Week5_key.R

willav

2021-09-29

```
# Data visualization with ggplot
# Willa & Elena
# 9/28/21
library(tidyverse)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## Registered S3 method overwritten by 'rvest':
##   method            from
##   read_xml.response xml2
```

```
## -- Attaching packages --------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## Warning: package 'tibble' was built under R version 3.6.2
```

```
## Warning: package 'tidyr' was built under R version 3.6.2
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
## -- Conflicts ------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidylog)
```

```
## Warning: package 'tidylog' was built under R version 3.6.2
```

```
##
## Attaching package: 'tidylog'
```

```
## The following objects are masked from 'package:dplyr':
##
##     add_count, add_tally, anti_join, count, distinct,
##     distinct_all, distinct_at, distinct_if, filter, filter_all,
##     filter_at, filter_if, full_join, group_by, group_by_all,
##     group_by_at, group_by_if, inner_join, left_join, mutate,
##     mutate_all, mutate_at, mutate_if, relocate, rename,
##     rename_all, rename_at, rename_if, rename_with, right_join,
##     sample_frac, sample_n, select, select_all, select_at,
##     select_if, semi_join, slice, slice_head, slice_max, slice_min,
##     slice_sample, slice_tail, summarise, summarise_all,
##     summarise_at, summarise_if, summarize, summarize_all,
##     summarize_at, summarize_if, tally, top_frac, top_n, transmute,
```

```
##     transmute_all, transmute_at, transmute_if, ungroup

## The following objects are masked from 'package:tidyr':
##
##     drop_na, fill, gather, pivot_longer, pivot_wider, replace_na,
##     spread, uncount

## The following object is masked from 'package:stats':
##
##     filter
```

```
################ #### Warm-up #### ##################
# From the practice questions last week:

# 1. Load in world-happiness_2020.csv (what we worked with last week)

happiness <- read.csv("../data/world-happiness_2020.csv")

# 2. Pick two variables and summarize them in a new data frame. Get the mean,
# median, and sd.

summary1 <- happiness %>%
  summarize(ladder.mean = mean(Ladder_score, na.rm = TRUE),
            ladder.med = median(Ladder_score, na.rm = TRUE),
            ladder.sd = sd(Ladder_score, na.rm = TRUE),
            gen.mean = mean(Generosity, na.rm = TRUE),
            gen.med = median(Generosity, na.rm = TRUE),
            gen.sd = sd(Generosity, na.rm = TRUE))
```

```
## summarize: now one row and 6 columns, ungrouped
```

```
# mean(c(1, 2, 3, NA, 5), na.rm = TRUE)

# 3. In another new data frame, get the mean, median, sd for these variables by
# region.

summary2 <- happiness %>%
  group_by(Regional_indicator) %>%
  summarize(ladder.mean = mean(Ladder_score, na.rm = TRUE),
            ladder.med = median(Ladder_score, na.rm = TRUE),
            ladder.sd = sd(Ladder_score, na.rm = TRUE),
            gen.mean = mean(Generosity, na.rm = TRUE),
            gen.med = median(Generosity, na.rm = TRUE),
            gen.sd = sd(Generosity, na.rm = TRUE)) %>%
  ungroup()
```

```
## group_by: one grouping variable (Regional_indicator)

## summarize: now 10 rows and 7 columns, ungrouped

## ungroup: no grouping variables
```

```
# 4. In a third new data frame, get the mean, median, sd for these variables by
# region and population category.

summary3 <- happiness %>%
  group_by(Regional_indicator, country_size) %>%
  summarize(ladder.mean = mean(Ladder_score, na.rm = TRUE),
```

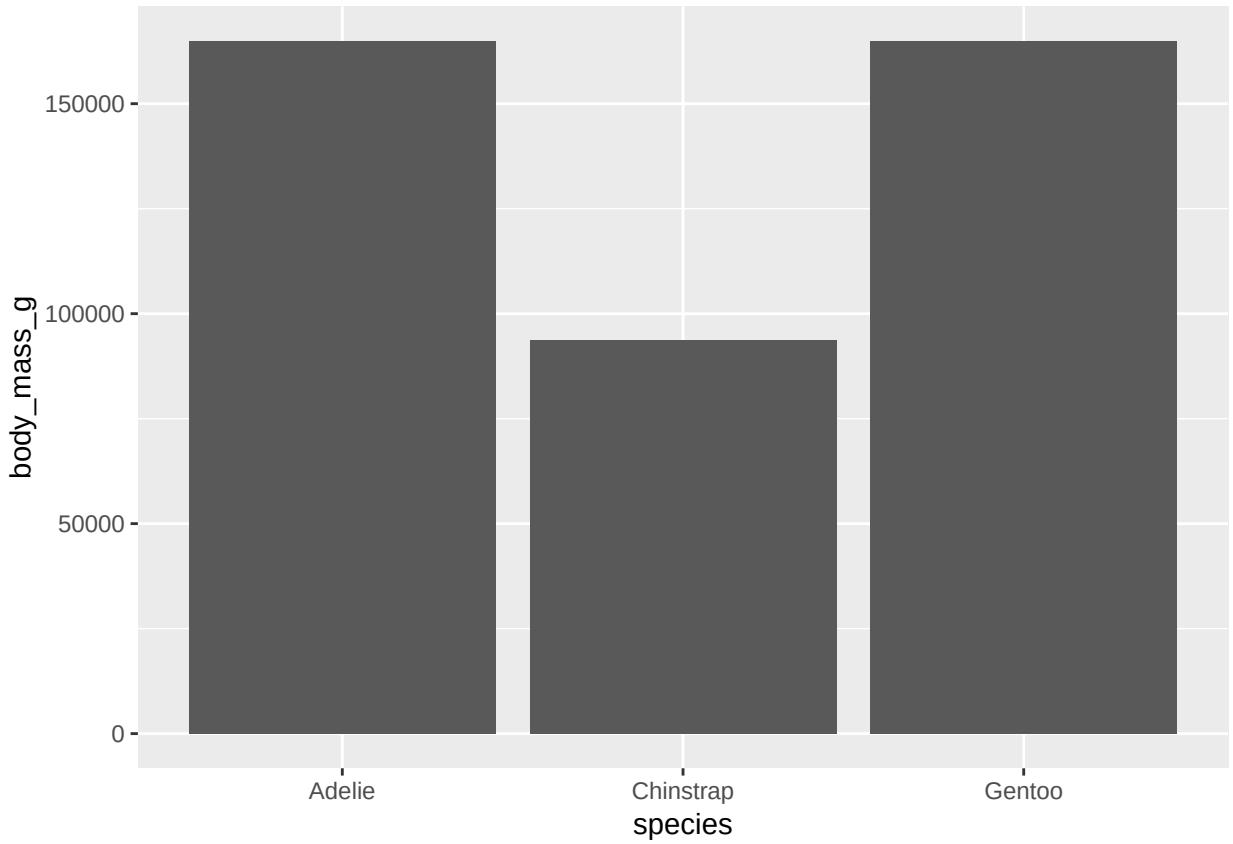
```
            ladder.med = median(Ladder_score, na.rm = TRUE),
            ladder.sd = sd(Ladder_score, na.rm = TRUE),
            gen.mean = mean(Generosity, na.rm = TRUE),
            gen.med = median(Generosity, na.rm = TRUE),
            gen.sd = sd(Generosity, na.rm = TRUE)) %>%
  ungroup()
```

```
## group_by: 2 grouping variables (Regional_indicator, country_size)
```

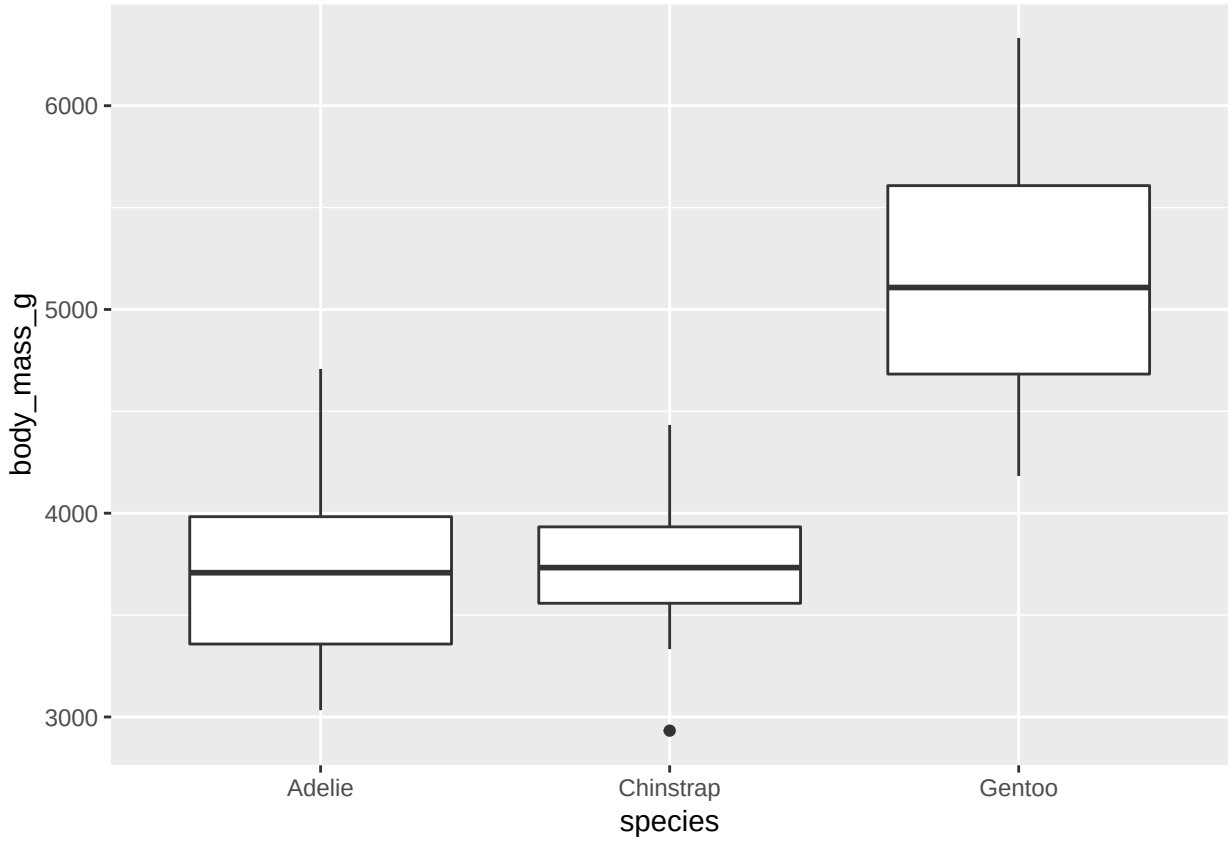
```
## summarize: now 22 rows and 8 columns, one group variable remaining (Regional_indicator)
```

```
## ungroup: no grouping variables
```

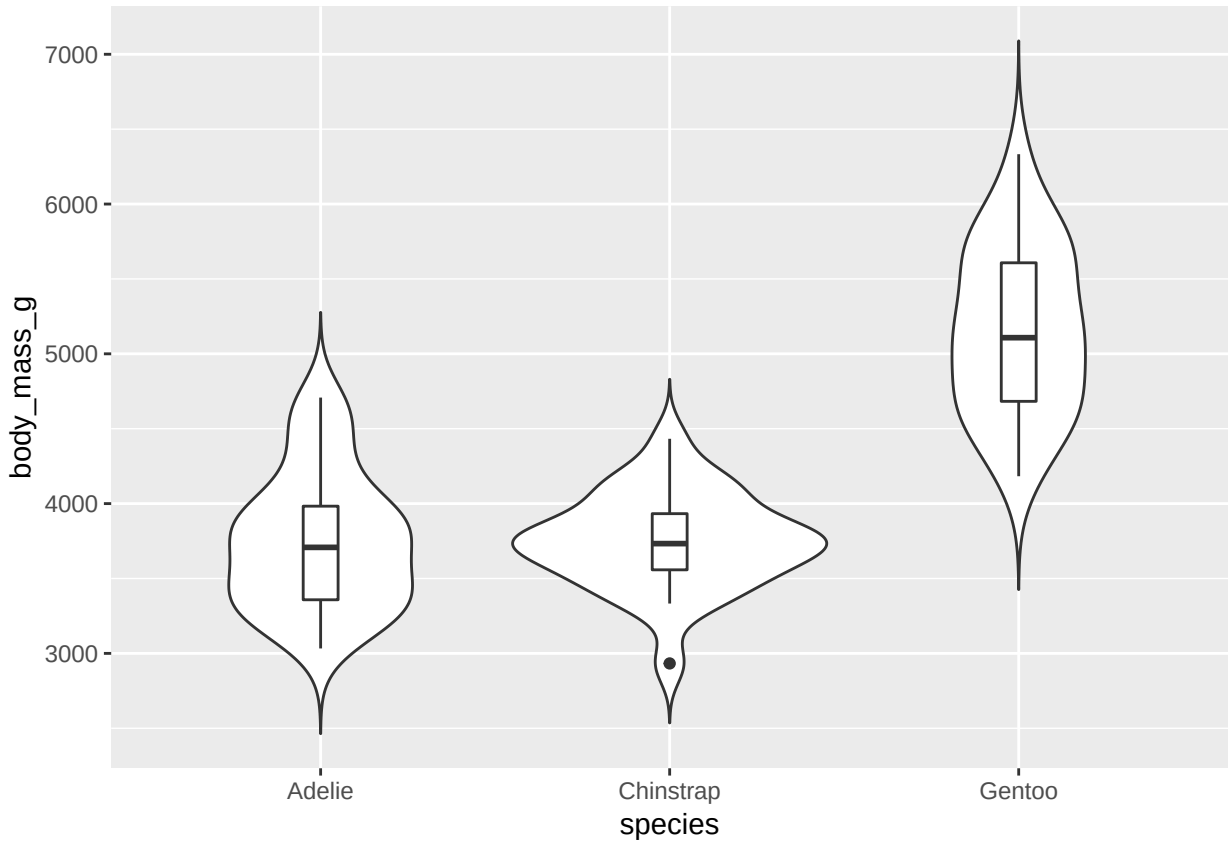
```
################ #### Data Viz demo #### ##################

#### Read in our data ####

penguins <- read.csv('../data/penguins_clean.csv')

#### Explore our data with some simple plots ####

## 1. Look at body mass by species

# Set up our ggplot and define our variables.
# aes = asthetic mapping. This tells ggplot how to map your variables to the plot.

ggplot(penguins, aes(x = species, y = body_mass_g)) +

# represent the data to a column object.
# geom = geometric object. It tells ggplot the geometric
# representation to use for your data.
  geom_col()
```




```
# In this case, bar charts aren't the best way to look at data
# because it doesn't tell us much about individual data points
#  or the distribution of data.

## Let's change the geometric representation of our data and use a boxplot.

ggplot(penguins, aes(x = species, y = body_mass_g)) +

  # represent the data to a boxplot object.
  geom_boxplot()
```

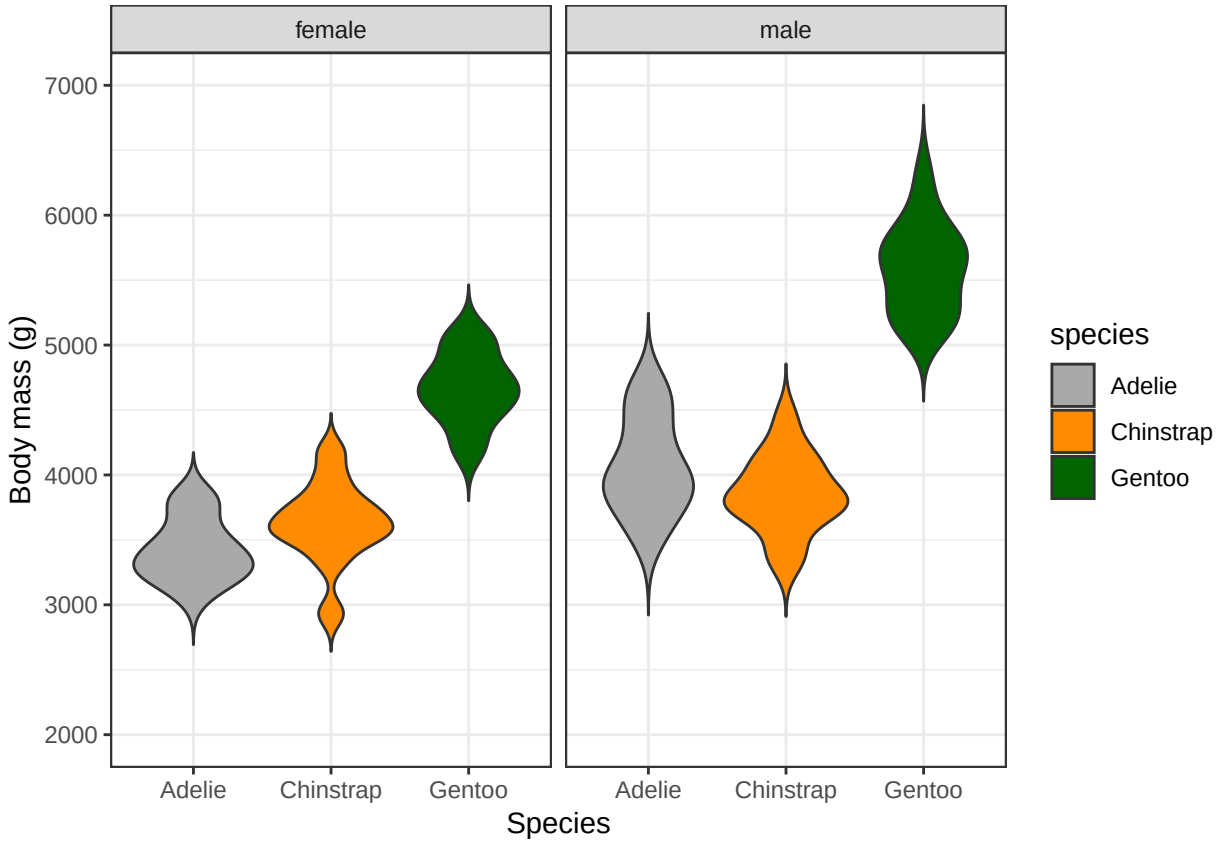



```
# This gives us some more descriptive stats about the data
# but we still don't have a good feel for what the distribution
# of the data points look like.


## Lets use a violin plot to visualize the data.
## Violin plots are a great way to visualize the distribution of your data.

ggplot(penguins, aes(x = species, y = body_mass_g)) +

  # Map the data to a violin plot. Options stop the ends from being trimmed.
  geom_violin(trim = FALSE) +

# Add a boxplot on top. Options make the boxplot small.
  geom_boxplot(width = 0.1)
```



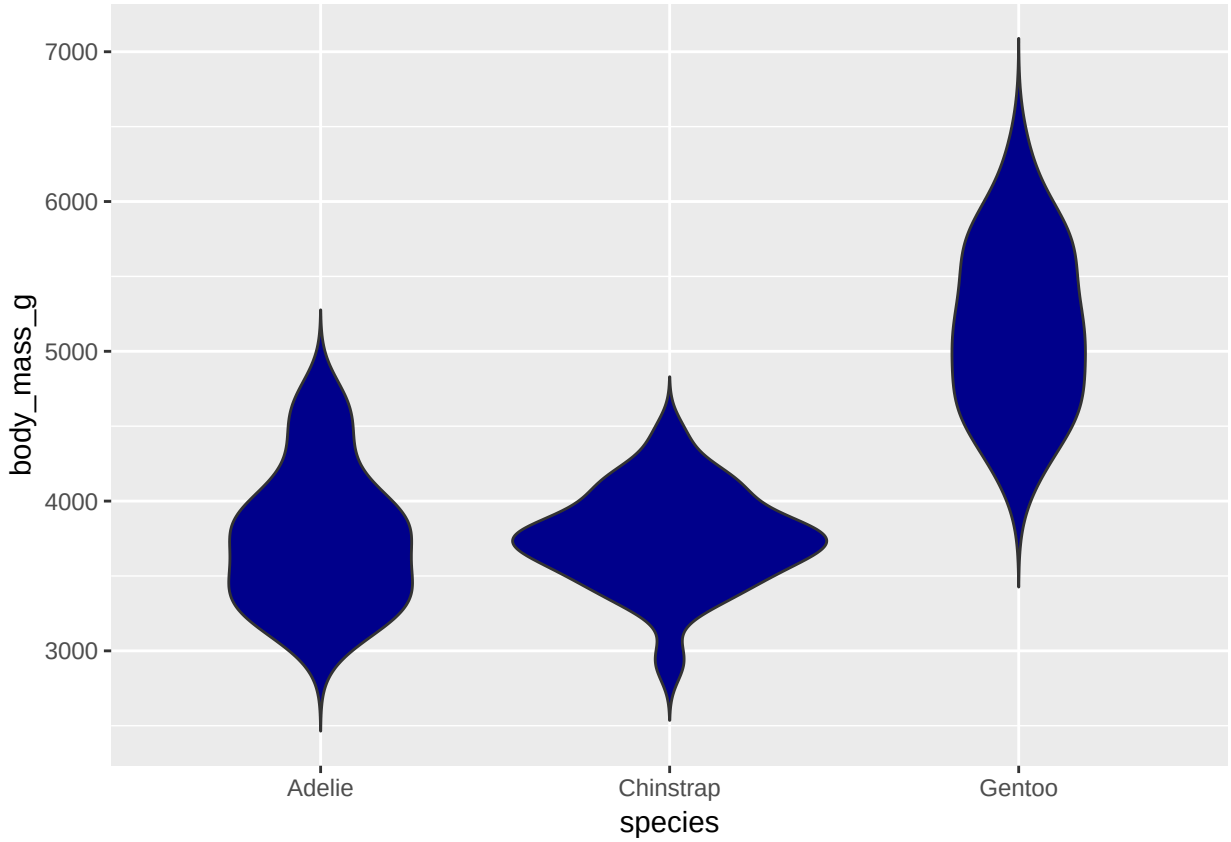

```
# Notice that the species are ordered alphabetically.
# We could change this by creating a factor.
# (eg. factor(species, levels = c("Chinstrap", "Adelie", "Gentoo"))


## Now lets see if there are differences between species.
# We can use "fill" as an additional aesthetic mapping.
# Fill = filled in color
# Color = colored outline.
# For some shapes (eg. lines, points) they only have a color attribute.
#  For other shapes (eg. boxes) they have both color and fill.

ggplot(penguins, aes(x = species, y = body_mass_g, fill = species)) +
  geom_violin(trim = FALSE) +

  # We can further split the plot by sex
  facet_wrap(~ sex) +

# Now that we are happy with our plot choice, lets customize it further

  ## change axis labels
  xlab( "Species") +
  ylab( "Body mass (g)") +

  ## change the y axis scale to start at 2 and end at 7000
  ylim(2000, 7000) +
```

```
  ## Change the color scheme.
  # Color scales and palettes allow you to change the color scheme
  # for  mapping variable, in this case, species.
  # There are a few ways to do this. Choose ONE

  # i.Some color names are built in
  scale_fill_manual(values = c("dark gray", "dark orange", " dark      green")) +
  # or if you were using "color" instead of fill
 # scale_color_manual(values = c("dark gray", "dark orange", " dark     green"))

  ##ii.We can give it hex values
  # scale_fill_manual(values=c ("#999999", "#E69F00"))

  ## iii. We can use color palettes
  # scale_fill_brewer(palette ="Dark2")

## change the overall theme
# Theme = overall look of the plot. Including, grid lines, font, font size, legend etc.

  # Some themes are built in and you can apply them in one command
  theme_bw()
```

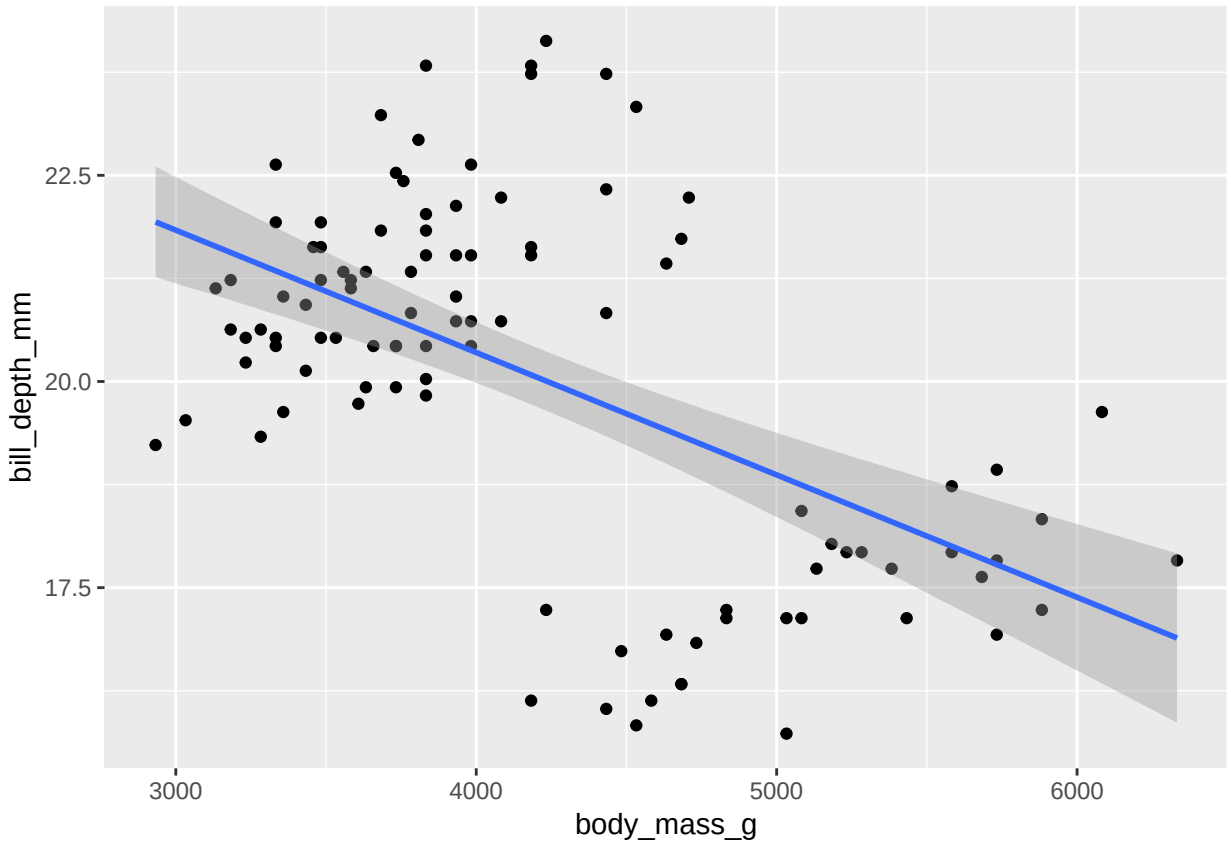



```
  # OR
  # theme_classic()

# You can also create your own custom theme and change any features of the plot using theme()
```

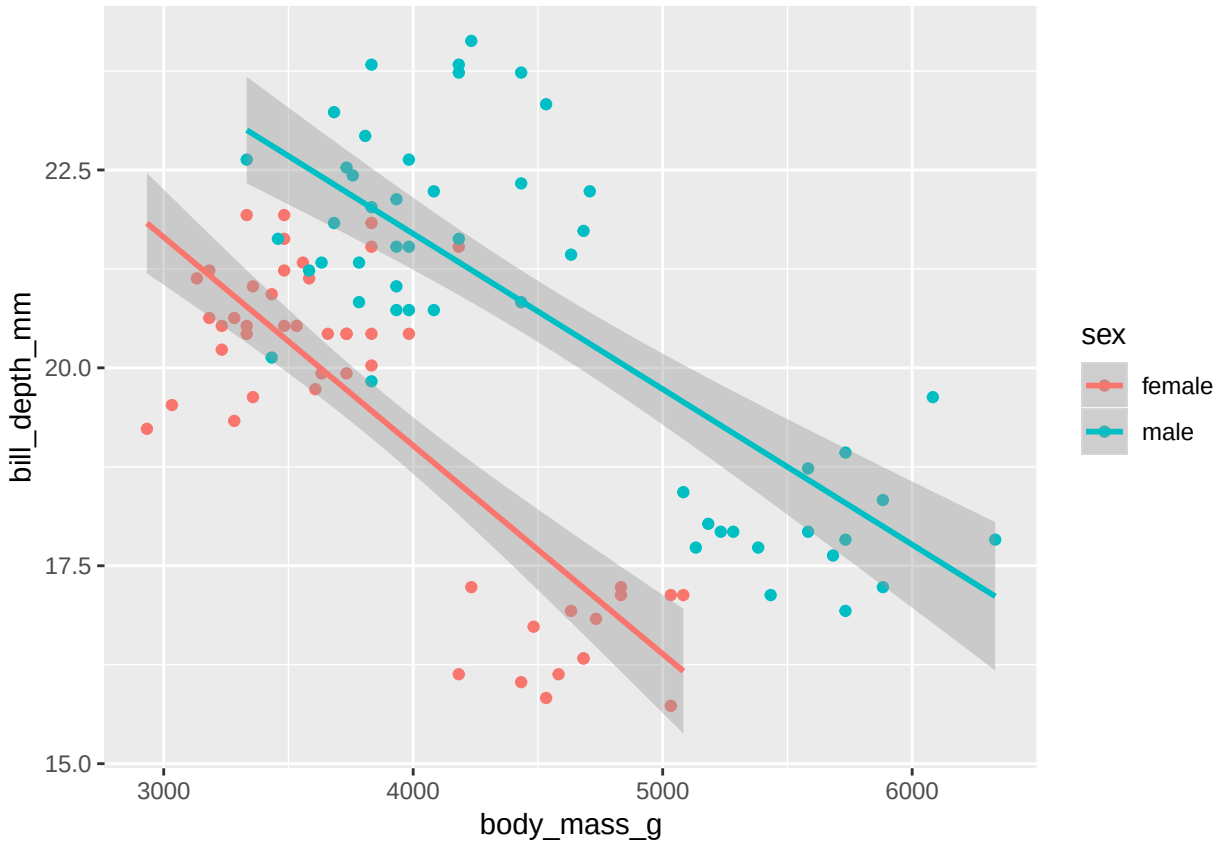
```r
# NOTE. Above we used color to represent variables. We can also just change the color of individual obj
  # eg.
ggplot(penguins, aes(x = species, y = body_mass_g)) +
  geom_violin(trim = FALSE, fill = "dark blue")
```



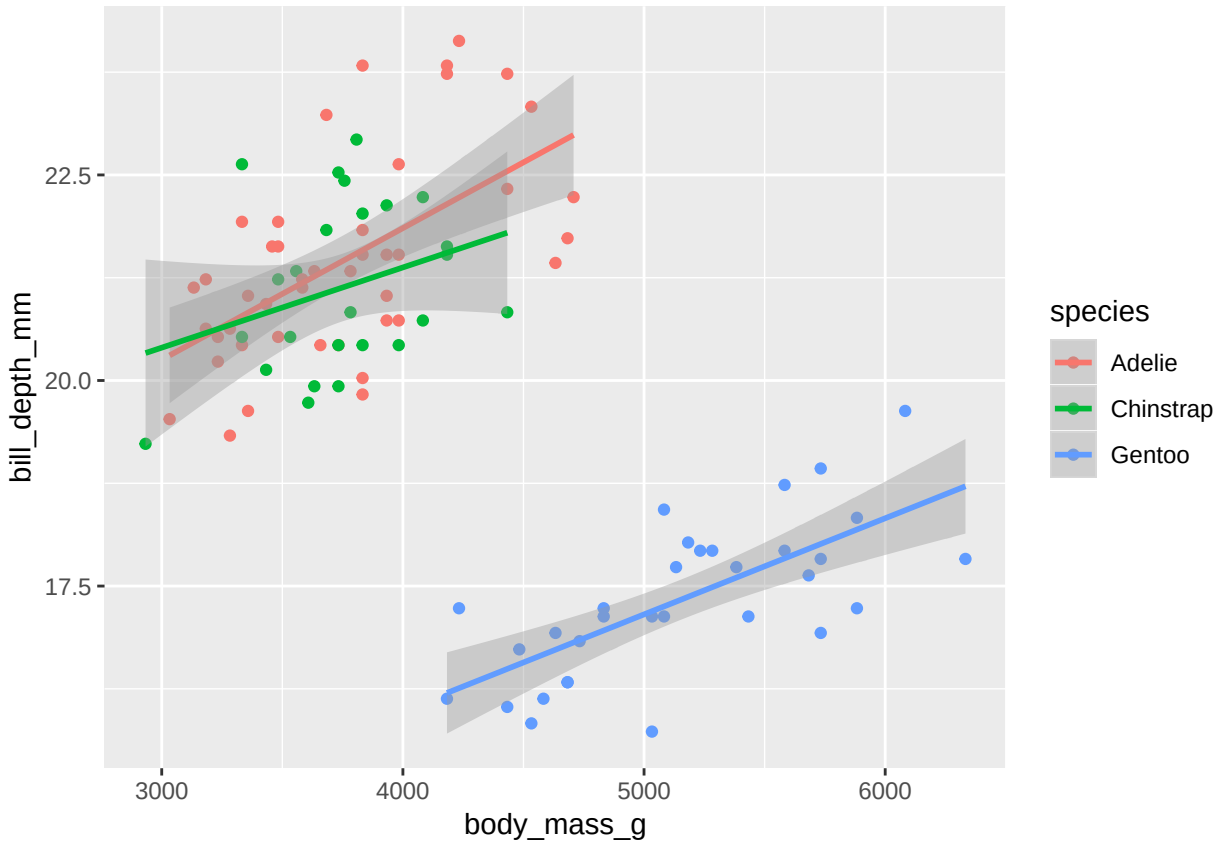

```r
## 6. Exploring continuous variables with scatter plots.  We can explore the relationships between two

# create a ggplot object with body mass and bill depth

# Set up our ggplot
ggplot(penguins, aes(body_mass_g, bill_depth_mm)) +

  # Represent the data as points
  geom_point() +

  # Add a  line. Option method = 'lm' gives you a linear regression line.
  geom_smooth(method = 'lm')
```



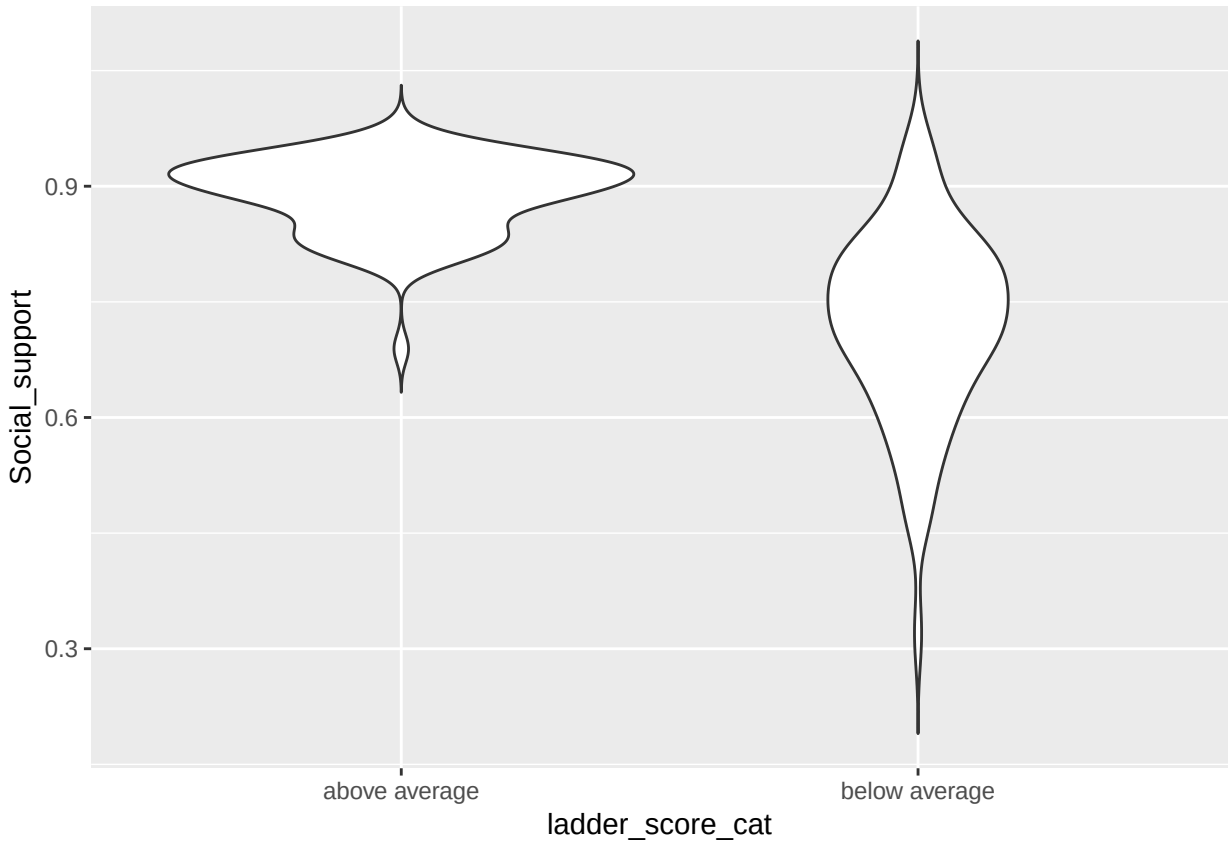

```
# There are two clear groups here. Given what we saw already about sex differences let's see if they gr
# Note: we will use "color" instead of "fill" here
# because geom_point() only has a "color" attribute.
ggplot(penguins, aes(body_mass_g, bill_depth_mm, color = sex)) +
  geom_point() +
  geom_smooth(method = 'lm')
```



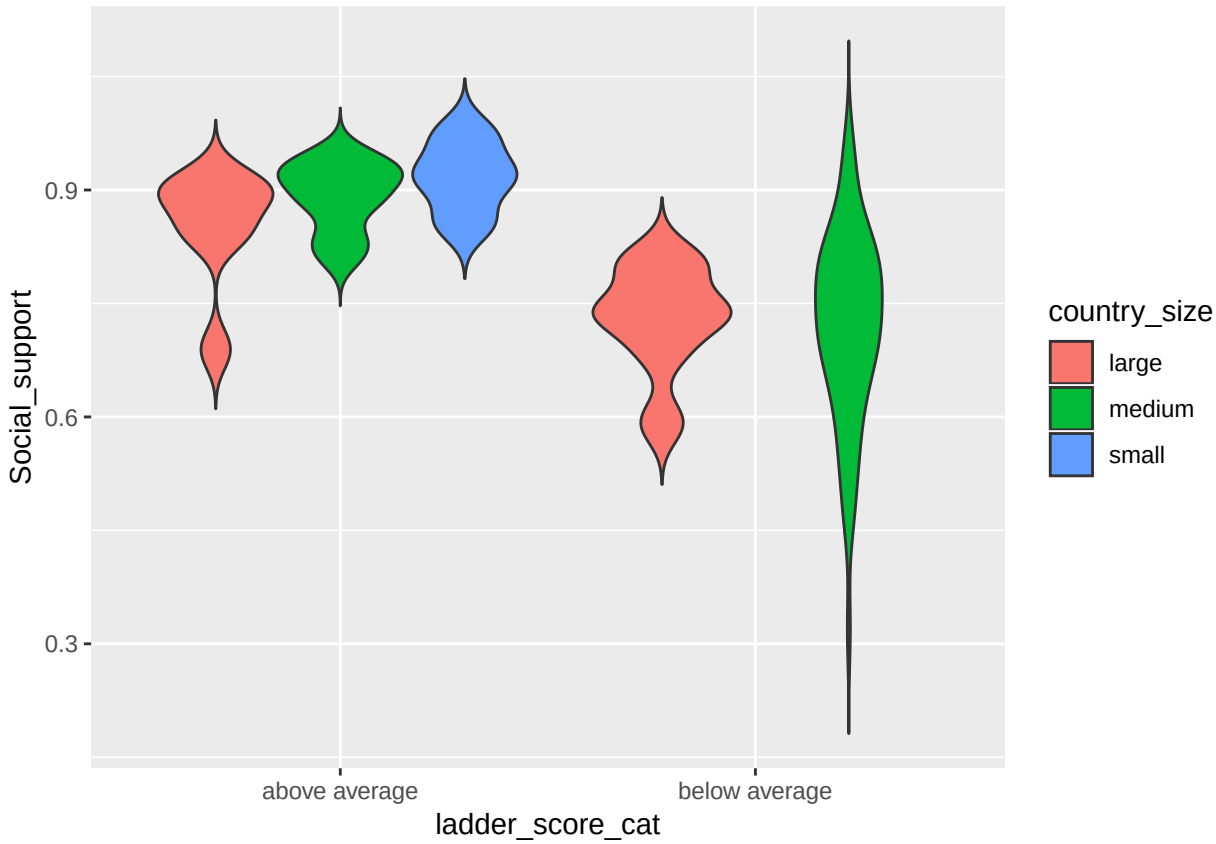

```
# There are sex differences but these don't explain the grouping.
# Lets try grouping by species instead.
ggplot(penguins, aes(body_mass_g, bill_depth_mm, color = species)) +
  geom_point() +
  geom_smooth(method = 'lm')
```



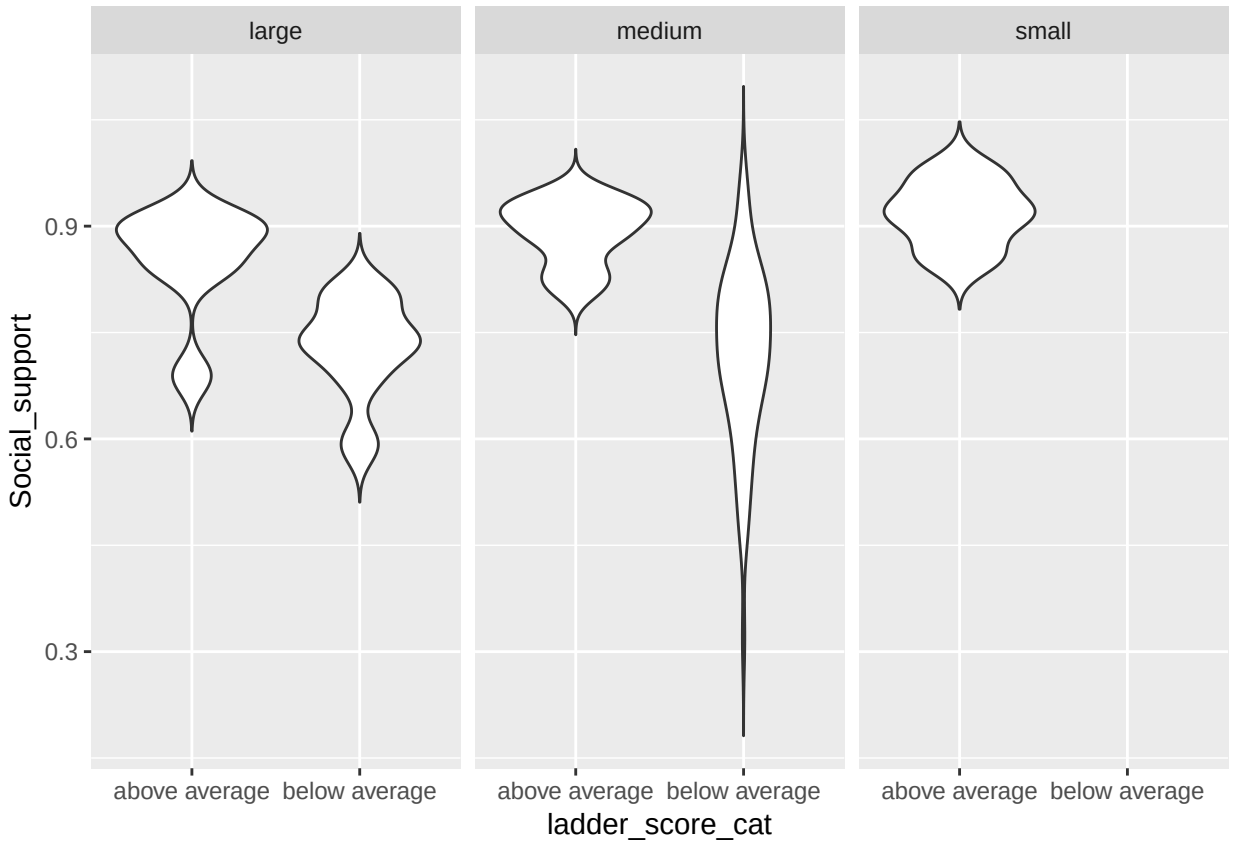

```
# We see that this split in the data is due to species and
# by accounting for species the relationship between bill depth
# and body mass becomes postive whereas before it was negative.
# This shows the value of looking at your data before running analyses.

############################ ##### Practice Key ##### ########################################

## 1. Load in the Happiness data

happiness <- read.csv('../data/world-happiness_2020.csv')

## 2. Create a violin plot relating happiness levels to social support. Use ladder_score_cat as the x-v

# create ggplot object and map variables
ggplot(happiness, aes(ladder_score_cat, Social_support)) +
  # violin plot
  geom_violin(trim = FALSE)
```




```
## 3. Is this relationship between social support and happiness the same for all country sizes? (hint:

ggplot(happiness, aes(ladder_score_cat, Social_support, fill = country_size)) +
  # violin plot
  geom_violin(trim = FALSE)
```




```
# Another option is to split by country size.
ggplot(happiness, aes(ladder_score_cat, Social_support)) +
  # violin plot
  geom_violin(trim = FALSE) +

  # split plot by country size
  facet_wrap(~ country_size)
```

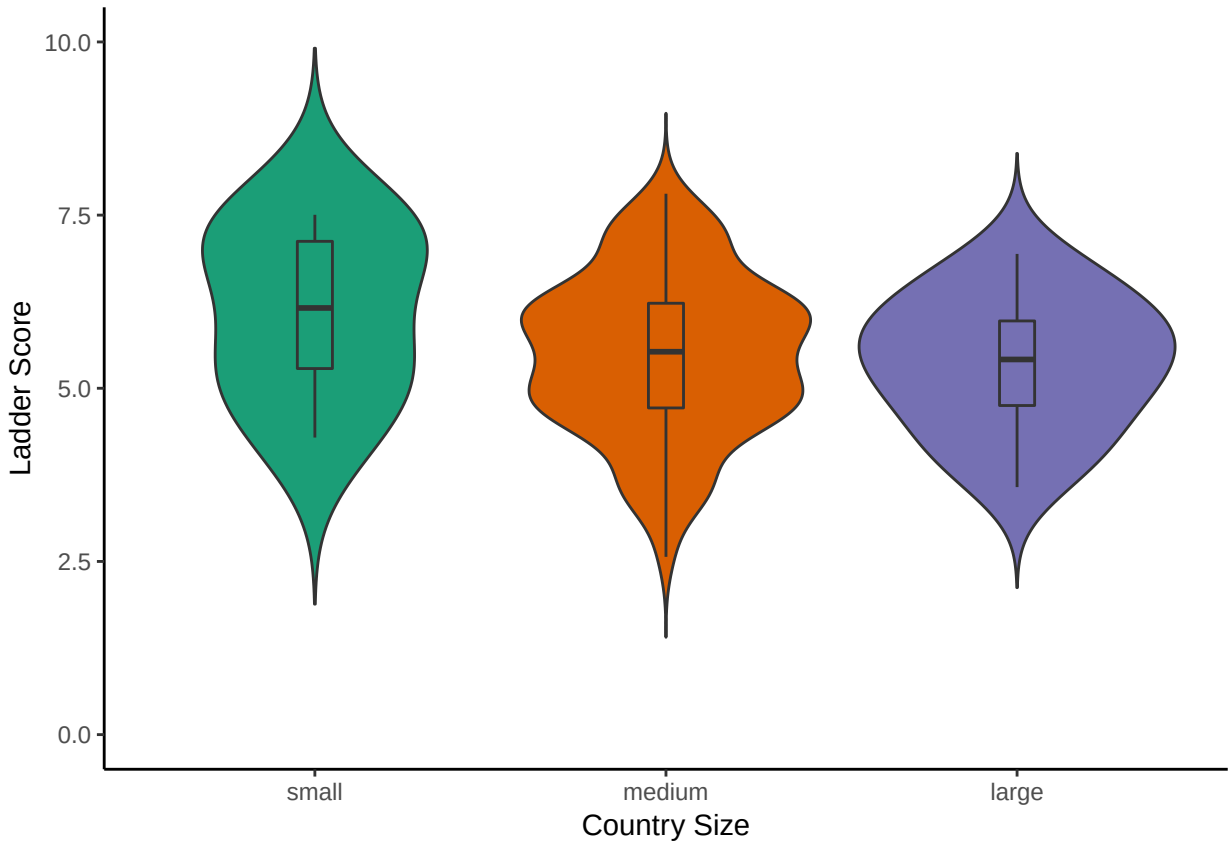



```
## 4. Recreate a violin plot of happiness by country size.
# make country size a factor and re-order the levels.
happiness <- happiness %>%
  mutate(country_size = factor(country_size,
                               levels = c("small", "medium", "large")))
```

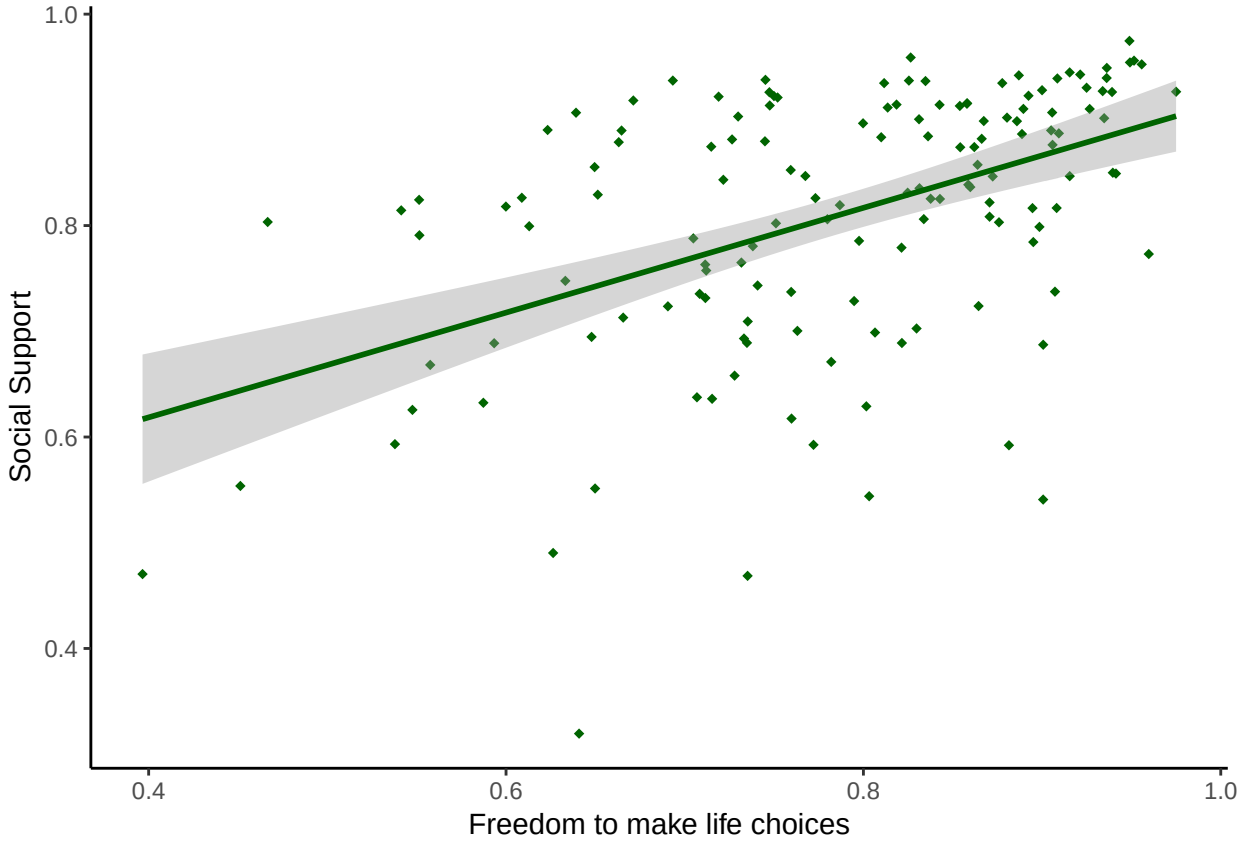
```
## mutate: changed 0 values (0%) of 'country_size' (0 new NA)
```

```
# set up variables to plot
plot = ggplot(happiness, aes(country_size, Ladder_score,
                      fill = country_size )) +

  # add a violin
  geom_violin(trim = FALSE) +

  # add a boxplot
  geom_boxplot(width = 0.1) +

  # Change axis labels

  xlab("Country Size") +
  ylab("Ladder Score") +

  # Change background color

  theme_classic()

## 5. customize the plot
```

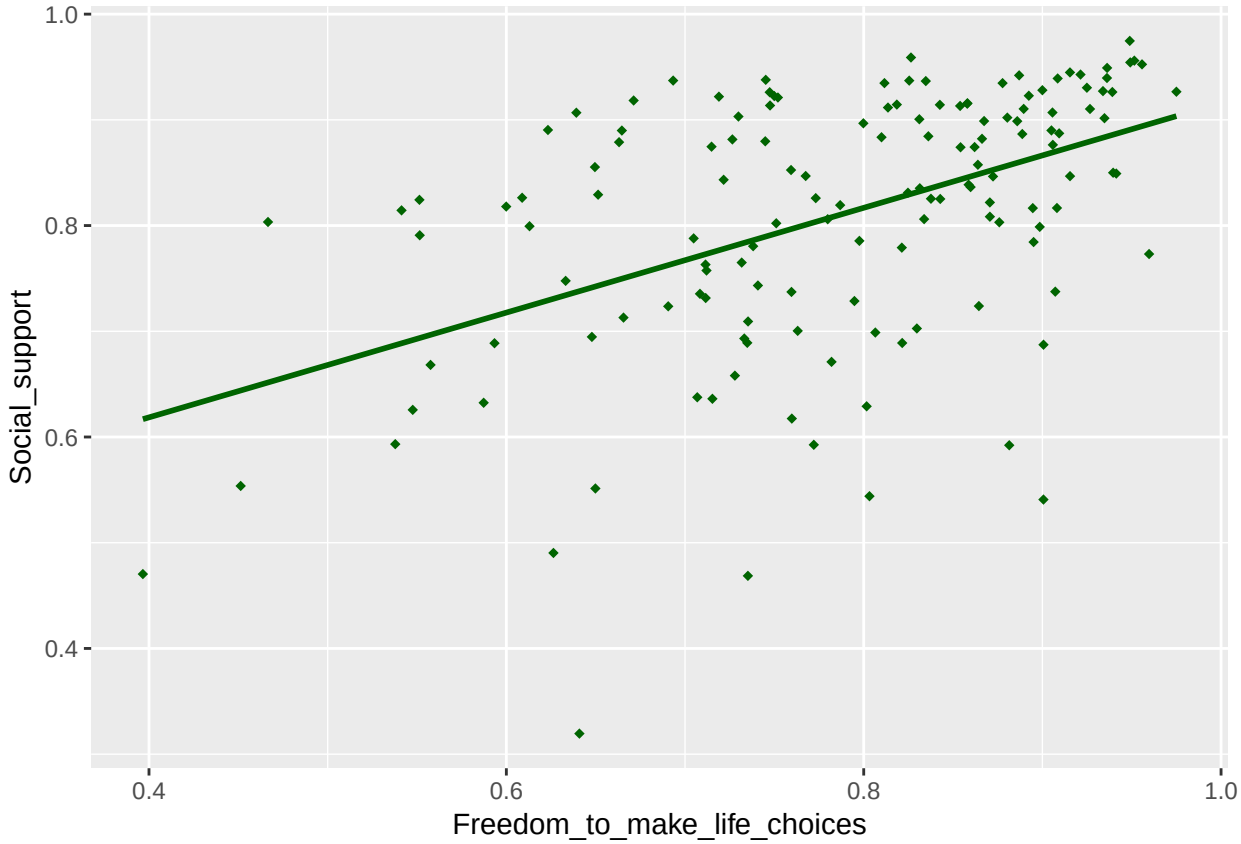
```
plot +
  # change y-axis limits
  ylim(0 , 10) +

  # Change the color scheme to a color of your choosing
  scale_fill_brewer(palette ="Dark2") +

  # remove the legend
  theme(legend.position = 'none')
```




```
## 6. Recreate the scatter plot
ggplot(happiness, aes(Freedom_to_make_life_choices, Social_support )) +

  # scatter plot
  geom_point(color = "dark green", shape = "diamond") +
  geom_smooth(method = 'lm', color = 'dark green') +

  # change axis labels

  xlab("Freedom to make life choices") +
  ylab("Social Support") +

  # set theme
  theme_classic()
```




```
## 6 ii) try se = FALSE
ggplot(happiness, aes(Freedom_to_make_life_choices, Social_support )) +

  # scatter plot
  geom_point(color = "dark green", shape = "diamond") +
  geom_smooth(method = 'lm', color = 'dark green', se = FALSE)
```




```
# se = FALSE removes the standard error from the regression line.
```