Introduction to ggplot2

Contents

Setup

First let us get the ggplot library, 'ggplot2'
If you don't have ggplot2 installed, get it like so:

```r
install.packages("ggplot2")
```

And then load it into your current environment,

```r
library(ggplot2)
```

Getting started

Get Data

We will be working with one of the example datasets available within the ggplot2 library, called msleep. This dataset contains statistics on sleep patterns for certain mammals.

```r
data("msleep", package = "ggplot2")
```

We can take a peek at the data, using the `head` command.

```r
kable(head(msleep))
```

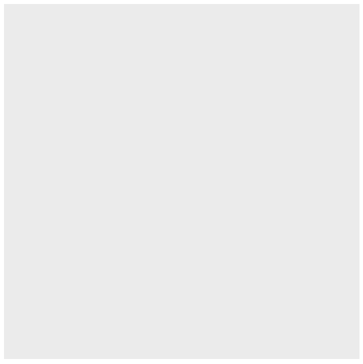
name	genus	vore	order	conservation	sleep_total	sleep_rem	sleep_cycle
Cheetah	Acinonyx	carni	Carnivora	lc	12.1	NA	NA
Owl monkey	Aotus	omni	Primates	NA	17.0	1.8	NA
Mountain beaver	Aplodontia	herbi	Rodentia	nt	14.4	2.4	NA
Greater short-tailed shrew	Blarina	omni	Soricomorpha	lc	14.9	2.3	0.1333333
Cow	Bos	herbi	Artiodactyla	domesticated	4.0	0.7	0.6666667
Three-toed sloth	Bradypus	herbi	Pilosa	NA	14.4	2.2	0.7666667

ggplot Grammar

When starting a new plot wiht ggplot, the `ggplot()` command initializes the plot structure. Then we can add other components to it. There are 3 *essential* components of a ggplot graphic, which we will review:

1. The particular dataset that we wish to plot a figure for

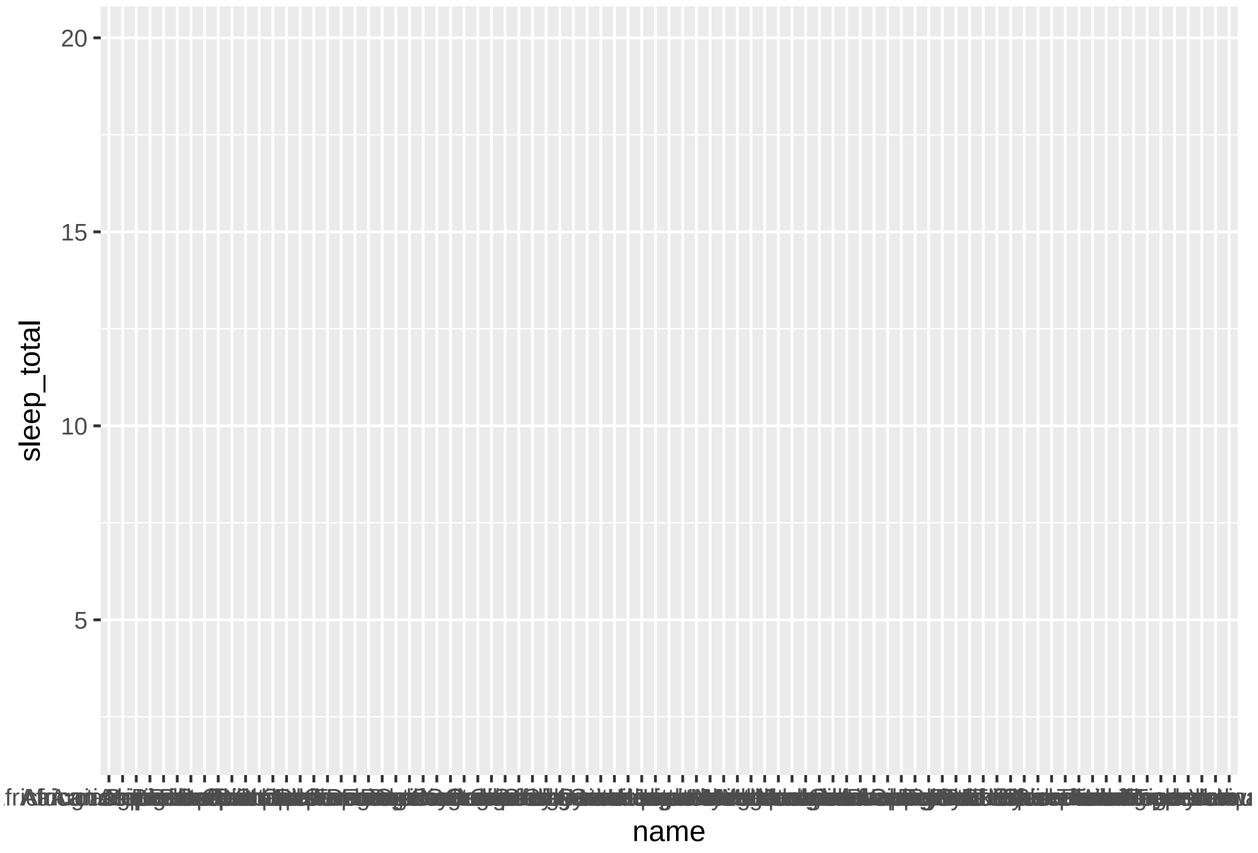
```
ggplot(msleep)
```



Notice how this is just an empty box! That's because we haven't really told ggplot what to do with the data (cue, component 2).

2. The aesthetics, defining the things from the data you wish to plot, and what parts of the graph they go on to. In this instance, we are defining the variables from our dataset, that map to the x and y coordinates.

```
ggplot(msleep, aes(x = name, y = sleep_total))
```



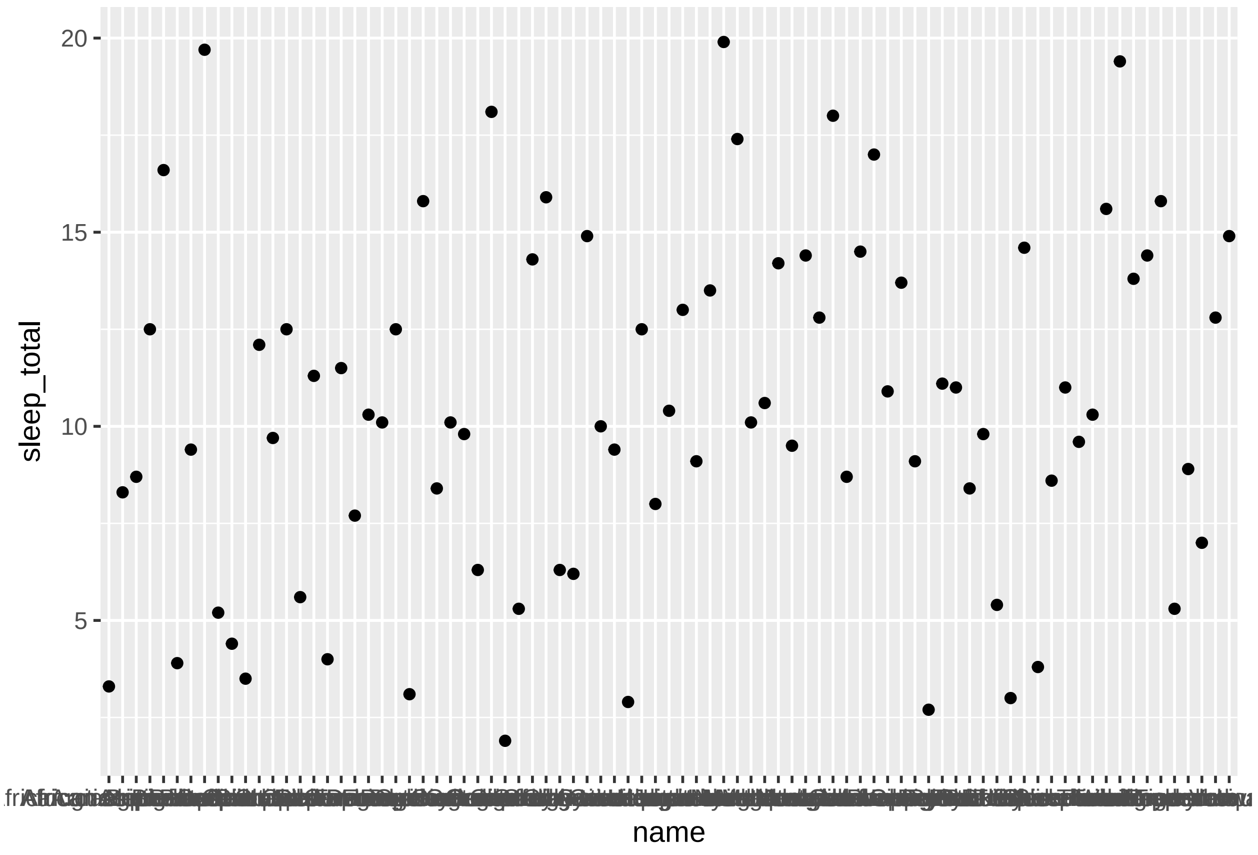
Well, now we have specified what aspects of the data to put on the plot. Now we have to tell it *how* we want to see the corresponding points (histogram? range of the values? scatterplot?).

3. The geometric object, also called the 'geom', specifies how your data is presented on the plot. This can be a geom_point(), geom_count(), geom_line() etc.

- We add this attribute using the '+' sign. You can tack on as many of these as you want, as long as they work within the bounds of the different variables you have defined.

Here, we are going to compare the different animals (as defined by the column 'name') against the total amount of sleep they have on average (as defined by the column 'sleep_total').

```
ggplot(msleep, aes(x = name, y = sleep_total)) + geom_point()
```



Modify a basic plot

Let us start with the figure we just plotted. This is a *scatter plot.*

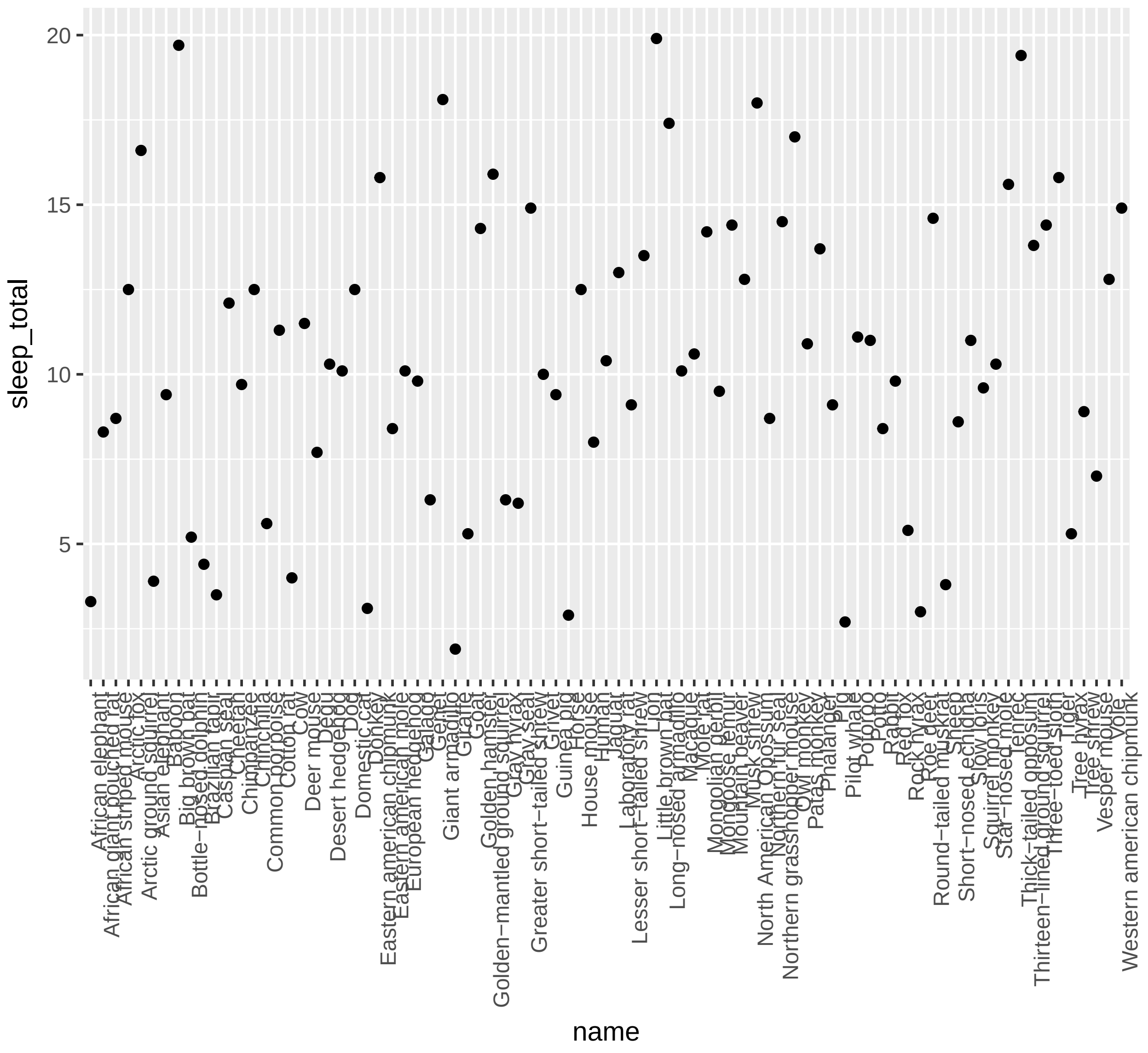
There are some immediate challenges we see here as a reader.

1. Changing the orientation of the labels.
   First off, we can't read the labels on the x-axis! This is the default in ggplot, and you will probably find yourself tweaking this quite often. There are two ways we can fix it.

i) Change the angle of the labels relative to the axis.

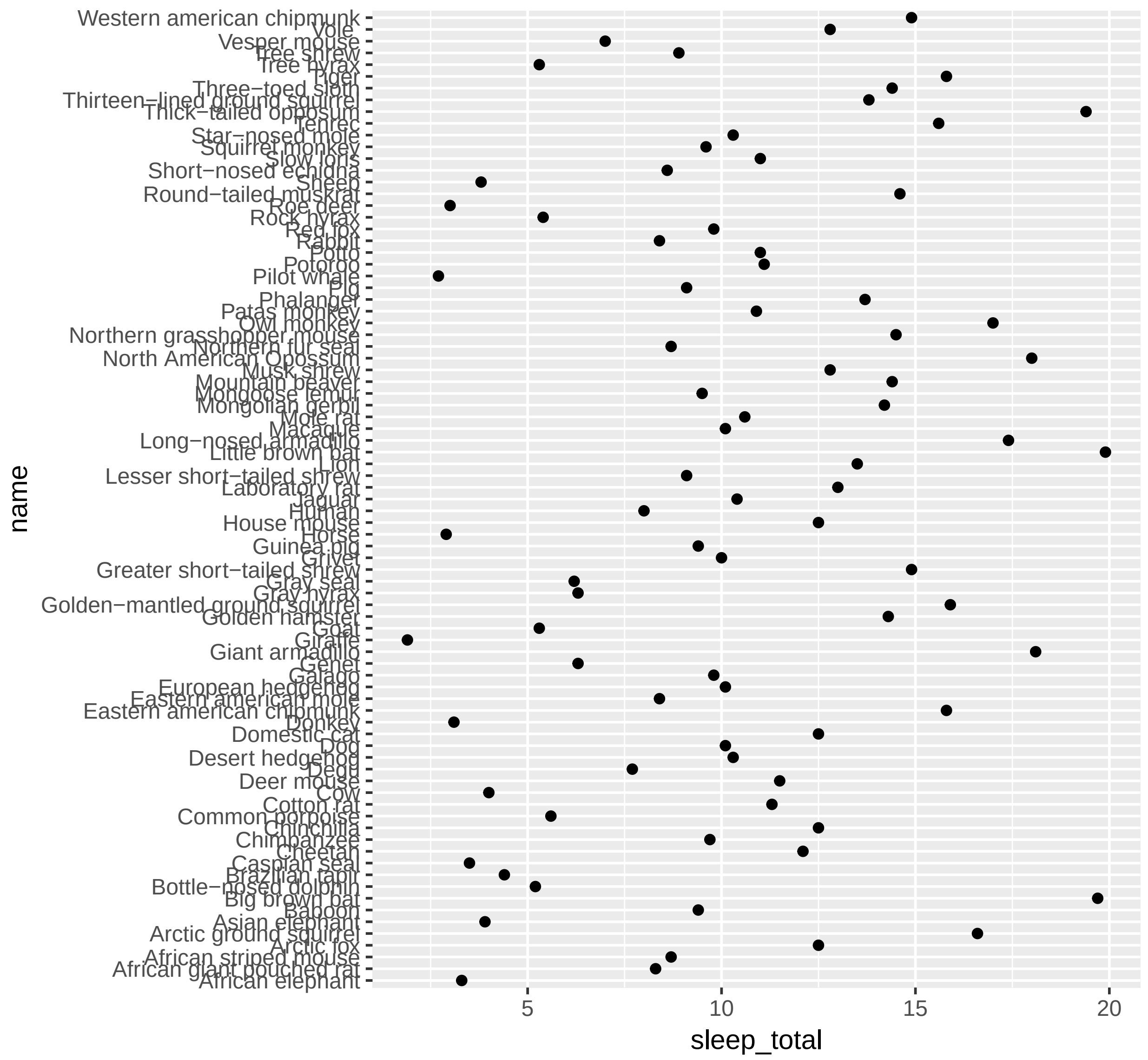
We modify this aspect using the theme object. We will be coming back to this one later for changing other settings too!

```r
ggplot(msleep, aes(x = name, y = sleep_total)) + geom_point() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



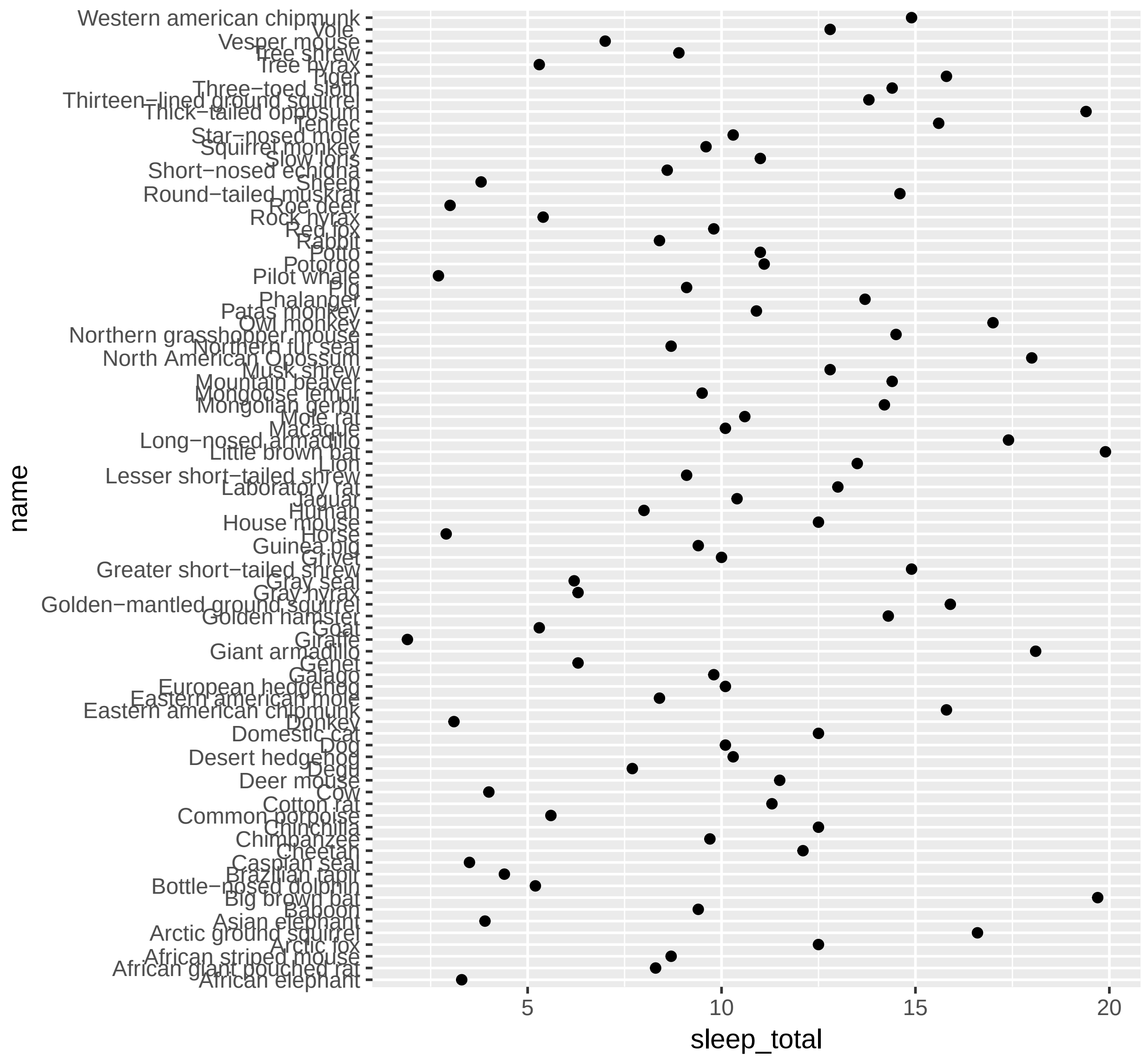
ii) Make the labels the y axis instead!
And the cool thing about ggplot is there is 2 ways you can do this too! If you're not interested in editing a lot of stuff, you can just add in a 'coord_flip()' object. Can you guess how we will do this?

```r
ggplot(msleep, aes(x = name, y = sleep_total)) + geom_point() +
    coord_flip()
```



Otherwise, you can just change the designation of what the 'x' axis is and what the 'y' axis is.

```
ggplot(msleep, aes(y = name, x = sleep_total)) + geom_point()
```

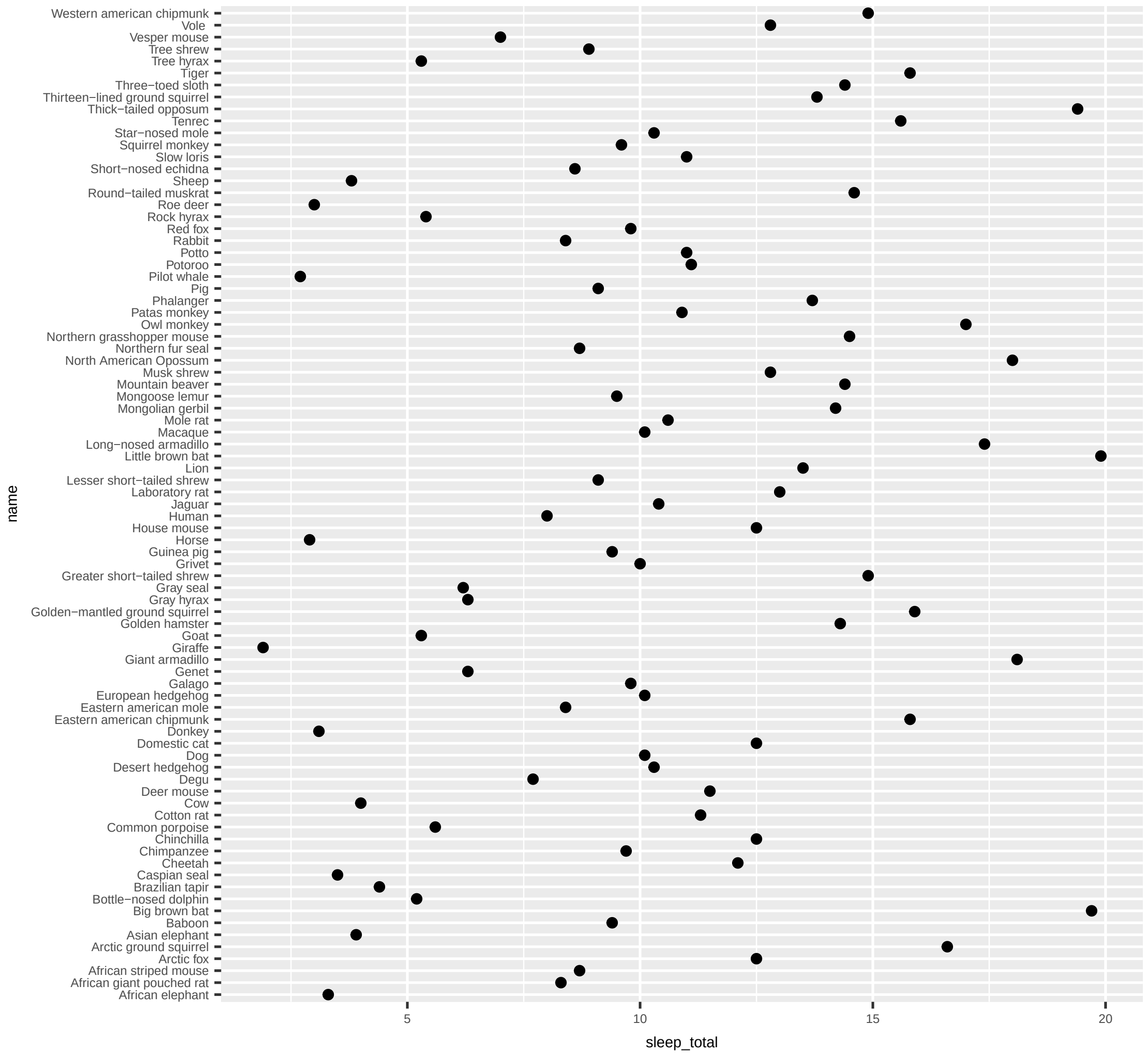


name
sleep_total
5
10
15
20
Western american chipmunk
Vole
Vesper mouse
Tree shrew
Tree hyrax
Tiger
Three−toed sloth
Thirteen−lined ground squirrel
Thick−tailed opposum
Tenrec
Star−nosed mole
Squirrel monkey
Slow loris
Short−nosed echidna
Sheep
Round−tailed muskrat
Roe deer
Rock hyrax
Red fox
Rabbit
Pudu
Potoroo
Pilot whale
Pig
Phalanger
Patas monkey
Owl monkey
Northern grasshopper mouse
Northern fur seal
North American Opossum
Musk shrew
Mountain beaver
Mongoose lemur
Mongolian gerbil
Mole rat
Macaque
Long−nosed armadillo
Little brown bat
Lion
Lesser short−tailed shrew
Laboratory rat
Jaguar
Human
House mouse
Horse
Guinea pig
Grivet
Greater short−tailed shrew
Gray seal
Gray hyrax
Golden−mantled ground squirrel
Golden hamster
Goat
Giraffe
Giant armadillo
Genet
Galago
European hedgehog
Eastern american mole
Eastern american chipmunk
Donkey
Domestic cat
Dog
Desert hedgehog
Degu
Deer mouse
Cow
Cotton rat
Common porpoise
Chinchilla
Chimpanzee
Cheetah
Caspian seal
Brazilian tapir
Bottle−nosed dolphin
Big brown bat
Baboon
Asian elephant
Arctic ground squirrel
Arctic fox
African striped mouse
African giant pouched rat
African elephant

2. Changing the size of the text

Secondly, you can't read the text because it is overlapping so much. Perhaps we should decrease the font size? As with all things in ggplot, there are several ways you can do this. Here is the simplest one (using our old friend, the theme component).

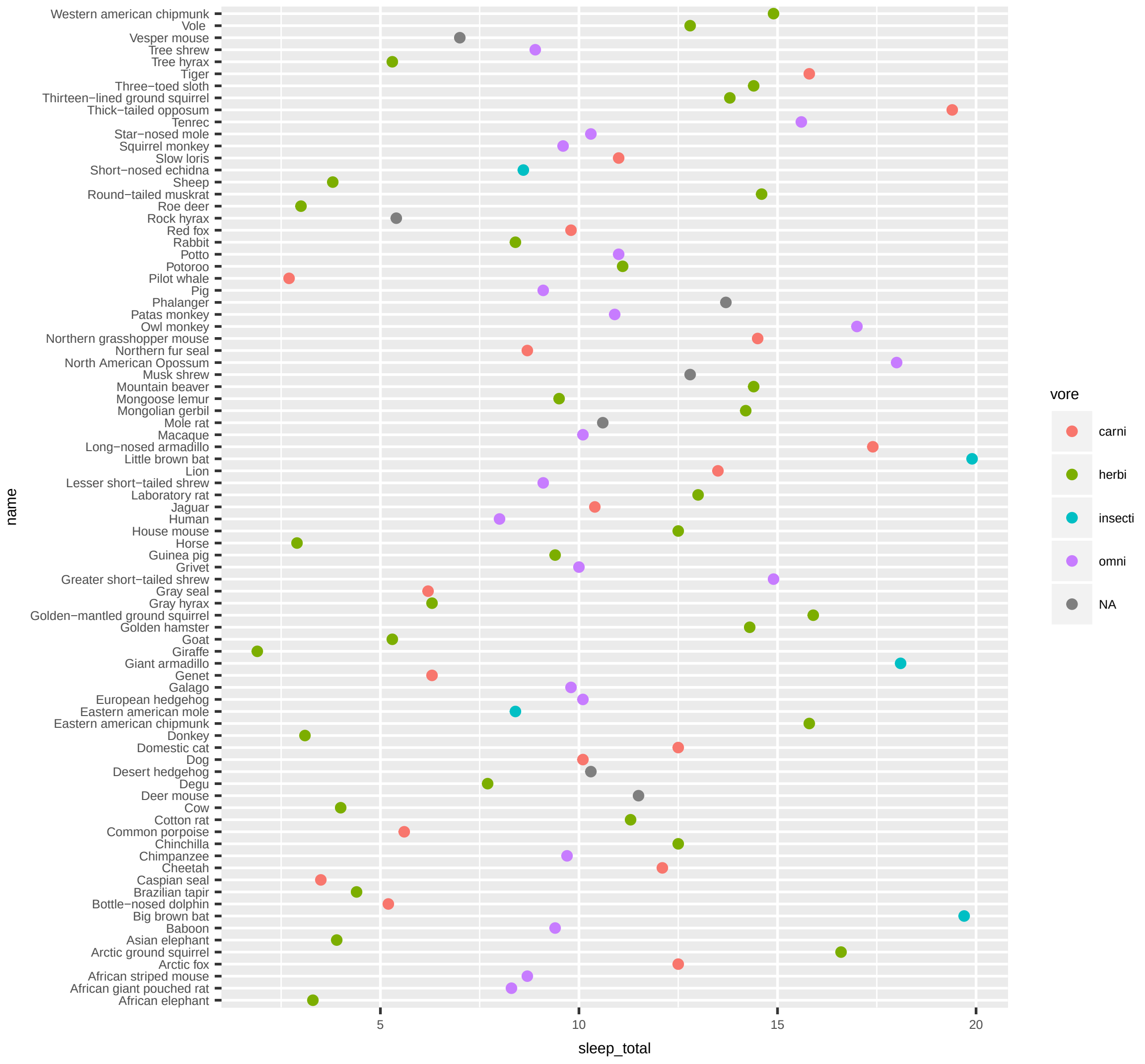
```r
ggplot(msleep, aes(y = name, x = sleep_total)) + geom_point() +
    theme(text = element_text(size = 6))
```



3. Adding a bit more detail (colours!)

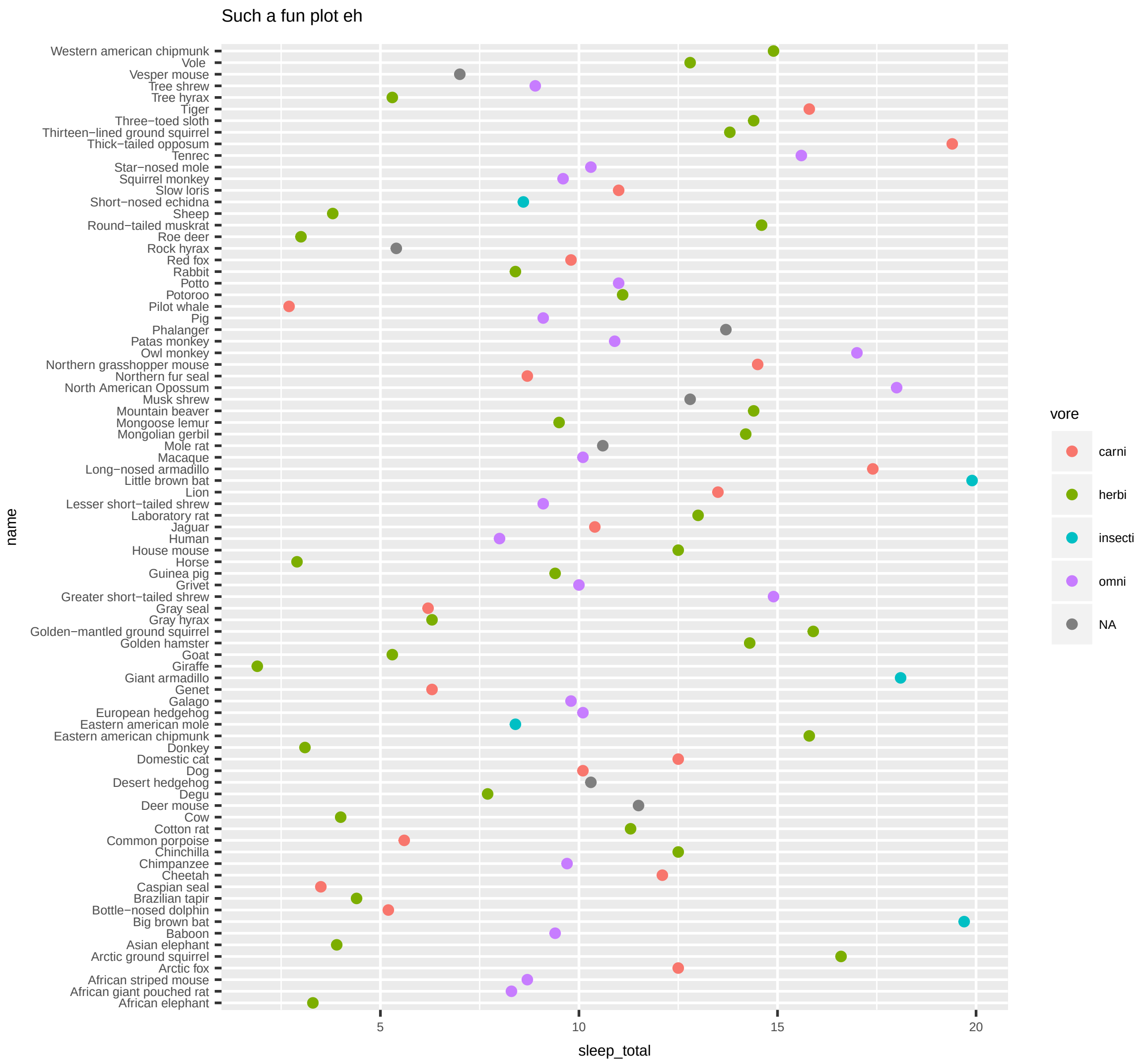
We can colour all these different points on our scatter plot which represent the different animals. Out of all the other columns we have in the dataframe (question - how would you check what's there?), the 'vore' column seems the most general, useful one.
We add it as part of the definition for aesthetics.

```r
ggplot(msleep, aes(y = name, x = sleep_total, colour = vore)) +
    geom_point() + theme(text = element_text(size = 6))
```



4. Adding a title
   We do this using the ggtitle object

```
ggplot(msleep, aes(y = name, x = sleep_total, colour = vore)) +
    geom_point() + theme(text = element_text(size = 6)) + ggtitle("Such a fun plot eh")
```



Notice how this is left justified. Terribly enough, this is the default in ggplot. We can fix this by going to our theme component, and adding an option for the title (remember how we modified the x axis earlier?). Notice how I've used the theme object twice. It is the lazy way to do it, but hey, it still works!

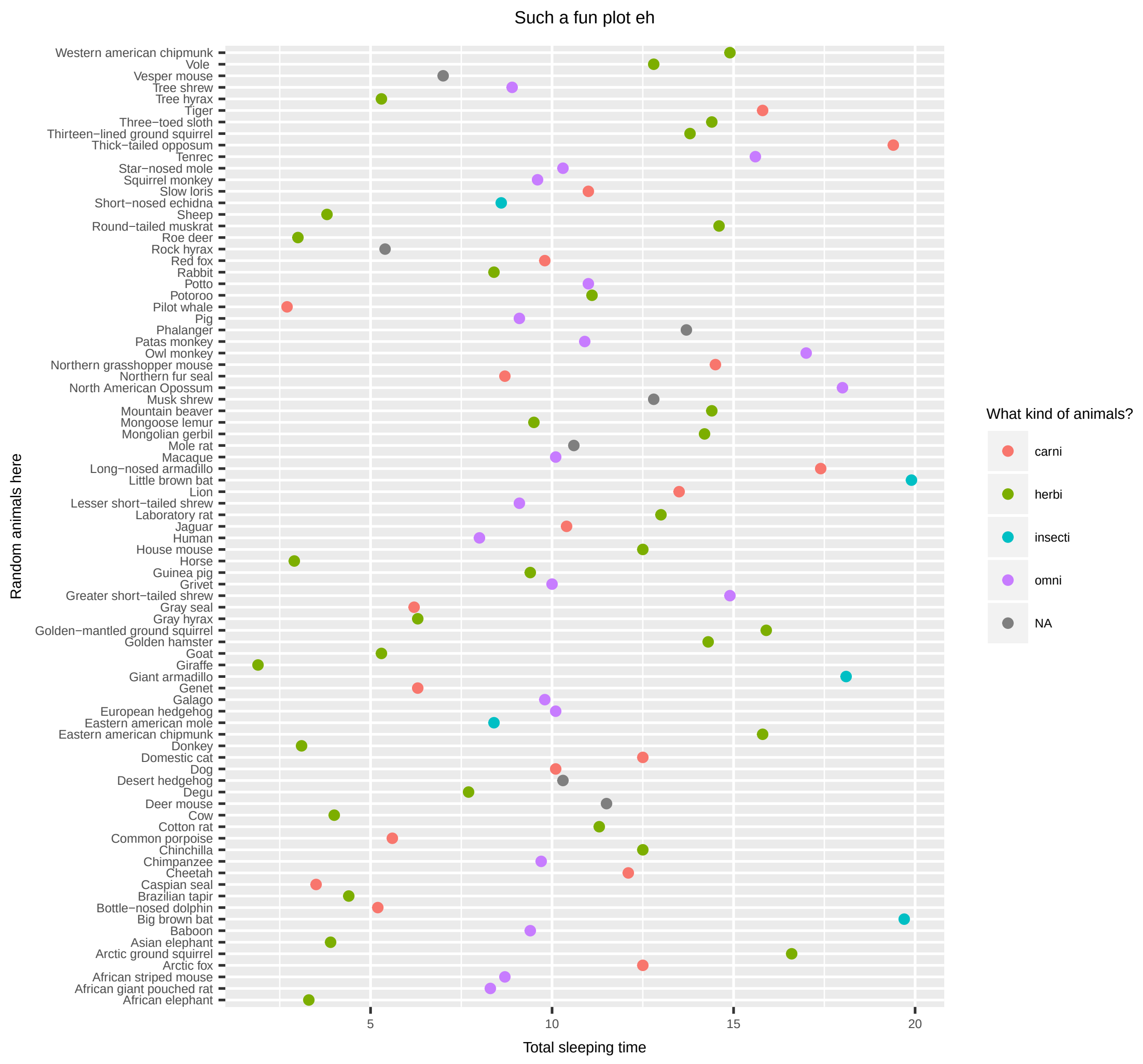
```
ggplot(msleep, aes(y = name, x = sleep_total, colour = vore)) +
    geom_point() + theme(text = element_text(size = 6)) + ggtitle("Such a fun plot eh") +
    theme(plot.title = element_text(hjust = 0.5))
```

With our original line of code where we modified the orientation of the x labels, this is how the command will look.

```
ggplot(msleep, aes(x = name, y = sleep_total, colour = vore)) +
    geom_point() + ggtitle("Such a fun plot eh") + theme(axis.text.x = element_text(angle = 90,
    hjust = 1), plot.title = element_text(hjust = 0.5))
```

5. Modifying labels
   Perhaps you also want to modify the text for the x axis and the y axis. Or perhaps for the legend for the different colours? We can define all these aspects with the 'labs' component that we add to our plot, like this:

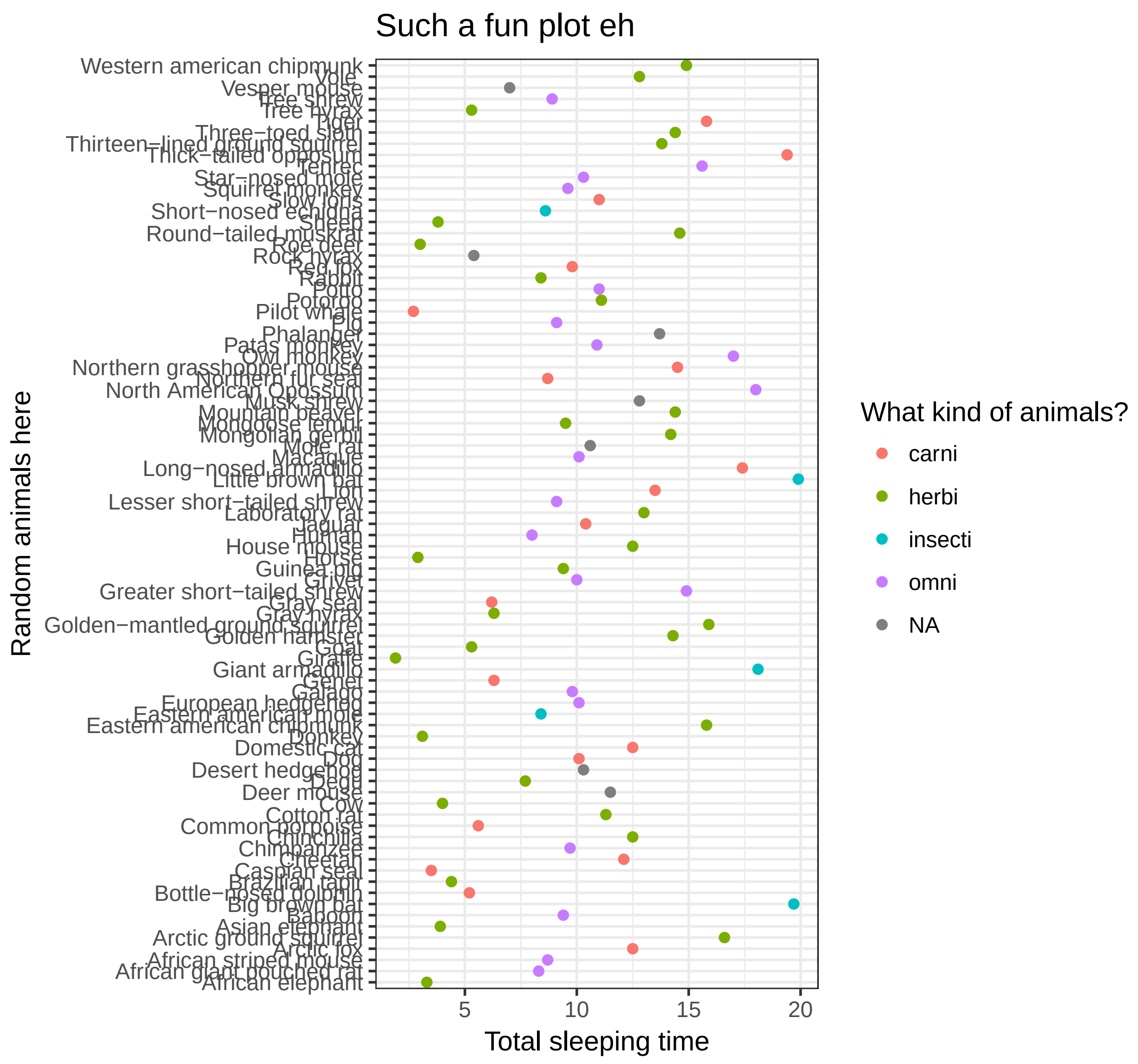
```
ggplot(msleep, aes(y = name, x = sleep_total, colour = vore)) +
    geom_point() + theme(text = element_text(size = 6)) + ggtitle("Such a fun plot eh") +
    theme(plot.title = element_text(hjust = 0.5)) + labs(x = "Total sleeping time",
    y = "Random animals here", colour = "What kind of animals?")
```



6. Getting rid of that awful grey background
   ggplot has this wonderful option where you can tack on a specific ‘theme’ to add different visual appeal to your plot. Consider these examples:

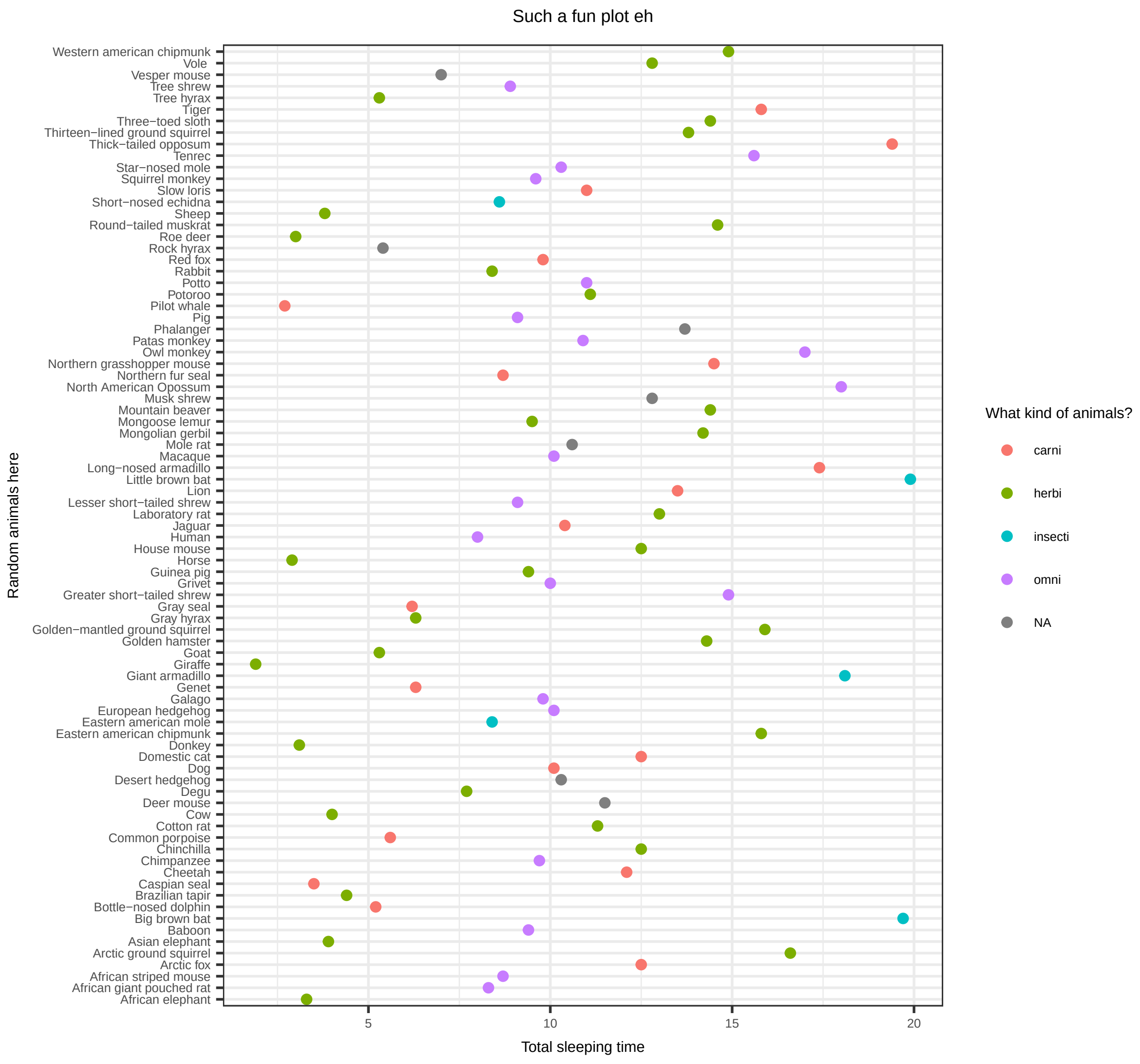
- theme_bw

```r
ggplot(msleep, aes(y = name, x = sleep_total, colour = vore)) +
    geom_point() + theme(text = element_text(size = 6)) + ggtitle("Such a fun plot eh") +
    theme(plot.title = element_text(hjust = 0.5)) + labs(x = "Total sleeping time",
    y = "Random animals here", colour = "What kind of animals?") +
    theme_bw()
```



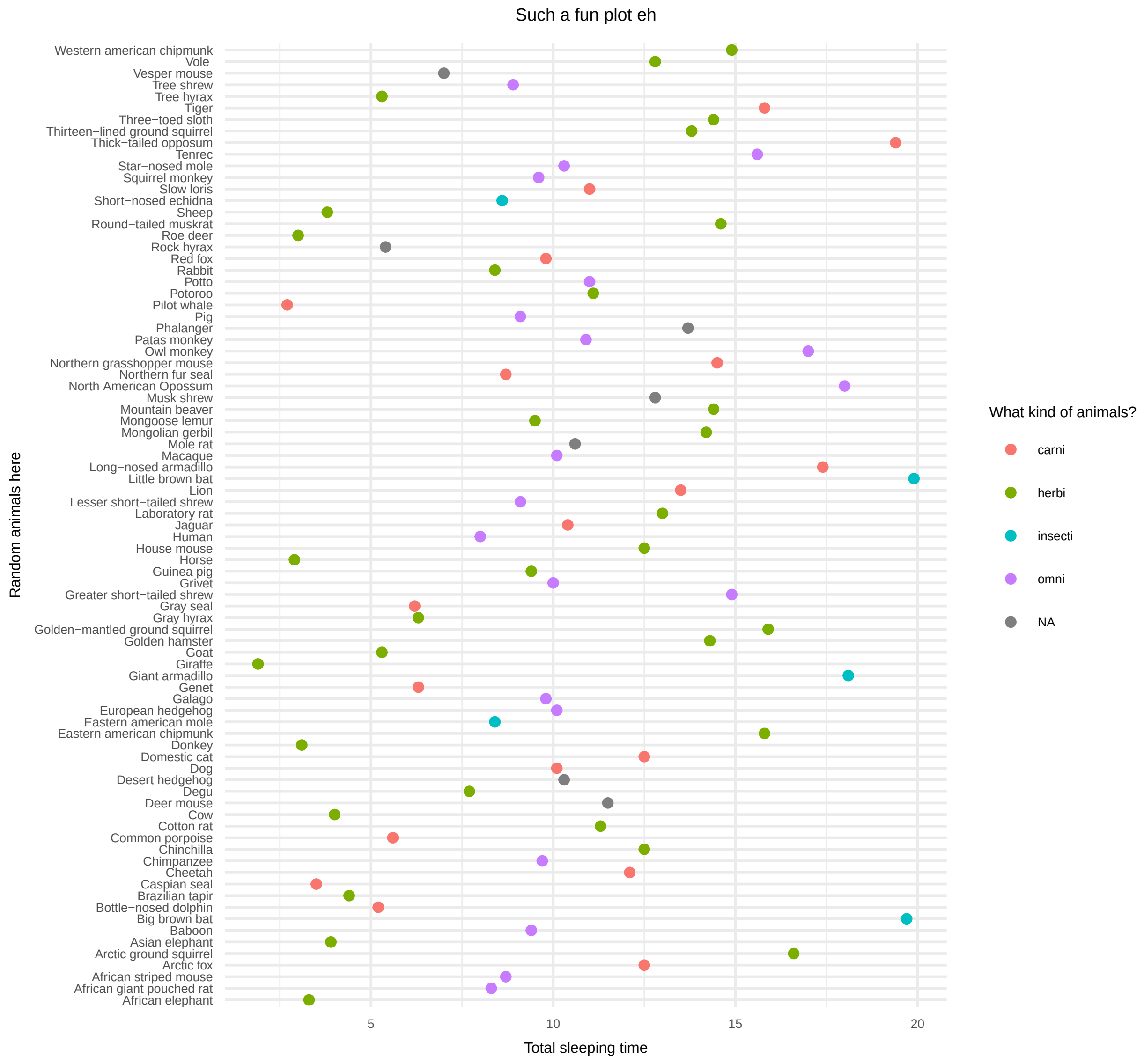
Notice how all our specifications for different text sizes went away! This is because we added in the theme_bw object at the end of our command. Here is what happens when we put it before we modify the text size.
- Components in a ggplot plot are evaluated 'in order' of how we add them.

```
ggplot(msleep, aes(y = name, x = sleep_total, colour = vore)) +
    geom_point() + theme_bw() + theme(text = element_text(size = 6)) +
    ggtitle("Such a fun plot eh") + theme(plot.title = element_text(hjust = 0.5)) +
    labs(x = "Total sleeping time", y = "Random animals here",
        colour = "What kind of animals?")
```



- theme_minimal

```r
ggplot(msleep, aes(y = name, x = sleep_total, colour = vore)) +
    geom_point() + theme_minimal() + theme(text = element_text(size = 6)) +
    ggtitle("Such a fun plot eh") + theme(plot.title = element_text(hjust = 0.5)) +
    labs(x = "Total sleeping time", y = "Random animals here",
        colour = "What kind of animals?")
```



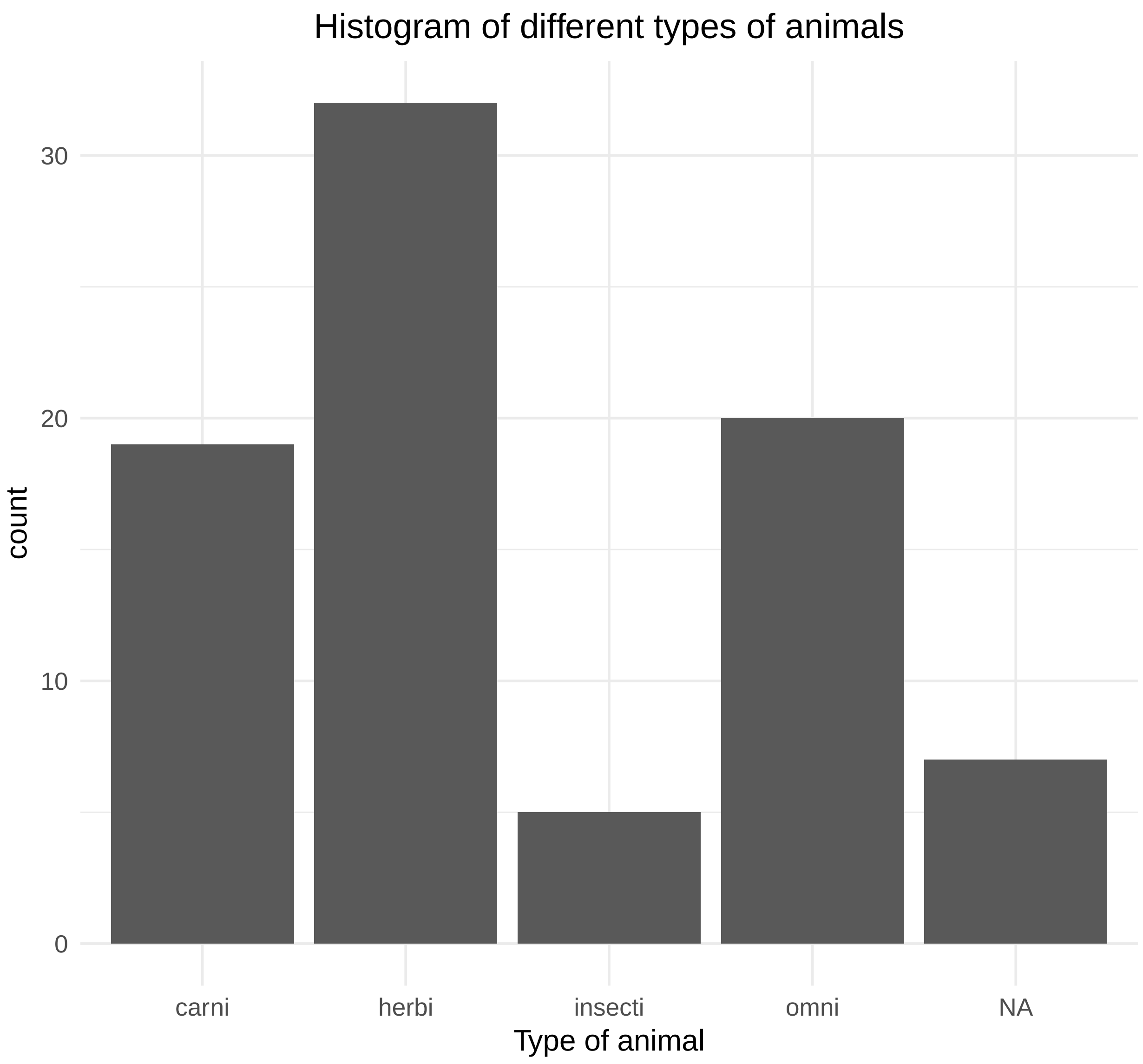
You can play around with additional themes here: http://ggplot2.tidyverse.org/reference/ggtheme.html

Basic plots in ggplot

Lastly, here are some additional basic plots in ggplot, besides the scatter plot.

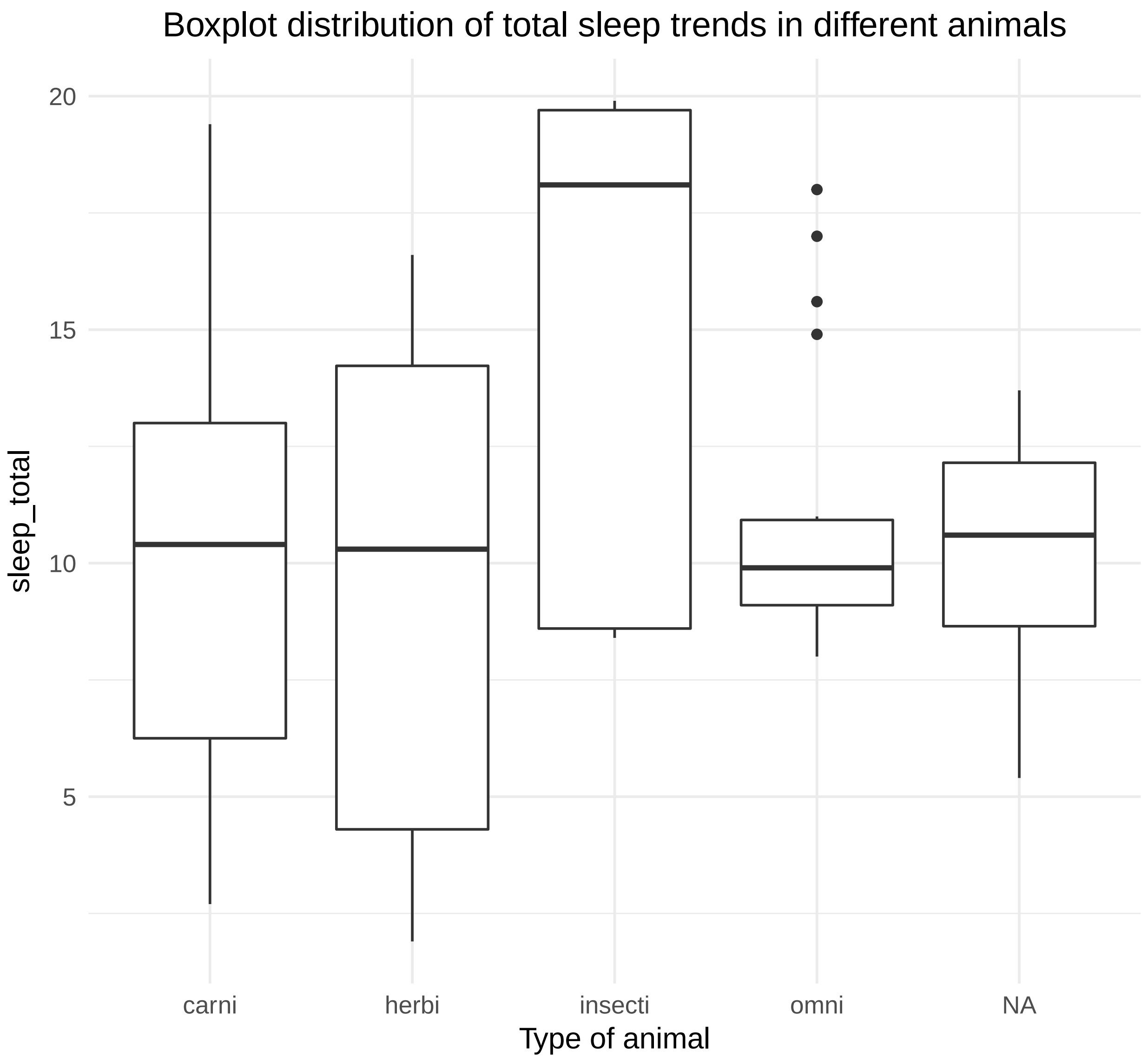
1. Histogram

```r
ggplot(msleep, aes(x = vore)) + geom_histogram(stat = "count") +
    theme_minimal() + theme(text = element_text(size = 12)) +
    ggtitle("Histogram of different types of animals") + theme(plot.title = element_text(hjust = 0.5))
    labs(x = "Type of animal")
```



2. Boxplot

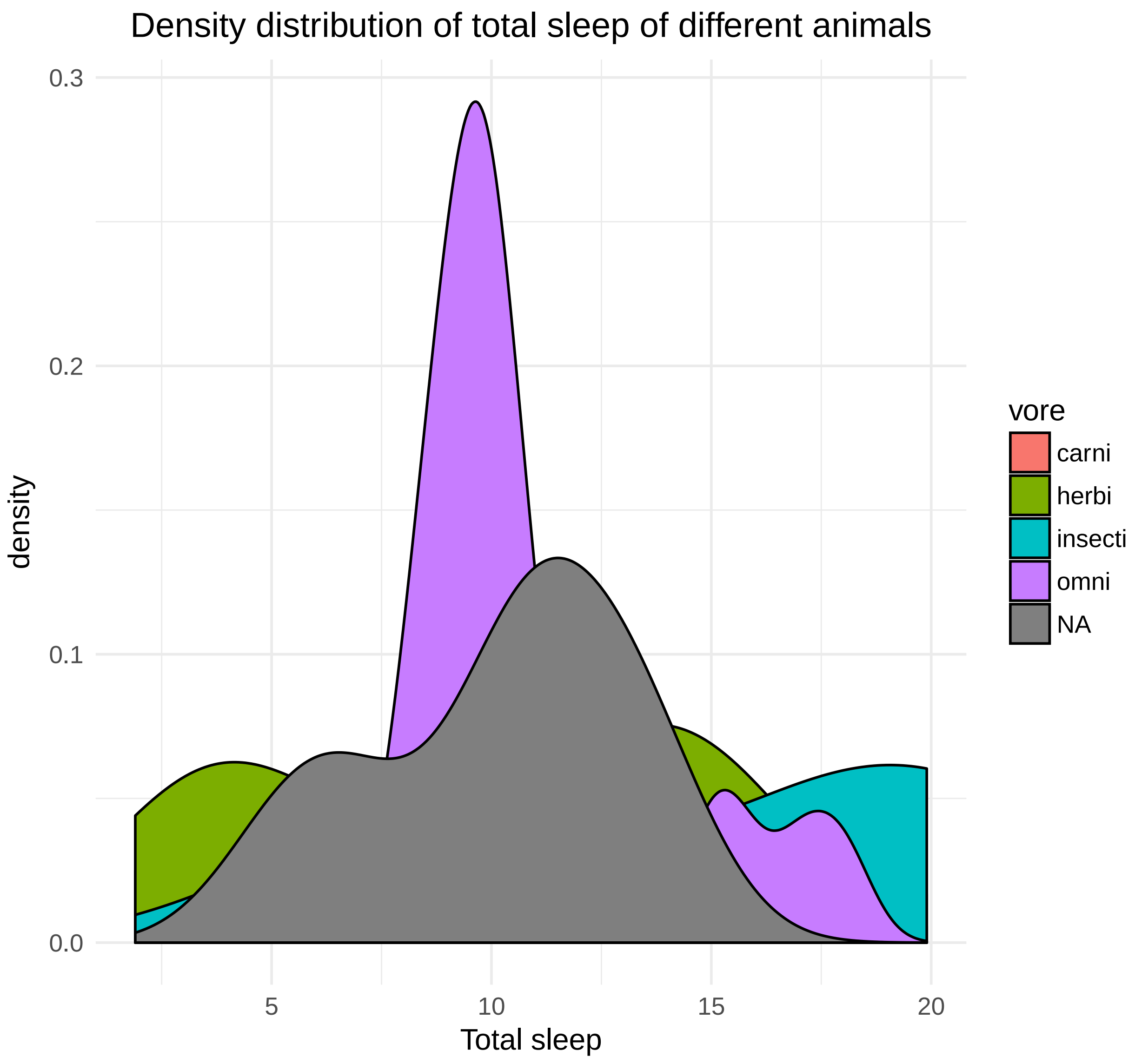
```r
ggplot(msleep, aes(x = vore, y = sleep_total)) + geom_boxplot() +
    theme_minimal() + theme(text = element_text(size = 12)) +
    ggtitle("Boxplot distribution of total sleep trends in different animals") +
    theme(plot.title = element_text(hjust = 0.5)) + labs(x = "Type of animal")
```



3. Density plot

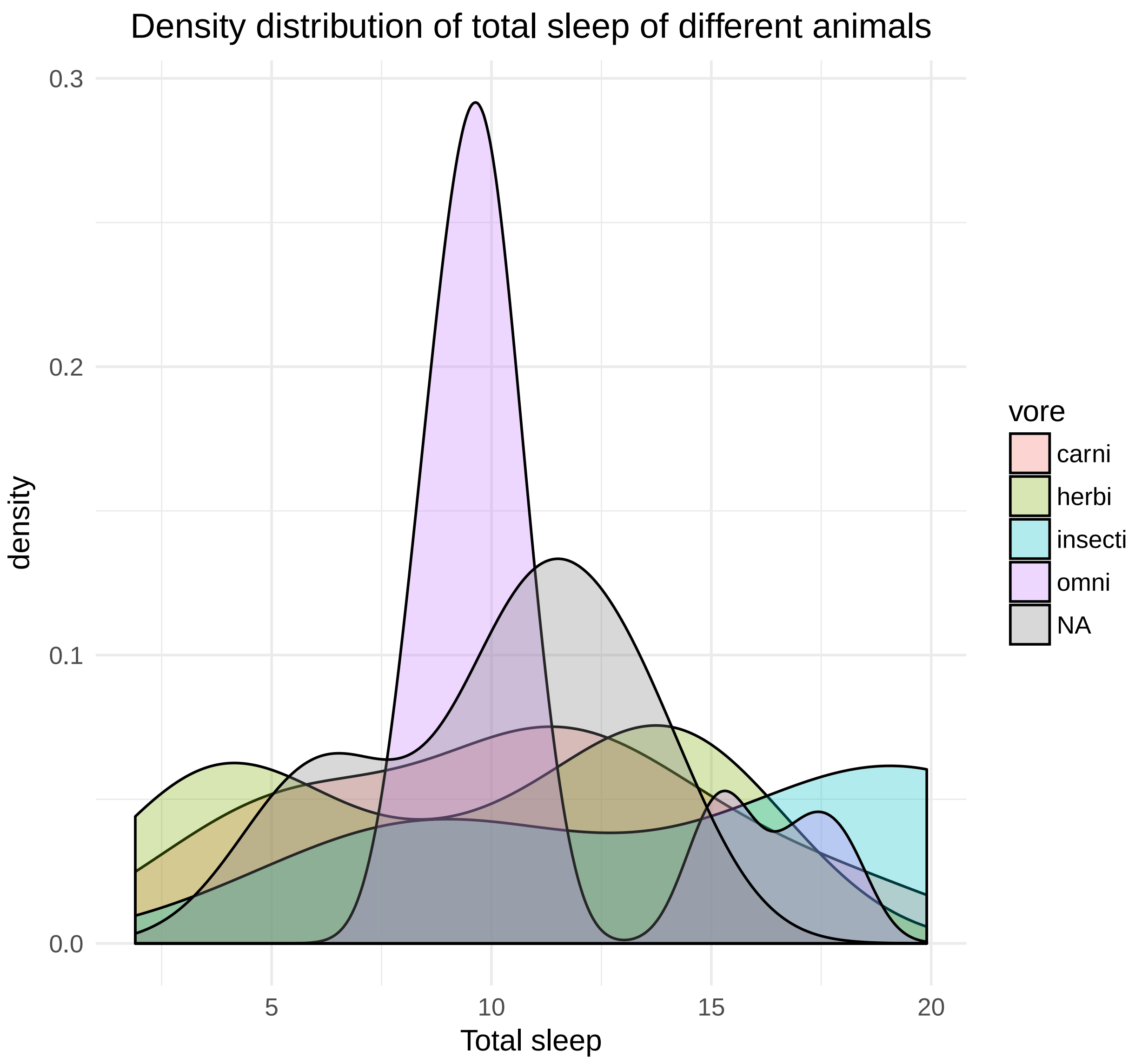
- We set the x to be a numeric observed variable, as we want to see the change in its density within our dataset.

```r
ggplot(msleep, aes(fill = vore, x = sleep_total)) + geom_density() +
    theme_minimal() + theme(text = element_text(size = 12)) +
    ggtitle("Density distribution of total sleep of different animals") +
    theme(plot.title = element_text(hjust = 0.5)) + labs(x = "Total sleep")
```



To make our density graphs more readable, we can modify the transparency of our geometric object (can you identify what this object is?)

```r
ggplot(msleep, aes(fill = vore, x = sleep_total)) + geom_density(alpha = 0.3) +
    theme_minimal() + theme(text = element_text(size = 12)) +
    ggtitle("Density distribution of total sleep of different animals") +
    theme(plot.title = element_text(hjust = 0.5)) + labs(x = "Total sleep")
```

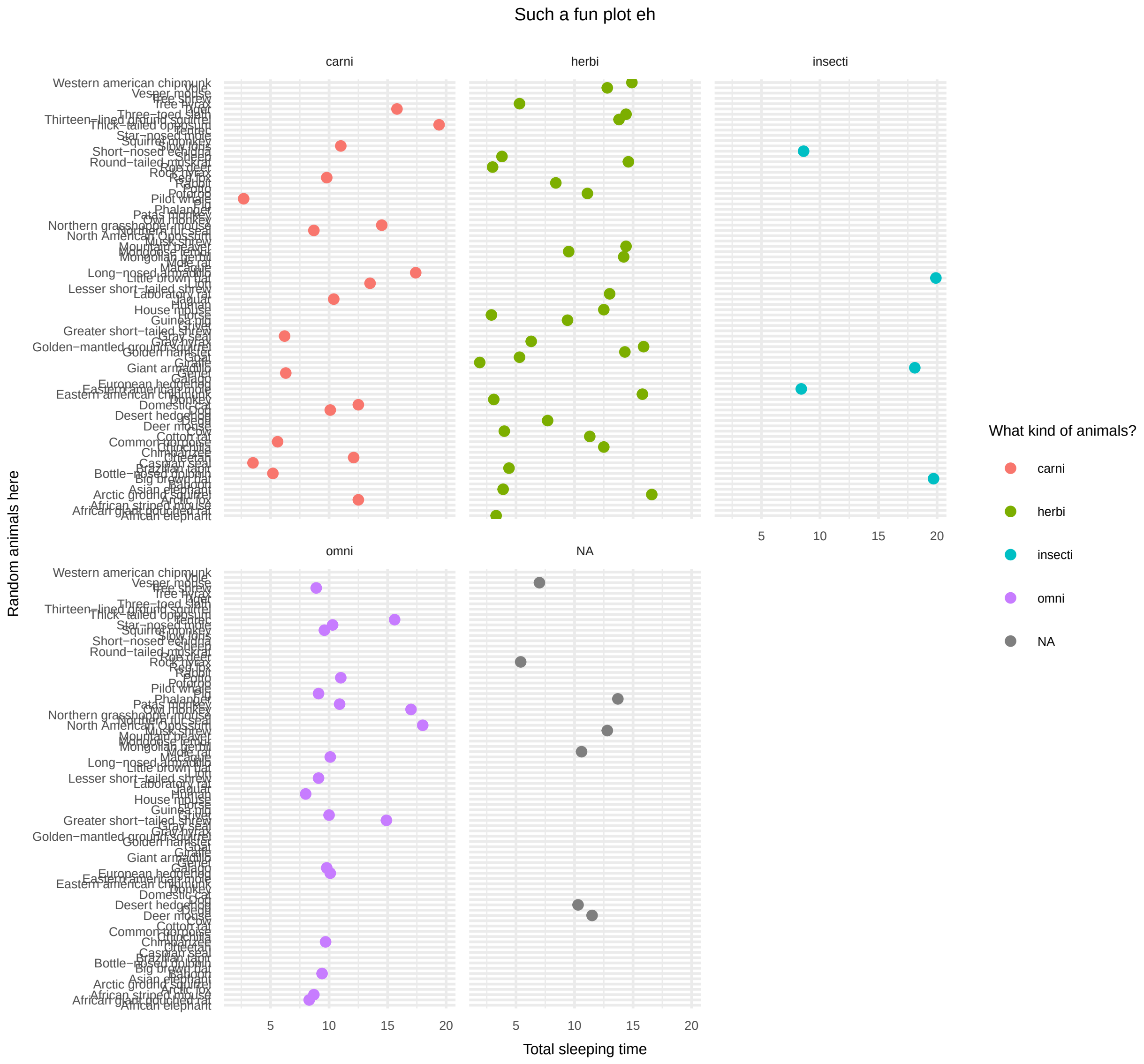

Density distribution of total sleep of different animals
density
0.0
0.1
0.2
0.3
5
10
15
20
Total sleep
vore
carni
herbi
insecti
omni
NA

Additional helpful little objects

FACETS

Perhaps you want to split the information in different panels. We can use the `facet_wrap` or `facet_grid` options for this.

```
ggplot(msleep, aes(y = name, x = sleep_total, colour = vore)) +
    geom_point() + theme_minimal() + theme(text = element_text(size = 6)) +
    ggtitle("Such a fun plot eh") + theme(plot.title = element_text(hjust = 0.5)) +
    labs(x = "Total sleeping time", y = "Random animals here",
        colour = "What kind of animals?") + facet_wrap(~vore)
```

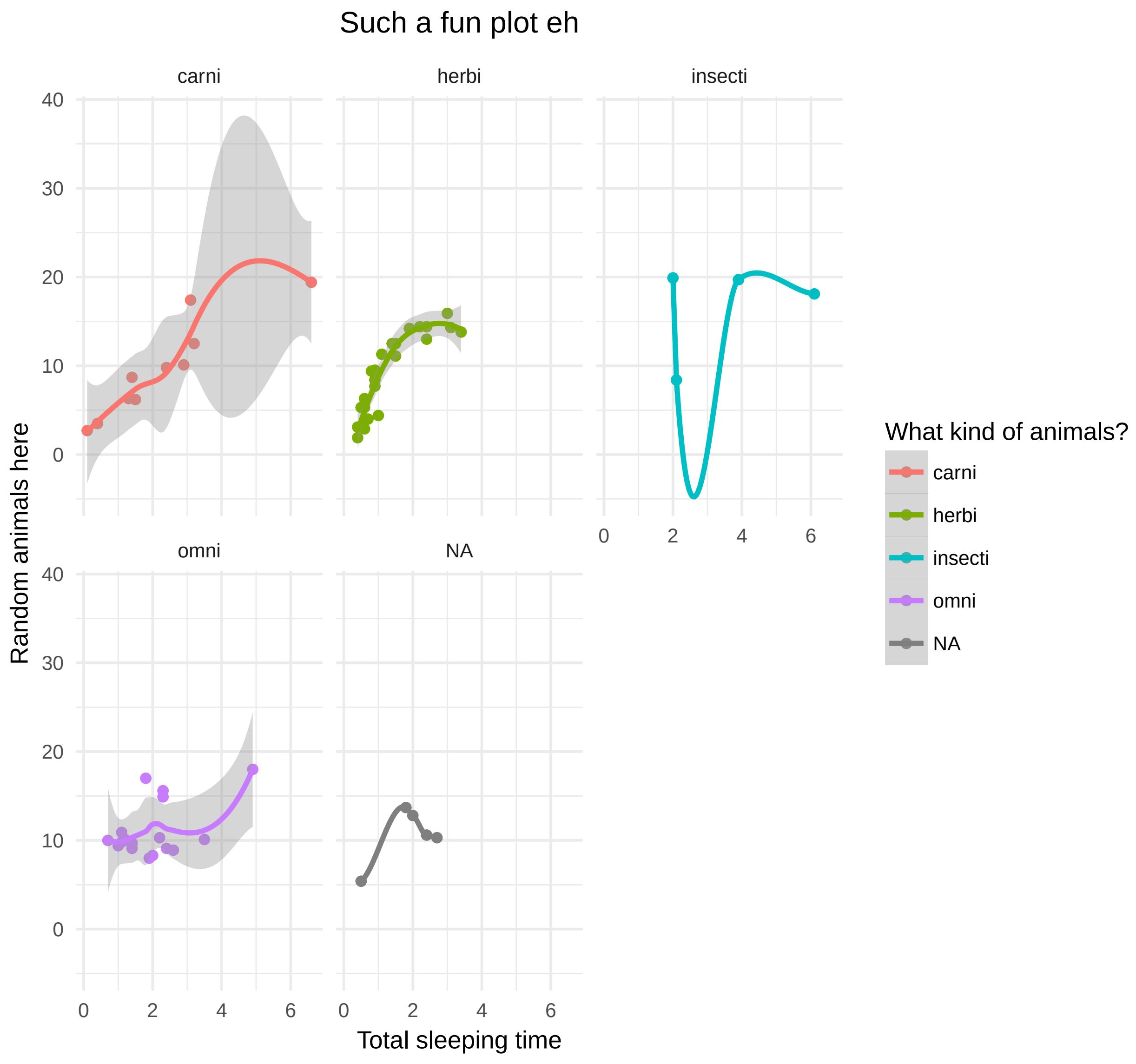


18

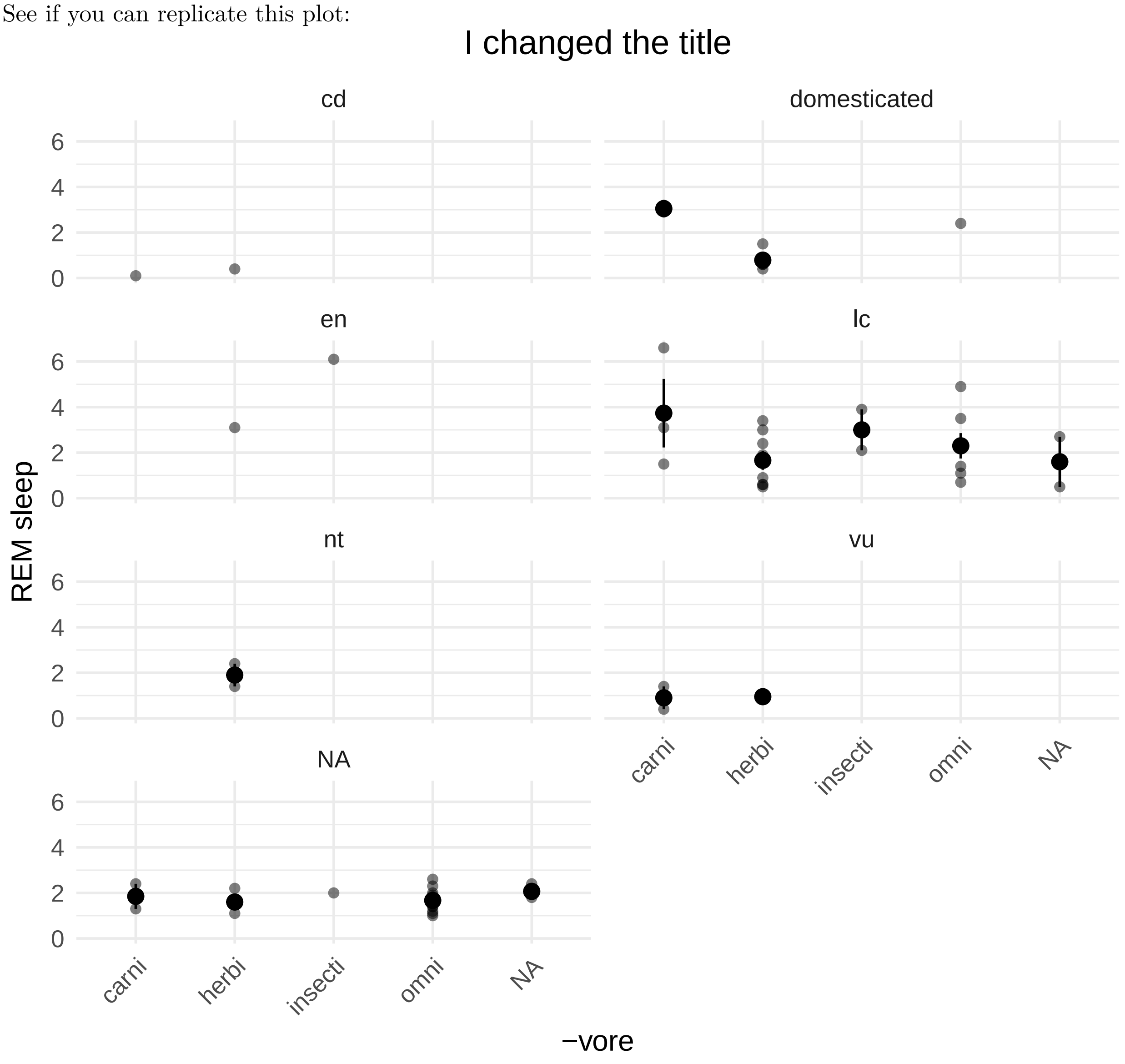
Smoothing function

We can also add in additional 'features' to our plots that will show what the trends are. The `stat_smooth` object by default does a loess fit on the ploints.
Here, let us plot the total sleep against the REM sleep, for all our entries in the msleep dataframe.

```
ggplot(msleep, aes(y = sleep_total, x = sleep_rem, colour = vore)) +
    geom_point() + theme_minimal() + theme(text = element_text(size = 10)) +
    ggtitle("Such a fun plot eh") + theme(plot.title = element_text(hjust = 0.5)) +
    labs(x = "Total sleeping time", y = "Random animals here",
        colour = "What kind of animals?") + facet_wrap(~vore) +
    stat_smooth()
```



EXERCISE
See if you can replicate this plot:



Hints:
- Look at the stat_summary function
- Can we fix the number of columns in the facet_wrap component?

Answer will be covered at the tutorial!

Extra readings

1. CHANGE THEMES

- Instead of always adding your theme of choice to the end of your ggplot command, you can change the default theme with the following command: `theme_set(theme_bw())`

2. You can also install other helpful packages for data science in R (including ggplot2), with the following command: `install.packages("tidyverse")`
   Look up www.tidyverse.org for more details!