Handout 02: Graphics in R

Taylor Arnold

One of the most important tasks in data analysis is visualizing the data. This is important from the first step of understanding a new dataset all the way through producing graphics for a final presentation. We will use a package called **ggplot** in order to produce visualizations this semester. The package provides a particular plotting paradigm that has a solid theoretical underpinning.

In this handout, I will use the `msleep` dataset in order to show various plots. This data is actually includes with the **ggplot2** package, and can be loaded as follows:

```
data(msleep)
```

The data contains information about the sleep patterns of 83 different mammals.[1]

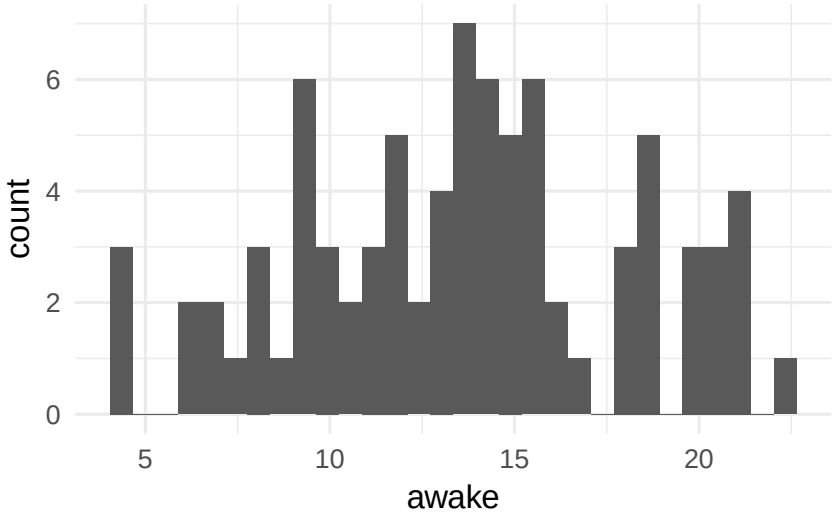
[1] V. M. Savage and G. B. West. A quantitative, theoretical framework for understanding mammalian sleep. Proceedings of the National Academy of Sciences, 104 (3):1051-1056, 2007.

Basic plotting

One numeric variable

The default plot for a single numeric variable is a *histogram*. To get the plot simply use the following:

```
qplot(awake, data = msleep)
```

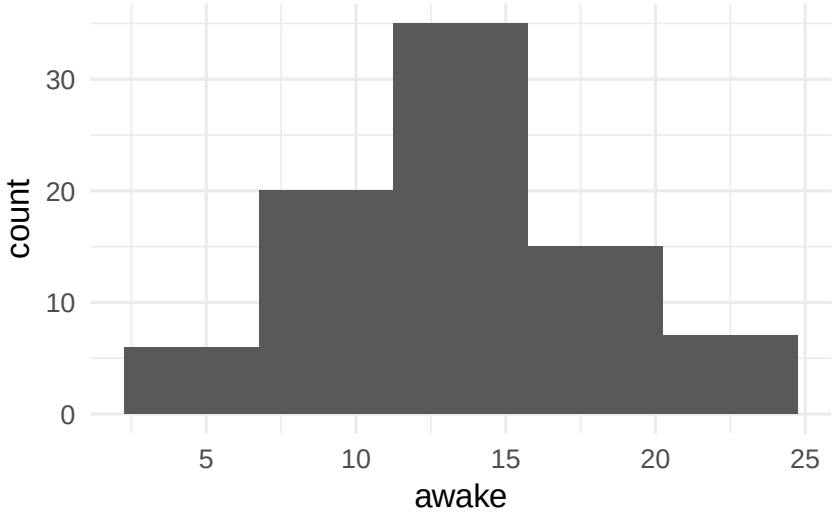


As described on Wikipedia, which I suggest looking at for more information if you are still unclear about the definition of a histogram:

A histogram is a graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable (quantitative variable) and was first introduced by Karl Pearson. To construct a histogram, the first step is to "bin" the range of values - that is, divide the entire range of values into a series of intervals - and then count how many values fall into each interval. The bins are usually specified as consecutive, non-overlapping intervals of a variable. The bins (intervals) must be adjacent, and are often (but are not required to be) of equal size.

By default the `qplot` function uses 30 bins, but this can be changed to any number you would like. Notice how changing the bins to include only 5 changes the plot:

```
qplot(awake, data = msleep, bins = 5)
```

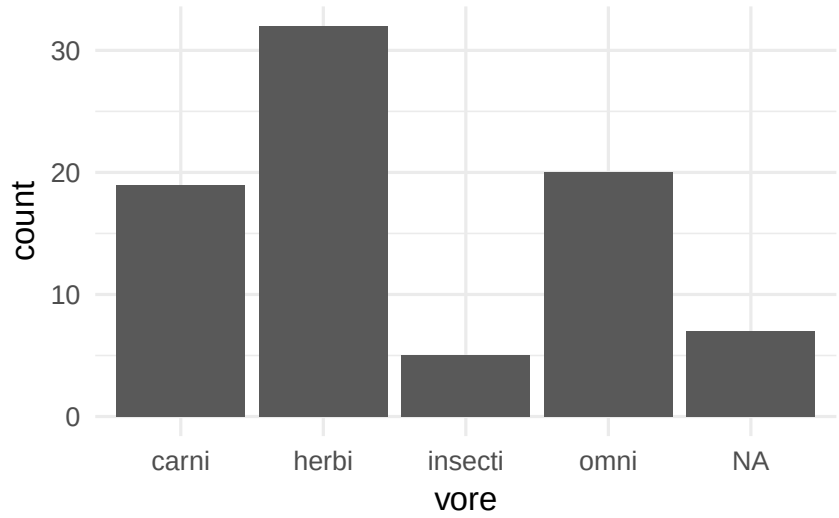


This makes the bins even more prominent. To test your knowledge, how many mammals are awake less than (about) 11 hours per day? You should be able to get this quickly from the plot.

One discrete variable

The default plot for a categorical variable is a *bar plot*. The code to generate it is exactly the same as for the histogram:

```
qplot(vore, data = msleep)
```



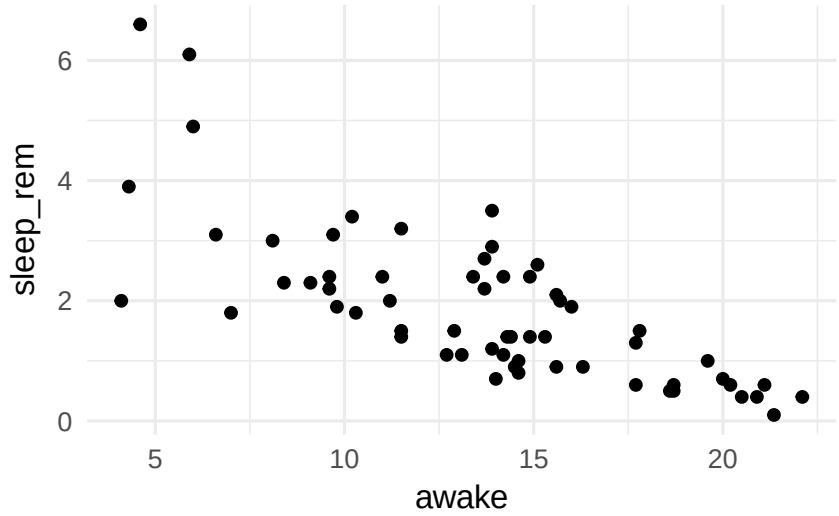
A bar plot is very similar to a histogram except that there is no need to bin the data into categories.

Two continuous variable

To plot two continuous variables, we simply supply both variables to the `qplot` function:

```
qplot(awake, sleep_rem, data = msleep)
```

```
## Warning: Removed 22 rows containing missing values
## (geom_point).
```

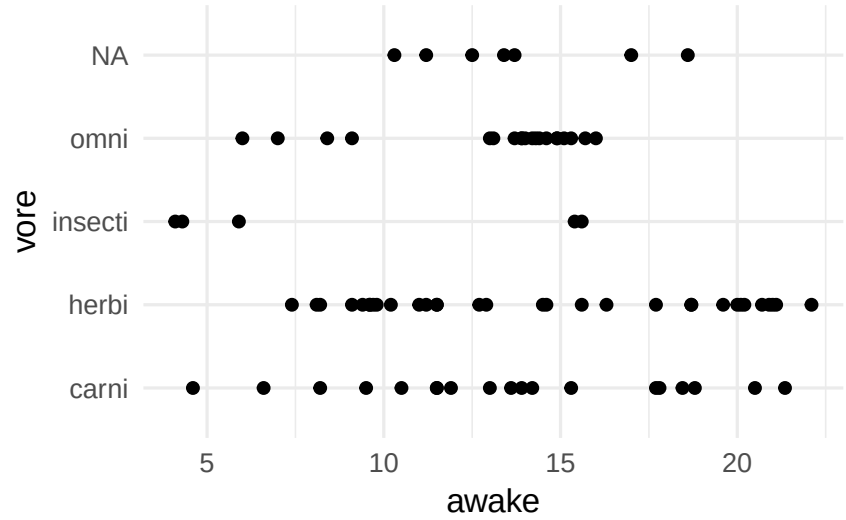


The result is a *scatter plot*, which you have almost certainly seen at some point. Each data point is represented by a point on the plot.

One continuous and one discrete variable

The default plot for two variables where one is discrete does not have a special name, but can be produced using a similar syntax:

```
qplot(awake, vore, data = msleep)
```



It resembles a scatter plot, except that one axis is a discrete categorical variable.

Aesthetics

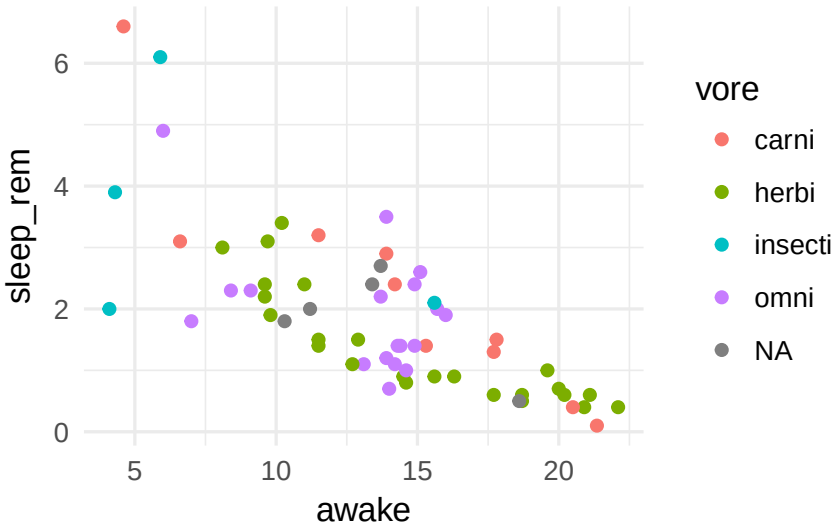
There are a number of graphical properties that we can manipulate on a plot, usually in the context of a scatter plot. The basic syntax behind these are discussed below. For each, there is a way of changing the properties either uniformly for every point or based on a third variable.

Color

To change the color of points based on a third variable, simply supply the color parameter to the function `qplot` with the variable name:

```
qplot(awake, sleep_rem, data = msleep, color = vore)
```

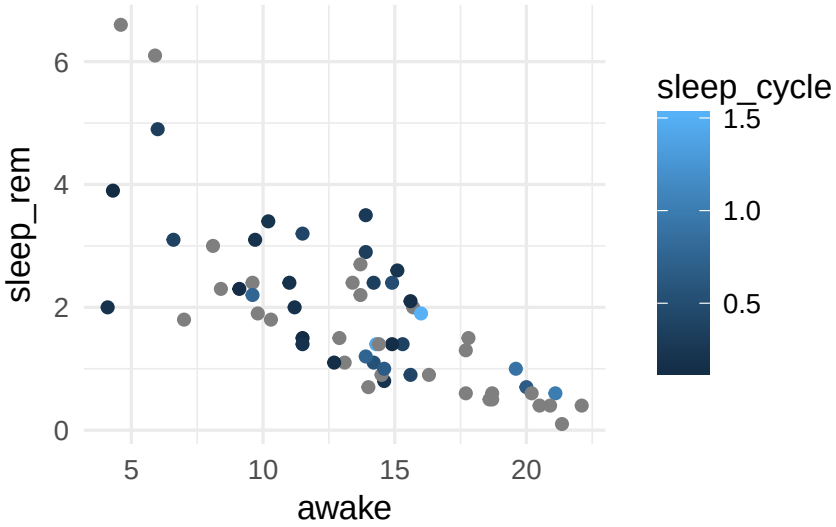
```
## Warning: Removed 22 rows containing missing values
## (geom_point).
```



Notice that a legend appears to describe what each color represents. We can also use a continuous variable to specify color with the same syntax:

```
qplot(awake, sleep_rem, data = msleep, color = sleep_cycle)
```

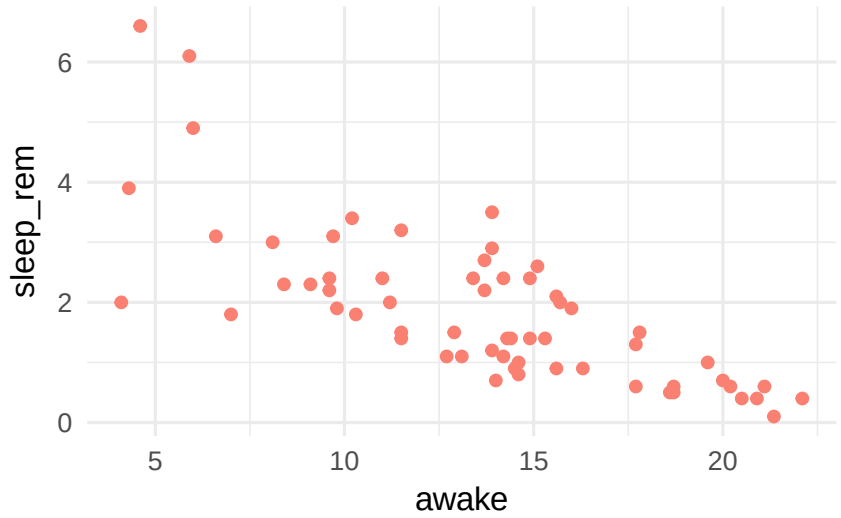
```
## Warning: Removed 22 rows containing missing values
## (geom_point).
```



If we instead want to change every color uniformly, we instead wrap the name of the color in quotes and the function `I`:

```
qplot(awake, sleep_rem, data = msleep, color = I("salmon"))
```

```
## Warning: Removed 22 rows containing missing values
## (geom_point).
```



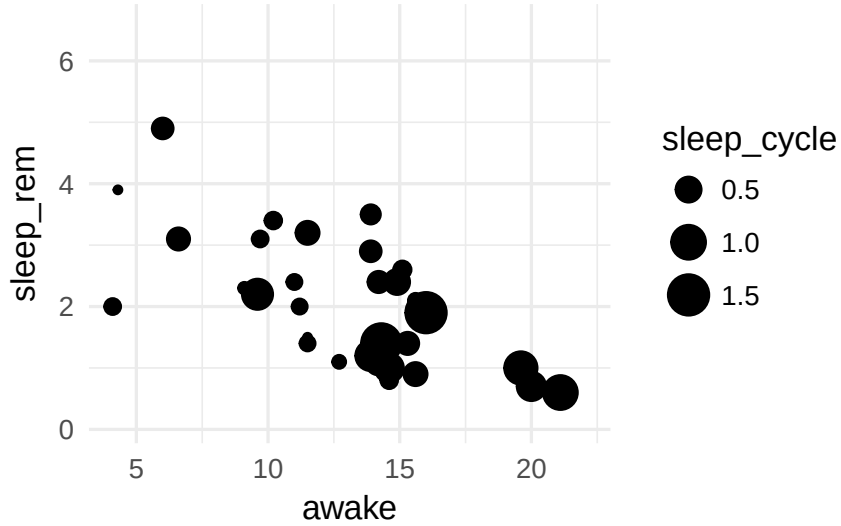
There are a lot of colors to choose from in R; to see all of the options type `colors()` in the console.

Size

It is also possible to change the size of the points in a similar fashion. Here is does not make sense to use a categorical variable, so we will stick to numeric variables:

```
qplot(awake, sleep_rem, data = msleep, size = sleep_cycle)
```

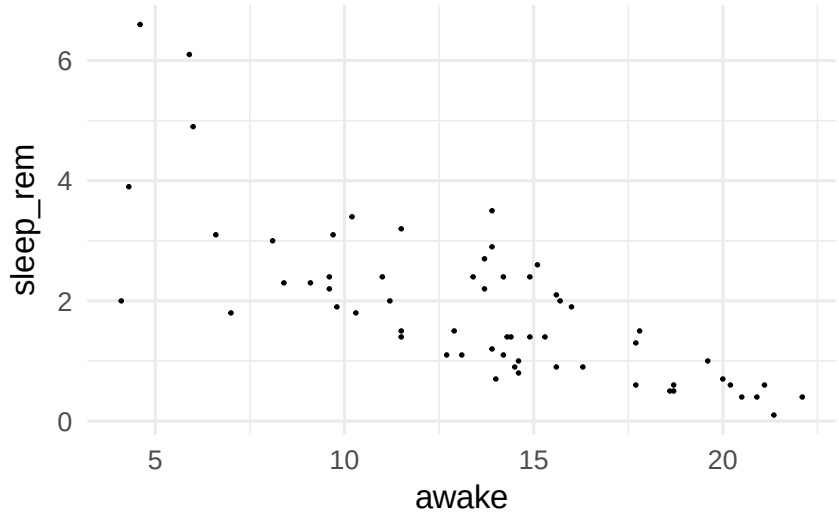
```
## Warning: Removed 51 rows containing missing values
## (geom_point).
```



This kind of scatter plot is called a *bubble chart*; we saw an example of this in Hans Roslin's video. It is also possible to change the size uniformly for all points, using the `I` function once again:

```
qplot(awake, sleep_rem, data = msleep, size = I(0.2))
```

```
## Warning: Removed 22 rows containing missing values
## (geom_point).
```



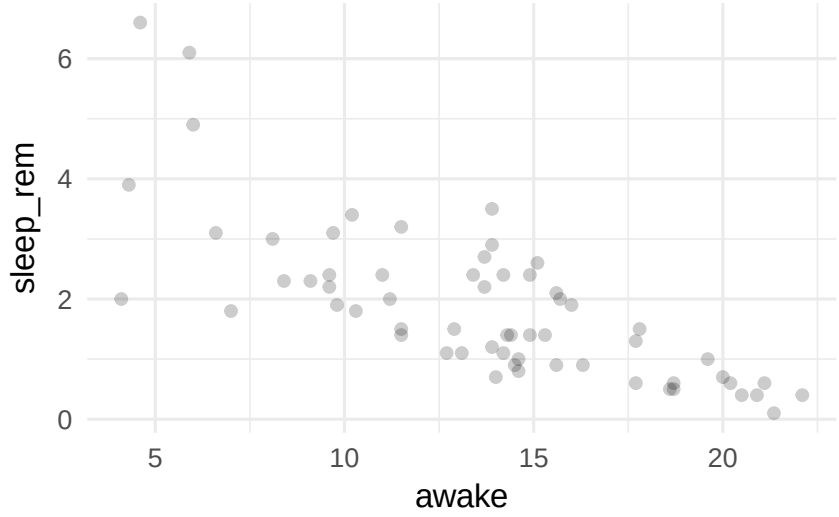
For reference, the default size is 1.

Alpha

Finally, we can also change the opacity of the points. That is, to what extent are the points see-through. The parameter that controls this is called `alpha`; when set to 1 (the default) the point is not at all see-through and when set to 0 it is invisible. Notice what happens when I set it to 0.2:

```
qplot(awake, sleep_rem, data = msleep, alpha = I(0.2))
```

```
## Warning: Removed 22 rows containing missing values
## (geom_point).
```



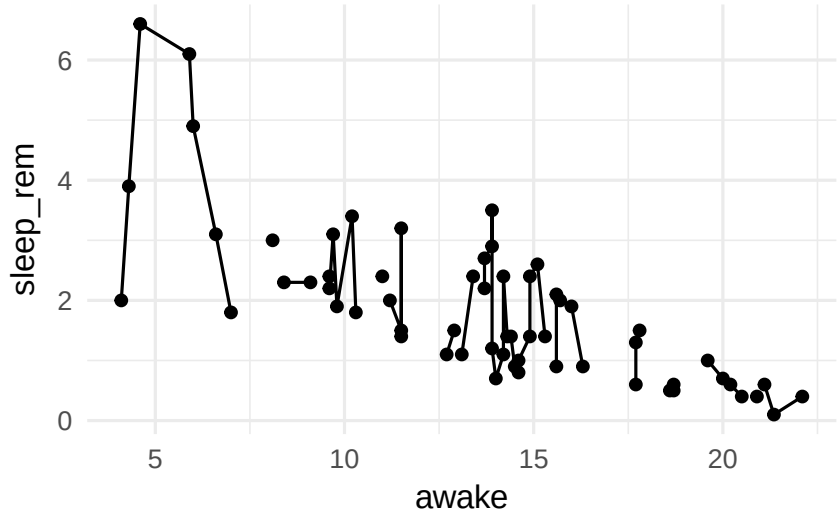
It is possible to set the alpha value to change with a third variable, though this is rarely very useful.

Adding layers

It is often useful to add additional plot tops over the top of the default types supplied by `qplot`. To do this we simply add functions that start with `geom_`. For example, a line plot (which actually does not make any sense here) can be produce by:

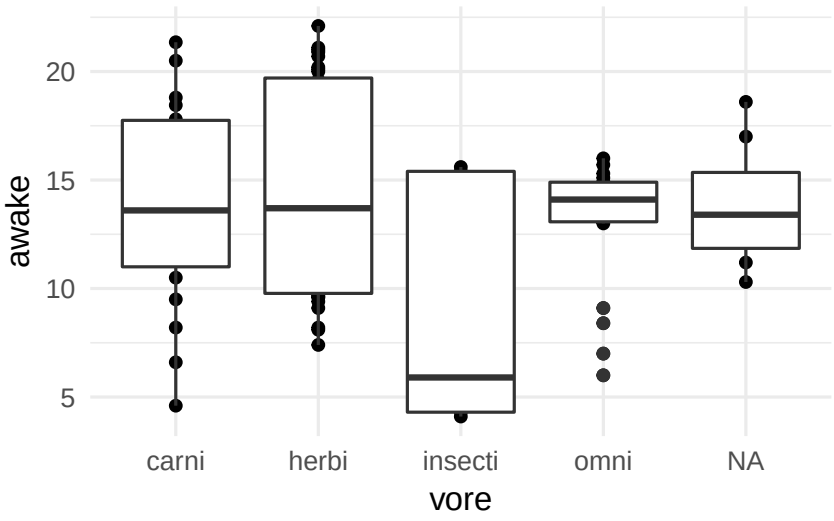
```
qplot(awake, sleep_rem, data = msleep) + geom_line()
```

```
## Warning: Removed 22 rows containing missing values
## (geom_point).
```



It looks strange in part because of missing values in the dataset. Another common alternative plot when we have both a continuous and a categorical variable is called a *box plot*:

```
qplot(vore, awake, data = msleep) + geom_boxplot()
```

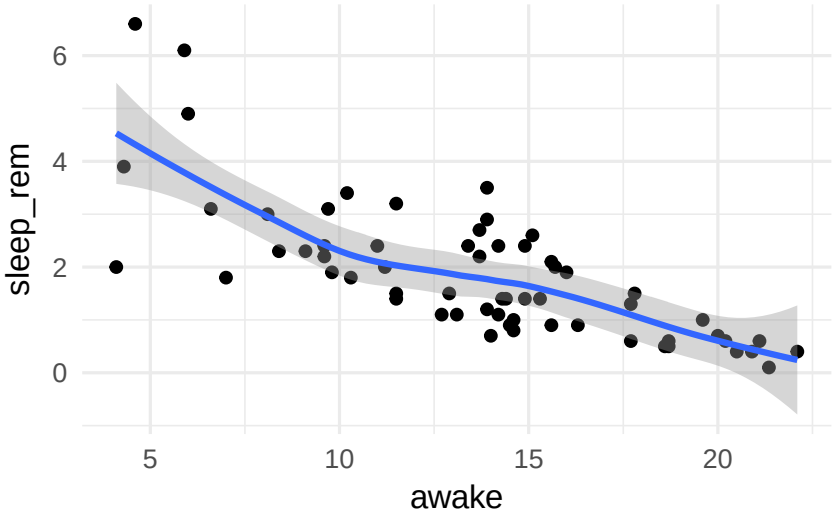


The details of this plot will be described in a future handout. We can also add a smoothing curve with:

```
qplot(awake, sleep_rem, data = msleep) + geom_smooth()
```

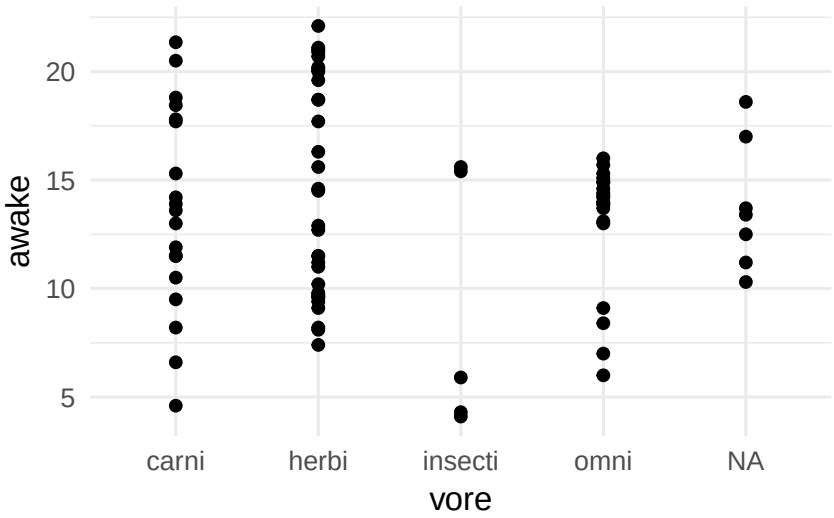
```
## Warning: Removed 22 rows containing non-finite
## values (stat_smooth).
```

```
## Warning: Removed 22 rows containing missing values
## (geom_point).
```



If we want a best fit line, the option `method` can be set to `lm`:

```
qplot(vore, awake, data = msleep) + geom_smooth(method = "lm")
```



These plots will be very helpful as we approach statistical modelling in a few weeks.