RNA-seq Exercises (Airway data)

Exercise 1

If we look at our metadata, we see that the control samples are SRR1039508, SRR1039512, SRR1039516, and SRR1039520. This bit of code will take the rawcounts data, `mutate()` it to add a column called `controlmean`, then `select()` only the gene name and this newly created column, and assigning the result to a new object called `meancounts`. (*Hint*: `rawcounts %>% mutate(...) %>% select(...)`)

```
meancounts <- rawcounts %>%
  mutate(controlmean = (SRR1039508+SRR1039512+SRR1039516+SRR1039520)/4) %>%
  select(ensgene, controlmean)
meancounts
```

```
## # A tibble: 64,102 × 2
##            ensgene controlmean
##              <chr>       <dbl>
## 1  ENSG00000000003      865.00
## 2  ENSG00000000005        0.00
## 3  ENSG00000000419      523.00
## 4  ENSG00000000457      250.25
## 5  ENSG00000000460       63.50
## 6  ENSG00000000938        0.75
## 7  ENSG00000000971     5331.25
## 8  ENSG00000001036     1487.25
## 9  ENSG00000001084      657.50
## 10 ENSG00000001167      469.00
## # ... with 64,092 more rows
```

1. Build off of this code, `mutate()` it once more (prior to the `select()`) function, to add another column called `treatedmean` that takes the mean of the expression values of the treated samples. Then `select()` only the `ensgene`, `controlmean` and `treatedmean` columns, assigning it to a new object called meancounts.

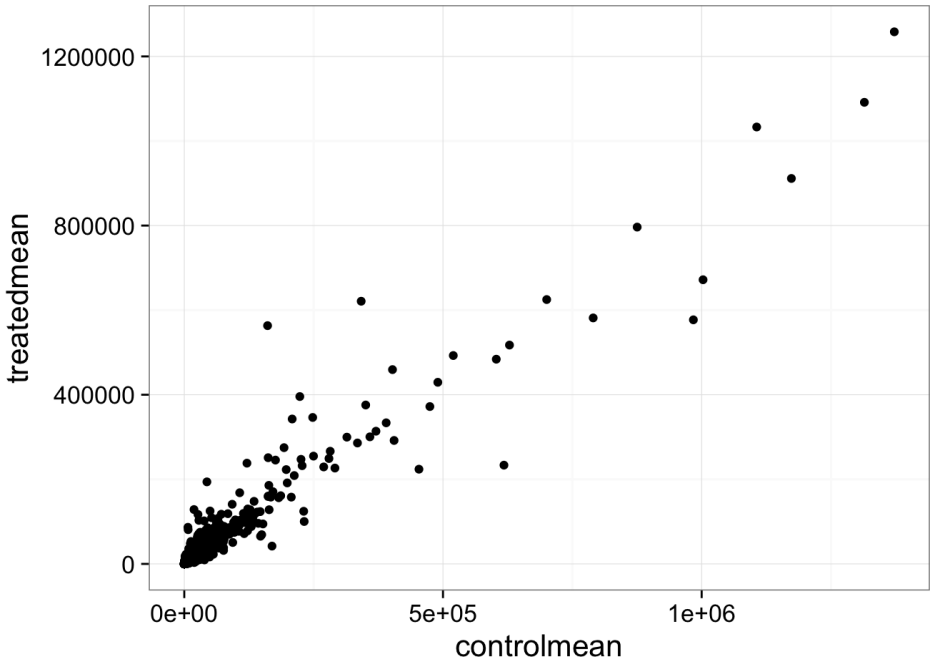
```
## # A tibble: 64,102 × 3
##            ensgene controlmean treatedmean
##              <chr>       <dbl>       <dbl>
## 1  ENSG00000000003      865.00      618.75
## 2  ENSG00000000005        0.00        0.00
## 3  ENSG00000000419      523.00      546.75
## 4  ENSG00000000457      250.25      233.75
## 5  ENSG00000000460       63.50       53.25
## 6  ENSG00000000938        0.75        0.00
## 7  ENSG00000000971     5331.25     6738.25
## 8  ENSG00000001036     1487.25     1122.75
## 9  ENSG00000001084      657.50      572.75
## 10 ENSG00000001167      469.00      316.00
## # ... with 64,092 more rows
```

2. Directly comparing the raw counts is going to be problematic if we just happened to sequence one group at a higher depth than another. Later on we'll do this analysis properly, normalizing by sequencing depth per sample using a better approach. But for now, `summarize()` the data to show the `sum` of the mean counts across all genes for each group. Your answer should look like this:

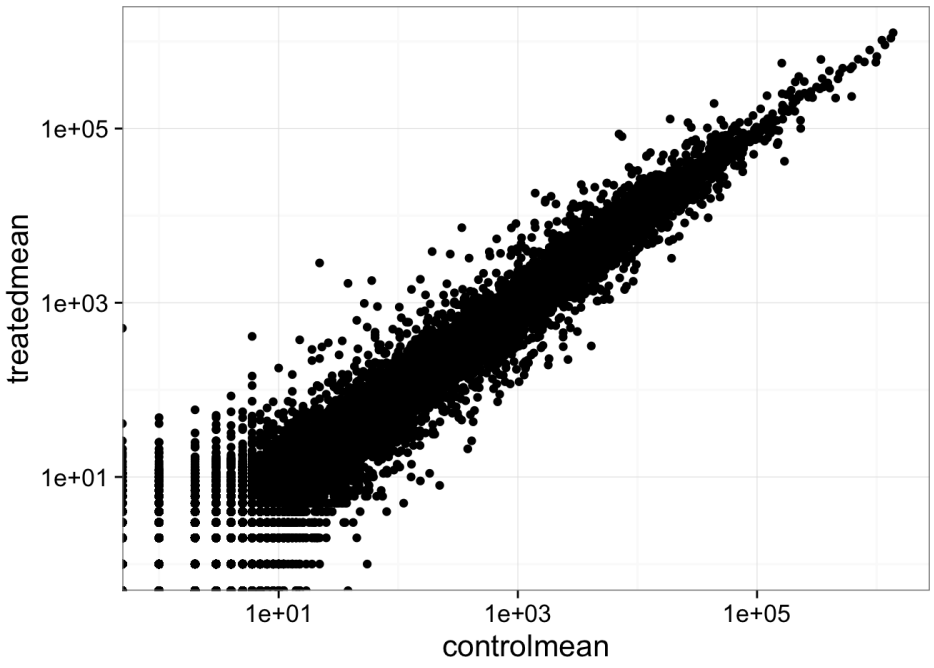
```
## # A tibble: 1 × 2
##   `sum(controlmean)` `sum(treatedmean)`
##                <dbl>              <dbl>
## 1           22390295           21488811
```

Exercise 2

1. Create a scatter plot showing the mean of the treated samples against the mean of the control samples.



2. Wait a sec. There are 60,000-some rows in this data, but I'm only seeing a few dozen dots at most outside of the big clump around the origin. Try plotting both axes on a log scale (*hint*: `... + scale_..._log10()`)



Exercise 3

Look up help on `?inner_join` or Google around for help for using **dplyr**'s `inner_join()` to join two tables by a common column/key. You downloaded annotables_grch37.csv from the data downloads page on bioconnector.org. Load this data with `read_csv()` into an object called `anno`. Pipe it to `View()` or click on the object in the Environment pane to view the entire dataset. This table links the unambiguous Ensembl gene ID to things like the gene symbol, full gene name, location, Entrez gene ID, etc.

```
anno <- read_csv("data/annotables_grch37.csv")
anno
```

```
## # A tibble: 67,416 × 9
##            ensgene entrez   symbol   chr     start       end strand
##              <chr>  <int>    <chr> <chr>     <int>     <int>  <int>
## 1  ENSG00000000003   7105   TSPAN6     X  99883667  99894988     -1
## 2  ENSG00000000005  64102     TNMD     X  99839799  99854882      1
## 3  ENSG00000000419   8813     DPM1    20  49551404  49575092     -1
## 4  ENSG00000000457  57147    SCYL3     1 169818772 169863408     -1
## 5  ENSG00000000460  55732 C1orf112     1 169631245 169823221      1
## 6  ENSG00000000938   2268      FGR     1  27938575  27961788     -1
## 7  ENSG00000000971   3075      CFH     1 196621008 196716634      1
## 8  ENSG00000001036   2519    FUCA2     6 143815948 143832827     -1
## 9  ENSG00000001084   2729     GCLC     6  53362139  53481768     -1
## 10 ENSG00000001167   4800     NFYA     6  41040684  41067715      1
## # ... with 67,406 more rows, and 2 more variables: biotype <chr>,
## #   description <chr>
```

1. Take our newly created `meancounts` object, and `arrange()` it `desc`ending by the absolute value (`abs()`) of the `log2fc` column. The first few rows should look like this:

```
## # A tibble: 27,450 × 4
##            ensgene controlmean treatedmean    log2fc
##              <chr>       <dbl>       <dbl>     <dbl>
## 1  ENSG00000109906        5.50      715.50  7.023376
## 2  ENSG00000250978        1.50      102.75  6.098032
## 3  ENSG00000128285       13.75        0.25 -5.781360
...
```

2. Continue on that pipeline, and `inner_join()` it to the `anno` data by the `ensgene` column. Either assign it to a temporary object or pipe the whole thing to `View` to take a look. What do you notice? Would you trust these results? Why or why not?

```
## # A tibble: 29,034 × 12
##            ensgene controlmean treatedmean    log2fc    entrez
##              <chr>       <dbl>       <dbl>     <dbl>     <int>
## 1  ENSG00000109906        5.50      715.50  7.023376      7704
## 2  ENSG00000250978        1.50      102.75  6.098032        NA
## 3  ENSG00000128285       13.75        0.25 -5.781360      2847
## 4  ENSG00000260802        0.25       12.00  5.584963    401613
## 5  ENSG00000171819        9.50      417.50  5.457705     10218
## 6  ENSG00000137673        0.25       10.25  5.357552      4316
## 7  ENSG00000127954       15.00      449.25  4.904484     79689
## 8  ENSG00000249364        0.50       14.75  4.882643 101928858
## 9  ENSG00000267339       55.50        2.00 -4.794416    148145
## 10 ENSG00000100033        3.75       93.75  4.643856      5625
## # ... with 29,024 more rows, and 7 more variables: symbol <chr>,
## #   chr <chr>, start <int>, end <int>, strand <int>, biotype <chr>,
## #   description <chr>
```

Exercise 4

1. Using a `%>%`, `arrange` the results by the adjusted p-value.

```
## # A tibble: 64,102 × 7
##                row   baseMean log2FoldChange      lfcSE      stat
##              <chr>      <dbl>          <dbl>      <dbl>     <dbl>
## 1  ENSG00000152583   997.4398       4.280694 0.19572061  21.87145
## 2  ENSG00000148175 11193.7188       1.434429 0.08325248  17.22987
## 3  ENSG00000179094   776.5967       2.981009 0.18833478  15.82825
## 4  ENSG00000109906   385.0710       5.095376 0.32987788  15.44625
## 5  ENSG00000134686  2737.9820       1.368175 0.08974798  15.24463
## 6  ENSG00000125148  3656.2528       2.126258 0.14207457  14.96579
## 7  ENSG00000120129  3409.0294       2.760597 0.18885833  14.61729
## 8  ENSG00000189221  2341.7673       3.039185 0.20995474  14.47543
## 9  ENSG00000178695  2649.8501      -2.372770 0.16979309 -13.97448
## 10 ENSG00000101347 12703.3871       3.406507 0.24761309  13.75738
## # ... with 64,092 more rows, and 2 more variables: pvalue <dbl>,
## #   padj <dbl>
```

2. Continue piping to `inner_join()`, joining the results to the `anno` object. See the help for `?inner_join`, specifically the `by=` argument. You'll have to do something like `... %>% inner_join(anno, by=c("row"="ensgene"))`. Once you're happy with this result, reassign the result back to `res`. It'll look like this.

```
##                row   baseMean log2FoldChange      lfcSE     stat
## 1 ENSG00000152583   997.4398       4.280694 0.19572061 21.87145
## 2 ENSG00000148175 11193.7188       1.434429 0.08325248 17.22987
## 3 ENSG00000179094   776.5967       2.981009 0.18833478 15.82825
## 4 ENSG00000179094   776.5967       2.981009 0.18833478 15.82825
## 5 ENSG00000109906   385.0710       5.095376 0.32987788 15.44625
## 6 ENSG00000134686  2737.9820       1.368175 0.08974798 15.24463
##          pvalue          padj    entrez  symbol chr     start       end
## 1 4.858346e-106 9.020004e-102      8404 SPARCL1   4  88394487  88452213
## 2  1.585139e-66  1.471484e-62      2040    STOM   9 124101355 124132531
## 3  1.986835e-56  1.229586e-52 102465532    PER1  17   8043790   8059824
## 4  1.986835e-56  1.229586e-52      5187    PER1  17   8043790   8059824
## 5  7.996137e-54  3.711407e-50      7704  ZBTB16  11 113930315 114121398
## 6  1.787492e-52  6.637314e-49      1912    PHC2   1  33789224  33896653
##   strand        biotype
## 1     -1 protein_coding
## 2     -1 protein_coding
## 3     -1 protein_coding
## 4     -1 protein_coding
## 5      1 protein_coding
## 6     -1 protein_coding
##                                                                 description
## 1                    SPARC-like 1 (hevin) [Source:HGNC Symbol;Acc:11220]
## 2                                 stomatin [Source:HGNC Symbol;Acc:3383]
## 3                period circadian clock 1 [Source:HGNC Symbol;Acc:8845]
## 4                period circadian clock 1 [Source:HGNC Symbol;Acc:8845]
## 5 zinc finger and BTB domain containing 16 [Source:HGNC Symbol;Acc:12930]
## 6       polyhomeotic homolog 2 (Drosophila) [Source:HGNC Symbol;Acc:3183]
```

3. How many are significant with an adjusted p-value <0.05? (Pipe to `filter()`).

```
## # A tibble: 2,851 × 15
... ... ... ... ... ...
```

Exercise 5

Look up the Wikipedia articles on MA plots and volcano plots. An MA plot shows the average expression on the X-axis and the log fold change on the y-axis. A volcano plot shows the log fold change on the X-axis, and the $-log_{10}$ of the p-value on the Y-axis (the more significant the p-value, the larger the $-log_{10}$ of that value will be).

1. Make an MA plot. Use a log_{10}-scaled x-axis, color-code by whether the gene is significant, and give your plot a title. It should look like this. What's the deal with the gray points?
2. Make a volcano plot. Similarly, color-code by whether it's significant or not.