Make new counts file with gene descriptions and new sample codes

Ann Loraine

2023-09-25

Prelude - Define variables

```
genome_version = "SL5"
annotations_fname = ifelse(genome_version=="SL5",
                           "../ExternalDataSets/S_lycopersicum_Jun_2022.bed.gz",
                           "../ExternalDataSets/S_lycopersicum_Sep_2019.bed.gz")
```

This Markdown will use gene annotations from file ../ExternalDataSets/S_lycopersicum_Jun_2022.bed.gz from the genome assembly called "SL5".

Introduction

This R Markdown document consumes a gene quantification (counts) file located in the current working directory, renames samples using a table from a Muday lab manual analysis, adds a new column with gene descriptions, and writes out a new data file to be used as input for differential expression analysis. Gene descriptions come from a gene model annotation file displayed in Integrated Genome Browser.

The gene expression quantification file contains "raw" counts indicating number of RNA-Seq fragments aligned per gene, per sample library, with one complication: the salmon software may split read/fragment counts into fractional values, for read sequences reported as aligning to more than one location in the genome assembly.

The sample renaming file provided by the Muday lab fixes a sample switching error that happened at the sequencing facility. What happened was that a spreadsheet was provided that listed all samples, and that spreadsheet was used to name the data files produced by the sequencer. However, the sequencing team assumed that the order of samples listed in the spreadsheet was the same as the order of RNA sample tubes provided in the physical box sent to the sequencing facility. The Muday lab figured this out by comparing a photograph of the box to the spreadsheet.

This Markdown performs rudimentary sanity-checking, asking:

- How many genes were measured?

The number of genes measured in the gene quantification input file should match the number of unique gene names from column 13 of the gene model annotation file.

- How many samples were measured?

We expect there will be 72 samples surveying 4 time points, 3 genotypes, 2 treatments, and 3 replicates per treatment.

Sample types include:

- Four time points: 15, 30, 45, and 75 minutes.
- Two temperatures: 28 degrees C (cool) or 34 degrees C (warm).

- Three genotypes: ARE, F3H-OX3, VF36

This Markdown also creates a sample renaming file for analysts to use to double-check that sample renaming performed here matches what the Muday lab's analysis recommends.

Results

Load a library we need:

```
library(readxl)
```

Define variables:

```
data_prefix="muday-144-"
data_suffix="_salmon.merged.gene_counts.tsv"
counts_fname=paste0(data_prefix,genome_version,data_suffix)
sample_sheet = './Documentation/muday-144_sample_sheet.xlsx'
sample_renaming_table = "./Documentation/Muday-lab-RNA-samples-for-sample-name-conversion.xlsx"
output_fname = paste0("./results/",data_prefix,genome_version,"_counts-salmon.txt")
```

The preceding code chunk defines variables indicating files used for:

- spreadsheet indicating correct names for samples: ./Documentation/Muday-lab-RNA-samples-for-sample-name-conversion.xlsx
- spreadsheet with sample codes and experimental metadata: ./Documentation/muday-144_sample_sheet.xlsx
- file to write out when done: ./results/muday-144-SL5_counts-salmon.txt

Read gene counts file:

```
counts=read.delim(counts_fname,header = T,row.names = "gene_id")
```

The counts data file contained 36,648 rows.

Column headings for the counts data file, after being read into a data frame, were:

```
names(counts)
```

```
##  [1] "gene_name"          "X7.OE3.15.min.28C"  "X7.OE3.15.min.34C"
##  [4] "X7.OE3.30.min.28C"  "X7.OE3.30.min.34C"  "X7.OE3.45.min.28C"
##  [7] "X7.OE3.45.min.34C"  "X7.OE3.75.min.28C"  "X7.OE3.75.min.34C"
## [10] "X7.VF36.15.min.28C" "X7.VF36.15.min.34C" "X7.VF36.30.min.28C"
## [13] "X7.VF36.30.min.34C" "X7.VF36.45.min.28C" "X7.VF36.45.min.34C"
## [16] "X7.VF36.75.min.28C" "X7.VF36.75.min.34C" "X7.are.15.min.28C"
## [19] "X7.are.15.min.34C"  "X7.are.30.min.28C"  "X7.are.30.min.34C"
## [22] "X7.are.45.min.28C"  "X7.are.45.min.34C"  "X7.are.75.min.28C"
## [25] "X7.are.75.min.34C"  "X8.OE3.15.min.28C"  "X8.OE3.15.min.34C"
## [28] "X8.OE3.30.min.28C"  "X8.OE3.30.min.34C"  "X8.OE3.45.min.28C"
## [31] "X8.OE3.45.min.34C"  "X8.OE3.75.min.28C"  "X8.OE3.75.min.34C"
## [34] "X8.VF36.15.min.28C" "X8.VF36.15.min.34C" "X8.VF36.30.min.28C"
## [37] "X8.VF36.30.min.34C" "X8.VF36.45.min.28C" "X8.VF36.45.min.34C"
## [40] "X8.VF36.75.min.28C" "X8.VF36.75.min.34C" "X8.are.15.min.28C"
## [43] "X8.are.15.min.34C"  "X8.are.30.min.28C"  "X8.are.30.min.34C"
## [46] "X8.are.45.min.28C"  "X8.are.45.min.34C"  "X8.are.75.min.28C"
## [49] "X8.are.75.min.34C"  "X9.OE3.15.min.28C"  "X9.OE3.15.min.34C"
## [52] "X9.OE3.30.min.28C"  "X9.OE3.30.min.34C"  "X9.OE3.45.min.28C"
## [55] "X9.OE3.45.min.34C"  "X9.OE3.75.min.28C"  "X9.OE3.75.min.34C"
## [58] "X9.VF36.15.min.28C" "X9.VF36.15.min.34C" "X9.VF36.30.min.28C"
```

```
## [61] "X9.VF36.30.min.34C" "X9.VF36.45.min.28C" "X9.VF36.45.min.34C"
## [64] "X9.VF36.75.min.28C" "X9.VF36.75.min.34C" "X9.are.15.min.28C"
## [67] "X9.are.15.min.34C"  "X9.are.30.min.28C"  "X9.are.30.min.34C"
## [70] "X9.are.45.min.28C"  "X9.are.45.min.34C"  "X9.are.75.min.28C"
## [73] "X9.are.75.min.34C"
```

As shown above, the counts columns include one column indicating gene name and 72 sample columns.

Also, the sample names as provided by the Muday lab start with a numeral, representing a replicate number. These names come from the prefixes for the fastq files we got from the sequencing provider.

When we write out the new data frame with gene descriptions, we'll modify the column names to use sample codes that are easier to understand and work with. We'll use the same sample codes as in the sample description sheet for this experiment, named `./Documentation/muday-144_sample_sheet.xlsx`.

Read gene model annotation file:

```
annotations = read.delim(annotations_fname,quote="",
                         header=F,as.is=T,sep="\t")[,13:14]
```

The gene model annotations file contained 43,752 rows.

Because this gene model annotation dataset contains many alternatively spliced genes, some genes are listed multiple times. As a result, the `annotations` data frame has many duplicate rows, rows that will mess up the next steps.

Remove the duplicate rows:

```
annotations = unique(annotations[,1:2])
names(annotations)=c("gene_name","description")
row.names(annotations)=annotations$gene
```

The gene model annotations data frame now contains 36,648 rows representing different SL5 gene model annotations.

As expected, the counts file and the annotations file contained the same number of rows.

Merge annotations and counts data:

```
new_counts=merge(counts,annotations,
                 by.x="gene_name",
                 by.y="gene_name")
```

Let's modify column headings to use sample codes as shown in `./Documentation/muday-144_sample_sheet.xlsx`.

As in the above-mentioned sample sheet file, samples codes for this experiment look like:

- [genotype].[temperature].[minutes].[replicate]

where:

- Genotype is one of A (ARE mutant), F (F3H-OX3), or V (VF36)
- Temperature is 28 or 34 degrees C
- Minutes is 15, 30, 45, or 75 minutes.
- Replicate is 7, 8, or 9

Example:

- A.28.15.9 is genotype ARE, 28 degrees C, 15 minutes of heat treatment, replicate 9.

Create a vector with desired column names:

```
genotype = rep(c(rep("F",8),rep("V",8),rep("A",8)),3)
temperature = rep(c("28","34"),36)
```

```
time = rep(c(rep("15",2),rep("30",2),rep("45",2),rep("75",2)),9)
replicate = c(rep("7",24),rep("8",24),rep("9",24))
new_colnames = paste(genotype,temperature,
                     time,replicate,sep=".")
new_colnames = c("gene_name",new_colnames,"description")
```

Sample codes and new column names will be:

```
new_colnames
```

```
##  [1] "gene_name"   "F.28.15.7"   "F.34.15.7"   "F.28.30.7"   "F.34.30.7"
##  [6] "F.28.45.7"   "F.34.45.7"   "F.28.75.7"   "F.34.75.7"   "V.28.15.7"
## [11] "V.34.15.7"   "V.28.30.7"   "V.34.30.7"   "V.28.45.7"   "V.34.45.7"
## [16] "V.28.75.7"   "V.34.75.7"   "A.28.15.7"   "A.34.15.7"   "A.28.30.7"
## [21] "A.34.30.7"   "A.28.45.7"   "A.34.45.7"   "A.28.75.7"   "A.34.75.7"
## [26] "F.28.15.8"   "F.34.15.8"   "F.28.30.8"   "F.34.30.8"   "F.28.45.8"
## [31] "F.34.45.8"   "F.28.75.8"   "F.34.75.8"   "V.28.15.8"   "V.34.15.8"
## [36] "V.28.30.8"   "V.34.30.8"   "V.28.45.8"   "V.34.45.8"   "V.28.75.8"
## [41] "V.34.75.8"   "A.28.15.8"   "A.34.15.8"   "A.28.30.8"   "A.34.30.8"
## [46] "A.28.45.8"   "A.34.45.8"   "A.28.75.8"   "A.34.75.8"   "F.28.15.9"
## [51] "F.34.15.9"   "F.28.30.9"   "F.34.30.9"   "F.28.45.9"   "F.34.45.9"
## [56] "F.28.75.9"   "F.34.75.9"   "V.28.15.9"   "V.34.15.9"   "V.28.30.9"
## [61] "V.34.30.9"   "V.28.45.9"   "V.34.45.9"   "V.28.75.9"   "V.34.75.9"
## [66] "A.28.15.9"   "A.34.15.9"   "A.28.30.9"   "A.34.30.9"   "A.28.45.9"
## [71] "A.34.45.9"   "A.28.75.9"   "A.34.75.9"   "description"
```

Let's check that the new column names correctly rename current column names.

Definite a function that returns `TRUE` if a proposed column name correctly renames a current column name, where the given variable `a_row` is a two-item vector with first item being a proposed new column name and the second item being the corresponding curent column name.

```
areSameSample = function(a_row) {
  new = a_row[1]
  old = a_row[2]
  if (new==old) {
    return(TRUE)
  }
  new_vals = unlist(strsplit(new,"\\."))
  new_genotype = new_vals[1]
  new_temperature = new_vals[2]
  new_time = new_vals[3]
  new_replicate = new_vals[4]
  old_vals = unlist(strsplit(old,"\\."))
  old_genotype = old_vals[2]
  old_replicate = old_vals[1]
  old_time = old_vals[3]
  old_temperature = old_vals[5]
  if (new_time != old_time) {
    print(new_time)
    print(old_time)
    return(FALSE)
  }
  if (new_temperature != substr(old_temperature,1,2)) {
    print(new_temperature)
    print(old_temperature)
```

```
    return(FALSE)
  }
  if (new_replicate!=substr(old_replicate,2,3)) {
    print(new_replicate)
    print(old_replicate)
    return(FALSE)
  }
  new_to_old_lookup = c("are","OE3","VF36")
  names(new_to_old_lookup)=c("A","F","V")
  if (new_to_old_lookup[new_genotype] !=old_genotype) {
    print(new_to_old_lookup[new_genotype])
    print(old_genotype)
    return(FALSE)
  }
  return(TRUE)
}
```

Apply the new function:

```
current_colnames = colnames(new_counts)
to_test = data.frame(new_colnames,current_colnames)
result = apply(to_test,1,areSameSample)
if (sum(result)==length(result)) {
  colnames(new_counts) = new_colnames
} else {
    print("WARNING: Can't rename column names.")
}
```

Did we correctly rename samples using their corresponding sample codes? `YES`.

Now, let's address the sample mislabeling problem that the Muday lab determined was due to a mismatch between the sample spreadsheet and the physical sample box delivered to the sequencing facility.

To start, let's read the sample renaming spreadsheet provided by the Muday lab and translate the Muday lab names to sample codes, the same sample codes used in `new_counts`.

Sample codes:

- [genotype].[temperature].[minutes].[replicate]

```
genotypes=c("V","F","A")
names(genotypes)=c("VF36","OE3","are")
translate = function(x) {
  x1 = sub("#",'',x)
  x2 = sub("C",'',x1)
  v = strsplit(x2," ")[[1]]
  v1 = c(genotypes[v[2]],v[5],v[3],v[1])
  sample_code=paste(v1,collapse = ".")
  return(sample_code)
}
key = read_excel(sample_renaming_table,skip=3)
key=subset(key,!is.na(key[,1]) & !is.na(key[,3]))
```

The sample renaming spreadsheet lists 72 sample names in column 2 and 72 sample names in column 3.

Use Muday lab sample renaming key to rename `new_counts` columns:

```
original=sapply(unlist(as.vector(key[,2])),translate)
desired = sapply(unlist(as.vector(key[,3])),translate)
names(desired)=original
current_names = names(new_counts)
corrected_names = rep(NA,length(current_names))
for (iter in seq(1,length(current_names))) {
  name = current_names[iter]
  if (name == "gene_name" | name == "description") {
    corrected_names[iter]=name
  }
  else {
    corrected_names[iter]=desired[name]
  }
}
names(new_counts)=corrected_names
```

Save file:

```
write.table(new_counts,file=output_fname,row.names = F,sep="\t", quote=F)
```

Wrote a new file: `./results/muday-144-SL5_counts-salmon.txt`.

Summarize Muday lab sample name revisions:

```
num_changed = sum(corrected_names!=current_names)
```

There were 48 sample names that changed using the Muday lab key.

To make it easier to review this work, let's also write out a fresh sample mapping file that an analyst can use to check that our sample re-mapping scheme worked properly.

```
df = data.frame("original"=current_names,"new"=colnames(new_counts))
to_remove = which(df[,1]==df[,2] & df[,1]%in%c("gene_name","description"))
df=df[-to_remove,]
same=(df$original==df$new)
notsame=!same
df$changed=notsame
row.names(df)=1:nrow(df)
fname = "results/sample_renaming_summary.txt"
write.table(df,file=fname,row.names = F,sep="\t", quote=F)
```

Wrote a file named results/sample_renaming_summary.txt that records sample renaming. An analyst can use this to check that the sample renaming done using the Muday lab key was successful.

Discussion

Sanity checks passed, in which we confirmed expectations about the experiment: (1) the counts data file contained measurements for every annotated gene and (2) the number of sample columns we observed matched the expected number.

We created a new file with the same data as the original counts data file, but with gene descriptions and with sample codes used in place of column names assigned by the nf-core rnaseq pipeline, which used fastq file name prefixes as column names, as specified in `muday-144_samples.csv`, the nf-core configuration file we used in running the pipeline.

Next, we used a key provided by the Muday lab to change sample names and thus correct the sample mislabeling error that happened at the sequencing facility.

The new file was written out. Also, we wrote a second file (`results/sample_renaming_summary.txt`) which records how this Markdown renamed samples. An analysis can look at this to double-check our work.

Conclusion

The counts file was created and is suitable for use in subsequent data exploration and analysis steps. The new counts file is saved as: `./results/muday-144-SL5_counts-salmon.txt`.

Validation / sanity-checking file for reviewing sample renaming is saved as: `results/sample_renaming_summary.txt`.