Handout 03: Data Types in R

Taylor Arnold

We have already discussed that variables in R have different *data types*, most likely one of the following:

- `int` stands for integers.
- `dbl` stands for doubles, or real numbers.
- `chr` stands for character vectors, or strings.

And we can see these when printing out the data, such as with the mammals sleep data:

```
data(msleep)
msleep
```

```
## # A tibble: 83 × 11
##                            name       genus
##                           <chr>       <chr>
## 1                       Cheetah    Acinonyx
## 2                   Owl monkey       Aotus
## 3              Mountain beaver  Aplodontia
## 4  Greater short-tailed shrew     Blarina
## 5                           Cow         Bos
## 6             Three-toed sloth    Bradypus
## 7           Northern fur seal Callorhinus
## 8                Vesper mouse     Calomys
## 9                           Dog       Canis
## 10                    Roe deer   Capreolus
## # ... with 73 more rows, and 9 more
## #   variables: vore <chr>, order <chr>,
## #   conservation <chr>, sleep_total <dbl>,
## #   sleep_rem <dbl>, sleep_cycle <dbl>,
## #   awake <dbl>, brainwt <dbl>, bodywt <dbl>
```

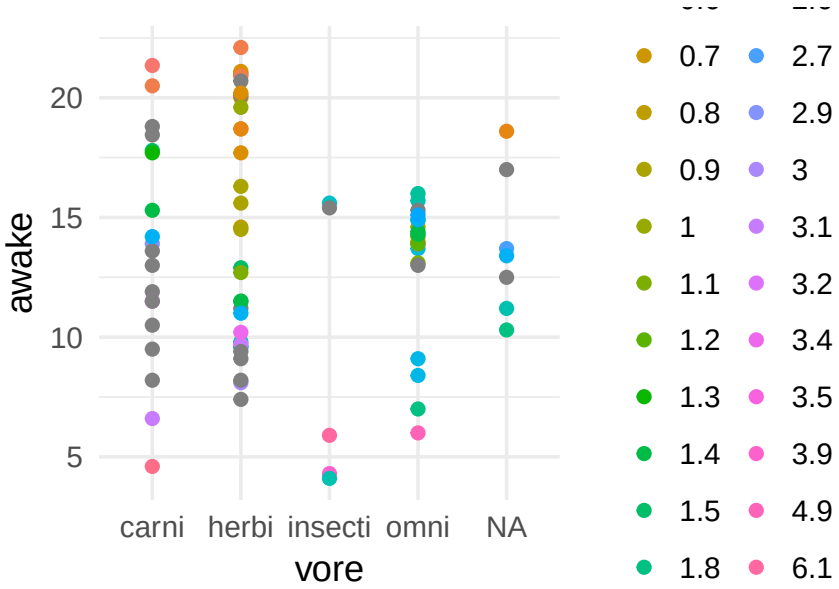
The names and genus are characters whereas numbers such as `awake` are doubles. This handout shows how to convert numeric variables into categorical ones and way to manipulate categorical variables. There is generally no straightforward way to convert a categorical variable into a number, so we do not discuss that here.

Numeric to Categorical

With factor()

To convert every unique values of a numeric variable to a category in a new categorical variable, we simply use the function `factor`. For example, we can convert `sleep_rem` into categories. This changes, amongst other things, the way color is used in a plot:

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



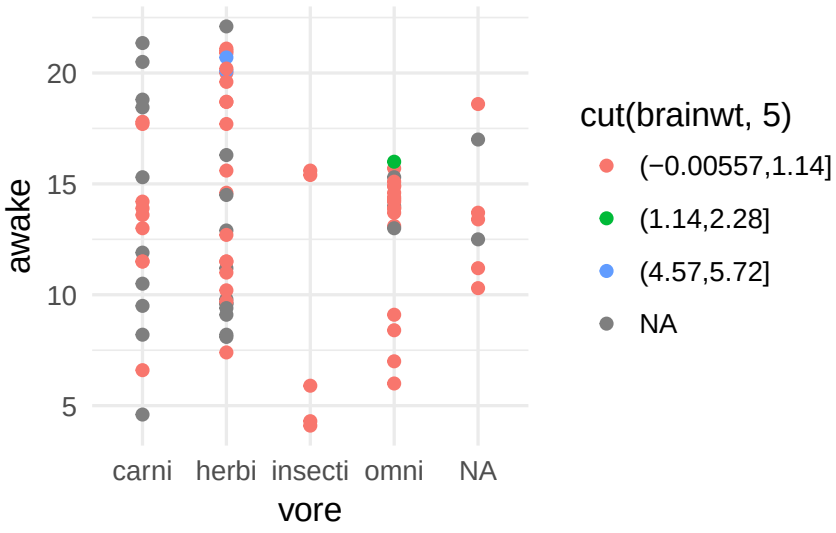
Notice that missing values became there own category. This is often very useful.

With cut() and bin()

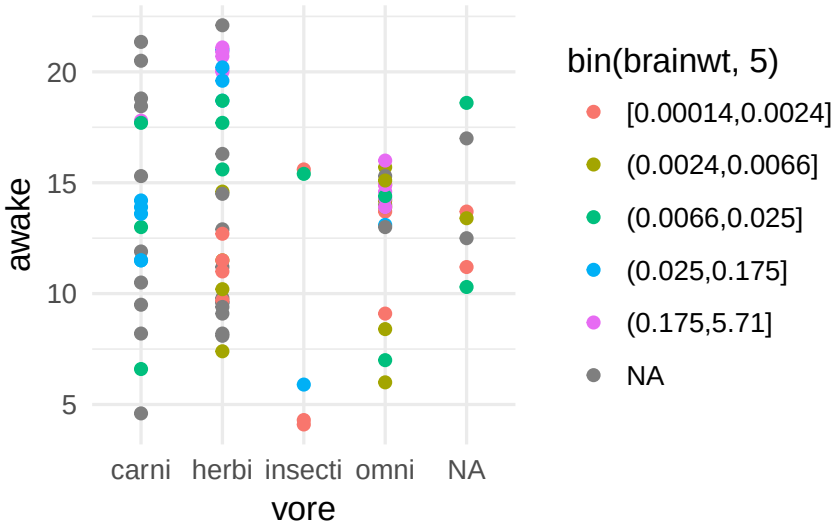
Often we do not want each numeric variable to be its own category but instead wish to group values that are close into a single category. There are two related functions for doing this: `cut` and `bin`. Both take the variable followed by the number of bins to use. The function `cut` splits the range of the variable into equally spaced buckets where as `bin` breaks the range up so that each bin has (roughly) the same proportion of data points.

For most variables the difference between these two functions is small; I usually use `cut` because the interval cutoffs print out nicer. However, when a variable is fairly skewed, I find that `bin` works better. For example, notice how different cutting and binning the `brainwt` variable is:

```
qplot(vore, awake, data = msleep, color = cut(brainwt, 5))
```




```
qplot(vore, awake, data = msleep, color = bin(brainwt, 5))
```



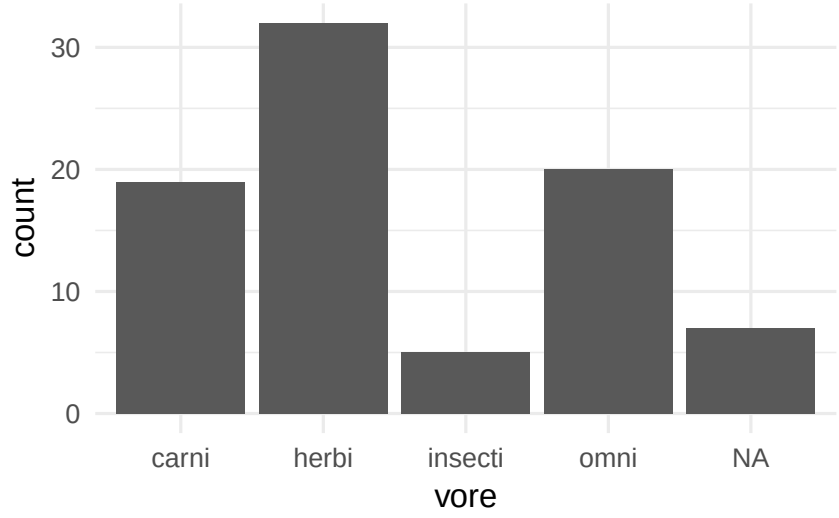
This variable is very skewed and the cutting algorithm puts almost all of the mammals in the same category.

Manipulating Categorical Data

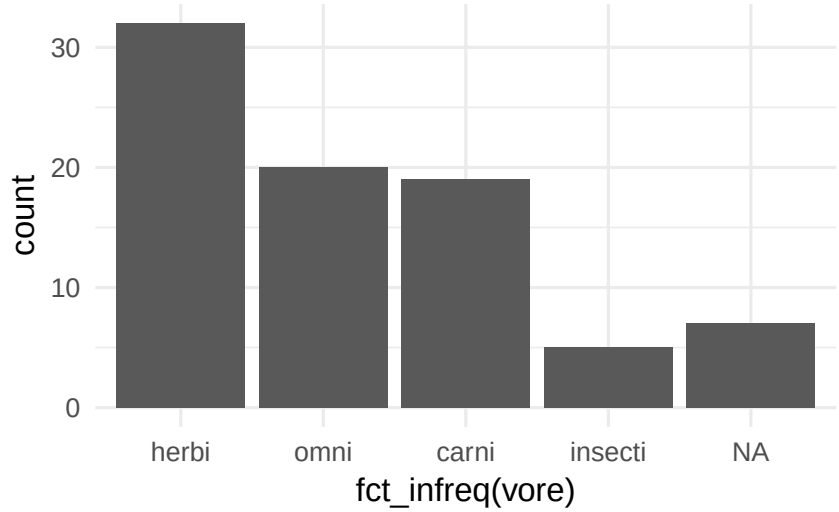
Likewise, it is sometimes useful to convert one categorical variable into another categorical variable by grouping or ordering the categories in a different way.

By default categories are order alphabetically. The function `fct_inorder` instead arranges them in their order of appearance. Compare, for example the following:

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

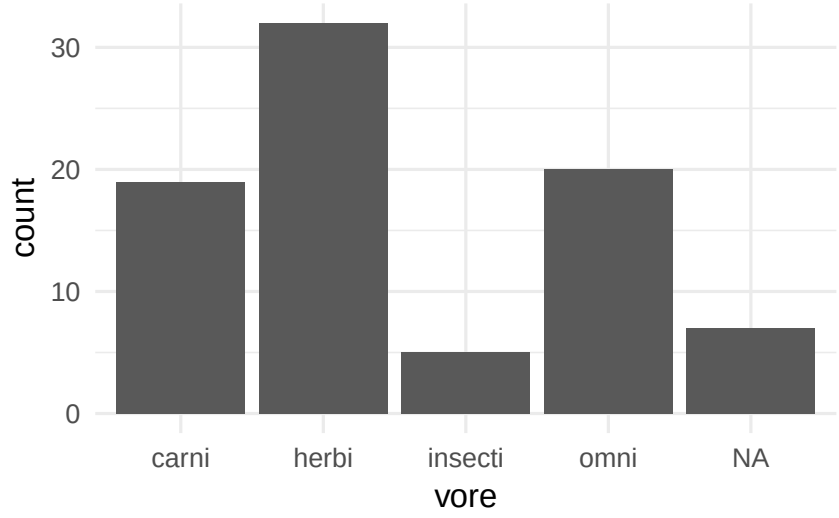



```
qplot(fct_infreq(vore), data = msleep)
```

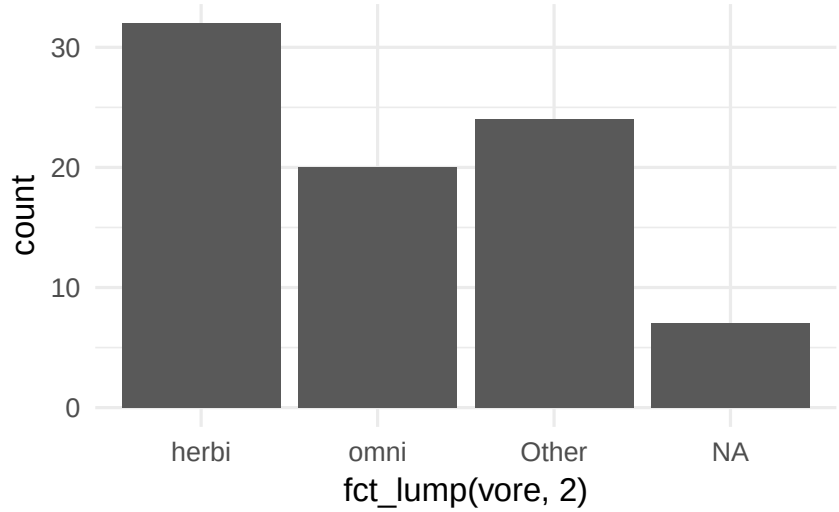


The function `fct_lump` take a second argument that gives the maximum number of categories allowed. The most frequent categories are included and all other categories are lumped into the `other` category.

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

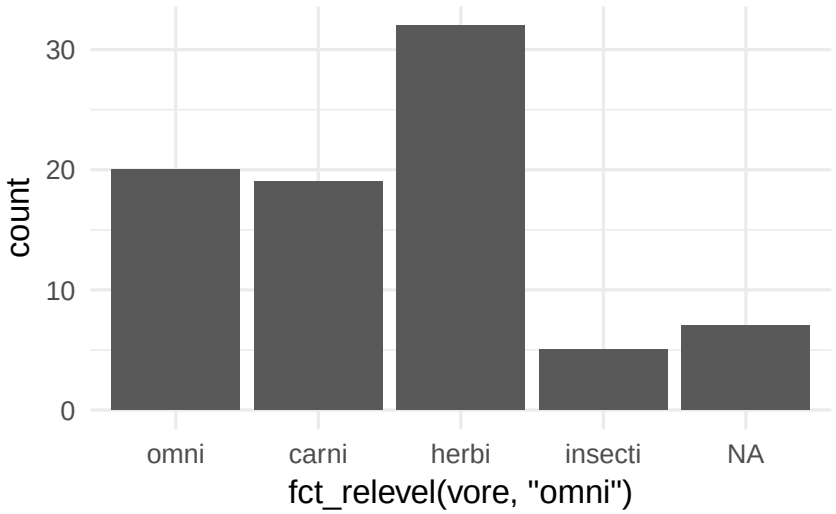



```
qplot(fct_lump(vore, 2), data = msleep)
```



Finally, the function `fct_relevel` takes a second argument that gives the category that should be first:

```
qplot(fct_relevel(vore, "omni"), data = msleep)
```



This latter function may seem odd here but will be indispensable when used with the modeling functions that will be introduced in a few weeks.