AMAAS: Data Visualization with ggplot2

Benjamin Utting

Jaap Saers (jpps2@cam.ac.uk) & Benjamin Utting (bju23@cam.ac.uk)

Analytical Methods in Anthropology and Archaeology Seminar (AMAAS)

Michaelmas 2018

The following packages are required for completing these exercises: *tidyverse*

Introduction

This R Markdown document contains exercises to accompany a workshop for learning ggplot2. Answers to these questions will be distributed online.

We will be using the 'msleep' dataset that is included with ggplot2. This file contains sleep habits of different animal species. It is a dataframe with 83 rows and 11 variables.

Original source: V. M. Savage and G. B. West. A quantitative, theoretical framework for understanding mammalian sleep. Proceedings of the National Academy of Sciences, 104 (3):1051-1056, 2007.

A First Look at our Dataset

Before we get started, we should do some preliminary data exploration.

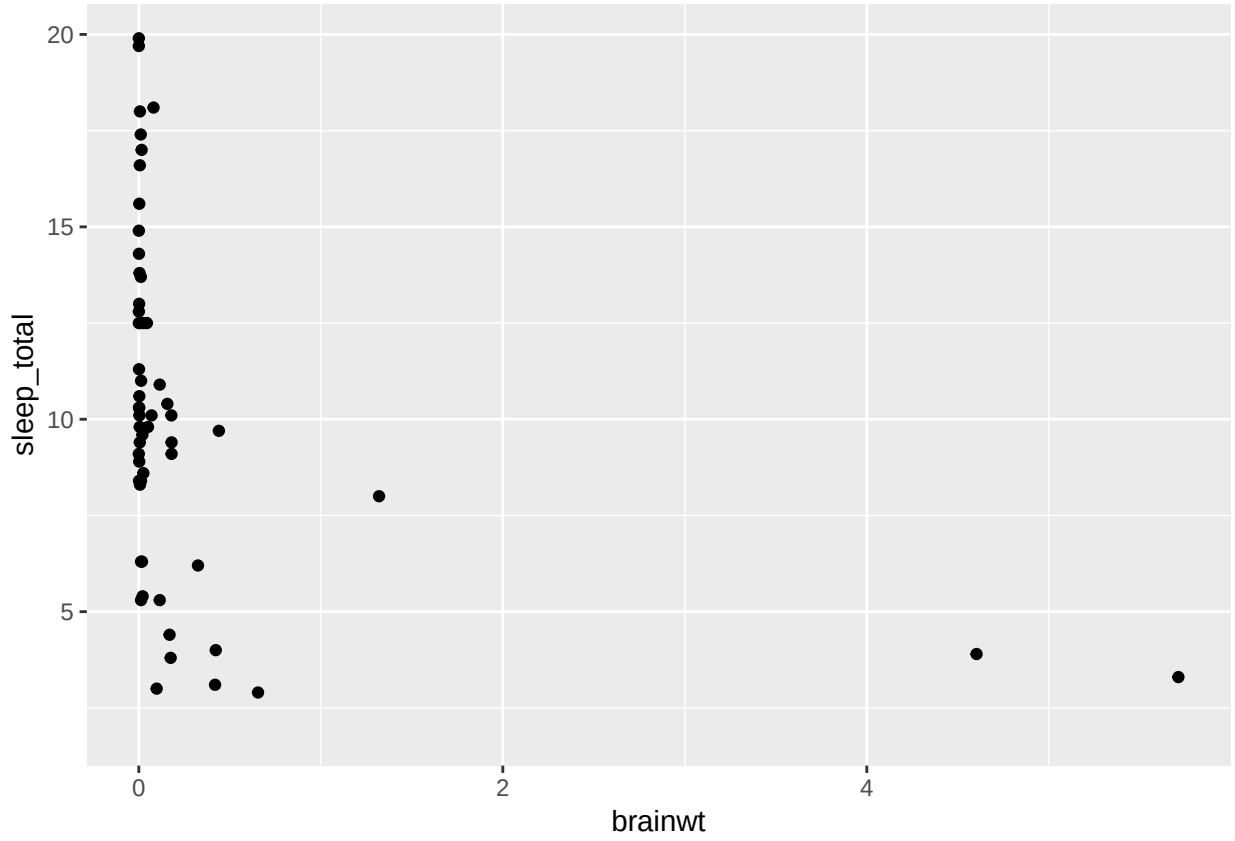
```
#Variable definitions:
?msleep

#View the full dataframe:
View(msleep)

#A look at the first 10 observations:
head(msleep, 10)

#Structure of the dataset:
str(msleep)

#Summary statistics for the dataset:
summary(msleep)
```

Please use this format for the following exercises: MyGraph <- ggplot(MyData, aes(variable for x axis, variable for y axis))+geom()

Reviewing scatterplots

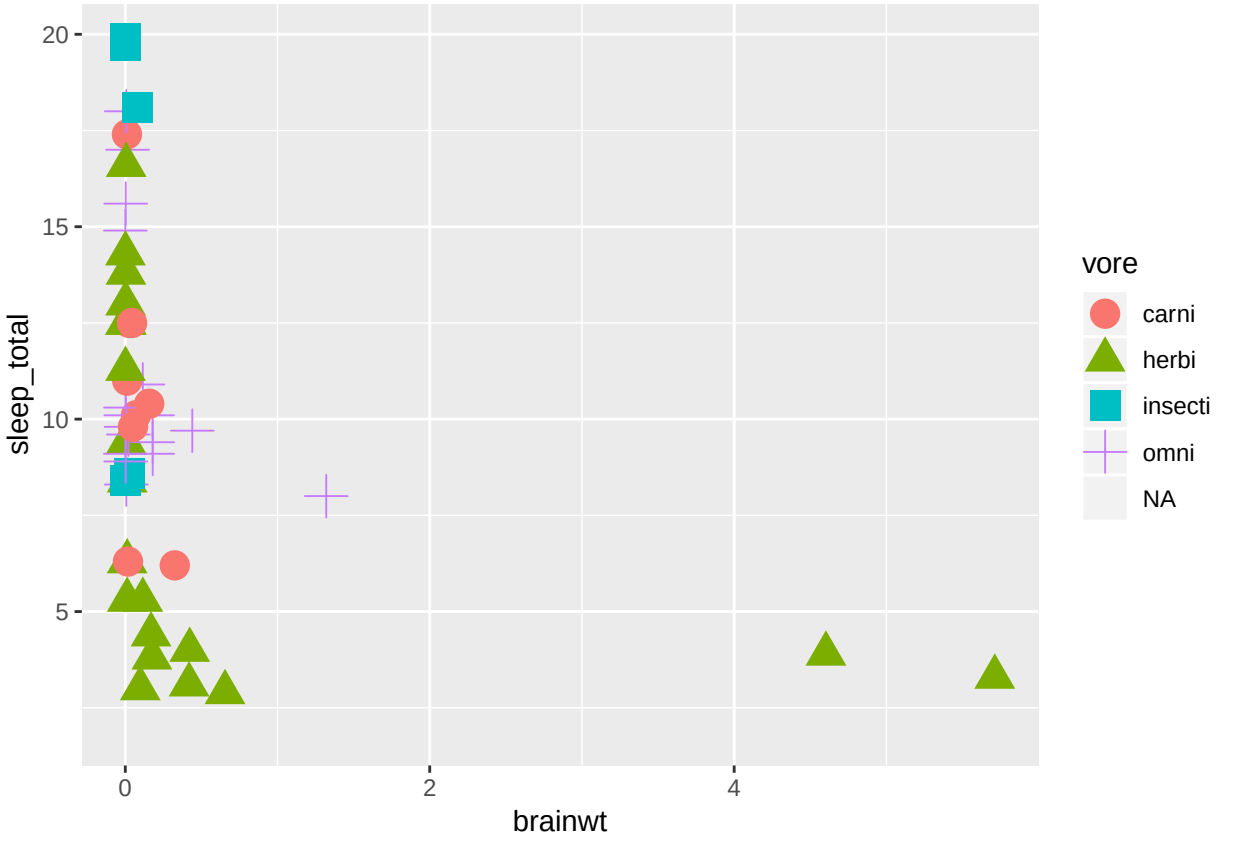
Say you want to look at the relationship between numbers of hours slept per day (sleep_total) and the weight of the animals brain (brainwt).

```
scatterplot<- ggplot(data = msleep, aes(x = brainwt, y = sleep_total)) +
  geom_point()

scatterplot
```



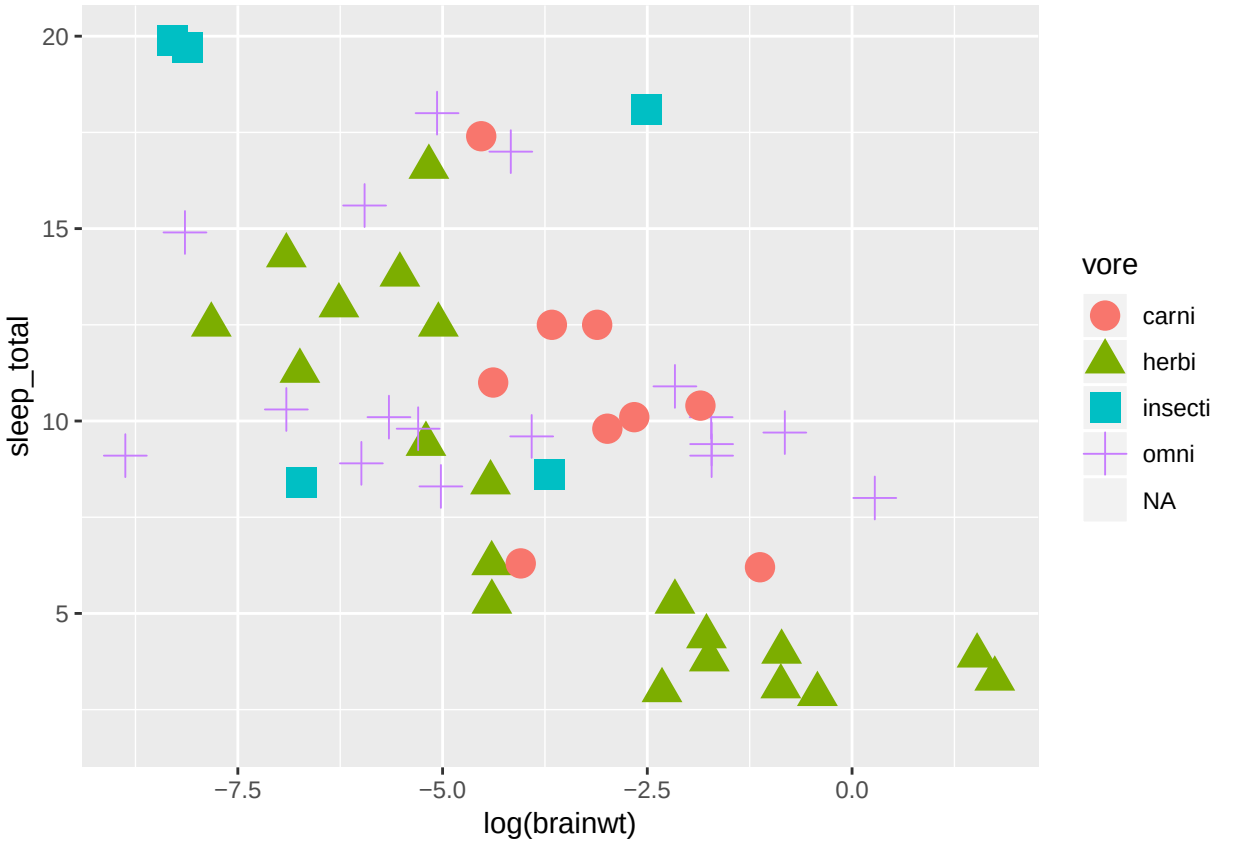
Next, you can add a different colour for each trophic level, and change the size of the points

```
scatterplot<- ggplot(data = msleep, aes(x = brainwt, y = sleep_total,
                                        colour = vore, shape = vore)) +
  geom_point(size=5)

scatterplot
```



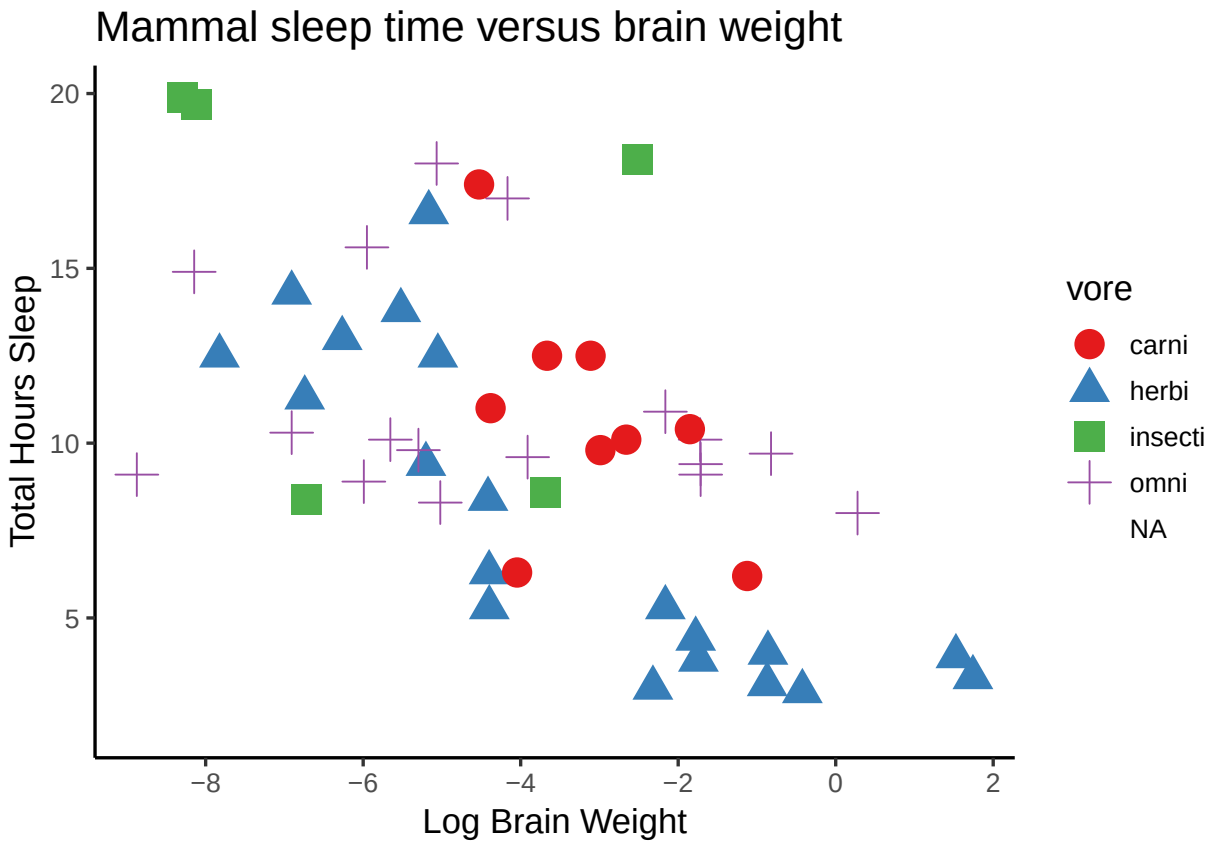
It looks like there are a few orders of magnitude spread in brain weight in these animals. Perhaps a log transform will make the data linear:

```
scatterplot <- ggplot(data = msleep, aes(x = log(brainwt), y = sleep_total,
                                         colour = vore, shape = vore)) +
  geom_point(size=5)

scatterplot
```



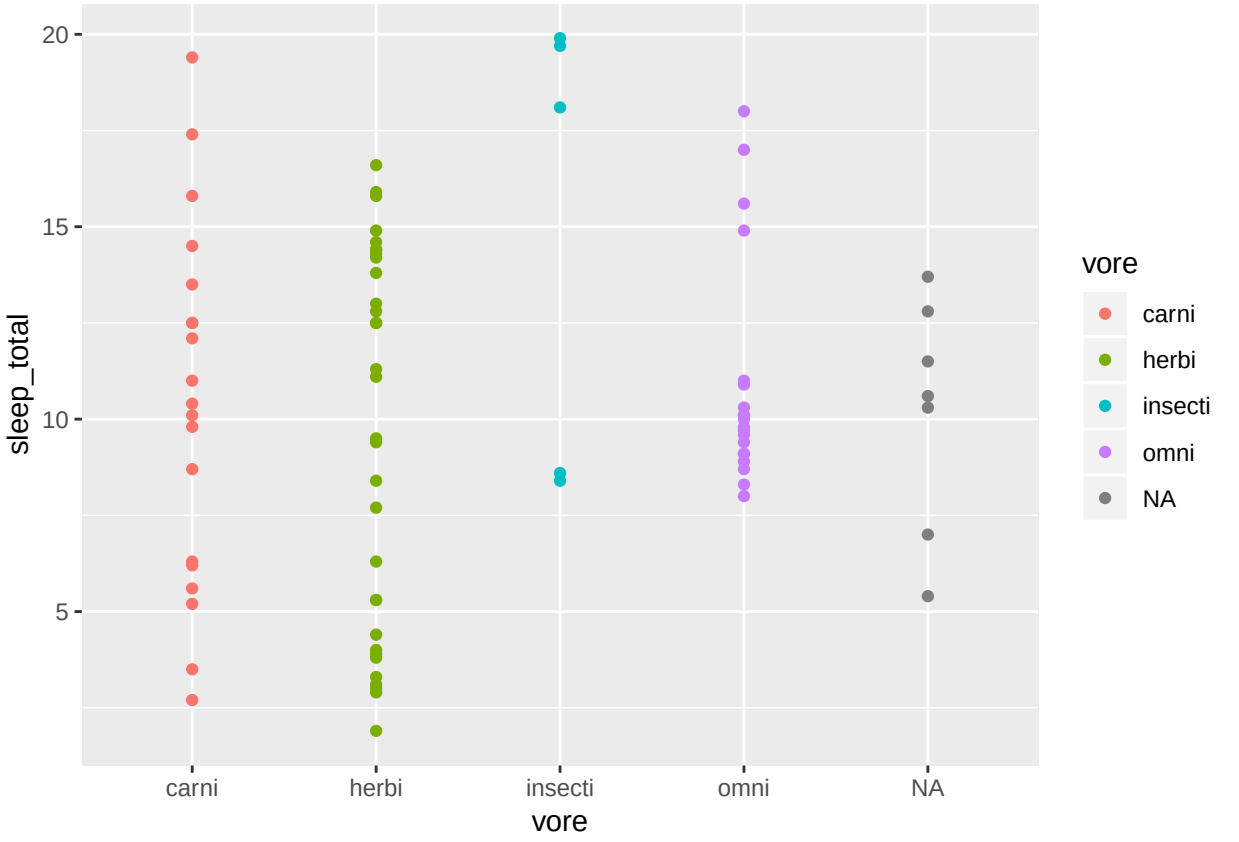
Next, we'll add some cosmetic touches to make our plot look more professional:

```
scatterplot<- ggplot(data = msleep, aes(x = log(brainwt), y = sleep_total,
                                        colour = vore, shape = vore)) +
  geom_point(size=5)

scatterplot <- scatterplot + xlab("Log Brain Weight") + ylab("Total Hours Sleep") +
ggtitle("Mammal sleep time versus brain weight") #axis titles

scatterplot <- scatterplot + scale_x_continuous(breaks = seq(-10, 4, 2))
#set x-axis breaks from -10  to 4 in steps of 2

scatterplot <- scatterplot + scale_colour_brewer(palette="Set1")    #you can set your own
#colour scheme or use premade "sets"

scatterplot <- scatterplot + theme_classic(base_size = 13)  #I think the grey doesn't
#look very nice, theme_classic() or theme_bw() are popular
#there are many preset themes to choose from and customize things such as font or size

scatterplot
```



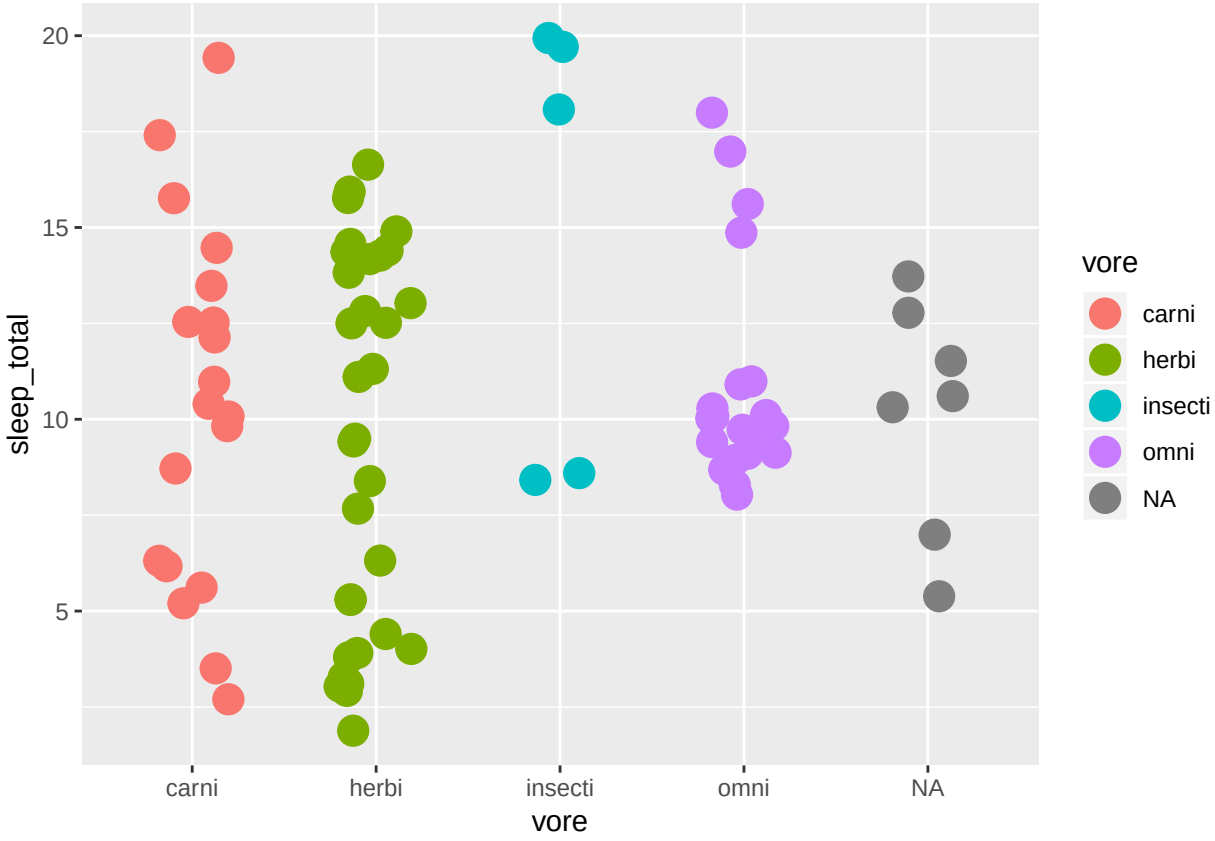
Now lets try plotting a categorical variable on the x-axis. We want to look at the average amount of sleep (sleep_total) per trophic level (vore).

```
sleepyvore <- ggplot(msleep, aes(x = vore, y = sleep_total, col = vore)) +
  geom_point()

sleepyvore
```



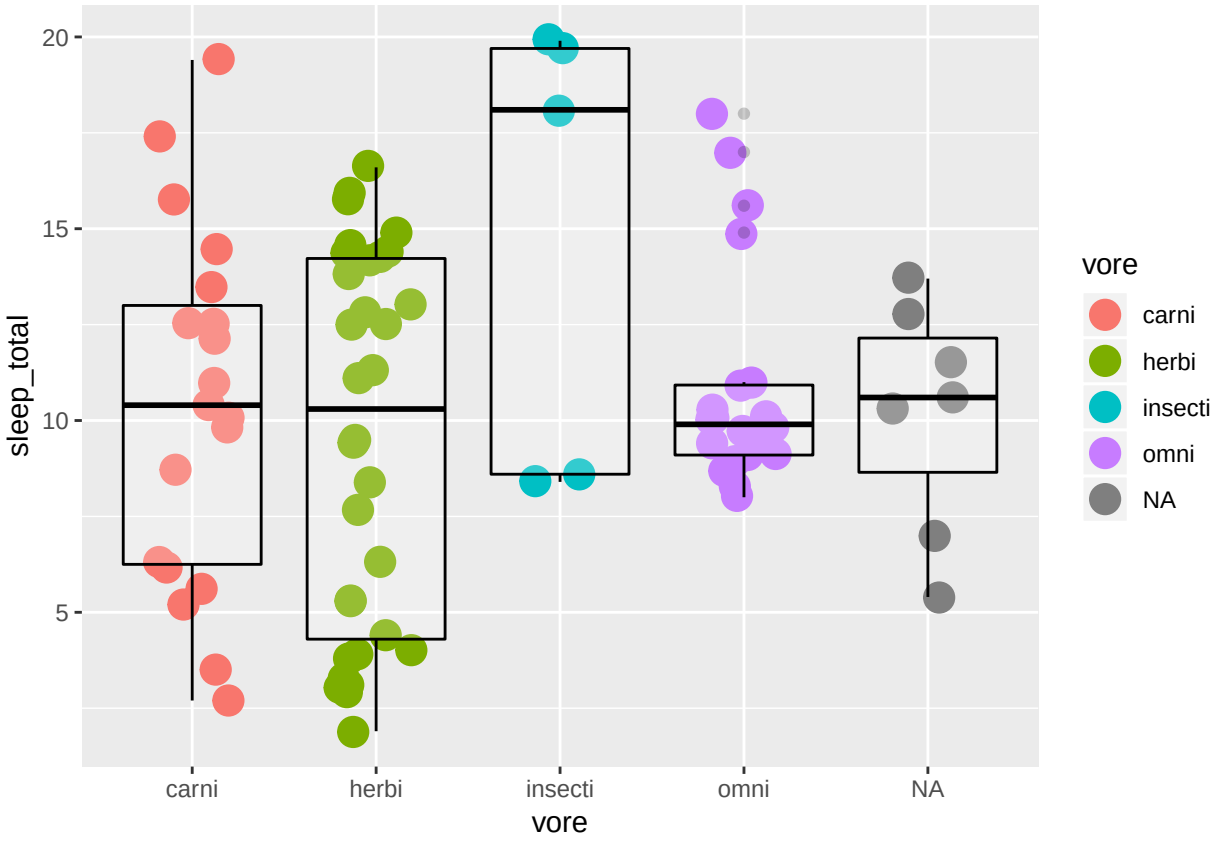
To prevent overlapping datapoints obscuring eachother, add some random noise using geom_jitter().

```
sleepyvore <- ggplot(msleep, aes(x = vore, y = sleep_total,
                                 col = vore)) +
  geom_jitter(position = position_jitter(width = 0.2), size = 5)
#width determins how wide the jitter is

sleepyvore
```



We can even overlay a boxplot if we want. Since we are overlaying the boxplots we need to make them transparant by specifying alpha between 0 and 1

```
sleepyvore <- sleepyvore + geom_boxplot(alpha = .2, colour = "black")

sleepyvore
```



Exercise 1A

A you want to know the distribution of body mass in your dataset. Visualise this using a geom_histogram() inside the geom, set an appropriate binwidth, colour to black, and fill to blue.

Exercise 1B

Everyone loves normal distributions, and bodywt is definitely not normal. When plotting things over several orders of magnitude a log-transform works wonders. Generate the same plotm but this time log-transformed bodywt.

Exercise 2

You want to see if there are differences in sleep time between primates and non-primate mammals. Assess using a boxplot.

Hint: First we have to create a new column for presence/absence of primate order using dplyr.

Exercise 3A

Scatter plot log(brainwt) to sleep_total in the same scatter plot for primates and non primates. add a linear model using geom_smooth(method = "lm").

Exercise 3B

3B use facet_grid(Primate~vore) to visualize the same relationship for trophic levels.

Exercise 4A

Violin/bean plots are a variation on the theme of the boxplot. They have the same structure but data density is mirrored by the shape of the body. Make a violinplot (geom_violin) of log(body weight) for each trophic level.

Exercise 4B

Plenty of room for improvement here. As you can see the data on the edges are trimmed by default. Turn off trim inside the geom using trim = FALSE and fill the geom with a nice colour using fill= "xx" and a black edge using colour= "xx".

Exercise 4C

Lets divide plots between primates and non-primates again. This time, fill the violin plots with a different color based on trophic level.

Exercise 5A

Plot the log of body weight on the x axis to brain weight on the y axis using a scatter plot. In geom_smooth, Use different colours based on the taxonomic order and shape based on whether is is a primate or not.

Exercise 5B

In the same plot, add a regression line for Primates and non-primates using geom_smooth().

Exercise 5C

Finally, let's make the plot a bit prettier. Add appropriate labels to the axes and give the plot a title. Have a look at different themes and select one you like.

Saving Plots

You can save plots using the export button above your plots in RStudio. However, you will have much more control over the plot properties if you use ggsave() which lets you set resolution, image name, type of image, etc. ggsave() takes the following arguments: ggsave(filename, plot = last_plot(), device = NULL, path = NULL,scale = 1, width = NA, height = NA, units = c("in", "cm", "mm"),dpi = 300, limitsize = TRUE, ...)

Here's an example:

```
ggsave("bodybrain.png", width = 10, height = 7, units = "in", dpi=300)
```