Chapter 8 examples

Brooke Anderson

4/7/2020

Load some general packages you'll need:

```r
library(tidyverse)
```

Pasilla dataset

Install the `pasilla` package and find the file address of the example data file you'll be using. The `system.file` function looks for a file that was included with a package. If you want to share data using something other than an R binary data file, this is how the data is included in the package. The output (`fn`) will be the filepath to the file "pasilla_gene_counts.tsv", which you downloaded as part of the `pasilla` package.

```r
# Uncomment and run the following line if you need the `pasilla` package
# BiocManager::install(pkgs = "pasilla")
fn <- system.file("extdata", "pasilla_gene_counts.tsv",
                  package = "pasilla", mustWork = TRUE)
fn
```

```
## [1] "/Library/Frameworks/R.framework/Versions/3.6/Resources/library/pasilla/extdata/pasilla_gene_cou
```

Now that you know where the file is, you need to read it in:

```r
counts <- fn %>%
  read_tsv()
counts
```

```
## # A tibble: 14,599 x 8
##    gene_id untreated1 untreated2 untreated3 untreated4 treated1 treated2
##    <chr>        <dbl>      <dbl>      <dbl>      <dbl>    <dbl>    <dbl>
##  1 FBgn00~          0          0          0          0        0        0
##  2 FBgn00~         92        161         76         70      140       88
##  3 FBgn00~          5          1          0          0        4        0
##  4 FBgn00~          0          2          1          2        1        0
##  5 FBgn00~       4664       8714       3564       3150     6205     3072
##  6 FBgn00~        583        761        245        310      722      299
##  7 FBgn00~          0          1          0          0        0        0
##  8 FBgn00~         10         11          3          3       10        7
##  9 FBgn00~          0          1          0          0        0        1
## 10 FBgn00~       1446       1713        615        672     1698      696
## # ... with 14,589 more rows, and 1 more variable: treated3 <dbl>
```

Again, this data is "transposed" compared to what we'll want for a lot of statistical modeling functions. Each column is an observation, while each row is the measures for a specific gene (which one is given in the first column, `gene_id`).

There is a vignette for the `pasilla` package at: https://bioconductor.org/packages/release/data/experimen t/vignettes/pasilla/inst/doc/create_objects.html The paper that the data originally came from is available here.

You can explore the data a bit:

```r
# See how many rows and columns the data has
counts %>%
  dim()
```

```
## [1] 14599     8
```

```r
# See a summary of each column
counts %>%
  summary()
```

```
##    gene_id            untreated1         untreated2       untreated3
##  Length:14599       Min.   :     0.0   Min.   :     0   Min.   :     0.0
##  Class :character   1st Qu.:     0.0   1st Qu.:     1   1st Qu.:     0.0
##  Mode  :character   Median :    27.0   Median :    46   Median :    18.0
##                     Mean   :   957.1   Mean   :  1501   Mean   :   572.5
##                     3rd Qu.:   637.0   3rd Qu.:   990   3rd Qu.:   351.0
##                     Max.   :232141.0   Max.   :360330   Max.   :131242.0
##    untreated4          treated1           treated2           treated3
##  Min.   :     0.0   Min.   :     0.0   Min.   :     0.0   Min.   :     0.0
##  1st Qu.:     0.0   1st Qu.:     1.0   1st Qu.:     0.0   1st Qu.:     0.0
##  Median :    19.0   Median :    47.0   Median :    20.0   Median :    22.0
##  Mean   :   674.1   Mean   :  1278.9   Mean   :   655.6   Mean   :   708.5
##  3rd Qu.:   413.5   3rd Qu.:   902.5   3rd Qu.:   416.0   3rd Qu.:   456.0
##  Max.   :167116.0   Max.   :253500.0   Max.   :146390.0   Max.   :164148.0
```

```r
# See a random sample of 5 rows
counts %>%
  sample_n(size = 5)
```

```
## # A tibble: 5 x 8
##   gene_id untreated1 untreated2 untreated3 untreated4 treated1 treated2 treated3
##   <chr>        <dbl>      <dbl>      <dbl>      <dbl>    <dbl>    <dbl>    <dbl>
## 1 FBgn00~          0          0          0          0        0        0        0
## 2 FBgn00~          3          0          1          1        0        2        0
## 3 FBgn00~          1          6          1          1        6        2        3
## 4 FBgn00~          0          0          0          0        0        0        0
## 5 FBgn00~          0          2          0          1        0        0        0
```

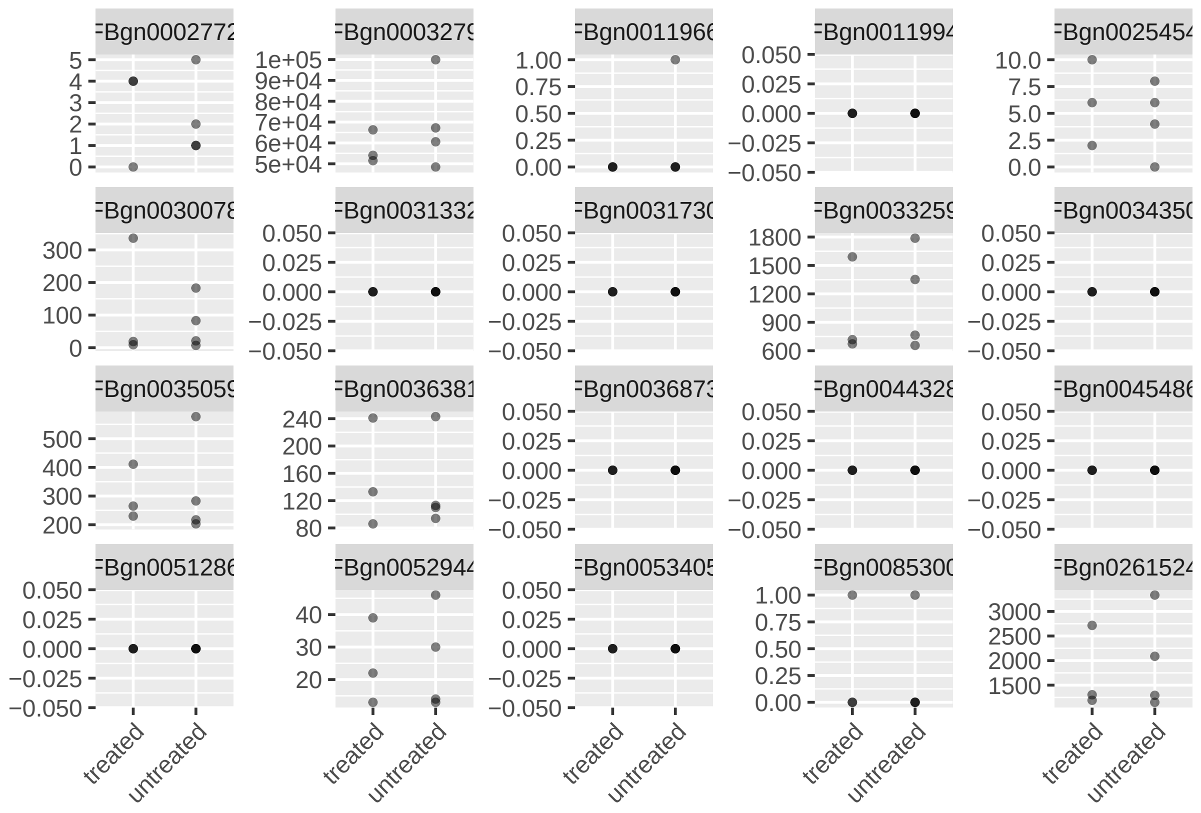
```r
# Check out the unique letters in the gene names
# (i.e., once you take out all the digits at the end)
counts %>%
  pull(gene_id) %>% # Extract just the `gene_id` column as a vector
  str_remove("[0-9].+") %>% # Remove the first digit and anything after
                            # using regular expressions
  unique() # Look at just the unique values
```

```
## [1] "FBgn"
```

It looks like every gene id starts with "FBgn" and then some 7-digit numeric ID.

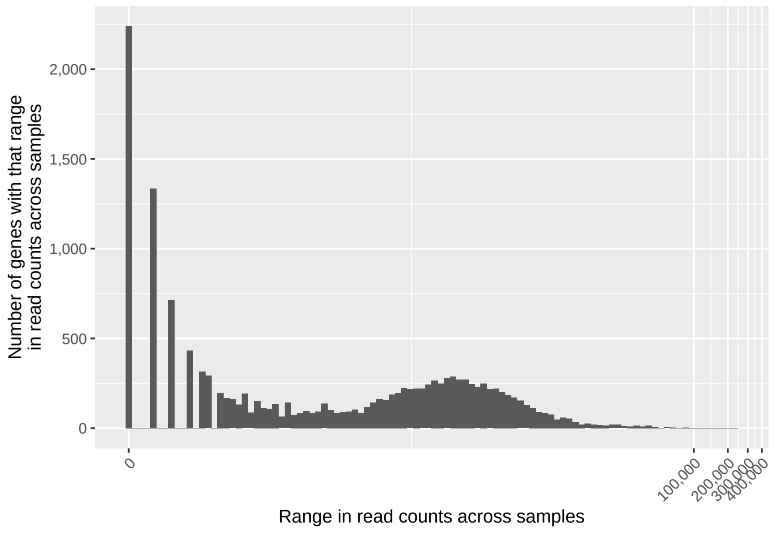
Next, it might be helpful to sample a few of the genes and then visualize how the values recorded for each gene vary across the samples, showing treated / untreated with color:

```r
counts %>%
  sample_n(size = 20) %>%
  pivot_longer(cols = -gene_id, names_to = "sample_id", values_to = "value") %>%
  mutate(treatment = str_remove(sample_id, "[0-9]")) %>% # Extract untreated / treated
                                                         # from sample IDs by getting
                                                         # rid of the number in each
  ggplot(aes(x = treatment, y = value)) +
  labs(x = "", y = "") +
  geom_point(alpha = 0.5, size = 1) + # Use some transparency (alpha) to see if points
                                      # are plotted on top of each other.
  facet_wrap(~ gene_id, scales = "free_y") + # Since scales are very different across
                                             # genes, set scales as free on y-axis
  # Rotate the x-axis labels a bit (otherwise, they overlap)
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



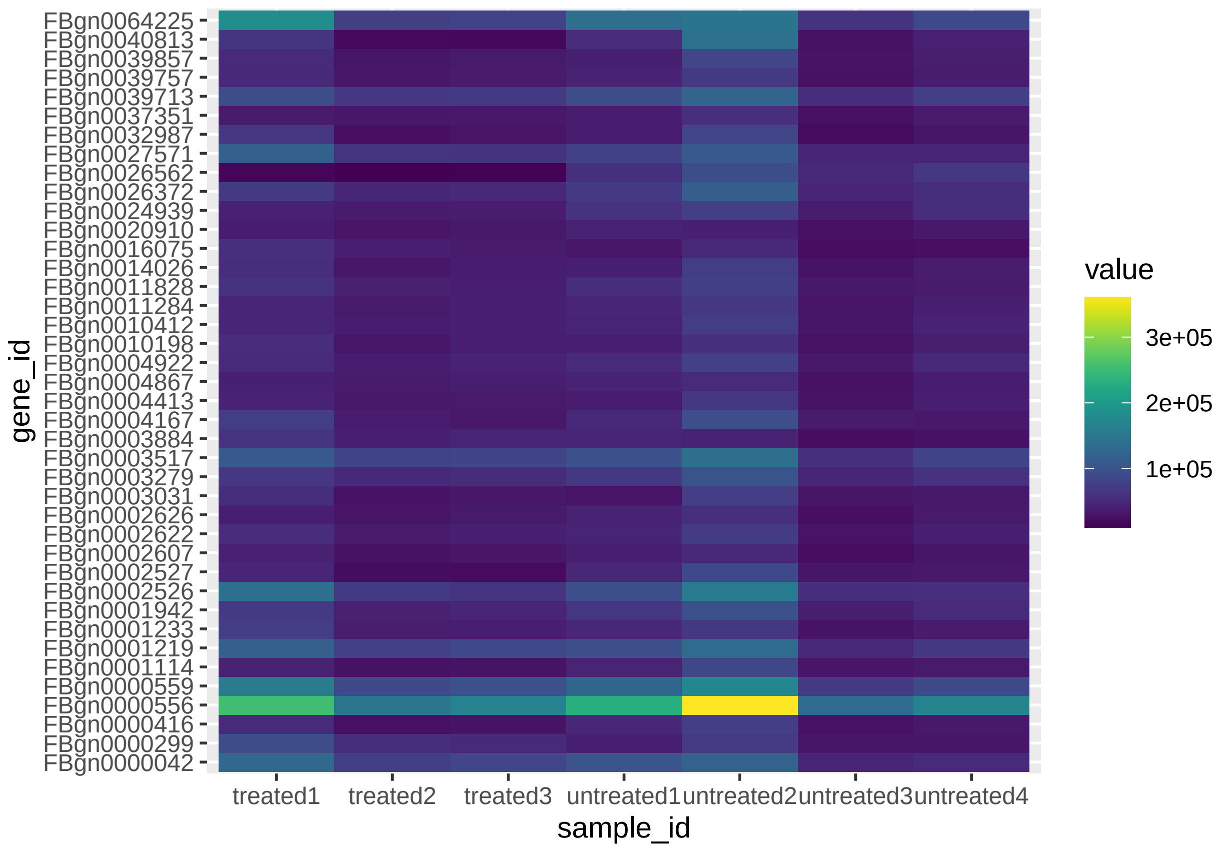
Check out how much the range in the measured levels for genes varies across genes:

```
library("scales")
counts %>%
  # Rearrange the data shape to making grouping and summarizing easier
  pivot_longer(-gene_id, names_to = "sample_id", values_to = "value") %>%
  # Group by gene and then calculate the minimum and maximum levels of each gene
  # and the difference between those (range)
  group_by(gene_id) %>%
  summarize(min_value = min(value),
            max_value = max(value),
            range = max_value - min_value) %>%
  # Plot a histogram of these ranges
  ggplot(aes(x = range)) +
  geom_histogram(bins = 100) +
  labs(x = "Range in read counts across samples",
       y = "Number of genes with that range\nin read counts across samples") +
  # Use a transformation on the x-axis---otherwise, with a few outliers, it's
  # hard to see variation among the low values. You can't just use "log" in this
  # case because you have a lot of zero values.
  scale_x_continuous(trans = "pseudo_log", labels = comma) +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Check typical count levels (for genes with highest levels) by sample.

```
library(purrr)
library(viridis)
counts %>%
  pivot_longer(-gene_id, names_to = "sample_id", values_to = "value") %>%
  # Get top 40 genes in terms of mean expression
  group_by(gene_id) %>%
  nest() %>%
  mutate(mean = map_dbl(data, ~ mean(.x$value))) %>%
  ungroup() %>%
  top_n(n = 40) %>%
  select(-mean) %>%
  unnest(data) %>%
  ggplot(aes(x = sample_id, y = gene_id, fill = value)) +
  geom_tile() +
  scale_fill_viridis()
```



Read in the meta-data for these data. The meta-data are available in a different file in the `padilla` package ("pasilla_sample_annotation.csv").

```
annotationFile <- system.file("extdata", "pasilla_sample_annotation.csv",
                              package = "pasilla", mustWork = TRUE)
pasillaSampleAnno <- annotationFile %>%
  read_csv()
pasillaSampleAnno
```

```
## # A tibble: 7 x 6
##   file      condition type      `number of lane~ `total number of r~ `exon counts`
##   <chr>     <chr>     <chr>                <dbl> <chr>                       <dbl>
## 1 treated~ treated   single-~                 5 35158667                 15679615
## 2 treated~ treated   paired-~                 2 12242535 (x2)            15620018
## 3 treated~ treated   paired-~                 2 12443664 (x2)            12733865
## 4 untreat~ untreated single-~                 2 17812866                 14924838
## 5 untreat~ untreated single-~                 6 34284521                 20764558
## 6 untreat~ untreated paired-~                 2 10542625 (x2)            10283129
## 7 untreat~ untreated paired-~                 2 12214974 (x2)            11653031
```

This file gives some general information about each sample. This includes the file name where the sample's data was saved (probably from the sampling equipment), the treatment or condition (treated / untreated), the type of read (single-read or paired-end), the number of lanes, total number of reads, and exon counts.

It can be hard to work with column names with spaces in the middle, so I recommend renaming the columns (replacing every space with an underscore will work):

```
# Current column names (they have spaces)
pasillaSampleAnno %>%
  colnames()
```

```
## [1] "file"                    "condition"              "type"
## [4] "number of lanes"         "total number of reads" "exon counts"
```

```
# Rename columns to get rid of spaces
pasillaSampleAnno <- pasillaSampleAnno %>%
  # \\s stands for a space in R regular expressions
  rename_all(.funs = str_replace_all, "\\s", "_")

# New column names (no more spaces)
pasillaSampleAnno %>%
  colnames()
```

```
## [1] "file"                    "condition"              "type"
## [4] "number_of_lanes"         "total_number_of_reads" "exon_counts"
```

We can see some summaries of the data by condition (treated / untreated) and type (single-read / pair-ended):

```
pasillaSampleAnno %>%
  # Group the data and use `count` to count the number of rows with each combo
  group_by(condition, type) %>%
  count() %>%
  ungroup() %>%
  # Reshape the data to make a prettier table
  pivot_wider(names_from = type, values_from = n) %>%
  # Use `kable` from the `knitr` package to output as a pretty table
  knitr::kable()
```

condition	paired-end	single-read
treated	2	1
untreated	2	2

Create a `DESeqDataSet` object with both the measurements (from `counts`) and the meta-data (from `pasillaSampleAnno`). This *data container* is a special type of object class that will help keep everything in order as you move through the analysis (and helps with managing the size of the data).

```
library("DESeq2")

# Need to remember what the column names of `counts` are (so we can
# use the same ones for the meta-data)
colnames(counts)
```

```
## [1] "gene_id"    "untreated1" "untreated2" "untreated3" "untreated4"
## [6] "treated1"   "treated2"   "treated3"
```

```
# Clean up the meta-data a bit before we add it to this special object class
pasillaSampleAnno <- pasillaSampleAnno %>%
  rename(file = "sample_id") %>%
  # Use regular expressions to clean up sample_id name (so it matches
  # the column names in `counts`)
  mutate(sample_id = str_remove(sample_id, "fb")) %>%
  # Evidently, the "-"s in the `type` column may cause problems in DESeq,
  # so shorten those labels
  mutate(type = str_remove(type, "-.+")) %>%
  # Convert the `condition` and `type` columns to factors and
```

```r
  # set the level order by hand
  mutate(condition = condition %>%
           as_factor() %>%
           fct_relevel("untreated", "treated"),
         type = type %>%
           as_factor() %>%
           fct_relevel("single", "paired")) %>%
  # Finally, you need to make sure the rows of this dataframe
  # are in the same order as the columns in the read count data matrix.
  # One way to do this is by making the column names of the count data
  # matrix (without the gene_id column) into a dataframe, with numbers
  # giving the order, join that with this data, then reorder by that
  # order number
  full_join(counts %>%
              # Get the column names from `counts`
              colnames() %>%
              # Get rid of the first column name (which is the `gene_id` column)
              `[`(-1) %>%
              # `enframe` will convert to a dataframe (from a vector)
              enframe(name = "order"),
            by = c("sample_id" = "value")) %>%
  # Rearrange by order, then get rid of that column
  arrange(order) %>%
  select(-order)

# Here's what the meta-data looks like now:
pasillaSampleAnno
```

```
## # A tibble: 7 x 6
##   sample_id  condition type   number_of_lanes total_number_of_reads exon_counts
##   <chr>      <fct>     <fct>            <dbl> <chr>                       <dbl>
## 1 untreated1 untreated single               2 17812866                 14924838
## 2 untreated2 untreated single               6 34284521                 20764558
## 3 untreated3 untreated paired               2 10542625 (x2)            10283129
## 4 untreated4 untreated paired               2 12214974 (x2)            11653031
## 5 treated1   treated   single               5 35158667                 15679615
## 6 treated2   treated   paired               2 12242535 (x2)            15620018
## 7 treated3   treated   paired               2 12443664 (x2)            12733865
```

```r
# Put all the required components into a DESeqDataSet object so you can
# run DESeq on it
pasilla <- counts %>%
  # For the read data, it's all in counts, but we need to take off the
  # first column (with the gene IDs) and then convert to a matrix to
  # input it into this object class
  column_to_rownames("gene_id") %>%
  as.matrix() %>%
  # Put everything into a DESeqDataSet object
  DESeqDataSetFromMatrix(colData = pasillaSampleAnno,
                         design = ~ condition)

class(pasilla)
```

```
## [1] "DESeqDataSet"
## attr(,"package")
## [1] "DESeq2"
```

DESeq

Apply `DESeq` algorithm to the data:

```
pasilla <- pasilla %>%
  DESeq()
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

You can use *extractor functions* like `results` to pull out specific data from the resulting object. For example, you can pull out gene-specific estimates of differences between treated and untreated samples with:

```
pasilla %>%
  results()
```

```
## log2 fold change (MLE): condition treated vs untreated
## Wald test p-value: condition treated vs untreated
## DataFrame with 14599 rows and 6 columns
##                      baseMean      log2FoldChange             lfcSE
##                     <numeric>           <numeric>         <numeric>
## FBgn0000003 0.171568715207063    1.02601368333522  3.80551160374507
## FBgn0000008  95.1440789963134 0.00215175720349084 0.223883696572144
## FBgn0000014  1.05657219346166  -0.496735176118498  2.16026588878143
## FBgn0000015 0.846723274987709   -1.88276477012506  2.10643312162068
## FBgn0000017   4352.5928987935  -0.240025038806395 0.126024438560778
## ...                       ...                 ...               ...
## FBgn0261571 0.087343676946538    0.90026855885989  3.81017301615324
## FBgn0261572  6.19713652050888  -0.959130034959993 0.777016744083823
## FBgn0261573  2240.98398636611  0.0126159820947047 0.112700631633334
## FBgn0261574  4857.74267170989  0.0152573285663809  0.19314843580021
## FBgn0261575  10.6835537573787   0.163562434452904 0.938909701933571
##                           stat             pvalue              padj
##                      <numeric>          <numeric>         <numeric>
## FBgn0000003   0.269612548895004  0.787458345134478                NA
## FBgn0000008 0.00961104911360736   0.99233161035707 0.996928202137134
## FBgn0000014  -0.229941683890912  0.818137084970592                NA
## FBgn0000015  -0.893816542666432  0.371420056652208                NA
## FBgn0000017   -1.90459121696969 0.0568332291795643 0.282363564717396
## ...                        ...                ...               ...
## FBgn0261571   0.236280230594043  0.813215226384258                NA
## FBgn0261572   -1.23437498903695  0.217063204412214                NA
## FBgn0261573   0.111942425804233  0.910869057061802 0.982036790151162
## FBgn0261574  0.0789927627587049  0.937038379558298 0.988141534972512
## FBgn0261575   0.174204648344848  0.861704632545053 0.967913643037464
```

If you want to work more easily with this output, you can convert it to a tibble with `as_tibble`:

```
pasilla %>%
  results() %>%
  as.data.frame() %>%
  rownames_to_column(var = "gene_id") %>%
  as_tibble()
```

```
## # A tibble: 14,599 x 7
##    gene_id      baseMean log2FoldChange lfcSE     stat pvalue   padj
##    <chr>           <dbl>          <dbl> <dbl>    <dbl>  <dbl>  <dbl>
##  1 FBgn0000003    0.172         1.03    3.81   0.270   0.787  NA
##  2 FBgn0000008   95.1           0.00215 0.224  0.00961 0.992   0.997
##  3 FBgn0000014    1.06         -0.497   2.16  -0.230   0.818  NA
##  4 FBgn0000015    0.847        -1.88    2.11  -0.894   0.371  NA
##  5 FBgn0000017 4353.           -0.240   0.126 -1.90    0.0568  0.282
##  6 FBgn0000018  419.           -0.105   0.148 -0.707   0.480   0.824
##  7 FBgn0000022    0.0797       -0.720   3.81  -0.189   0.850  NA
##  8 FBgn0000024    6.41          0.211   0.690  0.305   0.760  NA
##  9 FBgn0000028    0.439         1.41    2.78   0.509   0.611  NA
## 10 FBgn0000032  990.           -0.0919  0.148 -0.622   0.534   0.850
## # ... with 14,589 more rows
```

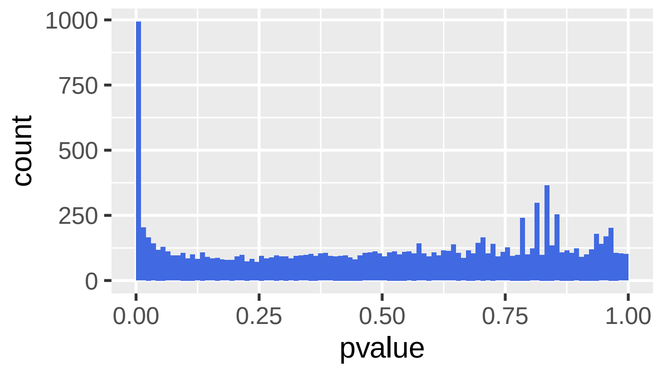
Check out the top few genes in terms of adjusted p-values for the Wald test comparing the untreated versus treated samples. Since smaller p-values are more interesting, pick out the smallest:

```
pasilla %>%
  results() %>%
  as.data.frame() %>%
  rownames_to_column(var = "gene_id") %>%
  filter(!is.na(padj)) %>%
  as_tibble() %>%
  arrange(padj) %>%
  dplyr::slice(1:5)
```

```
## # A tibble: 5 x 7
##   gene_id     baseMean log2FoldChange  lfcSE  stat    pvalue      padj
##   <chr>          <dbl>          <dbl>  <dbl> <dbl>     <dbl>     <dbl>
## 1 FBgn0039155     731.          -4.62 0.169  -27.4 4.89e-165 4.07e-161
## 2 FBgn0025111    1501.           2.90 0.127   22.8 1.53e-115 6.38e-112
## 3 FBgn0029167    3706.          -2.20 0.0970 -22.7 1.33e-113 3.69e-110
## 4 FBgn0003360    4343.          -3.18 0.144  -22.2 9.56e-109 1.99e-105
## 5 FBgn0035085     638.          -2.56 0.137  -18.6 1.29e- 77 2.14e- 74
```

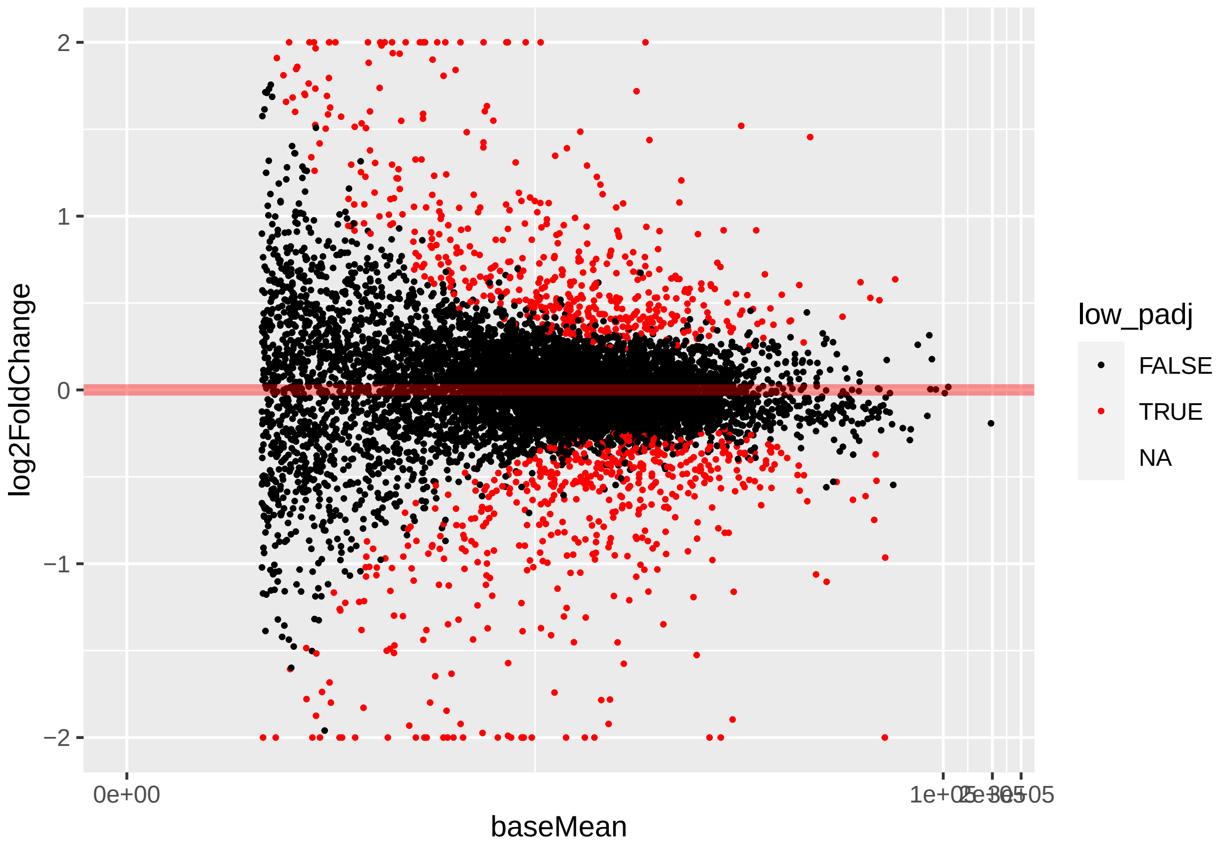
Histogram of p-values:

```
pasilla %>%
  results() %>%
  as_tibble() %>%
  ggplot(aes(x = pvalue)) +
  geom_histogram(binwidth = 0.01, fill = "Royalblue", boundary = 0)
```



MA plot (but using ggplot instead of `plotMA` from `DESeq2` package—the x-axis transform might not be identical...):

```
pasilla %>%
  results() %>%
  as_tibble() %>%
  mutate(low_padj = padj < 0.1) %>%
  mutate(log2FoldChange = ifelse(log2FoldChange > 2, 2, log2FoldChange),
         log2FoldChange = ifelse(log2FoldChange < -2, -2, log2FoldChange)) %>%
  ggplot(aes(x = baseMean, y = log2FoldChange, color = low_padj)) +
  geom_point(size = 0.5) +
  ylim(c(-2, 2)) +
  scale_color_manual(values = c("black", "red")) +
  scale_x_continuous(trans = "pseudo_log") +
  geom_hline(yintercept = 0, color = "red", size = 2, alpha = 0.4)
```


log2FoldChange
0e+00
1e+05
2e+05
3e+05
baseMean
low_padj
FALSE
TRUE
NA