# Build RSE object tutorial

*John R. Thompson*

*28 Dec 2015*

## Contents

## 1 Import and Build a RangedSummarizedExperiment

This file (Build_RSE_Object.rmd) shows how to create a "RangedSummarizedExperiment". Copy the .rmd file into the same folder with your text data files and edit a few lines in the setup code block.

Also described here is some background information on the SummarizedExperiment data structure including how to access the elements contained in the object.

**Code Block: Setup**

```r
#good practice to clear the workspace before starting a new analysis
rm(list=ls())
startTime = Sys.time()

library(magrittr)
library(DGE.Tools)
library(zFPKM)

#### Settings

#set the working dir to the folder containing count and annotation text files.
setwd("~/Fibrosis/IPF Cedar Sinai/RNA-Seq/RefGene2015")

#I like to segregate my output in a subfolder of the input file folder.
RdataFolder = "./RData"  # ./RData will be relative to the working dir

#set the filename to be used for the output (.RDS extension)
RSEfilename = "MyGene_RSE.RDS"

#Specify species (Human | Mouse | Rat | Other)
#Used to add proper EntrezIDs to the annotation which currently supports only
#human, mouse and rat.
Species = "Human"

#uncomment the next line to load Transcript Data instead of Gene data
#GeneData = TranscriptData
```

## 2 RangedSummarizedExperiment Description

The RangedSummarizedExperiment (RSE) is an object class defined in package Summarized-Experiment (Bioconductor 3.2 or higher).

An RSE object contains the following data structures:

- assays: matrices of R genes by C Samples; e.g. counts, FPKM, TPM etc.
- rowRanges: Gene or Transcript annotation; a dataframe
- colData: Sample annotations; a dataframe
- metadata: Additional experimental information not linked to rows or cols; e.g. Summarized QC stats

Conveniently, an RSE object can be subsetted like a matrix or dataframe and all the data objects contained in the RSE are properly handled.

e.g.: MySubsetRSE = RSE[1:100, c(2,4,6)]

This command returns the first 100 genes(rows) for samples 2, 4 and 6. All assay matrices contained in the RSE are properly subsetted. The rowRanges and colData are also properly subsetted. The metadata is carried along unchanged.

## 3 Loading Omicsoft Data

Function Build_RSE is called to import the Omicsoft pipeline output (Gene and Sample Annotation, Counts, FPKM, QC.Metrics) and formatted into a RangedSummarizedExperiment (RSE) R data object. Build_RSE will also, calculate and add the following data objects to the RSE object:

- RawLog2CPM (assay)
- RawLog2FPKM (assay)
- RawLog2TPM (assay)
- zFPKM (assay: use zFPKM > -3 to select expressed genes/transcripts)
- QC.Metrics.Summary (metadata: min, max, mean of aligned data QC)

## 4 Loading Data from other RNA-Seq pipelines (Data Structure Requirements)

Minmally, you need to provide **Count** data as well as **gene** and **sample** annotation, each as a separate text file.

The count data is N rows of genes or transcripts by M columns of samples.

Gene annotation is N rows of gene or transcript annotation, the fields **chr**, **start**, **end** and **strand** are required in the gene annotation. This information is used to create a GRanges object that enables you to display data in a genome browser context. If Xpress gene annotation does not provide the chromosome position information, we should request that it be added. In the meantime, the Annotables package is a facile source of that information from Ensembl data. Also, if you provide an **ExonLength** field in your gene annotation (non-redundant count of bases in all exons for a gene), then the output will contain TPM and FPKM data tables.

The sample annotation should have one row of sample annotation for each column of Count data. The first column must be a **SampleID** field that should uniquely identify each sample in a dataset. Other sample-related data and factors should be included here. In particular, if you plan to use duplicateCorrelation, a column called **block** should be present (although it can be added later if necessary).

The filenames built into the **GeneData** List data structure are the default file names for these datatypes coming from the Omicsoft RNA-Seq pipeline. Data from other pipelines can be most easily accommodated by renaming your data files to be consistent with the file names listed in the GeneData and TranscriptData Lists below. If you're more adventurous, you can also create a custom GeneData or TranscriptData list objects to point to the your file names. Note however, that the GeneData and TranscriptData lists are defined in DGE_Tools. So you must edit those variables **after** loading the DGE_Tools library. Do not change the **name** or **type** attributes, only the file name.

Note that the first four elements in these structures are required(Annotation, Design, Counts and Level). There is no need to modify this data structure if you're lacking the optional elements (FPKM and QC). The software is smart enough to ignore them if the files do not exist.

Again, for the adventurous, you can also define new **Assay** or **metadata** slots to hold additional datatypes if desired and they will be automatically imported. You should probably talk to me (JRT) before naming additional elements to avoid name clashes and confusion with data elements created later in the analysis workflow. Note that there is only one slot each available in a SummarizedExperiment for gene/transcript and sample annotation so you can not add additional objects in the row or column data slots. However, it is possible to add additional columns to existing row and column metadata slots.

If you provide a QC.Metrics file (e.g. containing sample-based alignment statistics), It should have one row per statistic (e.g. Pct exon reads, Pct aligned reads, etc) and one column per sample. This data will be transposed and appended (cbind) to the Sample metadata in colData. The column names in the QC data file need to match the **SampleID** field in the existing sample metadata, or else the merge won't work (the software will just skip adding the QC data to the RSE and continue).

```
GeneData = list(
  Annotation = list(name = "Annotation",
                    file = "RNA-Seq.Count.Annotation.txt",
                    type = "rowRanges"),
  Design = list(name = "Design",
                file = "RNA-Seq.Design.txt",
                type = "colData"),
  Counts = list(name = "Counts",
                file = "RNA-Seq.Count.Table.txt",
                type = "Assay"),
  Level = list(name = "Level",
               level = "Gene",
               type = "metadata"),
  FPKM = list(name = "FPKM",
              file = "RNA-Seq.FPKM.Table.txt",
              type = "Assay"),
  QC.Metrics = list(name = "QC.Metrics",
            file = "RNA-Seq.QCMetrics.Table.txt",
            type = "metadata"))

TranscriptData = list(
  Annotation = list(name = "Annotation",
                    file = "RNA-Seq.Transcript_Count.Annotation.txt",
                    type = "rowRanges"),
  Design = list(name = "Design",
                file = "RNA-Seq.Design.txt",
                type = "colData"),
  Counts = list(name = "Counts",
                file = "RNA-Seq.Transcript_Count.Table.txt",
                type = "Assay"),
  Level = list(name = "Level",
               level = "Transcript",
               type = "metadata"),
  FPKM = list(name = "FPKM",
              file = "RNA-Seq.Transcript_FPKM.Table.txt",
              type = "Assay"),
  QC.Metrics = list(name = "QC.Metrics",
            file = "RNA-Seq.QCMetrics.Table.txt",
            type = "metadata"))
```

## 5 Accessing data contained in a RSE object (Accessor Functions)

Once you have your data nicely contained in a SummarizedExperiment object, the RSE datastructure has handy accessor functions to access the data elements it contains.

**Examples:**

- print(RSE) #prints a summary of the contents
- MyCounts = assay(RSE, “Gene.Counts”) #returns a matrix
- MyTPM = assay(RSE, “Gene.TPM”) #returns a matrix
- MyzFPKM = assay(RSE, “Gene.zFPKM”) #returns a matrix
- MySampleAnnotation = colData(RSE) #Returns a DataFrame (behaves similar to dataframe; see ?DataFrame)
- MySampleAnnotation = colData(RSE) %>% as.data.frame #Returns a base dataframe (easier to view/inspect in RStudio)
- MyGeneAnnotation = rowRanges(RSE) #returns a Large GRanges object (but not very convenient to access geneID, genesym, description,etc.)
- MyGeneAnnotation = mcols(RSE) #Gets the gene metadata out as a DataFrame
- MyGeneAnnotation = mcols(RSE) %>% as.data.frame #gets the metadata out as a data.frame
- QC.Metrics.Summary = metadata(RSE)[[“QC.Metrics.Summary”]] #Returns a dataframe of alignment stats
- Created = metadata(RSE)[[“Created”]] #Returns a list of Session.Info and other creation details (use ls(Created) to see what’s there)

See SummarizedExperiment:: for more accessor functions
See ?SummarizedExperiment for documentation
See ?mcols for more details See ?DataFrame for more detail on this data structure. It’s different from a base data.frame in that RStudio won’t be able to open and view it. But it otherwise generally behaves like a regular data.frame. “The most notable exception is that the row names are optional. This means calling rownames(x) will return NULL if there are no row names.”

## 6 Workflow Best Practice Suggestion

Suggested workflow best practice is to NEVER edit/change any value in the original RSE object. Rather, use the accessor functions exemplified above to pull out individual data elements. If you filter the RSE, for example to remove non-expressed genes (as is the recommend practice), the modified RSE should be saved separately (if necessary) so the original complete dataset remains intact. In this way, the original RSE object remains pristine and complete so that you can reuse the RSE object in multiple scripts or analyses and it will always represent the unmodified original analysis-ready RNA-Seq results.

## 7 zFPKM Analysis and Plot (Filtering Non-expressed Genes)

We create a zFPKM dataframe while reading in the data and creating an RSE Object.

Inspect the zFPKM plot to make sure the fit distribution (blue line) overlaps with the right most peak in the bimodal FPKM distribution (red line).

If the plots look good, you can use zFPKM > -3 filter to select genes detected above threshold.

zFPKM Reference:
BMC Genomics 2013, 14:778
doi:10.1186/1471-2164-14-778;
http://www.biomedcentral.com/1471-2164/14/778

Using zFPKM is better than a simple count filter for low expression genes. A count filter is just an arbitrary empirical filter that is biased against short genes. zFPKM is statistically-based and calibrated with orthogonal ENCODE data.

Note that the zFPKM plotting function produces a faceted plot with one panel for each sample. With very many samples, you may break the plotting code or at a minimum, the plots may get too small to see if you have a large number of samples (JRT has used up to 173 plots and it's still intelligible).

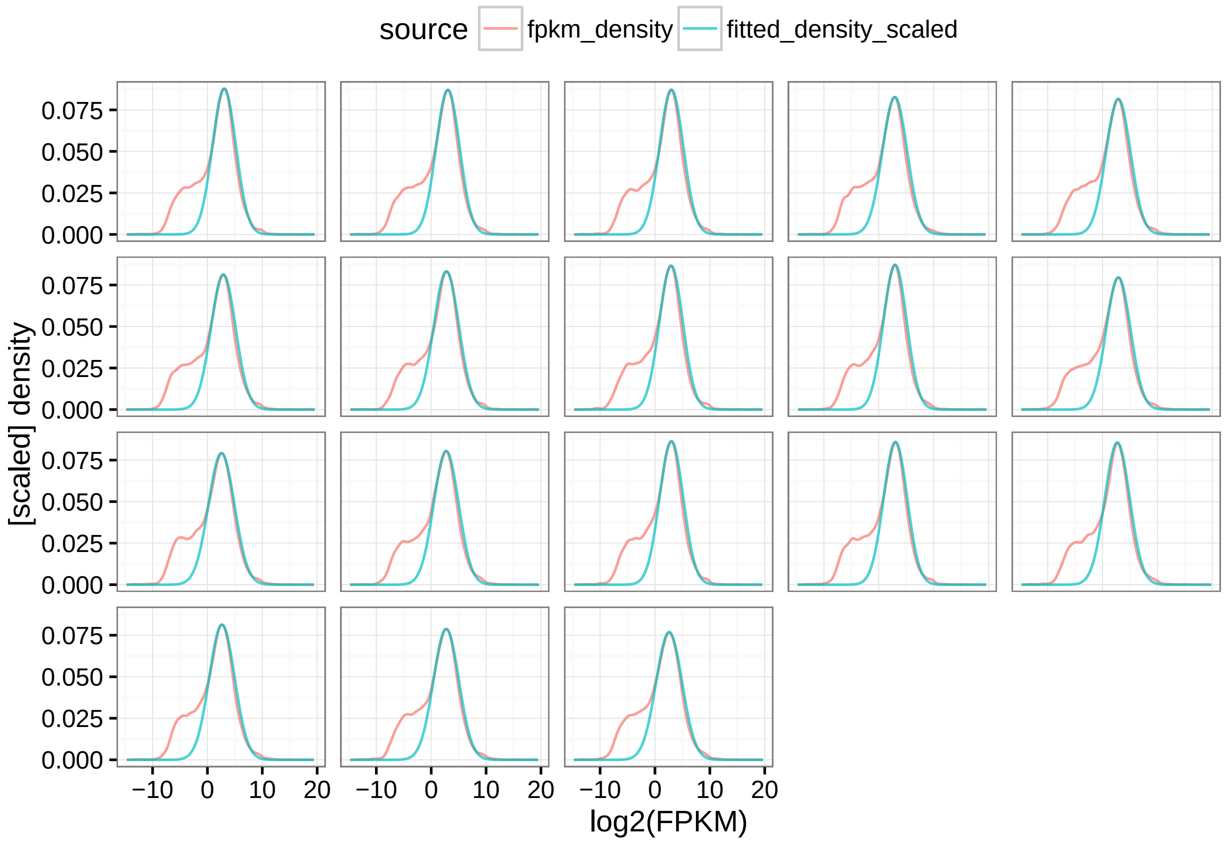
You can suppress the plots (but still calculate zFPKM by setting the showplot and saveplot parameters to FALSE in the function call to Build_RSE.

**Code Block: Build_RSE**

```
RSE = Build_RSE(GeneData, OutputPath=RdataFolder)

#create the output folder, if needed
if (!file.exists(RdataFolder)){
  dir.create(RdataFolder)
}

saveRDS(RSE,file=file.path(RdataFolder, RSEfilename))
```



**Output:**

The RSE object has been written to filename ./RData/MyGene_RSE.RDS.

## 8 Catalog of RSE Data Components

The data objects contained in a RSE object are detailed in the lists below. See section 5 for accessor functions to access the different components of an RSE.

**Assays:**

- RawLog2CPM: calculated by edgeR::cpm function with prior.count=2
- RawLog2FPKM: Calculated from RawLog2CPM if ExonLength data is present.
- RawLog2TPM: Calculated from RawLog2FPKM data if ExonLength data is present.
- fpkmToTpm function from Harold Pimentel used for this conversion.
- zFPKM: Calculated from RawLog2FPKM data if ExonLength is present. If ExonLength is absent but an FPKM datafile was imported (i.e. Omicosoft FPKM data), then we use the imported FPKM values instead.

**MetaData:**

- Level: One of Gene, Transcript or Exon
- QC.Metrics QC data in original format supplied (metrics in rows and samples in columns).
- QC.Metrics.Summary: Min, Max, Mean of QC.Metrics rows
- Created: A List of information about creation of the data
- Session.Info: R Session info
- ByFunction: What Function/Version created the object
- Date: Date of creation
- RVersion: String defining the R Version in use

**Annotation Data:**

- colData: Sample Annotation (includes QC.Metrics columns if provided)
- rowRanges: Gene Annotation as GRanges
- mcols: Gene Annotation as DataFrame

## 9 Session Info

***Time required to process this report:*** *24.20161 secs*

**R Session Info**

| Section | Name | Value |
|---|---|---|
| System | sysname | Windows |
| System | release | 7 |
| System | version | build 7601, Service Pack 1 |
| System | nodename | PF029MHB |
| System | machine | x86 |
| System | login | thompj27 |
| System | user | thompj27 |
| System | effective_user | thompj27 |
| System | Directory | C:/Users/thompj27/Documents/Fibrosis/IPF Cedar Sinai/RNA-Seq/RefGene |
| R | Version | R version 3.2.3 (2015-12-10) |
| Packages | knitr | 1.11 CRAN CRAN 2015-08-14 |
| Packages | envDocument | 2.1.1 BMS BMS 2015-10-24 |
| Packages | zFPKM | 0.0.0.9000 NA NA NA |
| Packages | DGE.Tools | 1.0 NA NA NA |
| Packages | magrittr | 1.5 CRAN CRAN 2014-11-22 19:15:57 |
| Packages | BiocInstaller | 1.20.1 NA NA NA |
| Script | Path | C:/Users/thompj27/Documents/Fibrosis/IPF Cedar Sinai/RNA-Seq/RefGene |
| Script | Modified | 2015-12-28 15:54:42 |