMSO_1","hct116_DMSO_2","hct116_nutlin_1",
"hct116_nutlin_2")
```
    c. The output directory where the R data will be stored
```
> outdir <- "/scratch/Users/USERNAME/RNA-seq"
```
    d. The genome annotation file containing genomic loci of exons and introns of each gene
```
> hg38gtf <- "/scratch/Workshop/hg38/hg38_refseq.gtf"
```
4. Similar to loading modules in terminal, we also need to load packages in R in order to use special functions or tools. **R package** is a collection of specific functions, data and compiled code. The direcotry that the R package is stored is called **library**. The package needed for counting reads is called "RSubRead". Load package by:
```
> library("Rsubread")
```

5. Next, invoke the featureCounts function from subRead package, and store the output to a new variable called count_matrix:

```
> count_matrix <- featureCounts(files=input,
                      annot.ext=hg38gtf,
                      isGTFAnnotationFile=TRUE,
                      GTF.featureType="exon",
                      GTF.attrType="gene_id",
                      useMetaFeatures=TRUE,
                      allowMultiOverlap=TRUE,
                      largestOverlap=TRUE,
                      countMultiMappingReads=TRUE,
                      isPairedEnd=FALSE,
                      nthreads=8)
```

6. The process should take less than a minute to run. If the counting is successful, the end of command prompt should say "Read assignment finished.", and the read counts will be stored in the variable **count_matrix**. We can examine the variable:

```
> summary(count_matrix)
           Length Class      Mode
counts     319994 -none-     numeric
annotation      6 data.frame list
targets         4 -none-     character
stat            3 data.frame list
```

Here, **counts** refers to the total number of unique genes that was in the GTF file used for counting, **annotation** refers to the genomic location of each feature being counted, **targets** are the input samples, and **stat** is the summary of how many reads are being used for counting.

7. By default, featureCounts use the full path to the input bam files as the sample name. To make downstream analysis easier, it's better to change the sample names to be something more descriptive:

```
> colnames(count_matrix$counts) <- sample_ID
```

8. Finally, as a good practice, always save the R image to your working directory. Here, we use paste command to append the output directory we specified earlier so the workspace is saved there:

```
> save.image(paste0(outdir, "/featureCounts_workspace", ".RData"))
```

Once the R image is saved, it is safe to exit R without losing track of your progress. Exit R by typing in the command:

```
> q()
```

R would prompt the question:

```
Save workspace image? [y/n/c]:
```

Since we have already saved the image, you can respond to that prompt with "n". And the saved R image can be reloaded once you re-launch R followed by typing in the following command in R:

```
> load("/scratch/Users/USERNAME/RNA-seq/featureCounts_workspace.RData")
```