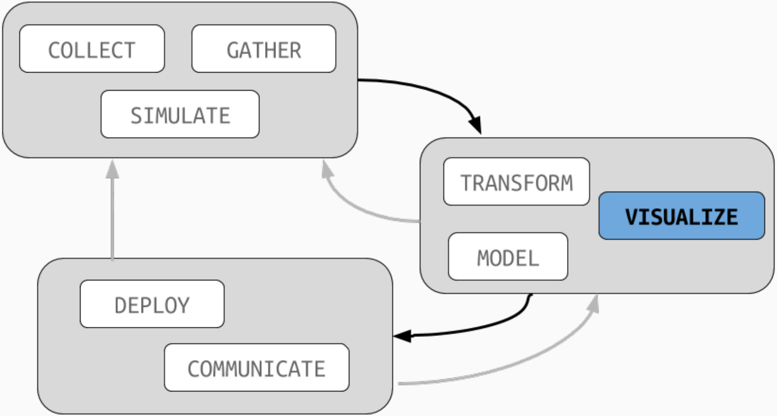
Lecture 03: Graphics with qplot

Taylor Arnold

Graphics in R


COLLECT
GATHER
SIMULATE
TRANSFORM
VISUALIZE
MODEL
DEPLOY
COMMUNICATE

Introduction

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 **ggplot2** in order to produce visualizations this semester. The package provides a particular plotting paradigm that has a solid theoretical underpinning.

Dataset

In todays notes, 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:

```r
library(ggplot2)
data(msleep)
```

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

Basic plotting

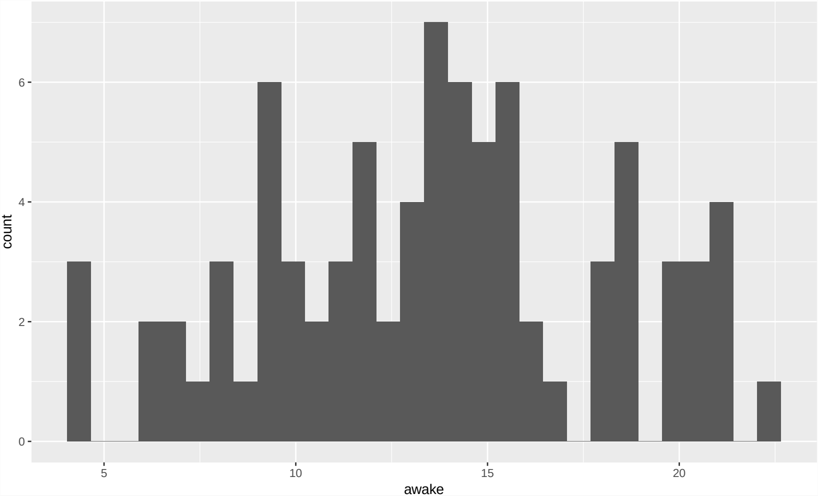
One numeric variable

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

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

One numeric variable

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



Histograms

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.*

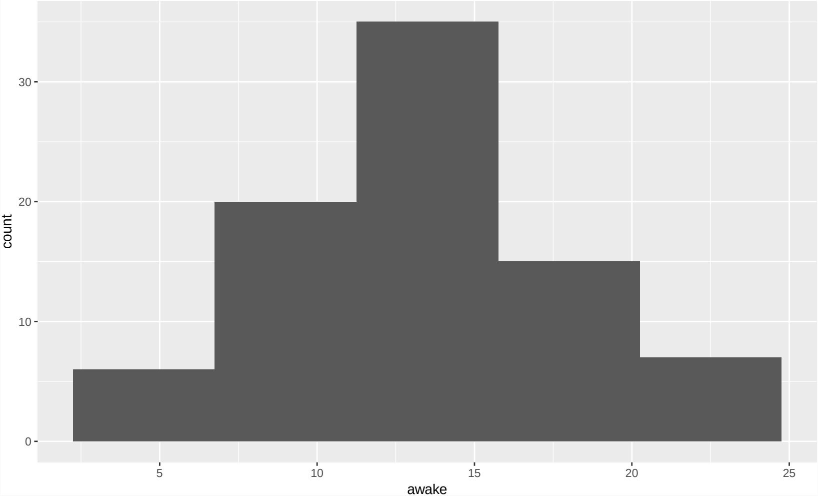
Bin size

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)
```

Bin size

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



Bin size

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.

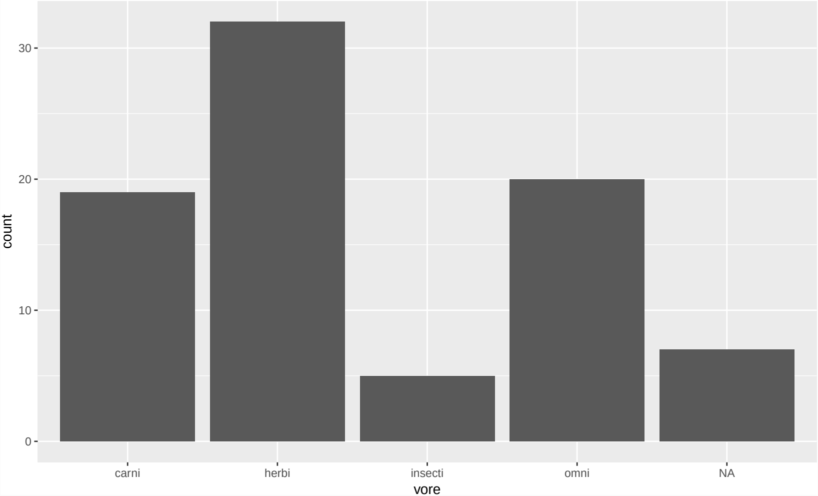
One discrete variable

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)
```

One discrete variable

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



Bar plots

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

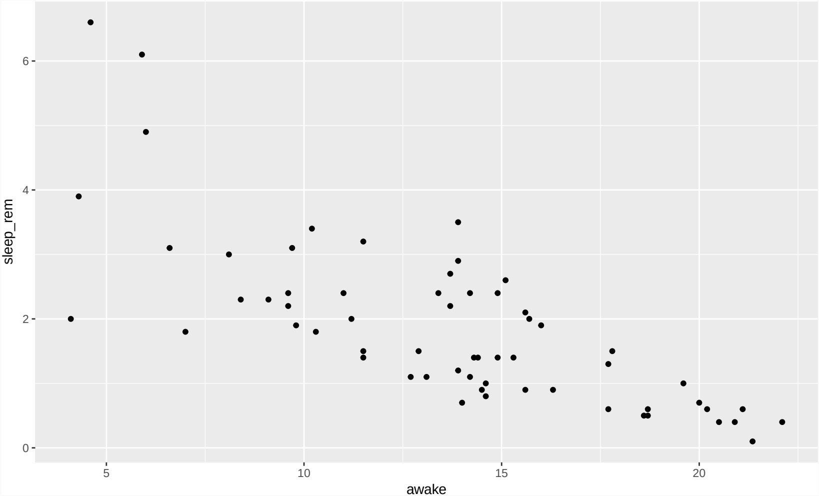
Two continuous variables

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

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

Two continuous variables

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



Scatter plot

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.

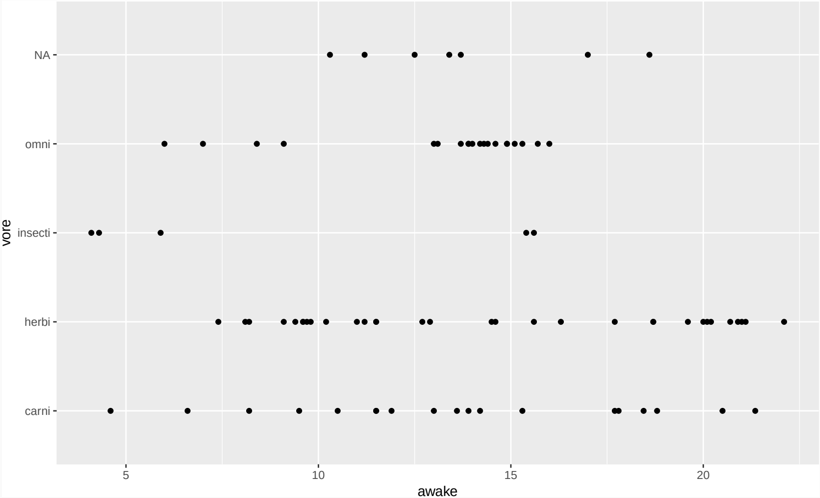
One continuous and one discrete variable

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

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

One continuous and one discrete variable

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



One continuous and one discrete variable

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

Aesthetics

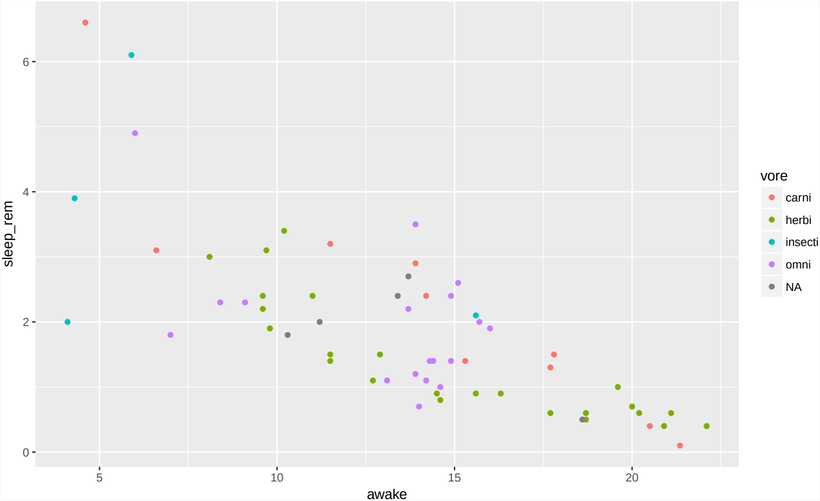
Color

To change the color of points based on a third variable, simply supply the color parameter to the function `qplot` with the variable name. Notice that a legend appears to describe what each color represents.

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

Color

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



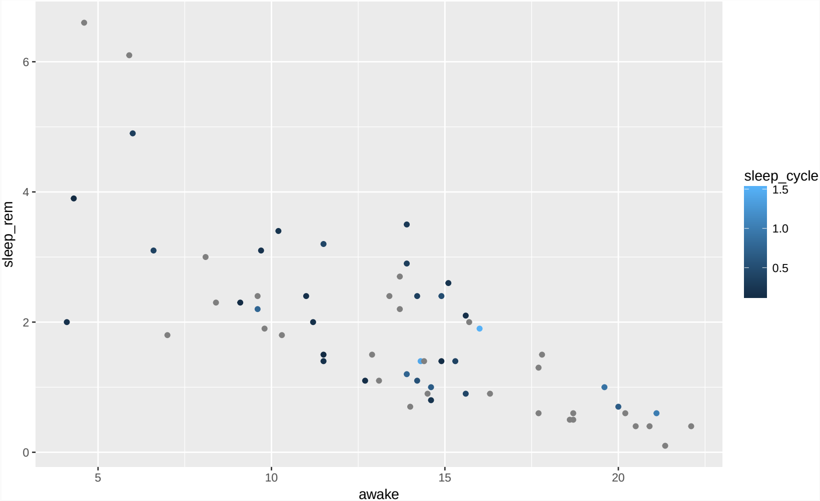
Continuous color

We can also use a continuous variable to specify color with the same syntax:

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

Continuous color

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



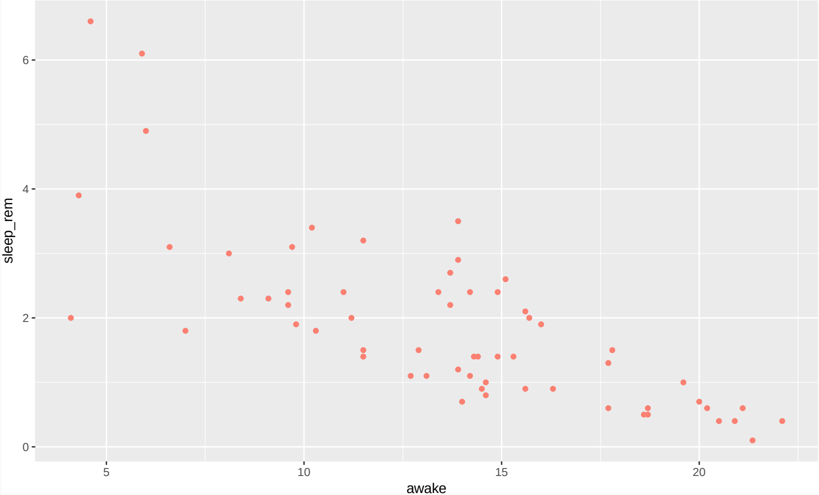
Fixed color

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

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

Fixed color

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



Fixed color

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

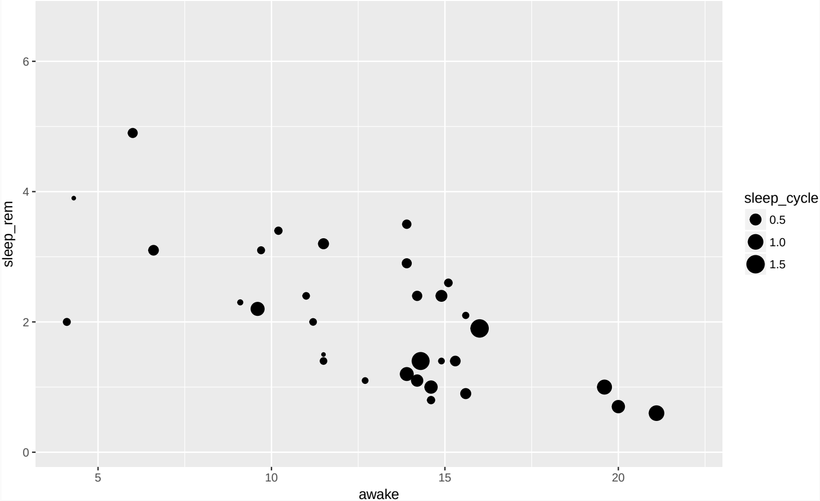
Size

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)
```

Size

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



Bubble chart

This kind of scatter plot is called a *bubble chart*; we saw an example of this in Hans Roslin's video.

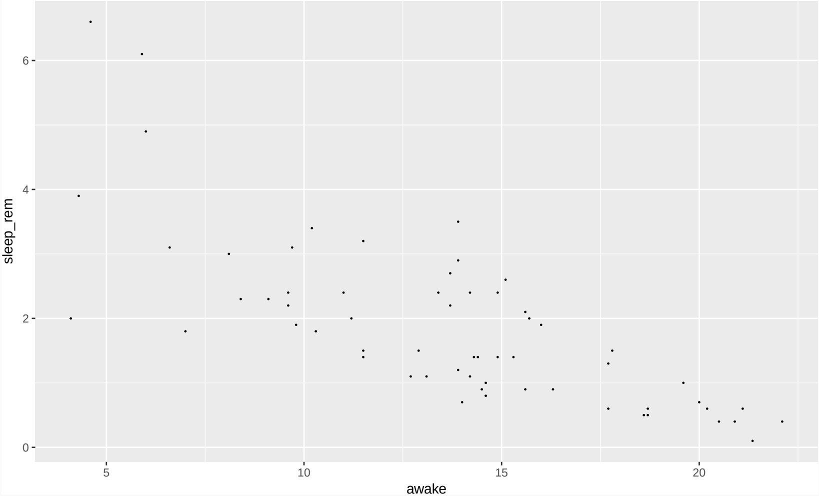
Uniform size

It is also possible to change the size uniformly for all points, using the I function once again. For reference, the default size is 1.

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

Uniform size

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



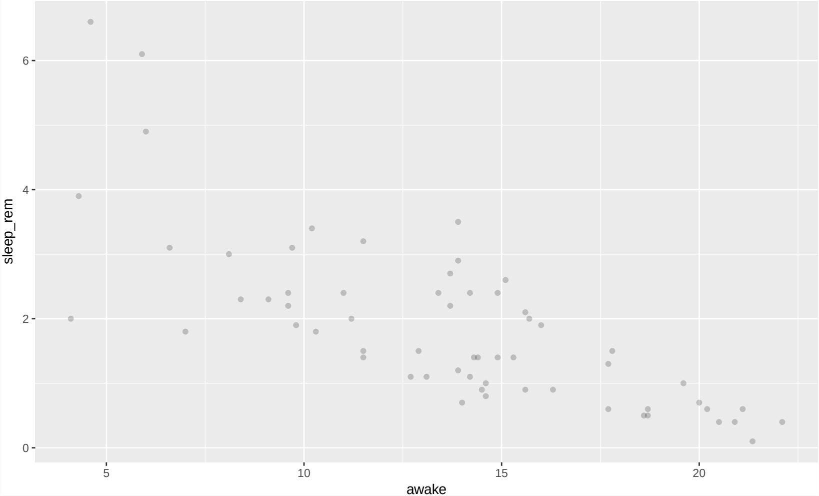
Alpha

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:

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

Alpha

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



Varying alpha

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