Exercises with ggplot2

Aubrey Odom-Mabey

2/14/2022

Adapted from Babraham Bioinformatics, version 2021-09

```r
suppressPackageStartupMessages({
  library(tidyverse)
})
```

Exercise 1: Simple point and line plots

1a) Weight Chart

Load the data from the weight_chart.txt file. This is a tab delimited text file, so we will use `read_delim()` to load the file and save it to the variable `wt`.

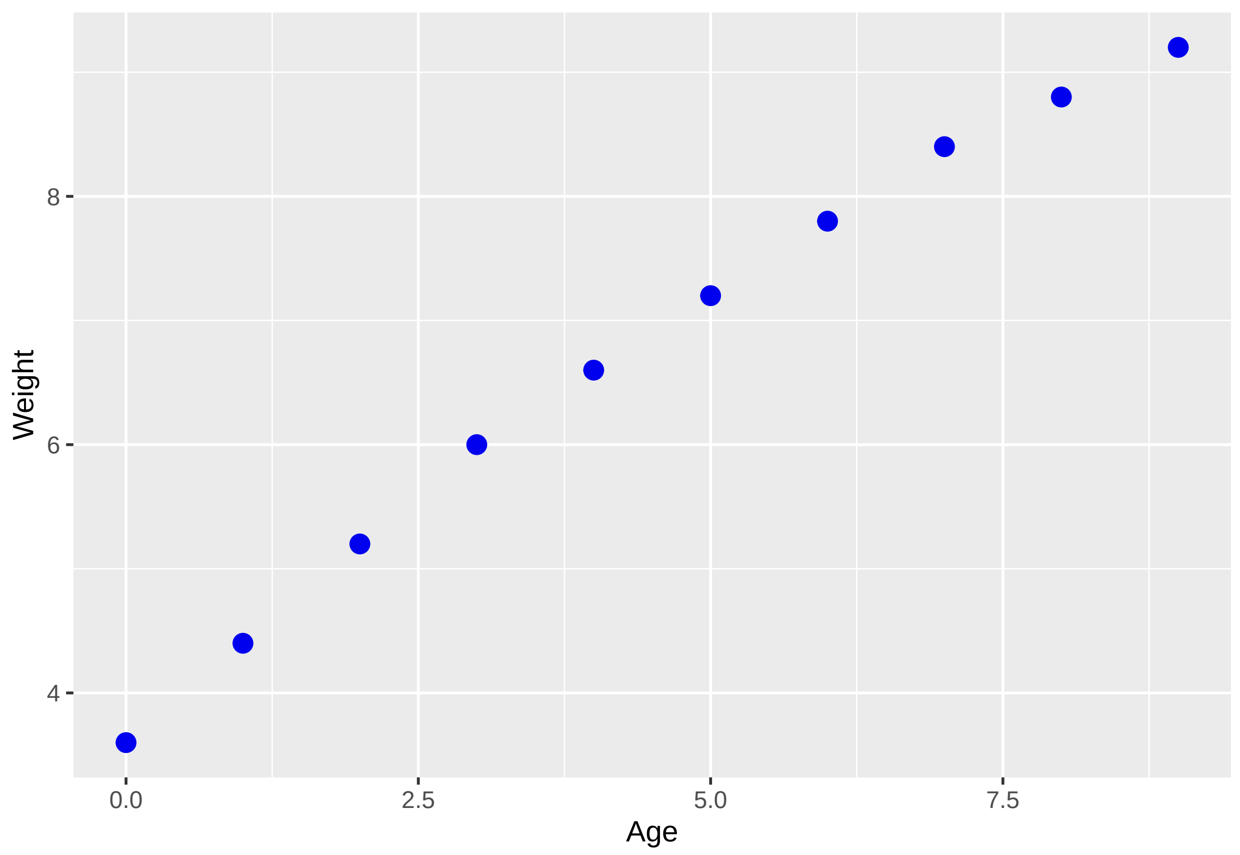
This file contains the details of the growth of a baby over the first few months of its life.

- Draw a scatterplot (using `geom_point`) of the `Age` vs `Weight`. When defining your aesthetics, the `Age` will be the `x` and `Weight` will be the `y`.
- Make all of the points filled with the color "blue2" by putting a fixed aesthetic into `geom_point()`. Also, change the points to be size 3.
- You will see that an obvious relationship exists between the two variables. Change the geometry to `geom_line()` to see another way to represent this plot.
- Combine the two plots by adding both a `geom_line()` and a `geom_point()` geometry to show both the individual points and the overall trend.

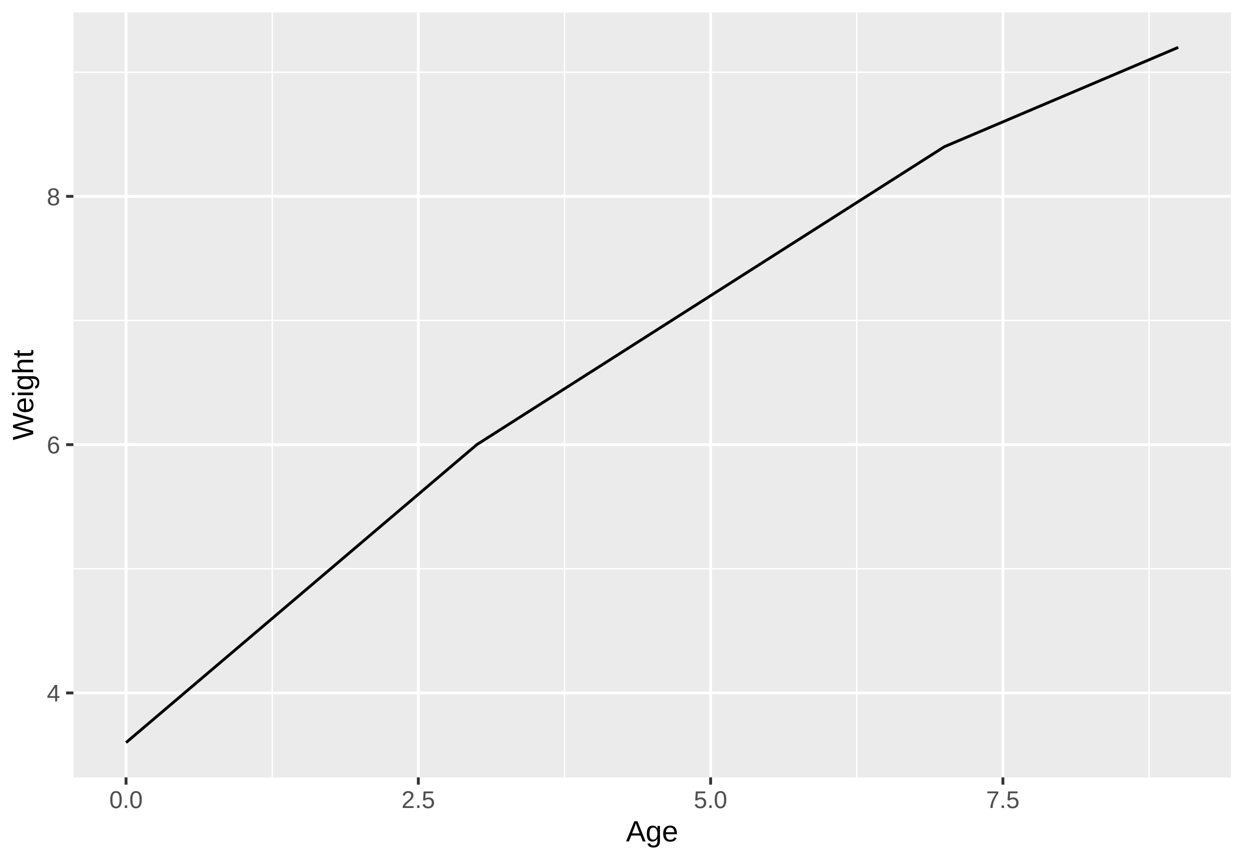
```r
wt <- read_delim("weight_chart.txt")
```

```
## Rows: 10 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## dbl (2): Age, Weight
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

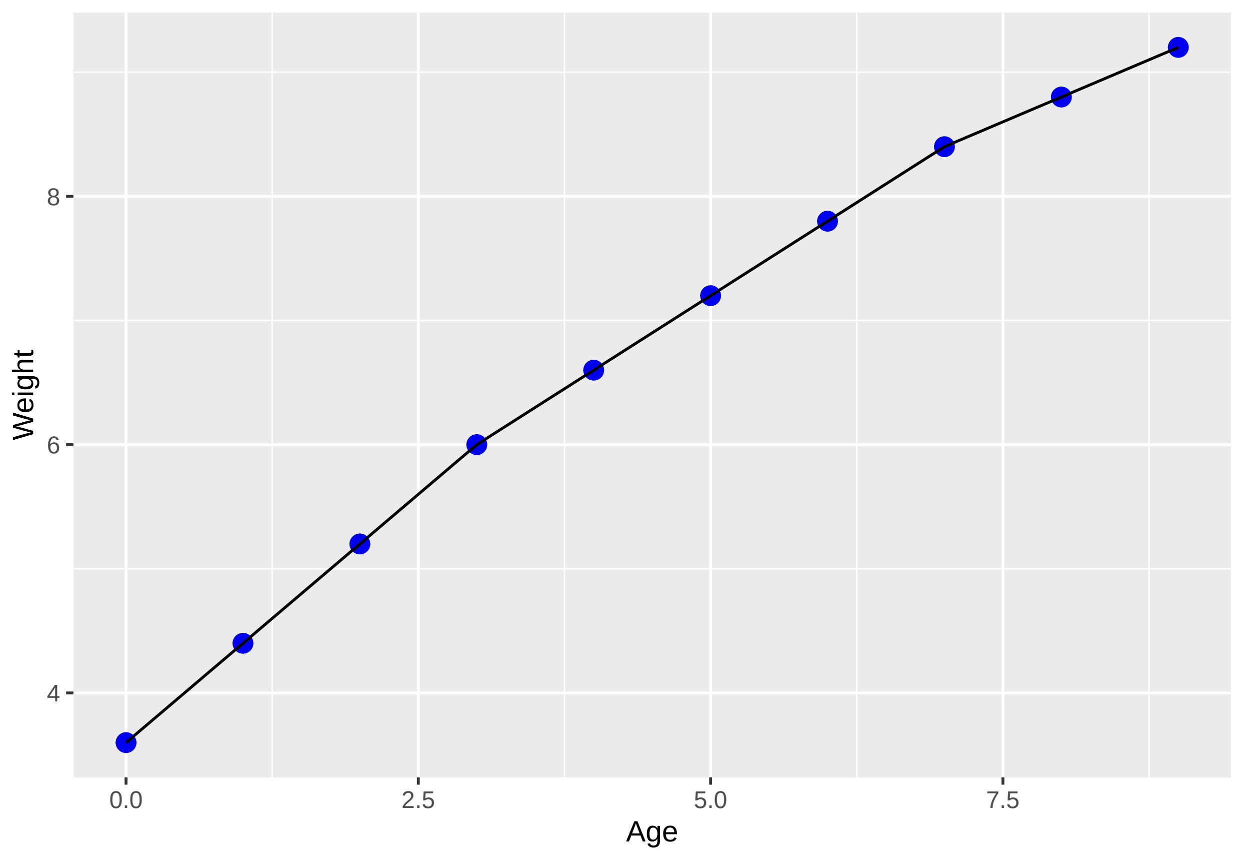
```r
p <- ggplot(data = wt, aes(x = Age, y = Weight))
p + geom_point(color = 'blue2', size = 3)
```



```r
p + geom_line()
```



```r
p + geom_point(color = 'blue2', size = 3) + geom_line()
```



1b) Chromosome Position

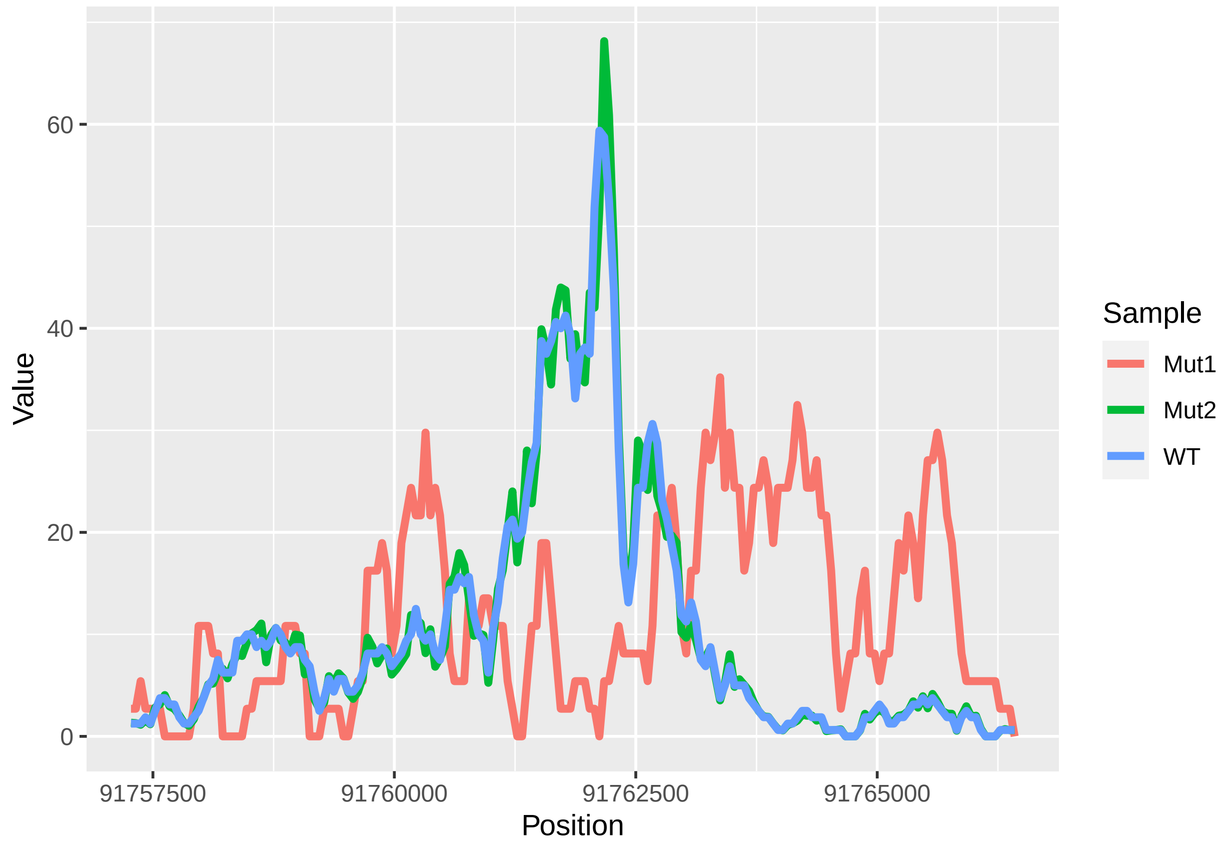
Load the data for the chromosome_position_data.txt file.

- Use `pivot_longer()` to put the data into tidy format, by combining the three data columns together. The options to `pivot_longer()` will be:
  - The columns to restructure: `cols = Mut1:WT`
  - The name of the new names column: `names_to = "Sample"`
  - The name of the values column: `values_to = "Value"`
- Draw a line (`geom_line`) graph to plot the position (`x = Position`) against the value (`y = Value`) and splitting the Samples by color (`color = Sample`). Use the size attribute in `geom_line()` to make the lines slightly thicker than their default width.

```
read_delim("chromosome_position_data.txt") %>%
  pivot_longer(cols = Mut1:WT, names_to = "Sample", values_to = "Value") ->
  chrom
```

```
## Rows: 184 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## dbl (4): Position, Mut1, Mut2, WT
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
ggplot(chrom, aes(x = Position, y = Value, color = Sample)) + geom_line(size = 1.4)
```



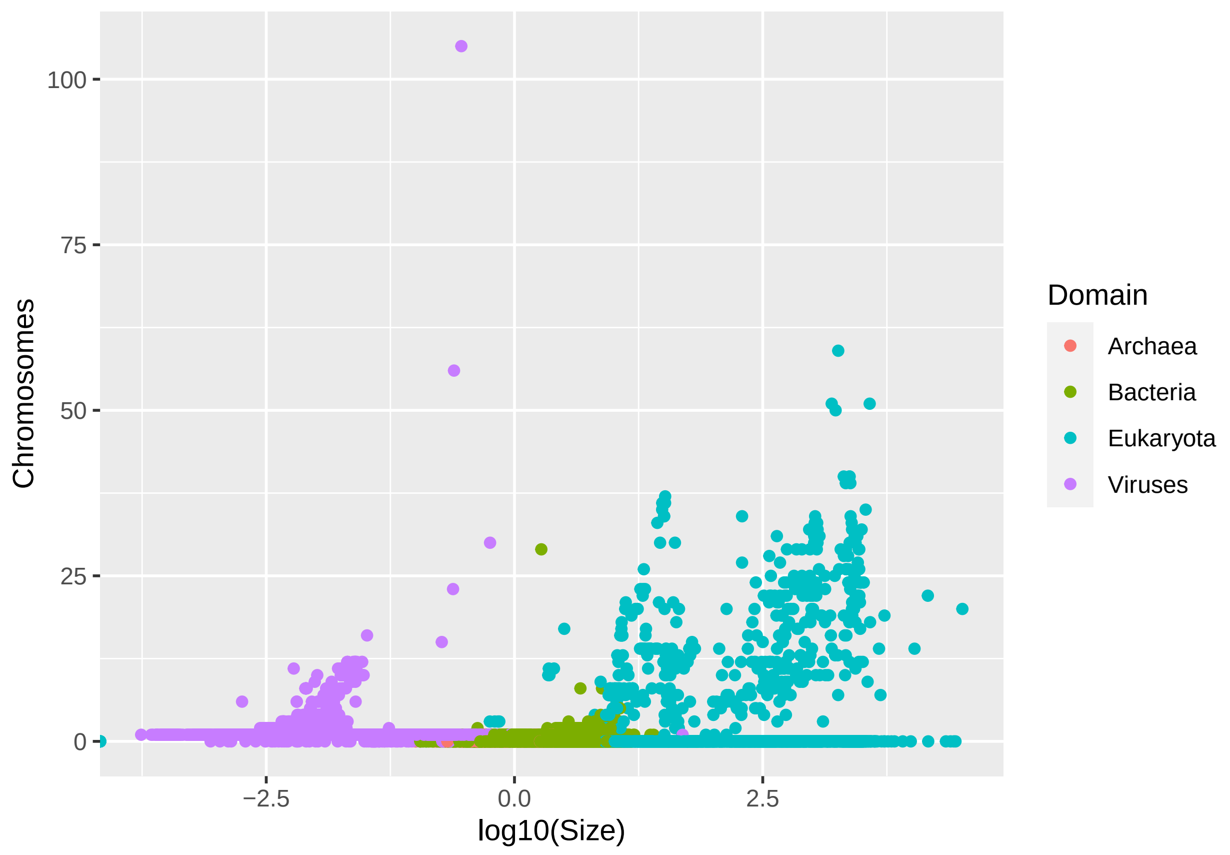
1c) If you have time...

- Load in the genomes.csv file and use the `separate` function to turn the `Groups` column into `Domain`, `Kingdom` and `Class` based on a semicolon delimiter.
- Plot a point graph of `log10(Size)` vs `Chromosomes` and color it by `Domain`.

```r
read_delim("genomes.csv") %>%
  separate(Groups, c("Domain", "Kingdom", "Class"), sep = ";") ->
  gen
```

```
## Rows: 47211 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Organism, Groups
## dbl (5): Size, Chromosomes, Organelles, Plasmids, Assemblies
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```r
ggplot(gen, aes(x = log10(Size), y = Chromosomes, color = Domain)) + geom_point()
```



Exercise 2: Barplots and Distributions

2a) Small File

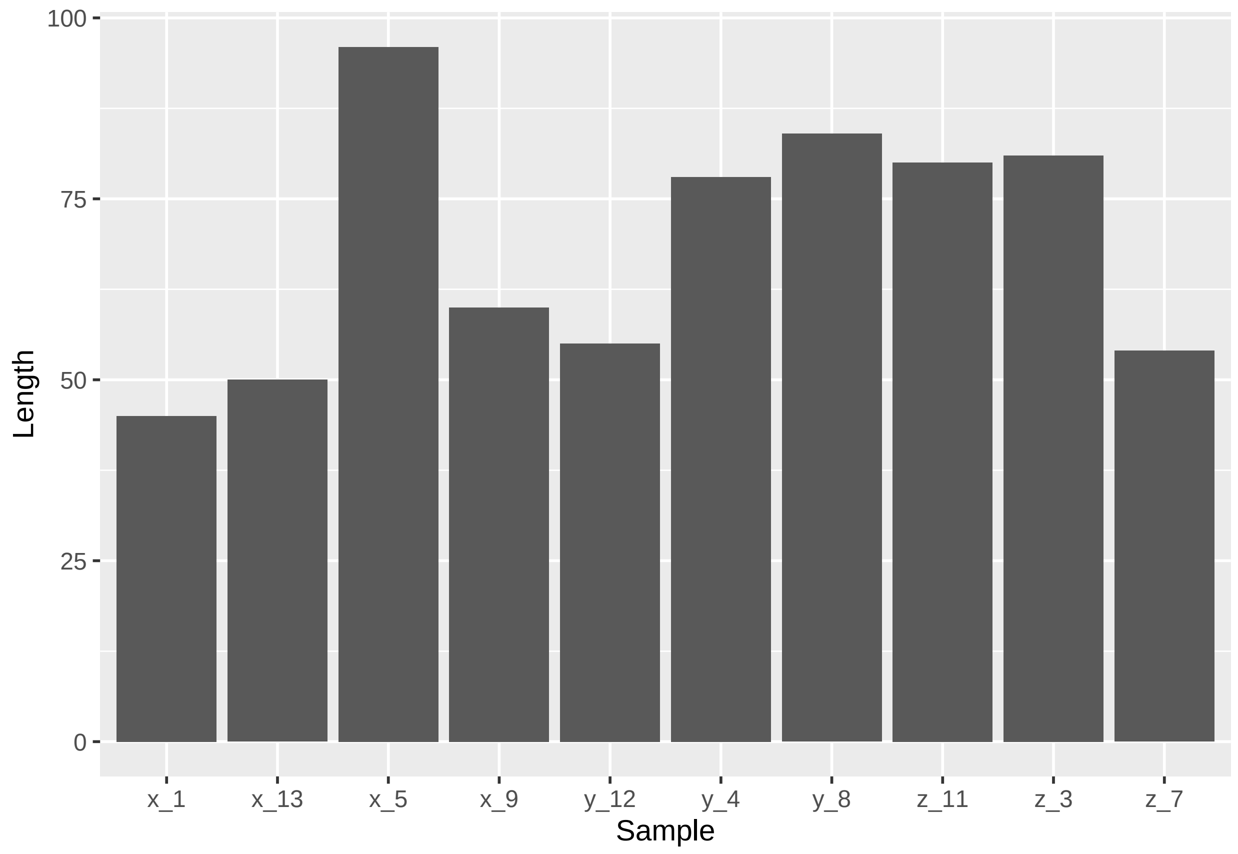
Load the data from small_file.txt using `read_delim()`. Save this as small_file.

- Plot out a barplot of the lengths of each sample from category A.
  - Start by filtering the data to keep only category A samples (starter code: `small %>% filter(Category == "A")`)
  - Pass this filtered `tibble` to `ggplot()`.
  - Your `x` aesthetic will be `Sample` and your `y` will be `length`.
  - Since the value in the data is the bar height you need to use `geom_col`.
- Plot out a barplot (using `geom_bar()`) of the mean length for each category in small_file.
  - You will need to set `stat = "summary"`, `fun = "mean"` in `geom_bar()` so it plots the mean value.
- Add a call to `geom_jitter()` to the last plot so you can also see the individual points.
  - Color the points by `Category` and decrease the width of the jitter columns to get better separation. Make sure height is set to 0 .
  - If you don't want to see the legend then you can set `show.legend = FALSE` in `geom_jitter()`.

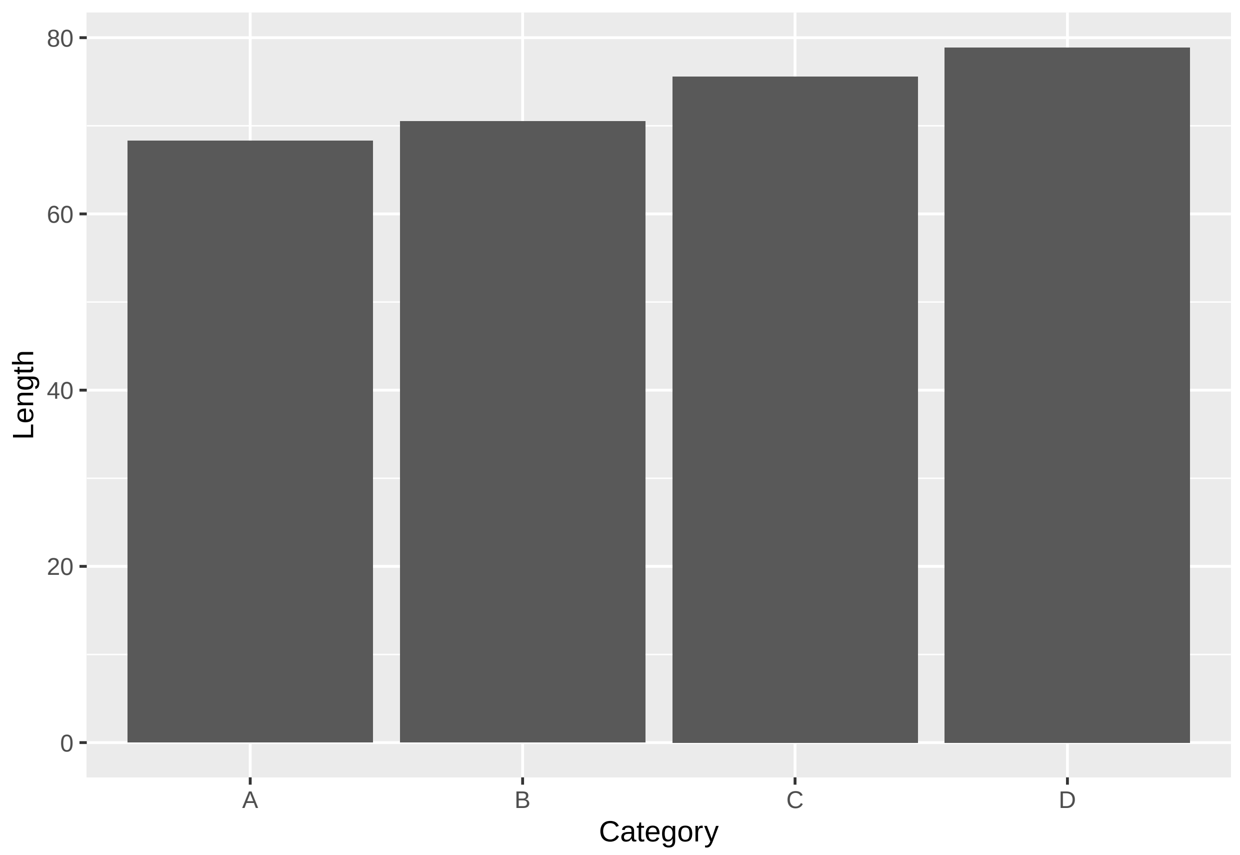
```r
small_file <- read_delim("small_file.txt")
```

```
## Rows: 40 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (2): Sample, Category
## dbl (1): Length
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

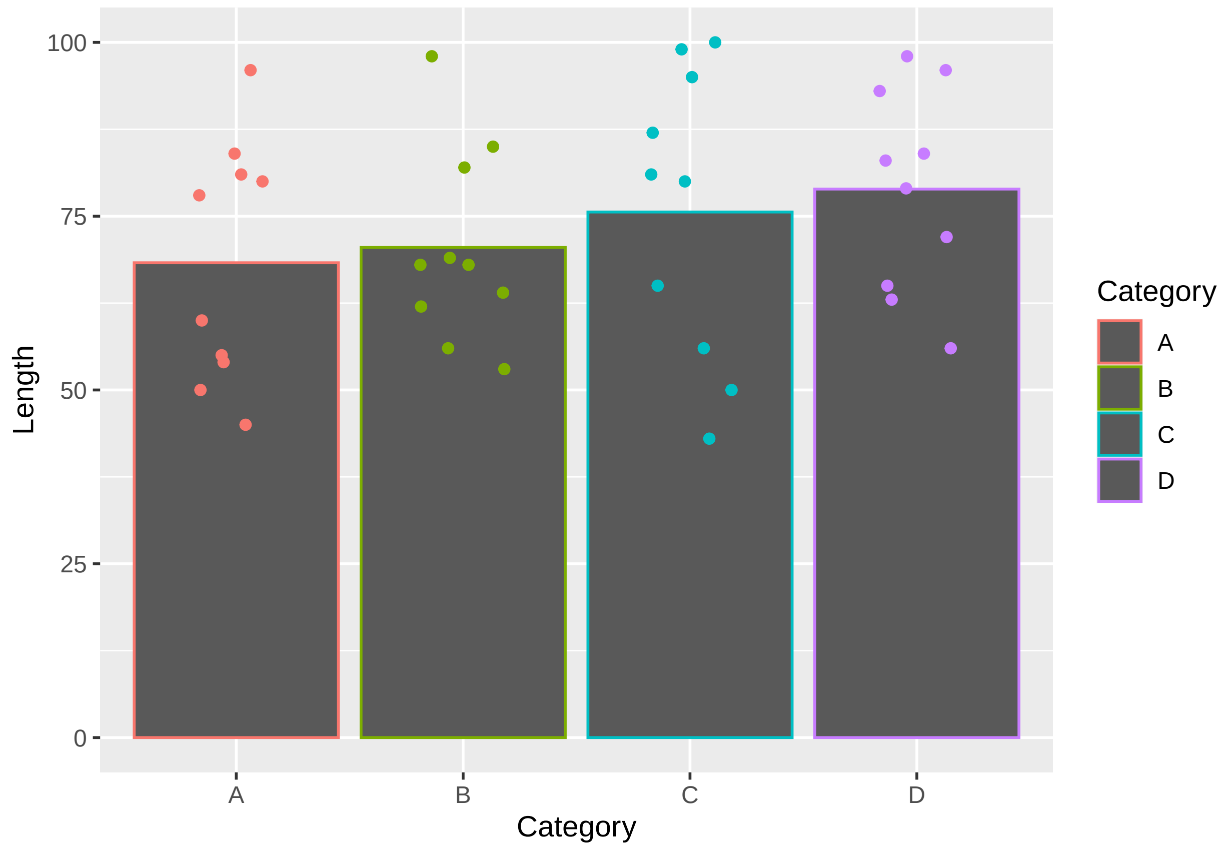
```r
#i
small_file %>%
  filter(Category == "A") %>%
  ggplot(aes(x = Sample, y = Length)) +
  geom_col()
```



```r
#ii
small_file %>%
  ggplot(aes(x = Category, y = Length)) +
  geom_bar(stat = "summary", fun = "mean")
```



```
#iii
small_file %>%
  ggplot(aes(x = Category, y = Length, color = Category)) +
  geom_bar(stat = "summary", fun = "mean") +
  geom_jitter(width = 0.2, height = 0, show.legend = FALSE)
```



2b) Expression Data

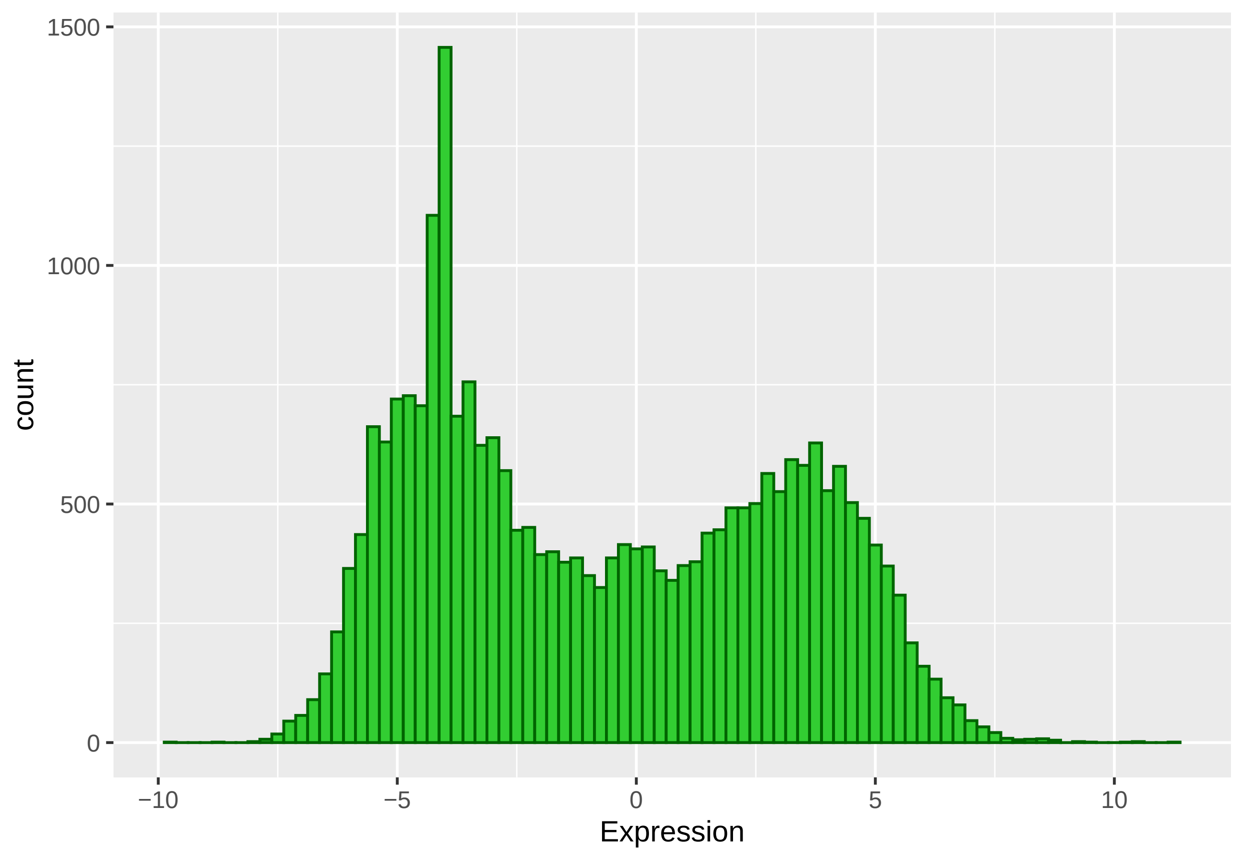
Load the data from expression.txt using `read_delim`.

- Plot out the distribution of `Expression` values in this data. You can try both `geom_histogram()` and `geom_density()`. Try changing the color and fill parameters to make the plot look prettier. In `geom_histogram()` try changing the binwidth parameter to alter the resolution of the distribution.

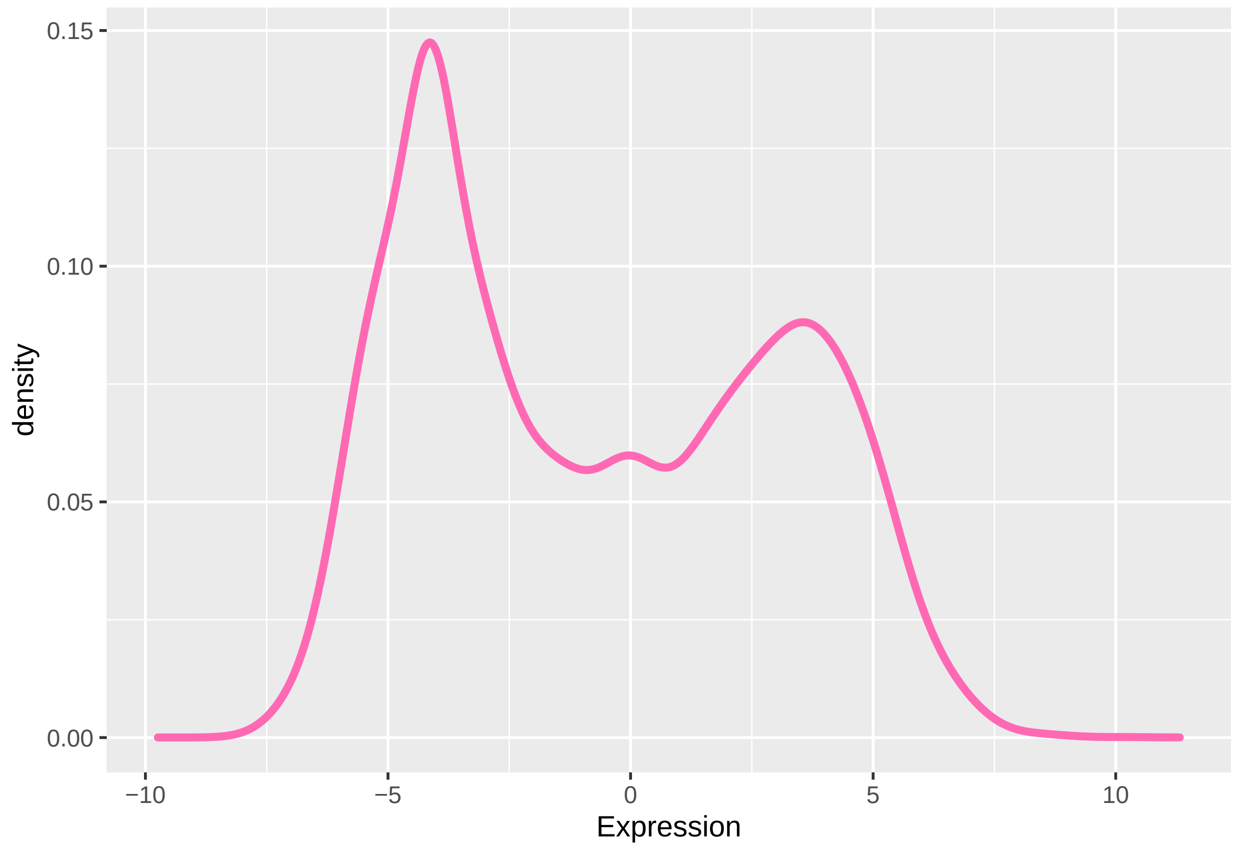
```
expr <- read_delim("expression.txt")
```

```
## Rows: 26127 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (1): Gene
## dbl (1): Expression
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#i
expr %>%
  ggplot(aes(x = Expression)) +
  geom_histogram(binwidth = 0.25, color = "darkgreen",
                 fill = "limegreen", size = 0.5)
```



```
#ii
expr %>%
  ggplot(aes(x = Expression)) +
  geom_density(color = "hotpink", size = 1.4)
```



2c) Cancer Stats

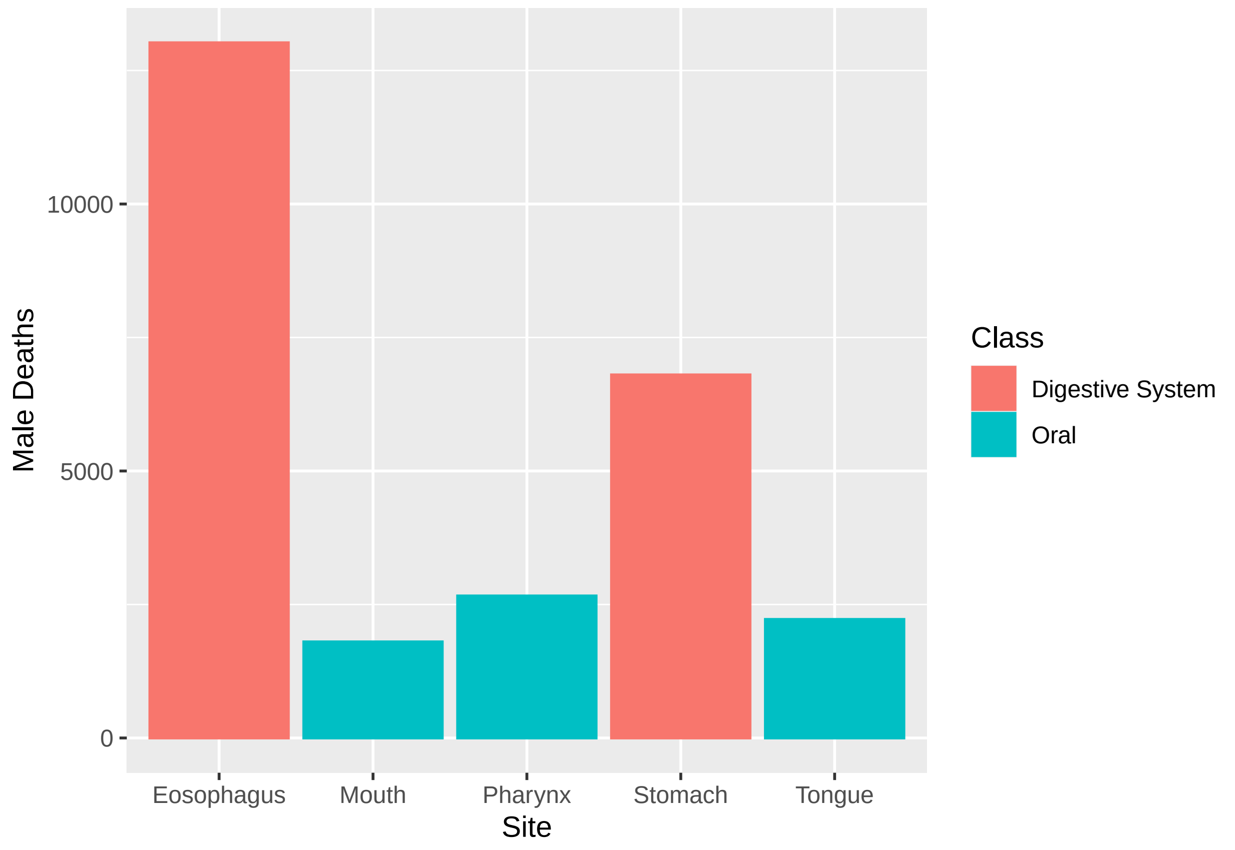
Load the data from cancer_stats.csv using `read_delim()`.

- Plot a barplot (`geom_col()`) of the number of male deaths for all sites. (`x = Site, y= Male Deaths`)
  - Make sure you let the RStudio auto-complete help you to fill in the Male Deaths column name so you get the correct backtick quotes around it.
- You won't be able to show all of the categories, so just show the first 5 (`cancer %>% dplyr::slice(1:5) %>% ggplot...`).

```r
cancer <- read_delim("cancer_stats.csv")
```

```
## Rows: 36 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Class, Site
## dbl (4): Male Cases, Female Cases, Male Deaths, Female Deaths
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```r
cancer %>%
  dplyr::slice(1:5) %>%
  ggplot(aes(x = Site, y = `Male Deaths`,
             fill = Class, color = Class)) +
  geom_col()
```



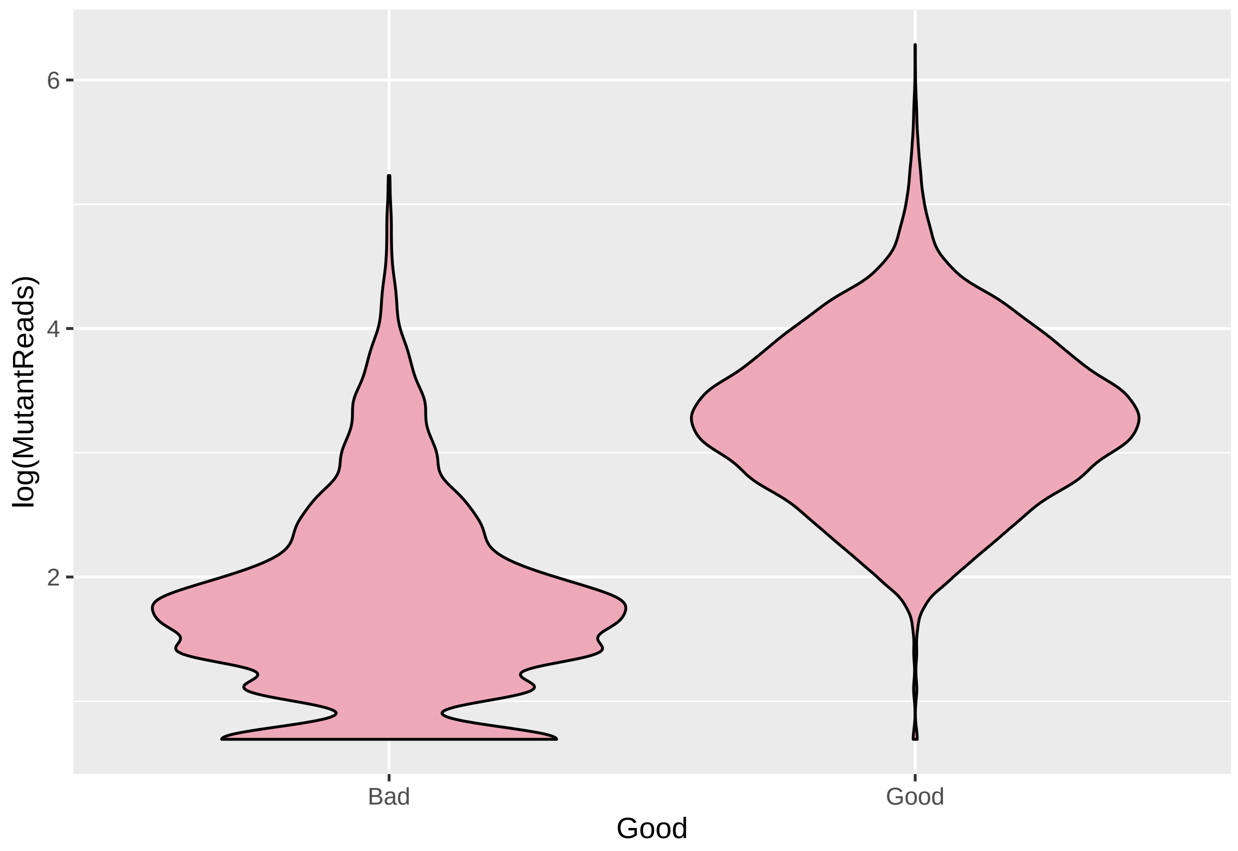
2d) If you have time…

- Load Child_Variants.csv and create a new variable called `Good` using `mutate()` and `if_else()`. The value should be "`GOOD`" if `QUAL == 200` otherwise it should be "`BAD`".
- Plot out a violin plot, using `geom_violin()` of the `MutantReads` for the two `Good` categories.

```r
child <- read_delim("Child_variants.csv")
```

```
## Rows: 25822 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): CHR, dbSNP, REF, ALT, GENE, ENST
## dbl (5): POS, QUAL, MutantReads, COVERAGE, MutantReadPercent
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
child %>%
  mutate('Good' = ifelse(QUAL == 200, 'Good', 'Bad')) %>%
  ggplot(aes(x = Good, y = log(MutantReads))) +
  geom_violin(fill = "pink2", color = "black")
```

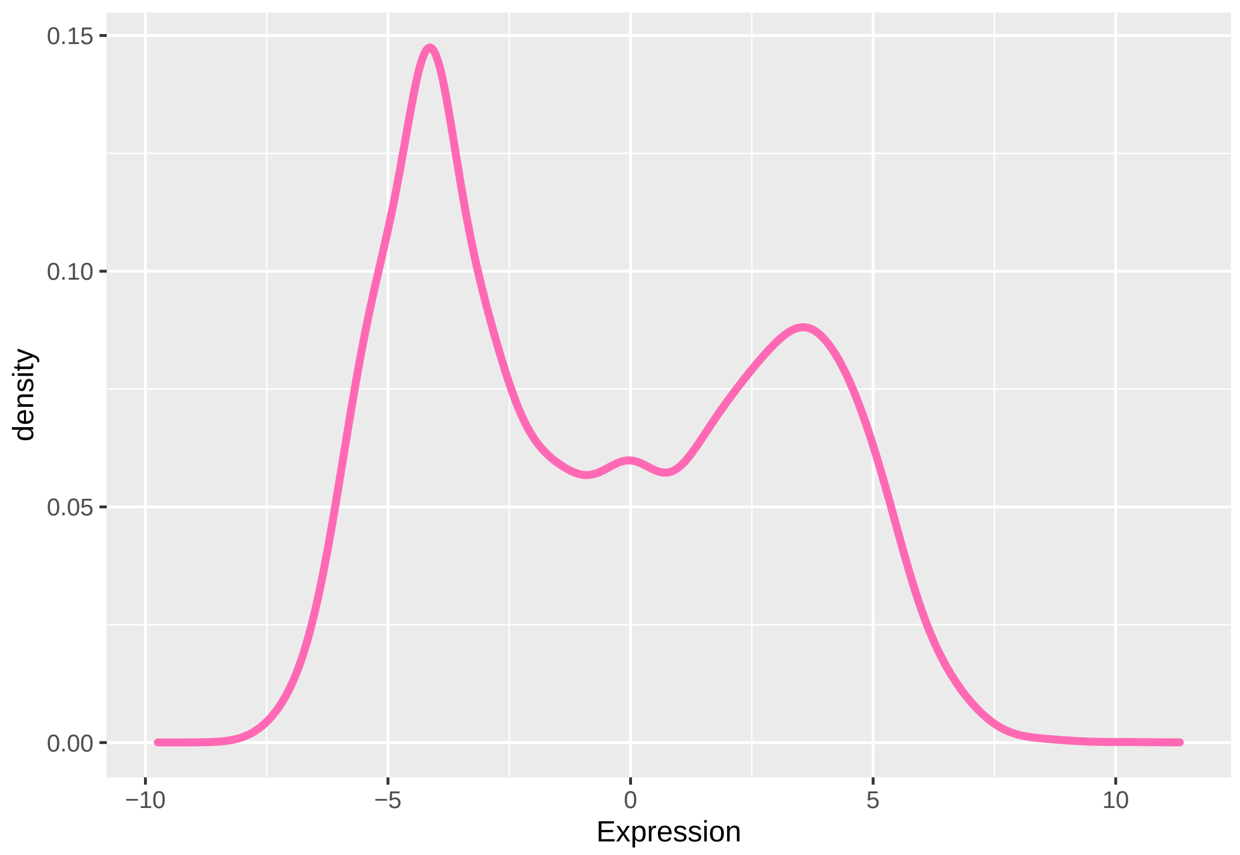


Exercise 3: Annotation, Scaling and Colors

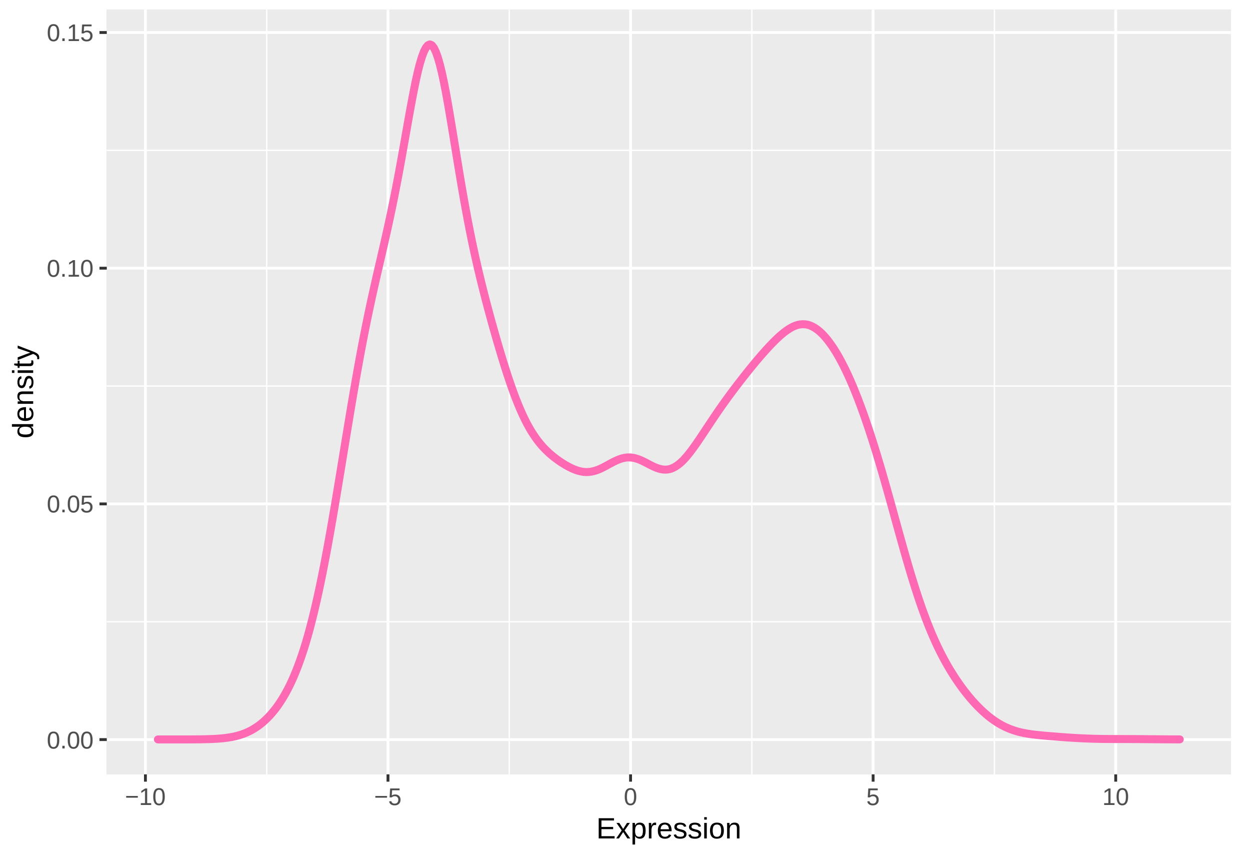
3a) Using themes

Use `theme_set()` to set your ggplot theme to be `theme_bw()` with a `base_size` of 12. Replot one of your earlier plots to see how its appearance changed.

```r
expr %>%
  ggplot(aes(x = Expression)) +
  geom_density(color = "hotpink", size = 1.4)
```



```
expr %>%
  ggplot(aes(x = Expression)) +
  geom_density(color = "hotpink", size = 1.4) +
  theme_set(theme_bw(base_size = 12))
```

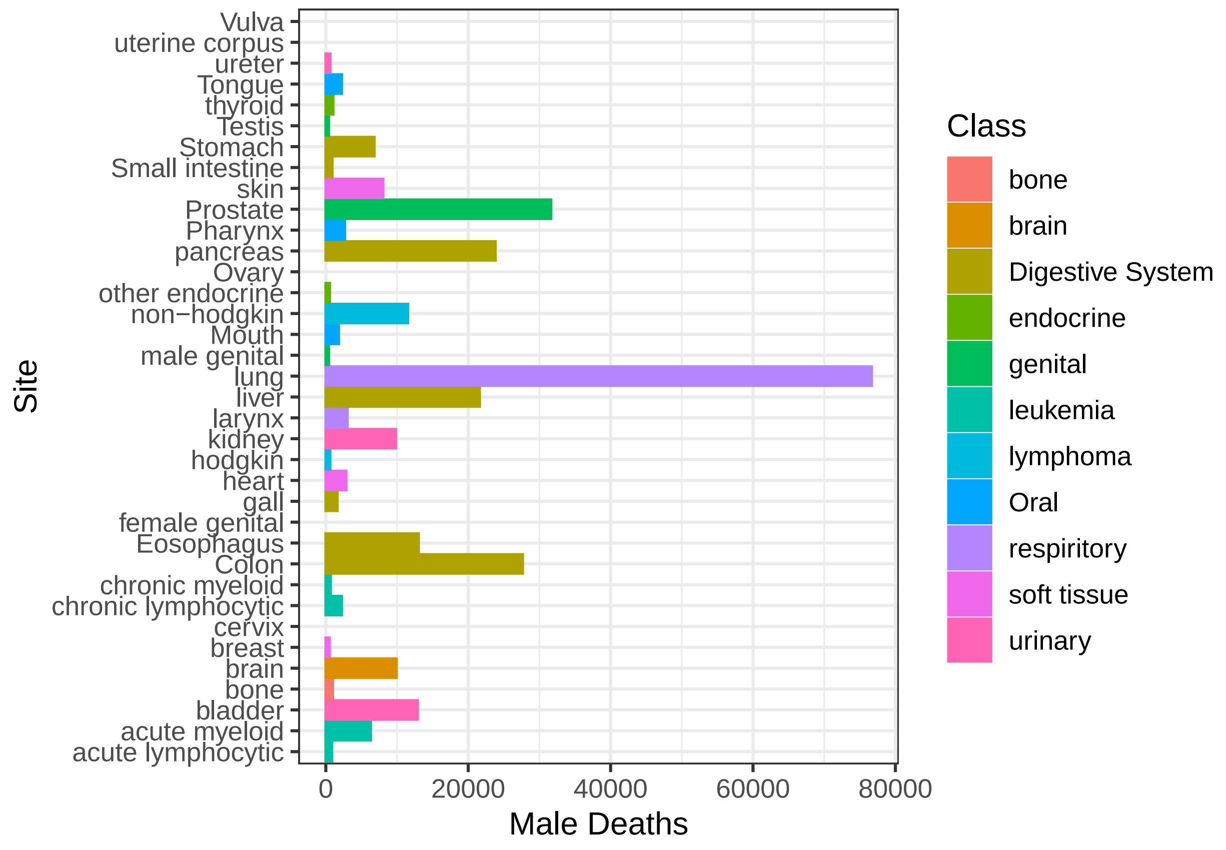


3b) Cancer

In the cancer barplot from exercise 2c, you had to exclude sites because you couldn't show them on the x axis. Use the `coord_flip` transformation to switch the x and y axes so you can remove the `slice()` function which restricted you to 5 sites, and show all of the sites again.

```r
cancer %>%
  arrange(Site) %>%
  ggplot(aes(x = Site, y = `Male Deaths`,
             fill = Class, color = Class)) +
  geom_col() +
  coord_flip()
```

```
## Warning: Removed 5 rows containing missing values (position_stack).
```



3c) Brains

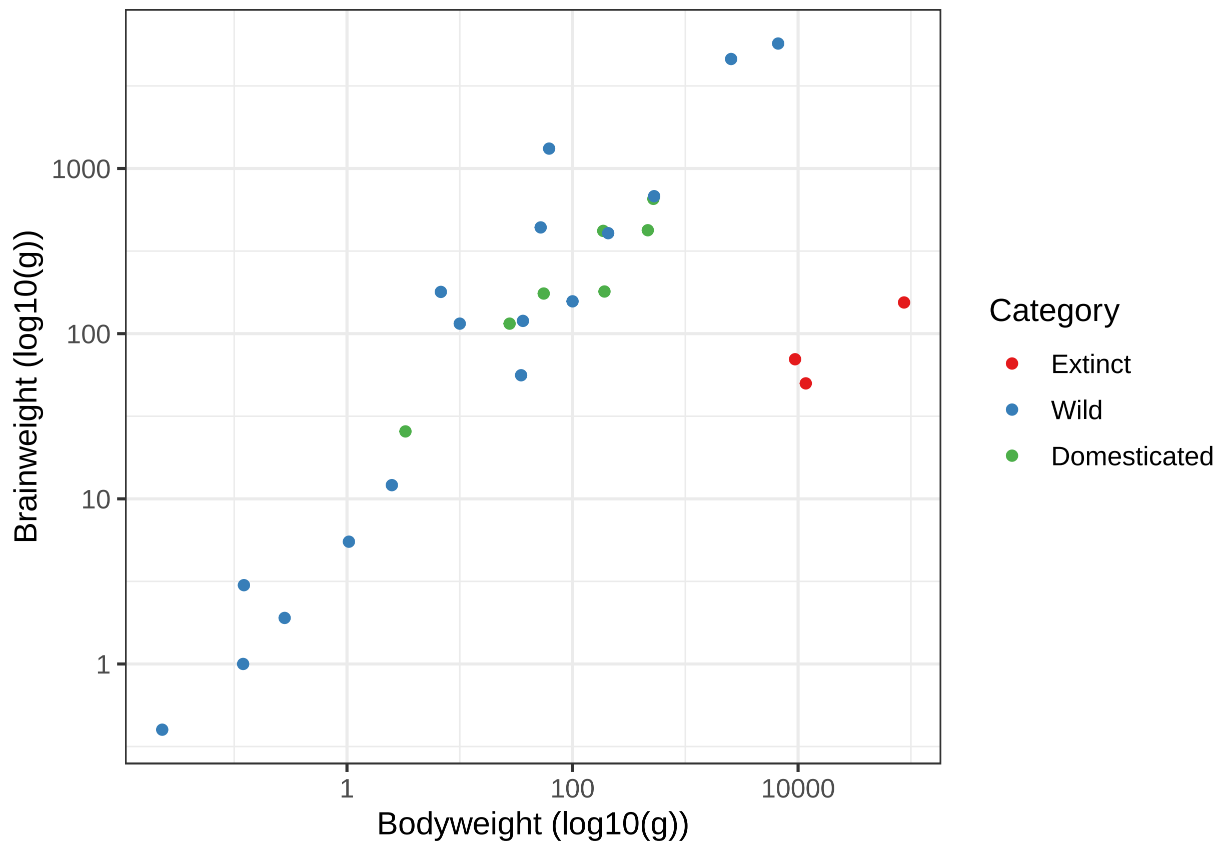
Load the data from brain_bodyweight.tsv

- Plot a scatterplot of the brain against the body.
- Change the axis labels (`xlab()` and `ylab()`) to say Brainweight (g) and Bodyweight (kg) and add a suitable title (`ggtitle()` or `labs`).
- Both brainweight and bodyweight are better displayed on a log scale – try implementing this in one of the ways below:
  - Turn the axes into log scale axes (`scale_x_log10()` and `scale_y_log10()`).
  - Modify the data to be log transformed when creating the aesthetic mapping (pass the column name into the `log10` function when defining the aesthetic mapping in `aes()`).
  - Use `mutate()` to modify the original data before passing it to `ggplot`.
- Color the plot by Category, and change the colors to use the ColorBrewer "Set1" palette (`scale_color_brewer()`).
- Change the ordering of the categories to be "Extinct", "Wild", "Domesticated".

```
brains <- read_delim("brain_bodyweight.txt")
```

```
## Rows: 27 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (2): Species, Category
## dbl (2): body, brain
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

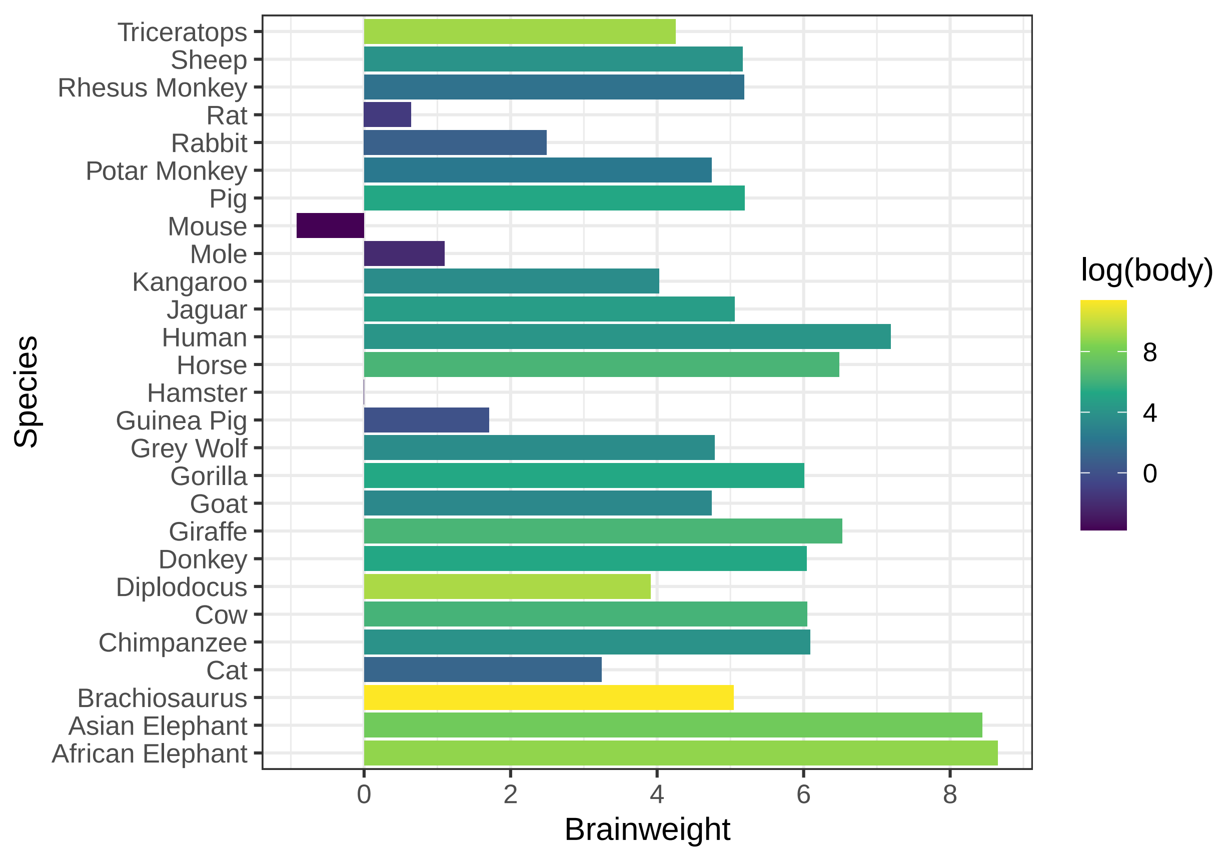
```r
brains %>%
  mutate(Category = factor(Category, levels = c("Extinct", "Wild", "Domesticated"))) %>%
  ggplot(aes(x = body, y = brain, color = Category)) +
  geom_point() +
  xlab("Bodyweight (log10(g))") +
  ylab("Brainweight (log10(g))") +
  scale_x_log10() +
  scale_y_log10() +
  scale_color_brewer(palette = "Set1")
```



3d) If you have time…

- Create a barplot of the brainweight of all species, colored by their bodyweight.
  - Use a custom color scheme for the coloring of the bars.
  - You will again need to use a log scale for the brain and bodyweight.

```
brains %>%
  ggplot(aes(x = Species, y = log(brain), fill = log(body))) +
  geom_col() +
  ylab("Brainweight") +
  coord_flip() +
  scale_fill_continuous(type = "viridis")
```


Triceratops
Sheep
Rhesus Monkey
Rat
Rabbit
Potar Monkey
Pig
Mouse
Mole
Kangaroo
Jaguar
Human
Horse
Hamster
Guinea Pig
Grey Wolf
Gorilla
Goat
Giraffe
Donkey
Diplodocus
Cow
Chimpanzee
Cat
Brachiosaurus
Asian Elephant
African Elephant
Species
Brainweight
log(body)