Tidy Data and Visualisation with R Exercises

Before you start, make sure you've read the document that describes the zebrafish dataset we're using in this practical. And make sure you've put the required file (`Amp.counts.tsv`) in your home directory.

Load the example dataset (`Amp.counts.tsv`) using `read_tsv`.

```
library(tidyverse)
results <- read_tsv('Amp.counts.tsv',
                    col_types = cols(Chr = col_character(),
                                     Strand = col_character()))
```

Tidy Data

1. Use `filter` to find out how many genes have an adjusted p-value less than 0.05.

```
filter(results, adjp < 0.05) %>%
  nrow()
```

```
## [1] 381
```

2. Find out which gene has the smallest p-value by sorting the data using `arrange`.

```
arrange(results, adjp) %>%
  select(., Gene, Name, adjp) %>%
  head(1)
```

```
## # A tibble: 1 x 3
##   Gene               Name      adjp
##   <chr>              <chr>    <dbl>
## 1 ENSDARG00000031683 fosab 1.05e-39
```

3. Make a new column in the data that is -log10 of the adjusted p-value column. You can use the `log10()` function to calculate this.

```
results <- mutate(results,
       log10p = -log10(adjp))

select(results, Gene, Name, adjp, log10p) %>%
  arrange(adjp) %>%
  head(5)
```

```
## # A tibble: 5 x 4
##   Gene               Name       adjp log10p
##   <chr>              <chr>     <dbl>  <dbl>
## 1 ENSDARG00000031683 fosab  1.05e-39   39.0
## 2 ENSDARG00000034503 per2   2.92e-28   27.5
## 3 ENSDARG00000104773 junbb  3.24e-24   23.5
## 4 ENSDARG00000055752 npas4a 1.42e-19   18.8
## 5 ENSDARG00000087440 ponzr4 7.20e-19   18.1
```

4. Make a new data.frame that contains the Gene, Name and all the normalised count columns.

```
normalised_counts <-
  select(results, Gene, Name, contains('normalised count'))
```

5. Make the new data.frame data tidy using `pivot_longer`.

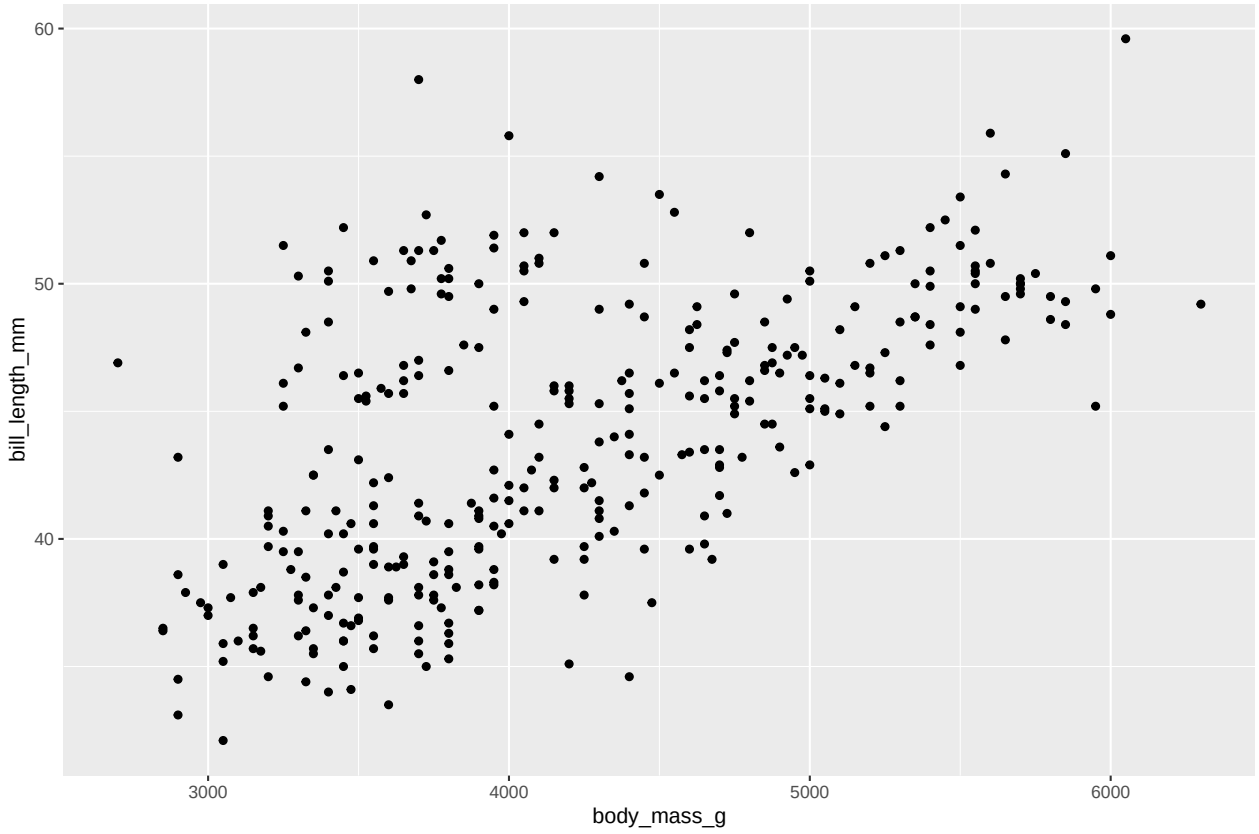
```
normalised_counts_long <-
  pivot_longer(normalised_counts, c(-Gene, -Name),
               names_to = 'sample',
               values_to = 'normalised count') %>%
  # this removes the " normalised count" suffix from
  # each count column name to leave just the sample name
  mutate(sample = sub(" normalised count", "", sample))
```

Plotting

Load the penguins dataset with `library(palmerpenguins)`.

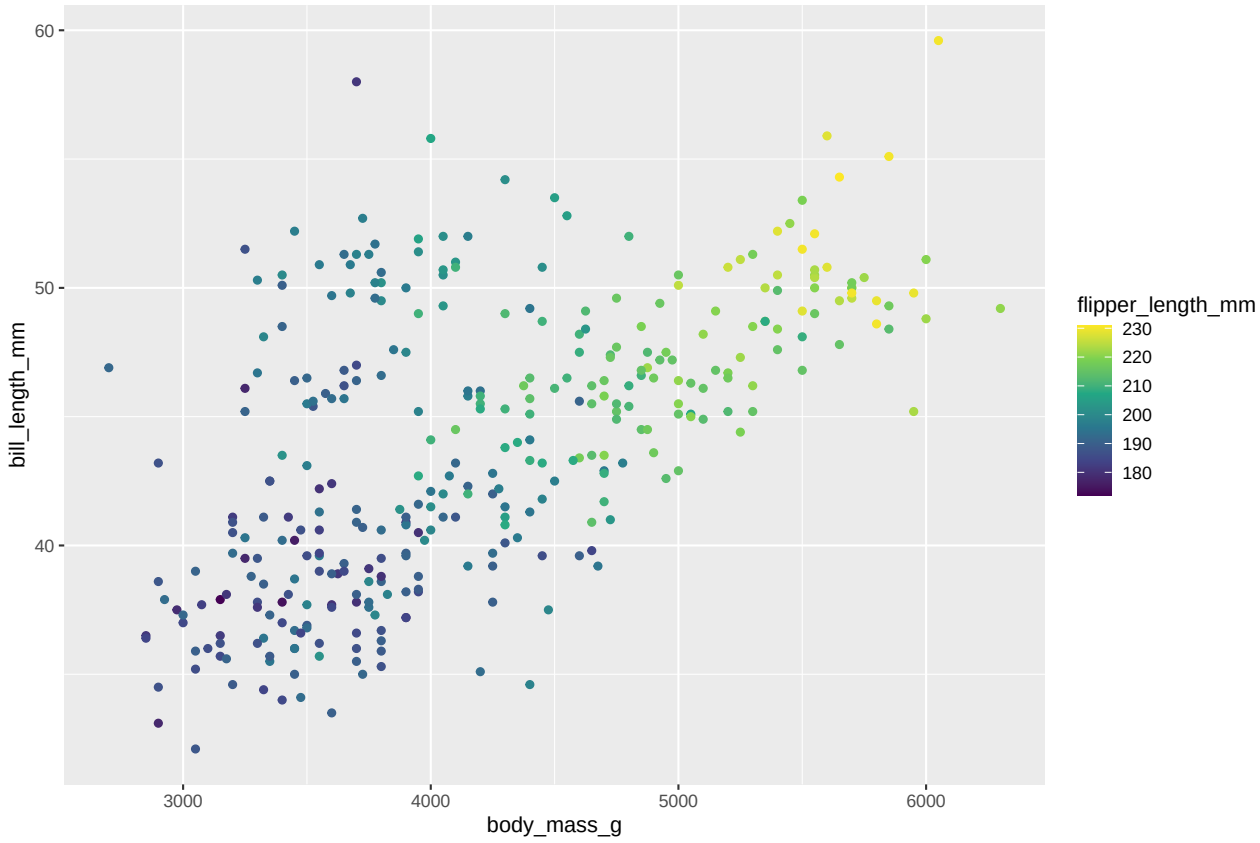
1. Using the `penguins` dataset, make a plot of body mass (body_mass_g) against bill length (bill_length_mm).

```
library(palmerpenguins)
ggplot(data = penguins) +
  geom_point(aes(x = body_mass_g, y = bill_length_mm))
```



2. Now colour the points by flipper length (flipper_length_mm) and use the `viridis` colour scale using `scale_colour_viridis_c`.

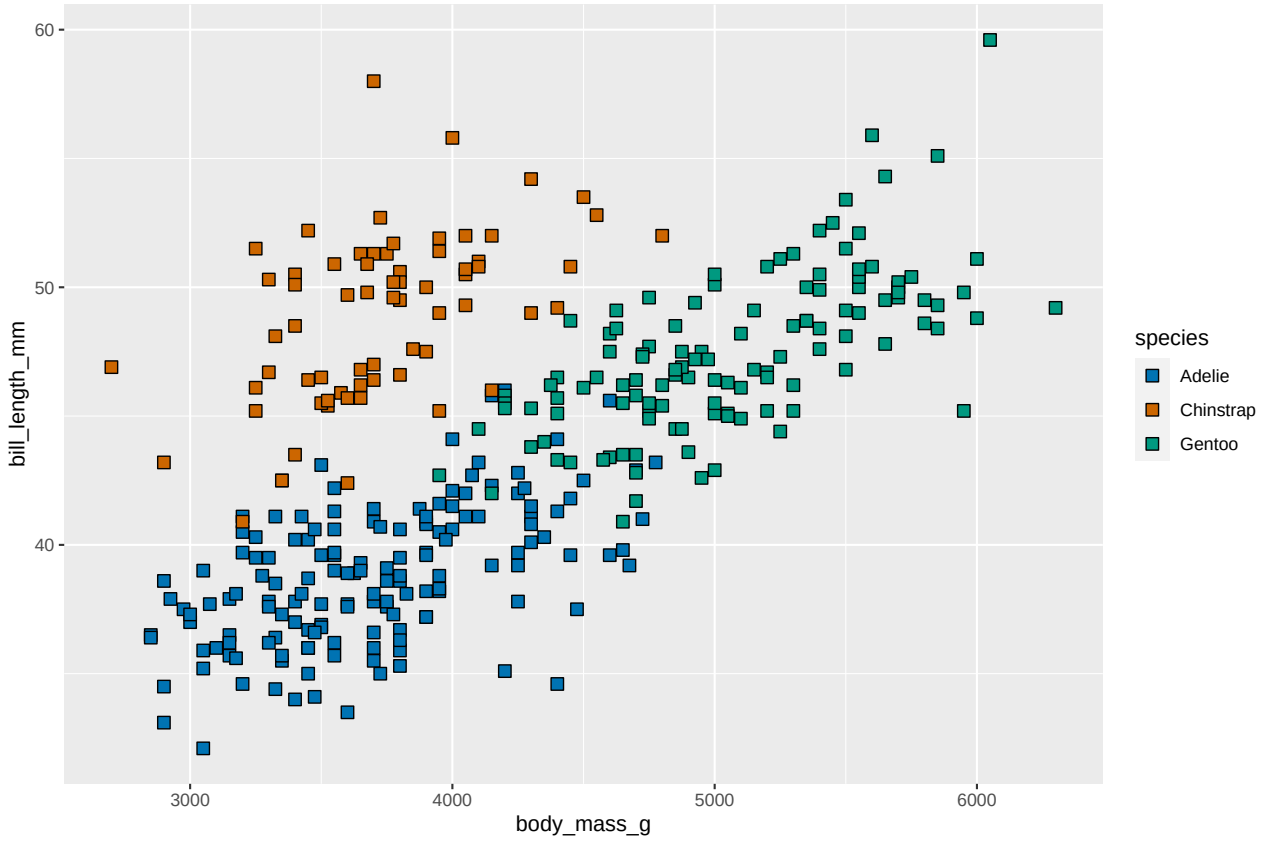
```
ggplot(data = penguins) +
  geom_point(aes(x = body_mass_g, y = bill_length_mm,
                 colour = flipper_length_mm)) +
  scale_colour_viridis_c()
```



3. Change the shape of the points to a hollow shape (one of 21-25).
   Make species the fill colour, and pick 3 colours to use with `scale_fill_manual`.

```
colour_blind_palette <-
  c("#0073B3", "#CC6600", "#009980", "#F2E640",
    "#59B3E6", "#CC99B3", "#000000", "#E69900")

ggplot(data = penguins) +
  geom_point(aes(x = body_mass_g, y = bill_length_mm,
                 fill = species),
             shape = 22, size = 3) +
  scale_fill_manual(values = colour_blind_palette)
```



Alternatively you could specify the fill colours as names

```
ggplot(data = penguins) +
  geom_point(aes(x = body_mass_g, y = bill_length_mm,
                 fill = species),
             shape = 22, size = 3) +
  scale_fill_manual(values = c('firebrick2', 'steelblue3',
                               'goldenrod1'))
```