BIMS 8382 Exercises

bioconnector.org/bims8382

Spring 2016

Contents

R Basics

Exercise 1

What are the values after each statement in the following?

```
mass <- 50              # mass?
age  <- 30              # age?
mass <- mass * 2        # mass?
age  <- age - 10        # age?
mass_index <- mass/age  # massIndex?
```

Exercise 2

See `?abs` and calculate the square root of the log-base-10 of the absolute value of `-4*(2550-50)`. Answer should be `2`.

Data Frames

1. What's the standard deviation expression (hint: get help on the `sd` function with `?sd`).
2. What's the range of rate represented in the data? (hint: `range()`).

Advanced Data Manipulation

Exercise 1

1. Display the data where the gene ontology biological process (the `bp` variable) is "leucine biosynthesis" (case-sensitive) *and* the limiting nutrient was Leucine. (Answer should return a 24-by-7 data frame – 4 genes × 6 growth rates).
2. Gene/rate combinations had high expression (in the top 1% of expressed genes)? *Hint:* see `?quantile` and try `quantile(ydat$expression, probs=.99)` to see the expression value which is higher than 99% of all the data, then `filter()` based on that. Try wrapping your answer with a `View()` function so you can see the whole thing. What does it look like those genes are doing? Answer should return a 1971-by-7 data frame.

Exercise 2

1. First, re-run the command you used above to filter the data for genes involved in the "leucine biosynthesis" biological process *and* where the limiting nutrient is Leucine.
2. Wrap this entire filtered result with a call to `arrange()` where you'll arrange the result of #1 by the gene symbol.
3. Wrap this entire result in a `View()` statement so you can see the entire result.

Exercise 3

Here's a warm-up round. Try the following.

Show the limiting nutrient and expression values for the gene ADH2 when the growth rate is restricted to 0.05. *Hint:* 2 pipes: `filter` and `select`.

```
## Source: local data frame [6 x 2]
##
##    nutrient expression
##       (chr)      (dbl)
## 1   Glucose       6.28
## 2   Ammonia       0.55
## 3 Phosphate      -4.60
## 4   Sulfate      -1.18
## 5   Leucine       4.15
## 6    Uracil       0.63
```

What are the four most highly expressed genes when the growth rate is restricted to 0.05 by restricting glucose? Show only the symbol, expression value, and GO terms. *Hint:* 4 pipes: `filter`, `arrange`, `head`, and `select`.

```
## Source: local data frame [4 x 4]
##
##   symbol expression                 bp                             mf
##    (chr)      (dbl)              (chr)                          (chr)
## 1   ADH2       6.28       fermentation* alcohol dehydrogenase activity
## 2  HSP26       5.86 response to stress*       unfolded protein binding
## 3   MLS1       5.64    glyoxylate cycle       malate synthase activity
## 4   HXT5       5.56    hexose transport  glucose transporter activity*
```

When the growth rate is restricted to 0.05, what is the average expression level across all genes in the "response to stress" biological process, separately for each limiting nutrient? What about genes in the "protein biosynthesis" biological process? *Hint:* 3 pipes: `filter`, `group_by`, `summarize`.

```
## Source: local data frame [6 x 2]
##
##    nutrient   meanexp
##       (chr)     (dbl)
## 1   Ammonia 0.9426667
## 2   Glucose 0.7426667
## 3   Leucine 0.8106667
## 4 Phosphate 0.9806667
## 5   Sulfate 0.7430769
## 6    Uracil 0.7313333
```

```
## Source: local data frame [6 x 2]
##
##    nutrient    meanexp
##       (chr)      (dbl)
## 1   Ammonia -1.6133514
## 2   Glucose -0.6911351
## 3   Leucine -0.5735676
## 4 Phosphate -0.7496216
## 5   Sulfate -0.9134807
## 6    Uracil -0.8799454
```

Exercise 4

That was easy, right? How about some tougher ones.

First, some review. How do we see the number of distinct values of a variable? Use `n_distinct()` within a `summarize()` call.

```
ydat %>% summarize(n_distinct(mf))
```

```
## Source: local data frame [1 x 1]
##
##   n_distinct(mf)
##            (int)
## 1           1086
```

Which 10 biological process annotations have the most genes associated with them? What about molecular functions? *Hint:* 4 pipes: `group_by`, `summarize` with `n_distinct`, `arrange`, `head`.

```
## Source: local data frame [10 x 2]
##
##                                                          bp     n
##                                                       (chr) (int)
## 1                                  biological process unknown   269
## 2                                        protein biosynthesis   182
## 3                           protein amino acid phosphorylation*    78
## 4                                       protein biosynthesis*    73
## 5                         cell wall organization and biogenesis*    64
## 6  regulation of transcription from RNA polymerase II promoter*    49
## 7                         nuclear mRNA splicing, via spliceosome    47
## 8                                                  DNA repair*    44
## 9                                    aerobic respiration*    42
## 10                                 ER to Golgi transport*    42
```

```
## Source: local data frame [10 x 2]
##
##                                     mf     n
##                                  (chr) (int)
## 1              molecular function unknown   886
## 2      structural constituent of ribosome   185
## 3                          protein binding   107
## 4                              RNA binding    63
## 5                         protein binding*    53
## 6                             DNA binding*    44
## 7            structural molecule activity    43
## 8                           GTPase activity    40
## 9  structural constituent of cytoskeleton    39
## 10          transcription factor activity    38
```

How many distinct genes are there where we know what process the gene is involved in but we don't know what it does? *Hint:* 3 pipes; `filter` where `bp!="biological process unknown" & mf=="molecular function unknown"`, and after `select`ing columns of interest, pipe the output to `distinct()`. The answer should be **737**, and here are a few:

```
## Source: local data frame [737 x 3]
##
##    symbol                                                                bp
##     (chr)                                                             (chr)
## 1    SFB2                                              ER to Golgi transport
## 2    EDC3                          deadenylylation-independent decapping
## 3    PER1                                  response to unfolded protein*
## 4   PEX25                       peroxisome organization and biogenesis*
## 5    BNI5                                                    cytokinesis*
## 6   CSN12 adaptation to pheromone during conjugation with cellular fusion
## 7   SEC39                                                 secretory pathway
## 8    ABC1                                           ubiquinone biosynthesis
## 9   PRP46                          nuclear mRNA splicing, via spliceosome
## 10   MAM3                      mitochondrion organization and biogenesis*
## ..    ...                                                               ...
## Variables not shown: mf (chr)
```

When the growth rate is restricted to 0.05 by limiting Glucose, which biological processes are the most upregulated? Show a sorted list with the most upregulated BPs on top, displaying the biological process and the average expression of all genes in that process rounded to two digits. *Hint:* 5 pipes: `filter`, `group_by`, `summarize`, `mutate`, `arrange`.

```
## Source: local data frame [881 x 2]
##
##                                          bp meanexp
##                                       (chr)   (dbl)
## 1                             fermentation*    6.28
## 2                          glyoxylate cycle    5.29
## 3  oxygen and reactive oxygen species metabolism    5.04
## 4                       fumarate transport*    5.03
## 5                  acetyl-CoA biosynthesis*    4.32
## 6                           gluconeogenesis    3.64
## 7                 fatty acid beta-oxidation    3.57
## 8                         lactate transport    3.48
## 9                     carnitine metabolism    3.30
## 10                     alcohol metabolism*    3.25
## ..                                     ...     ...
```

Group the data by limiting nutrient (primarily) then by biological process. Get the average expression for all genes annotated with each process, separately for each limiting nutrient, where the growth rate is restricted to 0.05. Arrange the result to show the most upregulated processes on top. The initial result will look like the result below. Pipe this output to a `View()` statement. What's going on? Why didn't the `arrange()` work? *Hint:* 5 pipes: `filter`, `group_by`, `summarize`, `arrange`, `View`.

```
## Source: local data frame [5,257 x 3]
## Groups: nutrient [6]
##
##    nutrient                    bp meanexp
##       (chr)                 (chr)   (dbl)
## 1   Ammonia   allantoate transport  6.6400
## 2   Ammonia  amino acid transport*  6.6400
## 3   Ammonia    allantoin transport  5.5600
## 4   Ammonia    proline catabolism*  5.1400
## 5   Ammonia         urea transport  5.1400
## 6   Ammonia asparagine catabolism*  4.7325
## 7   Ammonia  allantoin catabolism*  4.4400
## 8   Ammonia      peptide transport  3.9200
## 9   Ammonia       glyoxylate cycle  3.9100
## 10  Ammonia   sodium ion transport  3.2650
## ..      ...                   ...     ...
```

Let's try to further process that result to get only the top three most upregulated biolgocal processes for each limiting nutrient. Google search "dplyr first result within group." You'll need a `filter(row_number()......)` in there somewhere. *Hint:* 5 pipes: `filter`, `group_by`, `summarize`, `arrange`, `filter(row_number()....` *Note:* dplyr's pipe syntax used to be `%.%` before it changed to `%>%`. So when looking around, you might still see some people use the old syntax. Now if you try to use the old syntax, you'll get a deprecation warning.

```
## Source: local data frame [18 x 3]
## Groups: nutrient [6]
##
##     nutrient                                          bp meanexp
##        (chr)                                       (chr)   (dbl)
## 1    Ammonia                 allantoate transport   6.640
## 2    Ammonia                amino acid transport*   6.640
## 3    Ammonia                  allantoin transport   5.560
## 4    Glucose                       fermentation*   6.280
## 5    Glucose                   glyoxylate cycle   5.285
## 6    Glucose oxygen and reactive oxygen species metabolism   5.040
## 7    Leucine                       fermentation*   4.150
## 8    Leucine                  fumarate transport*   3.720
## 9    Leucine                     glyoxylate cycle   3.650
## 10 Phosphate          glycerophosphodiester transport   6.640
## 11 Phosphate           vacuole fusion, non-autophagic   4.195
## 12 Phosphate   regulation of cell redox homeostasis*   4.030
## 13   Sulfate                 protein ubiquitination   3.400
## 14   Sulfate                  fumarate transport*   3.270
## 15   Sulfate          sulfur amino acid metabolism*   2.690
```

```
## 16    Uracil                          fumarate transport*   4.320
## 17    Uracil                       pyridoxine metabolism   3.110
## 18    Uracil                     asparagine catabolism*   3.060
```

There's a slight problem with the examples above. We're getting the average expression of all the biological processes separately by each nutrient. But some of these biological processes only have a single gene in them! If we tried to do the same thing to get the correlation between rate and expression, the calculation would work, but we'd get a warning about a standard deviation being zero. The correlation coefficient value that results is `NA`, i.e., missing. While we're summarizing the correlation between rate and expression, let's also show the number of distinct genes within each grouping.

```
ydat %>%
  group_by(nutrient, bp) %>%
  summarize(r=cor(rate, expression), ngenes=n_distinct(symbol))
```

```
## Warning in cor(c(0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05,
## 0.05, : the standard deviation is zero
```

```
## Source: local data frame [5,286 x 4]
## Groups: nutrient [?]
##
##    nutrient                                       bp           r ngenes
##       (chr)                                    (chr)       (dbl)  (int)
## 1   Ammonia             'de novo' IMP biosynthesis*  0.31247162      8
## 2   Ammonia  'de novo' pyrimidine base biosynthesis -0.04817745      3
## 3   Ammonia 'de novo' pyrimidine base biosynthesis*  0.16699596      4
## 4   Ammonia       35S primary transcript processing  0.50795855     13
## 5   Ammonia      35S primary transcript processing*  0.42397321     30
## 6   Ammonia                    acetate biosynthesis  0.46768319      1
## 7   Ammonia                      acetate metabolism  0.92909260      1
## 8   Ammonia                     acetate metabolism* -0.68551933      1
## 9   Ammonia                 acetyl-CoA biosynthesis -0.85122895      1
## 10  Ammonia   acetyl-CoA biosynthesis from pyruvate  0.09509414      1
## ..      ...                                     ...         ...    ...
```

Take the above code and continue to process the result to show only results where the process has at least 5 genes. Add a column corresponding to the absolute value of the correlation coefficient, and show for each nutrient the singular process with the highest correlation between rate and expression, regardless of direction. *Hint:* 4 more pipes: `filter`, `mutate`, `arrange`, and `filter` again with `row_number()==1`. Ignore the warning.

```
## Source: local data frame [6 x 5]
```

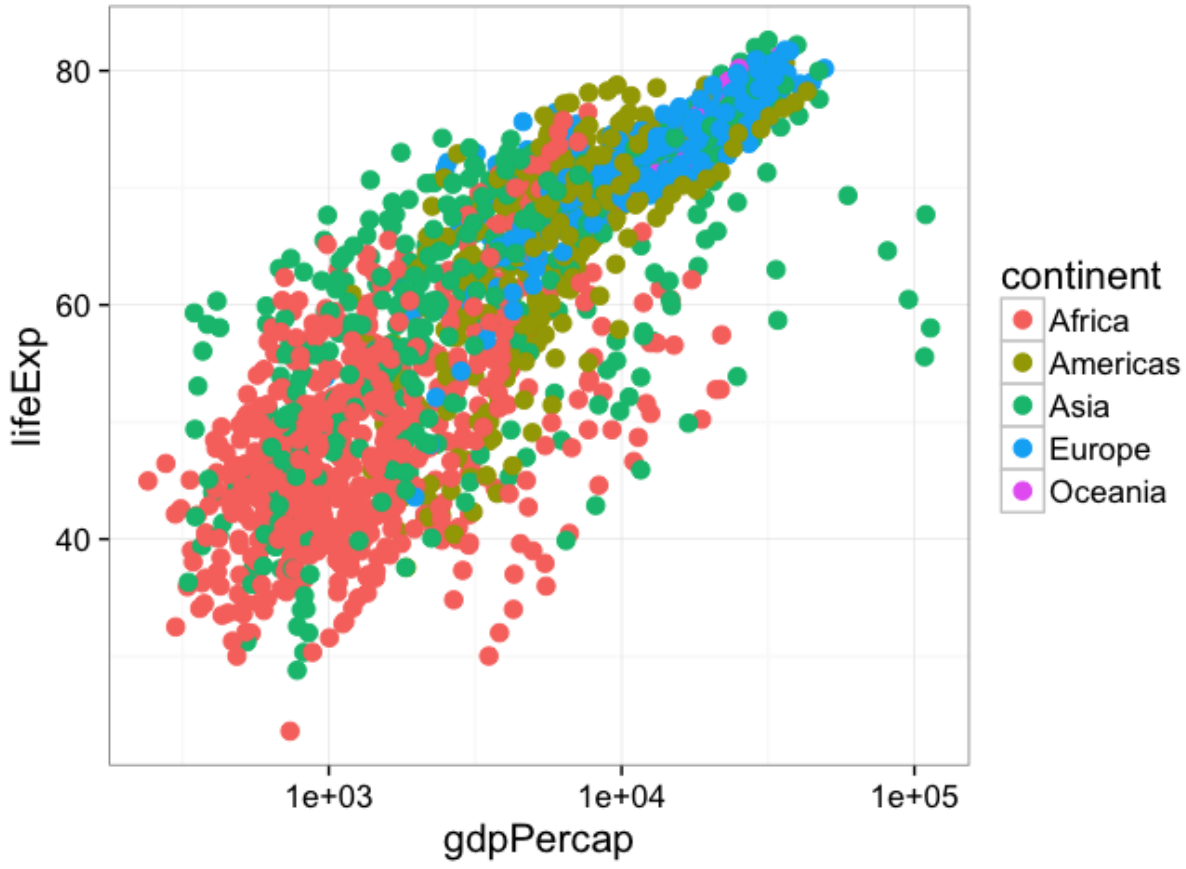
```
## Groups: nutrient [6]
##
##    nutrient                                          bp     r ngenes  absr
##       (chr)                                       (chr) (dbl)  (int) (dbl)
## 1   Ammonia telomerase-independent telomere maintenance -0.91      7  0.91
## 2   Glucose telomerase-independent telomere maintenance -0.95      7  0.95
## 3   Leucine telomerase-independent telomere maintenance -0.90      7  0.90
## 4 Phosphate telomerase-independent telomere maintenance -0.90      7  0.90
## 5   Sulfate                   translational elongation*  0.79      5  0.79
## 6    Uracil telomerase-independent telomere maintenance -0.81      7  0.81
```

Data Visualization

Exercise 1

Re-create this same plot from scratch without saving anything to a variable. That is, start from the `ggplot` call.

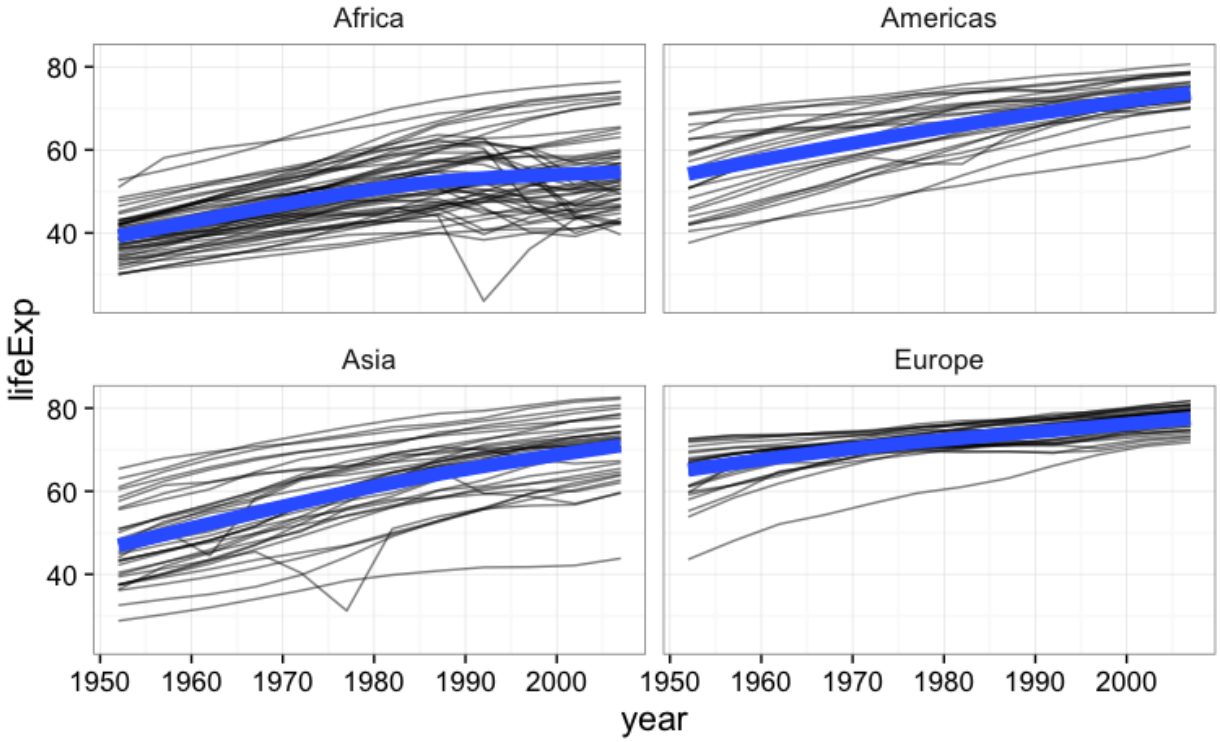
- Start with the `ggplot()` function.
- Use the gm data.
- Map `gdpPercap` to the x-axis and `lifeExp` to the y-axis.
- Add points to the plot
- Make the points size 3
- Map continent onto the aesthetics of the point
- Use a log10 scale for the x-axis.



Exercise 2

1. Make a scatter plot of `lifeExp` on the y-axis against `year` on the x.

2. Make a series of small multiples faceting on continent.
3. Add a fitted curve, smooth or lm, with and without facets.
4. **Bonus**: using `geom_line()` and and aesthetic mapping `country` to `group=`, make a "spaghetti plot", showing *semitransparent* lines connected for each country, faceted by continent. Add a smoothed loess curve with a thick (`lwd=3`) line with no standard error stripe. Reduce the opacity (`alpha=`) of the individual black lines. *Don't* show Oceania countries (that is, `filter()` the data where `continent!="Oceania"` before you plot it).



Exercise 3

1. Make a jittered strip plot of GDP per capita against continent.
2. Make a box plot of GDP per capita against continent.
3. Using a log10 y-axis scale, overlay semitransparent jittered points on top of box plots, where outlying points are colored.
4. **BONUS**: Try to reorder the continents on the x-axis by GDP per capita. Why isn't this working as expected? See `?reorder` for clues.

Exercise 4

1. Plot a histogram of GDP Per Capita.

2. Do the same but use a log10 x-axis.
3. Still on the log10 x-axis scale, try a density plot mapping continent to the fill of each density distribution, and reduce the opacity.
4. Still on the log10 x-axis scale, make a histogram faceted by continent *and* filled by continent. Facet with a single column (see `?facet_wrap` for help).
5. Save this figure to a 6x10 PDF file.

RNA-seq

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`.

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

```
## Source: local data frame [64,102 x 2]
##
##            ensgene controlmean
##              (chr)       (int)
## 1  ENSG00000000003        3460
## 2  ENSG00000000005           0
## 3  ENSG00000000419        2092
## 4  ENSG00000000457        1001
## 5  ENSG00000000460         254
## 6  ENSG00000000938           3
## 7  ENSG00000000971       21325
## 8  ENSG00000001036        5949
## 9  ENSG00000001084        2630
## 10 ENSG00000001167        1876
## ..             ...         ...
```

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. It should look like this.

```
## Source: local data frame [64,102 x 3]
##
##            ensgene controlmean treatedmean
##              (chr)       (int)       (int)
## 1  ENSG00000000003        3460        2475
## 2  ENSG00000000005           0           0
## 3  ENSG00000000419        2092        2187
## 4  ENSG00000000457        1001         935
```

```
## 5  ENSG00000000460         254         213
## 6  ENSG00000000938           3           0
## 7  ENSG00000000971       21325       26953
## 8  ENSG00000001036        5949        4491
## 9  ENSG00000001084        2630        2291
## 10 ENSG00000001167        1876        1264
## ..             ...         ...         ...
```

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. 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:

```
## Source: local data frame [1 x 2]
##
##   sum(controlmean) sum(treatedmean)
##              (int)            (int)
## 1         89561179         85955244
```

Exercise 2

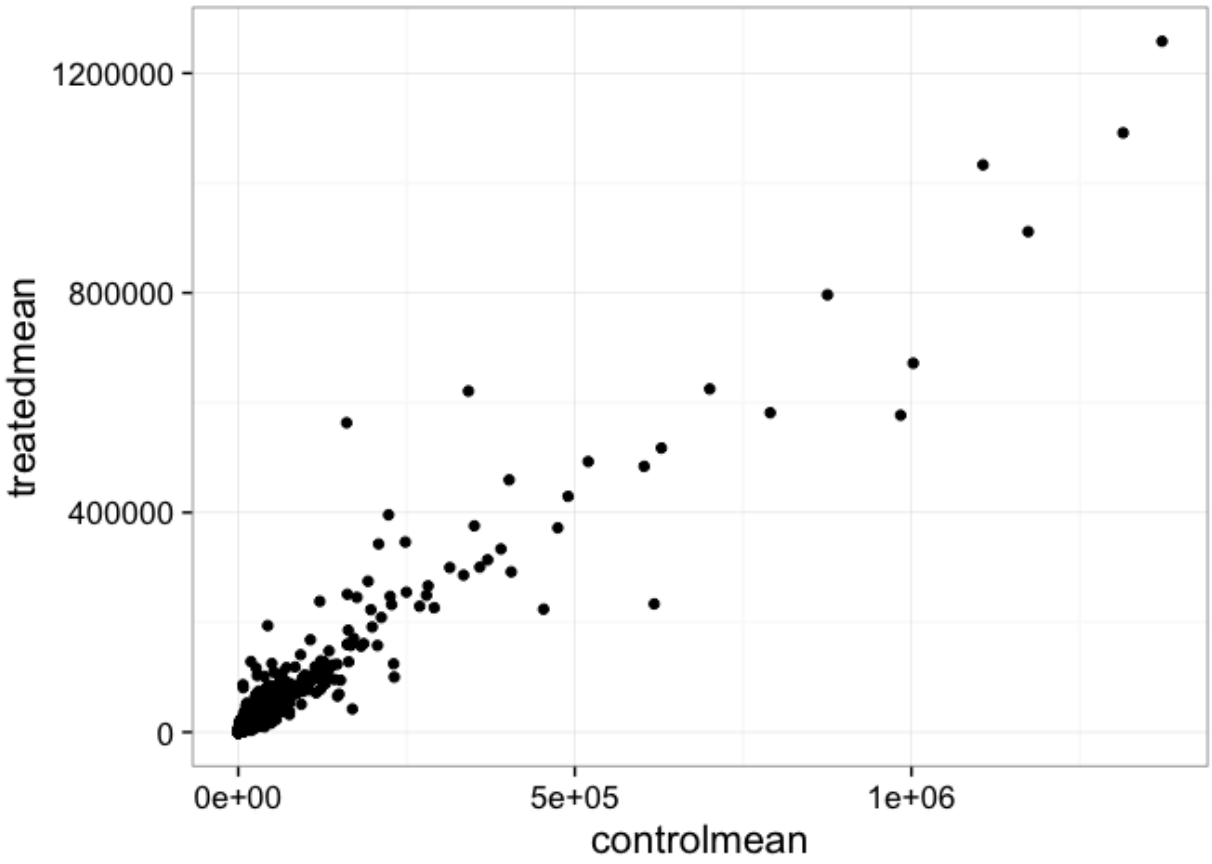
1. Create a scatter plot showing the mean of the treated samples against the mean of the control samples.

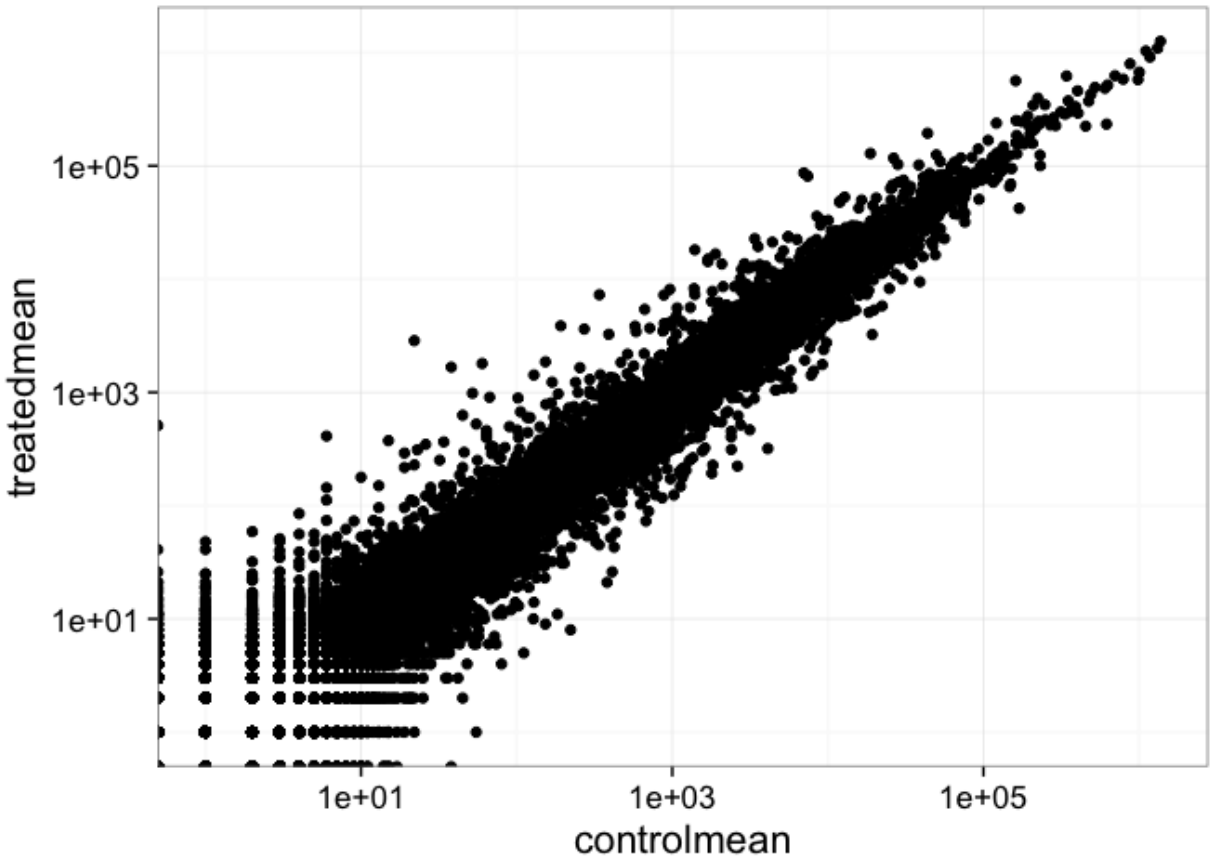
2. Wait a sec. There are 60,000-some rows in this data, but I'm only seeing a few dozen dots at most outside of the big clump around the origin. Try plotting both axes on a log scale (*hint*: `... + scale_..._log10()`)

Exercise 3

Go back and refresh your memory on using `inner_join()` to join two tables by a common column/key. You previously downloaded annotables_grch37.csv from bioconnector.org/data. 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
```


1200000
800000
400000
0
treatedmean
0e+00
5e+05
1e+06
controlmean


1e+05
1e+03
1e+01
treatedmean
1e+01
1e+03
1e+05
controlmean

```
## Source: local data frame [67,416 x 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
## ..             ...    ...      ...   ...       ...       ...    ...
## Variables not shown: 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 results should look like this:

```
## Source: local data frame [27,450 x 4]
##
##            ensgene controlmean treatedmean    log2fc
##              (chr)       (int)       (int)     (dbl)
## 1  ENSG00000109906          22        2862  7.023376
## 2  ENSG00000250978           6         411  6.098032
## 3  ENSG00000128285          55           1 -5.781360
## 4  ENSG00000260802           1          48  5.584963
## 5  ENSG00000171819          38        1670  5.457705
## 6  ENSG00000137673           1          41  5.357552
## 7  ENSG00000127954          60        1797  4.904484
## 8  ENSG00000249364           2          59  4.882643
## 9  ENSG00000267339         222           8 -4.794416
## 10 ENSG00000100033          15         375  4.643856
## ..             ...         ...         ...       ...
```

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?

```
## Source: local data frame [29,034 x 12]
##
##            ensgene controlmean treatedmean    log2fc    entrez
##              (chr)       (int)       (int)     (dbl)     (int)
```

```
## 1  ENSG00000109906          22        2862  7.023376      7704
## 2  ENSG00000250978           6         411  6.098032        NA
## 3  ENSG00000128285          55           1 -5.781360      2847
## 4  ENSG00000260802           1          48  5.584963    401613
## 5  ENSG00000171819          38        1670  5.457705     10218
## 6  ENSG00000137673           1          41  5.357552      4316
## 7  ENSG00000127954          60        1797  4.904484     79689
## 8  ENSG00000249364           2          59  4.882643 101928858
## 9  ENSG00000267339         222           8 -4.794416    148145
## 10 ENSG00000100033          15         375  4.643856      5625
## ..             ...         ...         ...       ...       ...
## Variables not shown: 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.

```
## Source: local data frame [64,102 x 7]
##
##                row   baseMean log2FoldChange      lfcSE      stat
##              (chr)      (dbl)          (dbl)      (dbl)     (dbl)
## 1  ENSG00000152583   997.4398       4.285847 0.19605831  21.86006
## 2  ENSG00000148175 11193.7188       1.434388 0.08411178  17.05336
## 3  ENSG00000179094   776.5967       2.984244 0.18864120  15.81968
## 4  ENSG00000109906   385.0710       5.137007 0.33077733  15.53010
## 5  ENSG00000134686  2737.9820       1.368176 0.09059205  15.10261
## 6  ENSG00000125148  3656.2528       2.127162 0.14255648  14.92154
## 7  ENSG00000120129  3409.0294       2.763614 0.18915513  14.61030
## 8  ENSG00000189221  2341.7673       3.043757 0.21020671  14.47983
## 9  ENSG00000178695  2649.8501      -2.374629 0.17015595 -13.95560
## 10 ENSG00000101347 12703.3871       3.414854 0.24787488  13.77652
## ..             ...        ...            ...       ...       ...
## Variables not shown: 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       4.285847 0.19605831 21.86006
## 2 ENSG00000148175 11193.7188       1.434388 0.08411178 17.05336
```

```
## 3 ENSG00000179094   776.5967       2.984244 0.18864120 15.81968
## 4 ENSG00000179094   776.5967       2.984244 0.18864120 15.81968
## 5 ENSG00000109906   385.0710       5.137007 0.33077733 15.53010
## 6 ENSG00000134686  2737.9820       1.368176 0.09059205 15.10261
##          pvalue          padj    entrez  symbol chr     start       end
## 1 6.235872e-106 1.119464e-101      8404 SPARCL1   4  88394487  88452213
## 2  3.300017e-65  2.962096e-61      2040    STOM   9 124101355 124132531
## 3  2.276377e-56  1.362184e-52 102465532    PER1  17   8043790   8059824
## 4  2.276377e-56  1.362184e-52      5187    PER1  17   8043790   8059824
## 5  2.170243e-54  9.740052e-51      7704  ZBTB16  11 113930315 114121398
## 6  1.556576e-51  5.588730e-48      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()`).

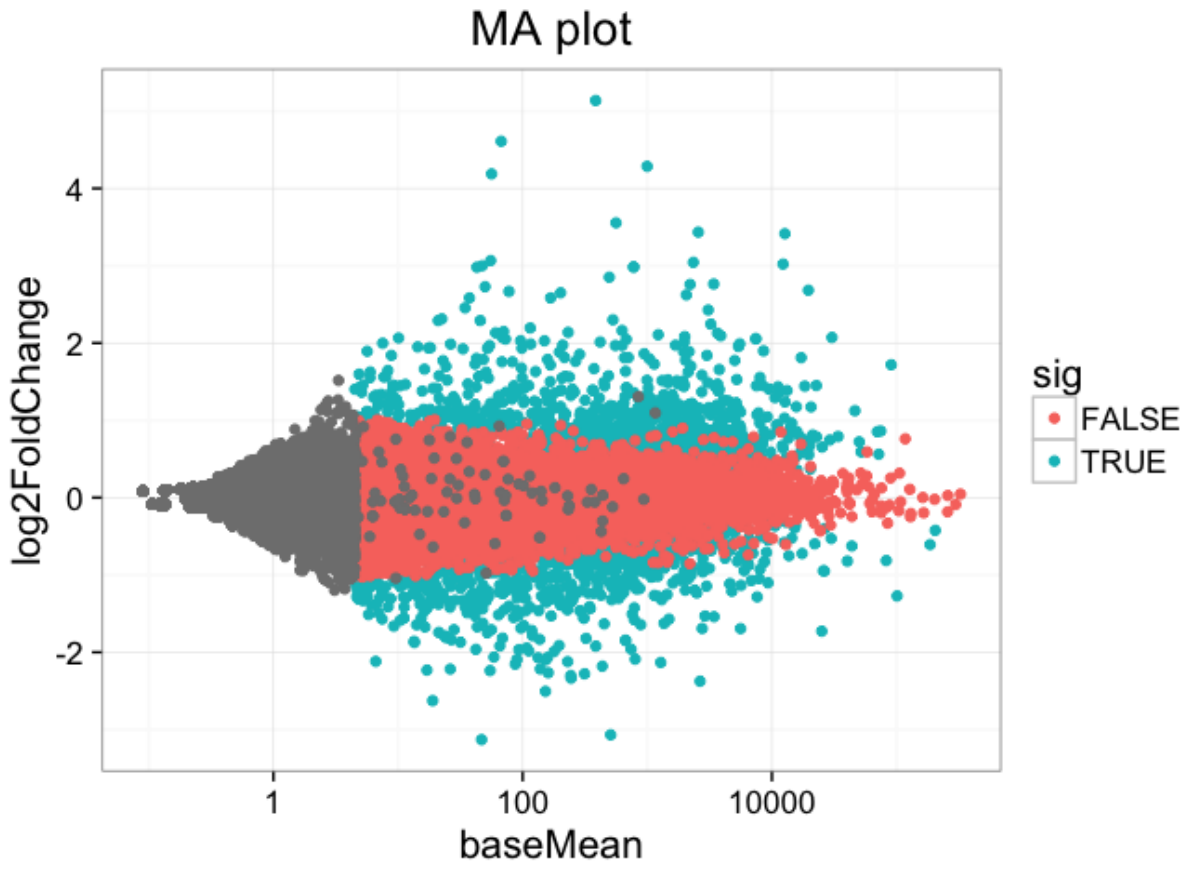
```
## Source: local data frame [2,852 x 15]
##
##                row   baseMean log2FoldChange      lfcSE      stat
##              (chr)      (dbl)          (dbl)      (dbl)     (dbl)
## 1  ENSG00000152583   997.4398       4.285847 0.19605831  21.86006
## 2  ENSG00000148175 11193.7188       1.434388 0.08411178  17.05336
## 3  ENSG00000179094   776.5967       2.984244 0.18864120  15.81968
## 4  ENSG00000179094   776.5967       2.984244 0.18864120  15.81968
## 5  ENSG00000109906   385.0710       5.137007 0.33077733  15.53010
## 6  ENSG00000134686  2737.9820       1.368176 0.09059205  15.10261
## 7  ENSG00000125148  3656.2528       2.127162 0.14255648  14.92154
## 8  ENSG00000120129  3409.0294       2.763614 0.18915513  14.61030
## 9  ENSG00000189221  2341.7673       3.043757 0.21020671  14.47983
## 10 ENSG00000178695  2649.8501      -2.374629 0.17015595 -13.95560
## ..             ...        ...            ...        ...       ...
## Variables not shown: pvalue (dbl), padj (dbl), entrez (int), symbol (chr),
```

```
##    chr (chr), start (int), end (int), strand (int), biotype (chr),
##    description (chr)
```

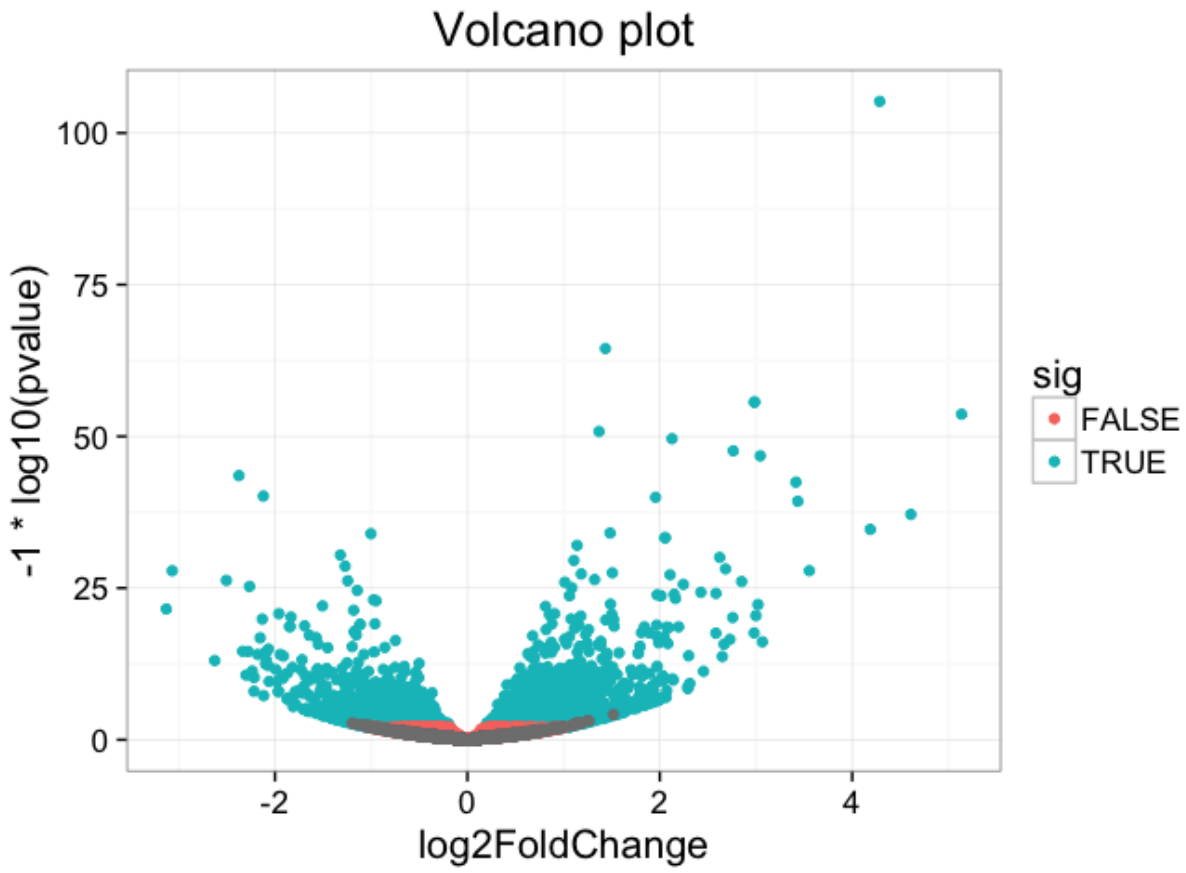
Exercise 5

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.


Volcano plot
-1 * log10(pvalue)
100
75
50
25
0
-2
0
2
4
log2FoldChange
sig
FALSE
TRUE