

# Lecture 04: Layered and Interactive Graphics

---

Taylor Arnold

# Grammar of Graphics

---

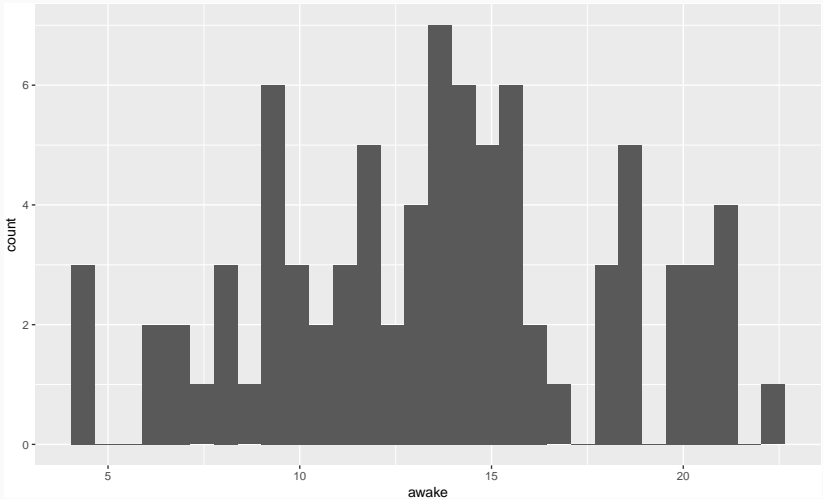
Last time, we used the **ggplot2** package to construct graphics using the `qplot` function.

The philosophy of the package is that graphics should be built by combining graphical elements called *layers*. We combine these in R by using the `+` sign.

For example, take one of the plots from last time:

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

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



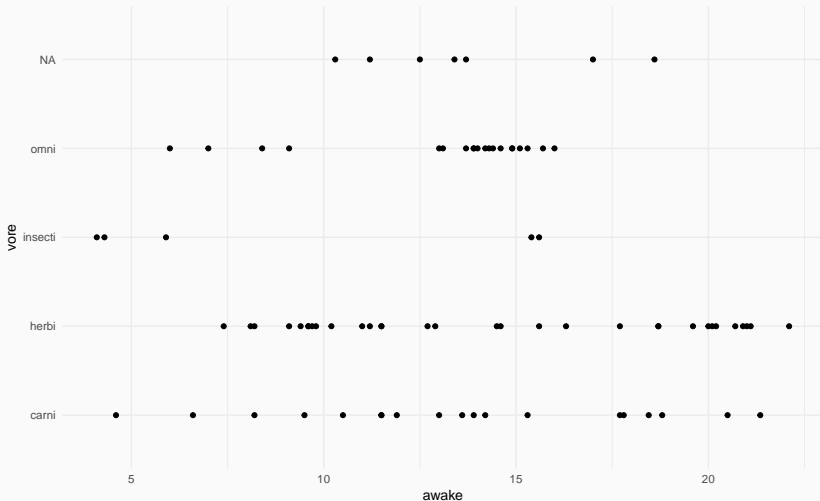
We can add a layer called a theme to the plot to change the way the plot looks. Personally, I prefer the minimal theme:

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

Most notably, output no longer has a grey background. Much of the fonts and other elements are also cleaned up a bit from the default.

# themes

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



Today we'll see some of the most useful layers that can be added to a plot.



## geom layers

---

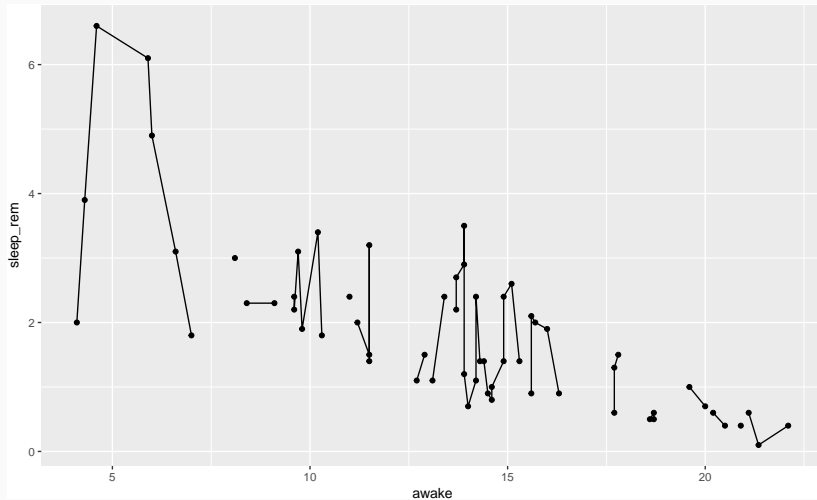
It is often useful to add additional plots over the top of the default types supplied by `qplot`. To do this we simply add functions that start with `geom_`. For example, a line plot (which actually does not make any sense here) can be produce by:

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

It looks strange in part because of missing values in the dataset.

# geom\_line

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

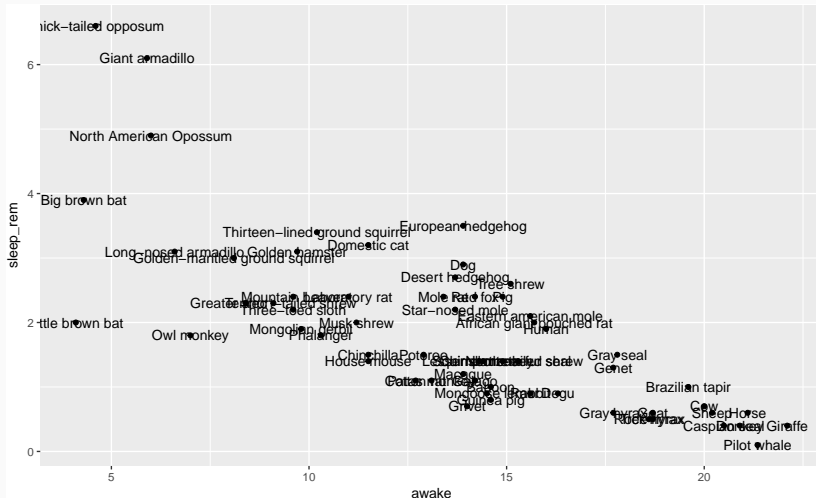


We might also want to display the names of the animals on the plot. To do, so we use the `geom_text` layer. However, this requires that we give `qplot` a `label` aesthetic.

```
qplot(awake, sleep_rem, data = msleep, label = name) +  
  geom_text()
```

# geom\_text

```
qplot(awake, sleep_rem, data = msleep, label = name) +  
  geom_text()
```



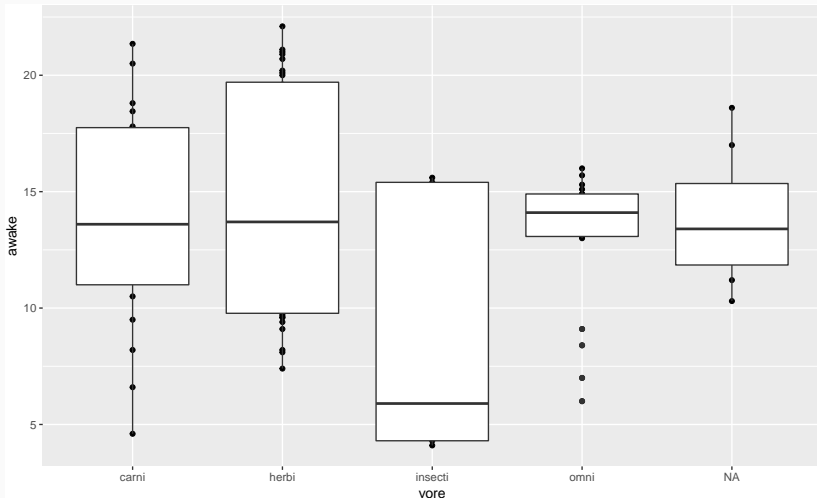
Another common alternative plot when we have both a continuous and a categorical variable is called a *box plot*:

```
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.

# geom\_boxplot

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



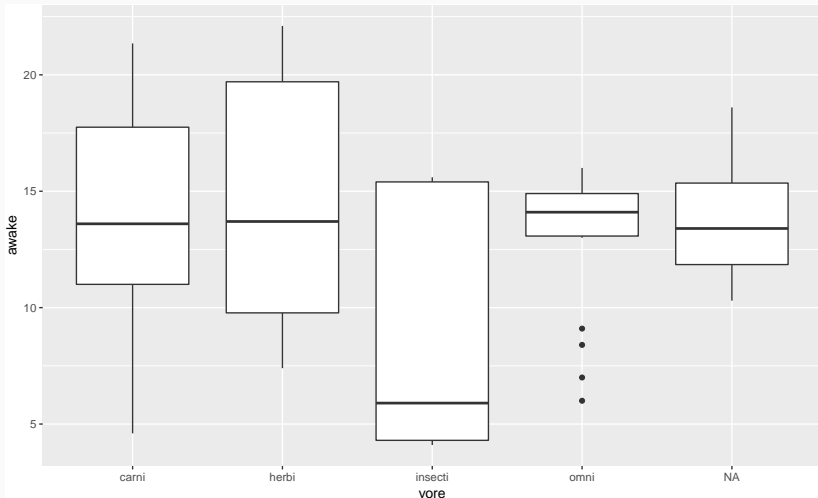
With the boxplot, we probably want to suppress the default `qplot` layer rather than just adding on top of it. To do this, set the `geom` argument of `qplot` to “blank”:

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



## geom\_blank

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



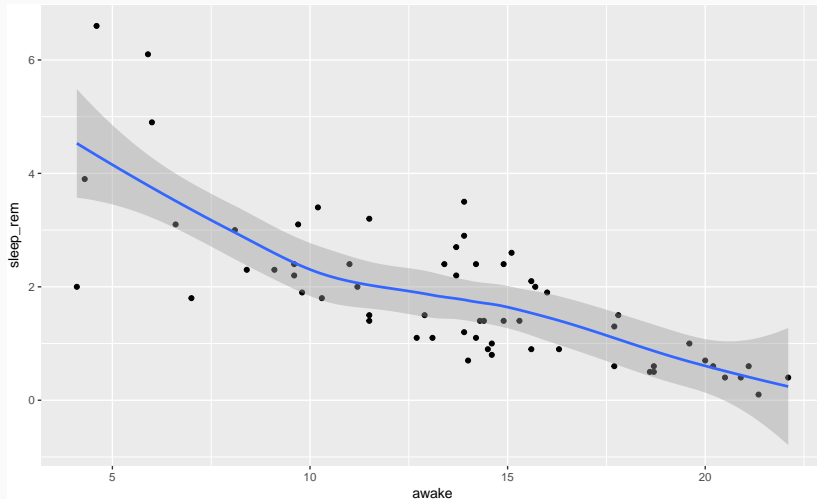
We can also add a smoothing curve with:

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

This adds a line that attempts to run through the data points without wiggling too much.

# geom\_smooth

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



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

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

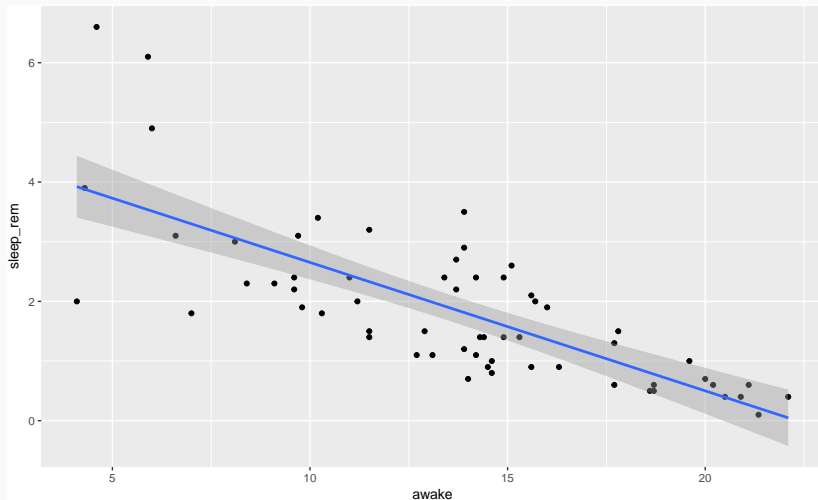
These plots will be very helpful as we approach statistical modelling in the upcoming weeks.

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

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

# geom\_smooth

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



## facets

---

One particularly powerful layer that we can add to plots is the `facet_wrap` layer. As an input we start with the `~` operator (more on this latter) followed by the name of a continuous variable.

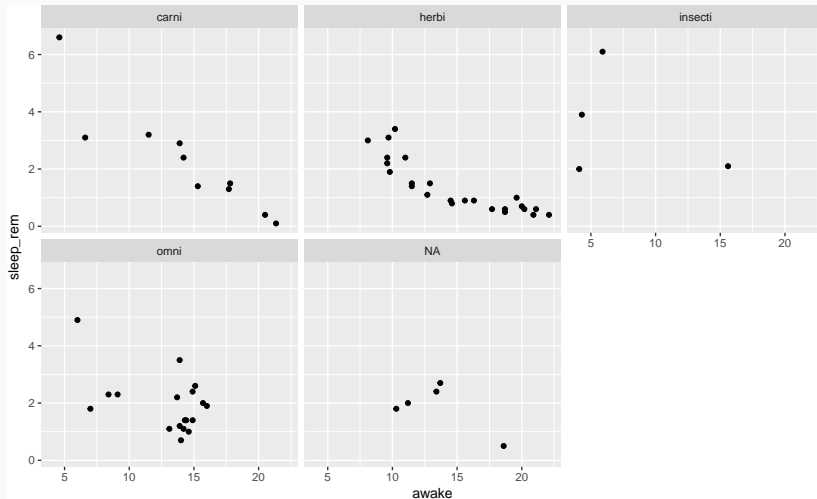
```
qplot(awake, sleep_rem, data = msleep) +  
  facet_wrap(~vore)
```

The package then create the requested plot for every single unique value of the discrete variable.



# facet\_wrap

```
qplot(awake, sleep_rem, data = msleep) +  
  facet_wrap(~vore)
```



**plotly**

---

The **plotly** package allows us to turn ordinary graphics into interactive ones. Simply load the library, run any standard ggplot graphic:

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

And then call the ggplotly function:

```
ggplotly()
```

# interactive graphics



By default, **plotly** includes any variables used for the plot in the tooltip (the box that comes up when you hover over a point).

Note that we can include a label name without actually using text labels to include them in the interactive plot.

For example:

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



- ▶ interactive graphics are a fantastic tool in doing data analysis
- ▶ use caution with larger datasets; consider sub-sampling or only working with a smaller collection
- ▶ when possible, try to replicate the most useful interactive views with static graphics for reproducibility and publication purposes