Efficiency in coding

Kasia Banas

The DRY principle

DONT REPEAT YOURSELF

First rule of coding

Don’t repeat yourself

Remember the CDC dataset from Week 1?

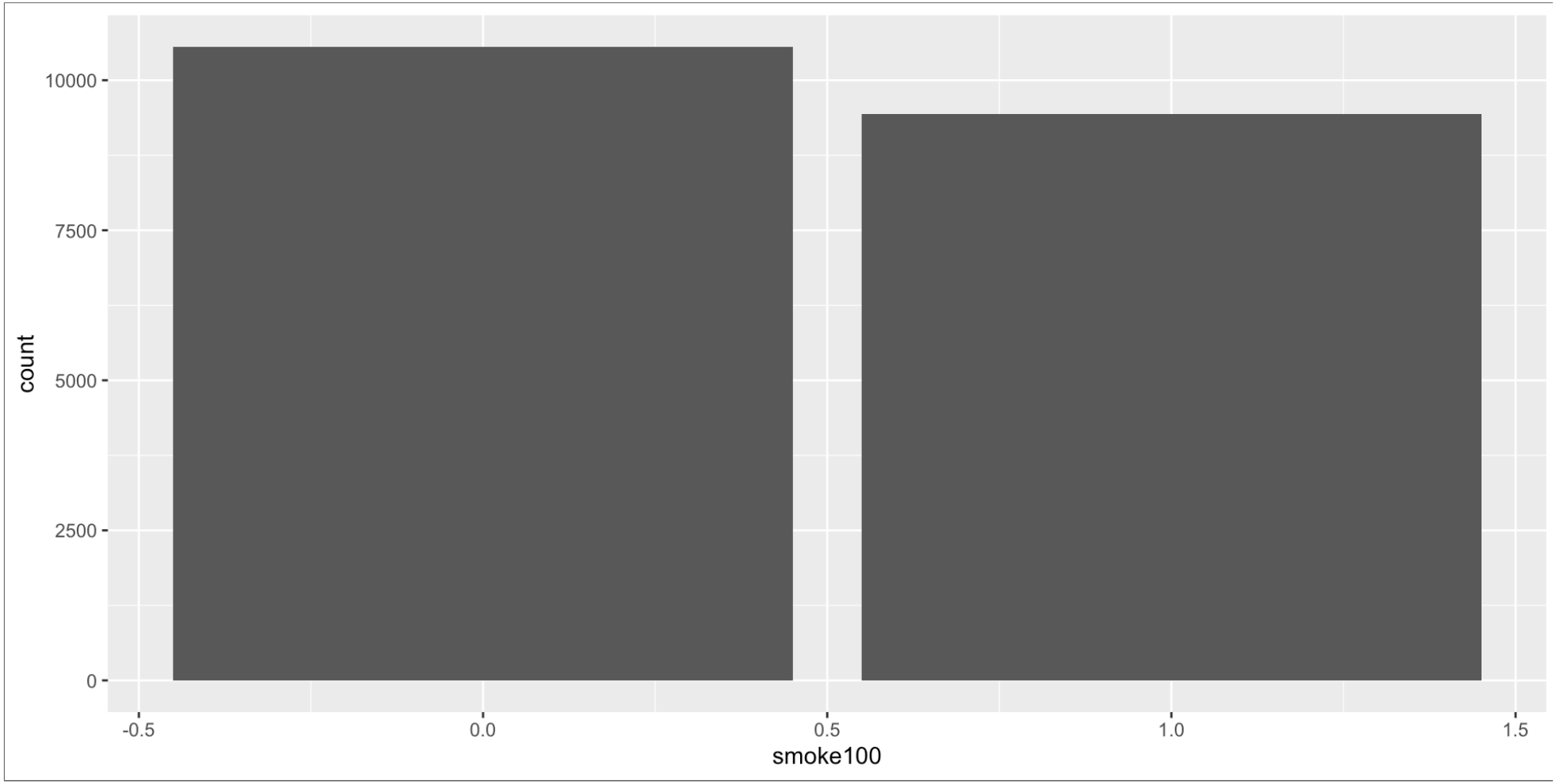
```
glimpse(cdc)
```

```
Rows: 20,000
Columns: 9
$ genhlth  <fct> good, good, good, good, very good, very good, very
good, very…
$ exerany  <dbl> 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
1, 1, 1…
$ hlthplan <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
1, 1, 1…
$ smoke100 <dbl> 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1, 0, 0…
$ height   <dbl> 70, 64, 60, 66, 61, 64, 71, 67, 65, 70, 69, 69, 66,
70, 69, 7…
$ weight   <int> 175, 125, 105, 132, 150, 114, 194, 170, 150, 180,
186, 168, 1…
$ wtdesire <int> 175, 115, 105, 124, 130, 114, 185, 160, 130, 170,
175, 148, 2…
```

First example

Fix this plot:

```
ggplot(data = cdc, aes(x = smoke100)) +
  geom_bar()
```


10000
7500
5000
2500
0
count
-0.5
0.0
0.5
1.0
1.5
smoke100

Change smoke100 into a factor

```
cdc <- cdc %>%
  mutate(smoke100 = factor(smoke100))
glimpse(cdc)
```

What do we expect to see in the glimpse?

Change smoke100 into a factor

```
Rows: 20,000
Columns: 9
$ genhlth  <fct> good, good, good, good, very good, very good, very
good, very…
$ exerany  <dbl> 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
1, 1, 1…
$ hlthplan <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
1, 1, 1…
$ smoke100 <fct> 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1, 0, 0…
$ height   <dbl> 70, 64, 60, 66, 61, 64, 71, 67, 65, 70, 69, 69, 66,
70, 69, 7…
$ weight   <int> 175, 125, 105, 132, 150, 114, 194, 170, 150, 180,
186, 168, 1…
$ wtdesire <int> 175, 115, 105, 124, 130, 114, 185, 160, 130, 170,
175, 148, 2…
```

Now, turn multiple variables into factors

How about this way?

```
cdc <- cdc %>%
  mutate(smoke100 = factor(smoke100),
         exerany = factor(exerany),
         hlthplan = factor(hlthplan))
```

Better way

Use the `across` helper within `mutate`:

```
cdc <- cdc %>%
  mutate(across(exerany:smoke100, factor))
glimpse(cdc)
```

Better way

```
Rows: 20,000
Columns: 9
$ genhlth  <fct> good, good, good, good, very good, very good, very
good, very…
$ exerany  <fct> 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0,
1, 1, 1…
$ hlthplan <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
1, 1, 1…
$ smoke100 <fct> 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1, 0, 0…
$ height   <dbl> 70, 64, 60, 66, 61, 64, 71, 67, 65, 70, 69, 69, 66,
70, 69, 7…
$ weight   <int> 175, 125, 105, 132, 150, 114, 194, 170, 150, 180,
186, 168, 1…
$ wtdesire <int> 175, 115, 105, 124, 130, 114, 185, 160, 130, 170,
175, 148, 2…
```

What if we want to give the factor levels labels?

The lambda notation (~):

```
cdc <- cdc %>%
  mutate(across(exerany:smoke100, ~factor(.x, labels = c("nc
glimpse(cdc)
```

What if we want to give the factor levels labels?

```
Rows: 20,000
Columns: 9
$ genhlth  <fct> good, good, good, good, very good, very good, very
good, very…
$ exerany  <fct> no, no, yes, yes, no, yes, yes, no, no, yes, yes,
yes, yes, y…
$ hlthplan <fct> yes, yes, yes, yes, yes, yes, yes, yes, yes, yes,
yes, yes, n…
$ smoke100 <fct> no, yes, yes, no, no, no, no, no, yes, no, yes, yes,
yes, yes…
$ height   <dbl> 70, 64, 60, 66, 61, 64, 71, 67, 65, 70, 69, 69, 66,
70, 69, 7…
$ weight   <int> 175, 125, 105, 132, 150, 114, 194, 170, 150, 180,
186, 168, 1…
$ wtdesire <int> 175, 115, 105, 124, 130, 114, 185, 160, 130, 170,
175, 148, 2…
```

Other select helpers

- everything()
- starts_with()
- contains()
- where()

And more: https://dplyr.tidyverse.org/reference/select.html

Quick recap of object types

What are these (in R context)?

- vector
- list
- matrix
- data frame
- tibble

Quick check

Each column in a data frame is a:

- vector
- list
- matrix
- tibble

Quick check

Each row in a data frame is a:

- vector
- list
- matrix
- tibble

Another challenge

The task: Create a numerical summary for `height` and `age`, and compute the interquartile range for each.
What did you do?

One approach - not DRY

```
IQR(cdc$height)
IQR(cdc$age)
```

Another approach - trying to be DRY

Why doesn't this work?

Let's read the error message to find out

```
cdc %>%
  select(height, age) %>%
  IQR()
```

```
Error in quantile(as.numeric(x), c(0.25, 0.75), na.rm = na.rm, names =
FALSE, : 'list' object cannot be coerced to type 'double'
```

The `map` approach - pretty DRY

```
cdc %>%
  select(height, age) %>%
  map(IQR)
```

```
$height
[1] 6

$age
[1] 26
```

Note the two arguments:

- a list or vector (piped in)
- a function to apply (in the brackets)

Note: the output is a list!

For dataframe output

```
cdc %>%
  select(height, age) %>%
  map_dfr(IQR)
```

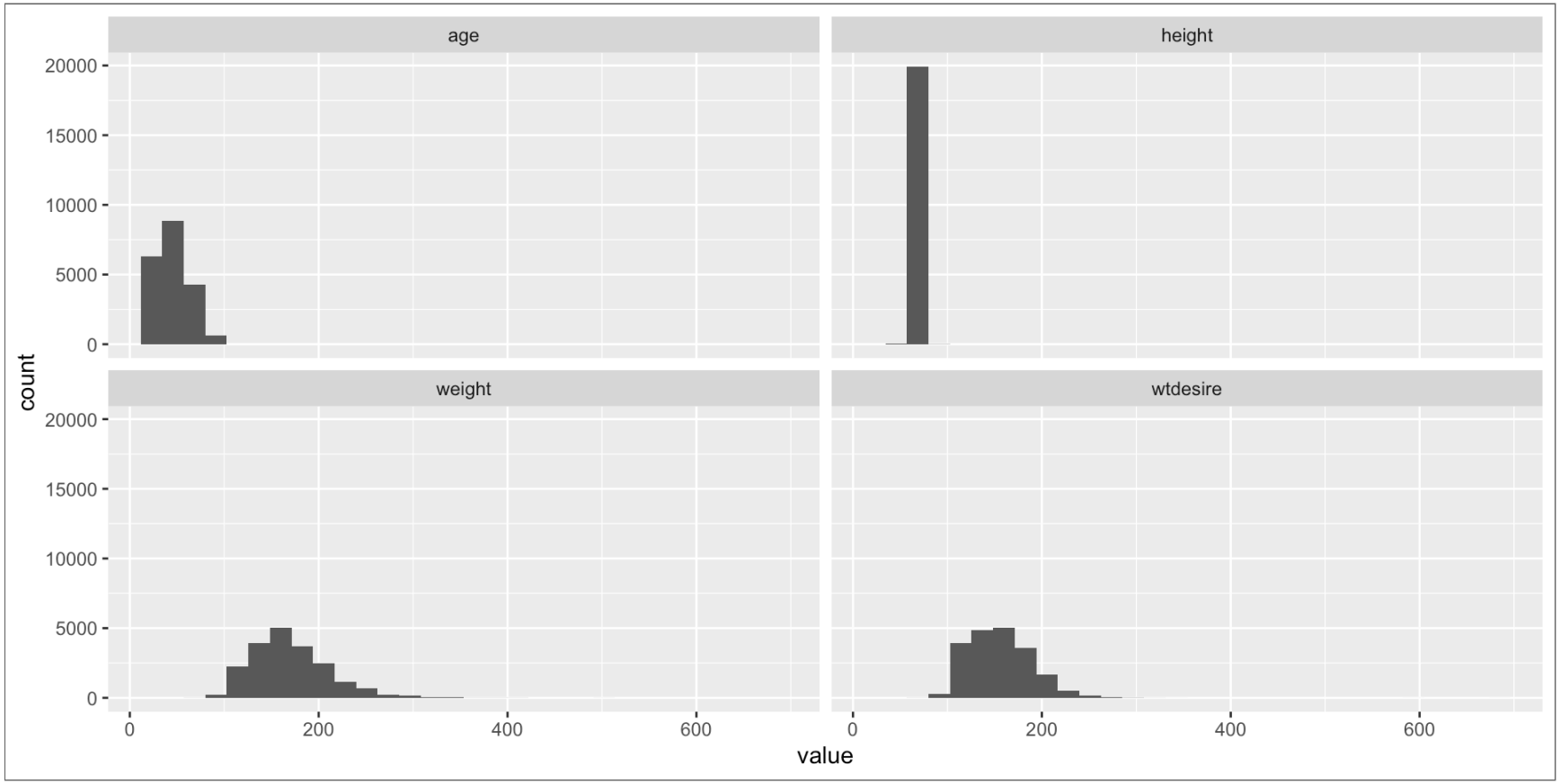
```
# A tibble: 1 × 2
  height   age
   <dbl> <dbl>
1      6    26
```

Sometimes all you need is a pivot

Create a histogram of every numeric variable in the cdc dataset:

```
cdc %>%
  select(where(is.numeric)) %>%
  pivot_longer(cols = everything()) %>%
  ggplot(aes(x = value)) +
  geom_histogram() +
  facet_wrap(~name) # try the scales argument
```

Sometimes all you need is a pivot



A few notes

A mean of a vector

What will be the output here?

```r
x <- c(1, 2, 3, 4, 5)
mean(x)
```

A mean of a vector

```
[1] 3
```

What about NA?

```
1 x <- c(1, 2, 3, 4, NA)
2 mean(x)
```

Let's check out `?mean`

What about NA?

```
[1] NA
```

What about NA?

```
x <- c(1, 2, 3, 4, NA)
mean(x, na.rm = TRUE)
```

```
[1] 2.5
```

Filtering out missing data

Let's look at a dataset with missing data:

```
data("msleep")
glimpse(msleep)
```

```
Rows: 83
Columns: 11
$ name         <chr> "Cheetah", "Owl monkey", "Mountain beaver",
"Greater shor…
$ genus        <chr> "Acinonyx", "Aotus", "Aplodontia", "Blarina",
"Bos", "Bra…
$ vore         <chr> "carni", "omni", "herbi", "omni", "herbi",
"herbi", "carn…
$ order        <chr> "Carnivora", "Primates", "Rodentia",
"Soricomorpha", "Art…
$ conservation <chr> "lc", NA, "nt", "lc", "domesticated", NA, "vu",
NA, "dome…
$ sleep_total  <dbl> 12.1, 17.0, 14.4, 14.9, 4.0, 14.4, 8.7, 7.0,
10.1, 3.0, 5…
$ sleep_rem    <dbl> NA, 1.8, 2.4, 2.3, 0.7, 2.2, 1.4, NA, 2.9, NA,
0.6, 0.8, …
```

Filtering out with `!=`

```
msleep %>%
  filter(vore != "omni")
```

```
# A tibble: 56 × 11
   name   genus vore  order conservation sleep_total sleep_rem
sleep_cycle awake
   <chr>  <chr> <chr> <chr> <chr>              <dbl>     <dbl>
<dbl> <dbl>
 1 Cheet… Acin… carni Carn… lc                  12.1      NA        NA
11.9
 2 Mount… Aplo… herbi Rode… nt                  14.4       2.4      NA
9.6
 3 Cow    Bos   herbi Arti… domesticated         4         0.7
0.667  20
 4 Three… Brad… herbi Pilo… <NA>                14.4       2.2
0.767   9.6
 5 North… Call… carni Carn… vu                   8.7       1.4
0.383  15.3
 6 Dog    Canis carni Carn… domesticated        10.1       2.9
```

Filtering out NA's - the wrong way

```r
msleep %>%
  filter(conservation != NA)
```

```
# A tibble: 0 × 11
# ℹ 11 variables: name <chr>, genus <chr>, vore <chr>, order <chr>,
#   conservation <chr>, sleep_total <dbl>, sleep_rem <dbl>,
sleep_cycle <dbl>,
#   awake <dbl>, brainwt <dbl>, bodywt <dbl>
```

Filtering out NA's - the right way

```
msleep %>%
  filter(!is.na(conservation))
```

```
# A tibble: 54 × 11
   name   genus vore  order conservation sleep_total sleep_rem
sleep_cycle awake
   <chr>  <chr> <chr> <chr> <chr>              <dbl>     <dbl>
<dbl> <dbl>
 1 Cheet… Acin… carni Carn… lc                  12.1      NA
NA  11.9
 2 Mount… Aplo… herbi Rode… nt                  14.4       2.4
NA   9.6
 3 Great… Blar… omni  Sori… lc                  14.9       2.3
0.133   9.1
 4 Cow    Bos   herbi Arti… domesticated         4         0.7
0.667  20
 5 North… Call… carni Carn… vu                   8.7       1.4
0.383  15.3
 6 Dog    Canis carni Carn… domesticated        10.1       2.9
```

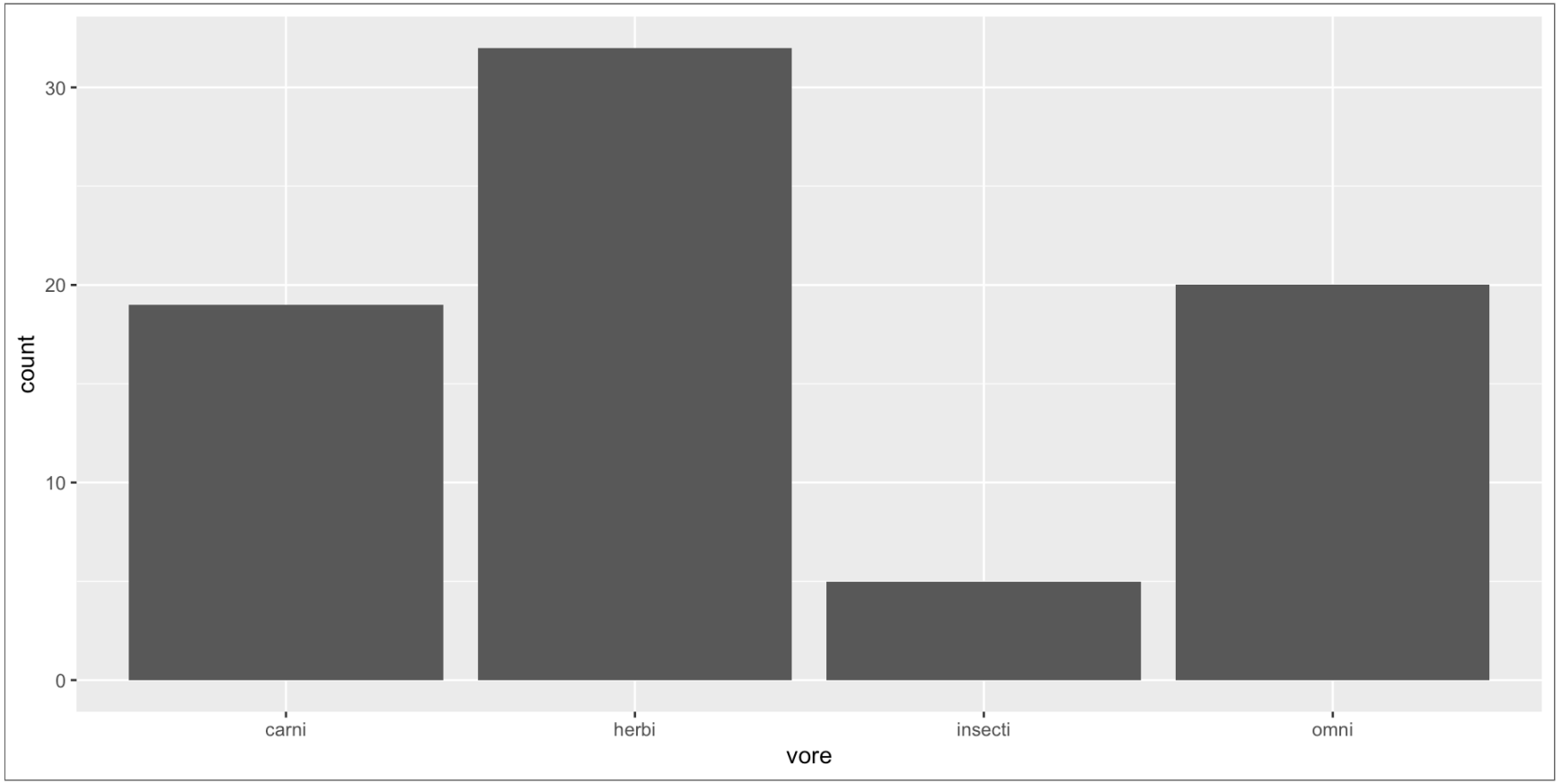
And now on plotting

Adding additional layers to a plot

```
vore_bar <- msleep %>%
  filter(!is.na(vore)) %>%
  ggplot(aes(x = vore)) +
  geom_bar()

vore_bar
```

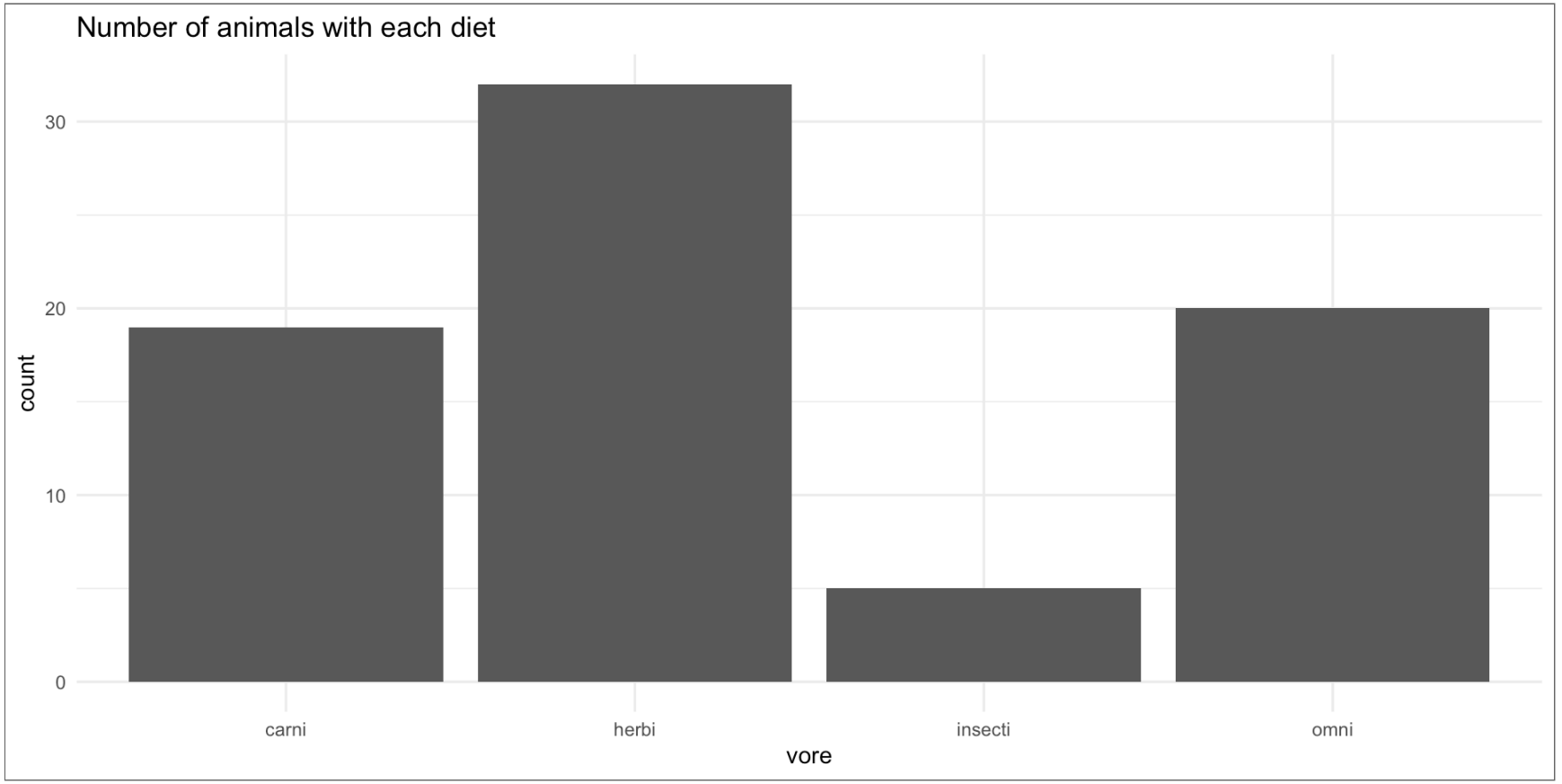
Adding additional layers to a plot



Adding additional layers to a plot

```
vore_bar +
  labs(title = "Number of animals with each diet") +
  theme_minimal()
```

Adding additional layers to a plot



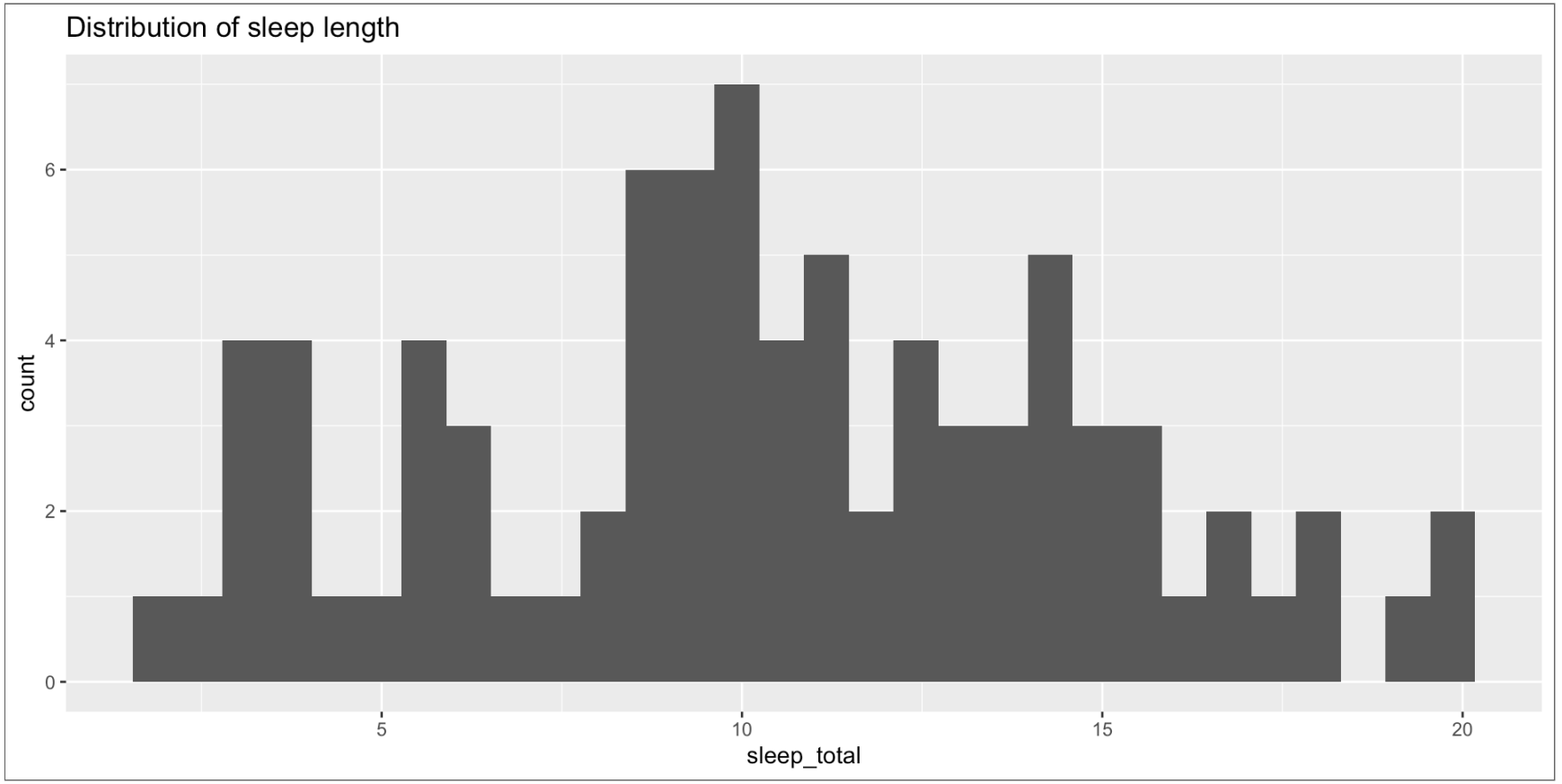
`facet_wrap` vs `facet_grid`

Both used to present a set of plots

Let's start with a histogram

```
1 sleep_hist <- msleep %>%
2   ggplot(aes(x = sleep_total))+
3   geom_histogram() +
4   labs(title = "Distribution of sleep length")
5
6 sleep_hist
```

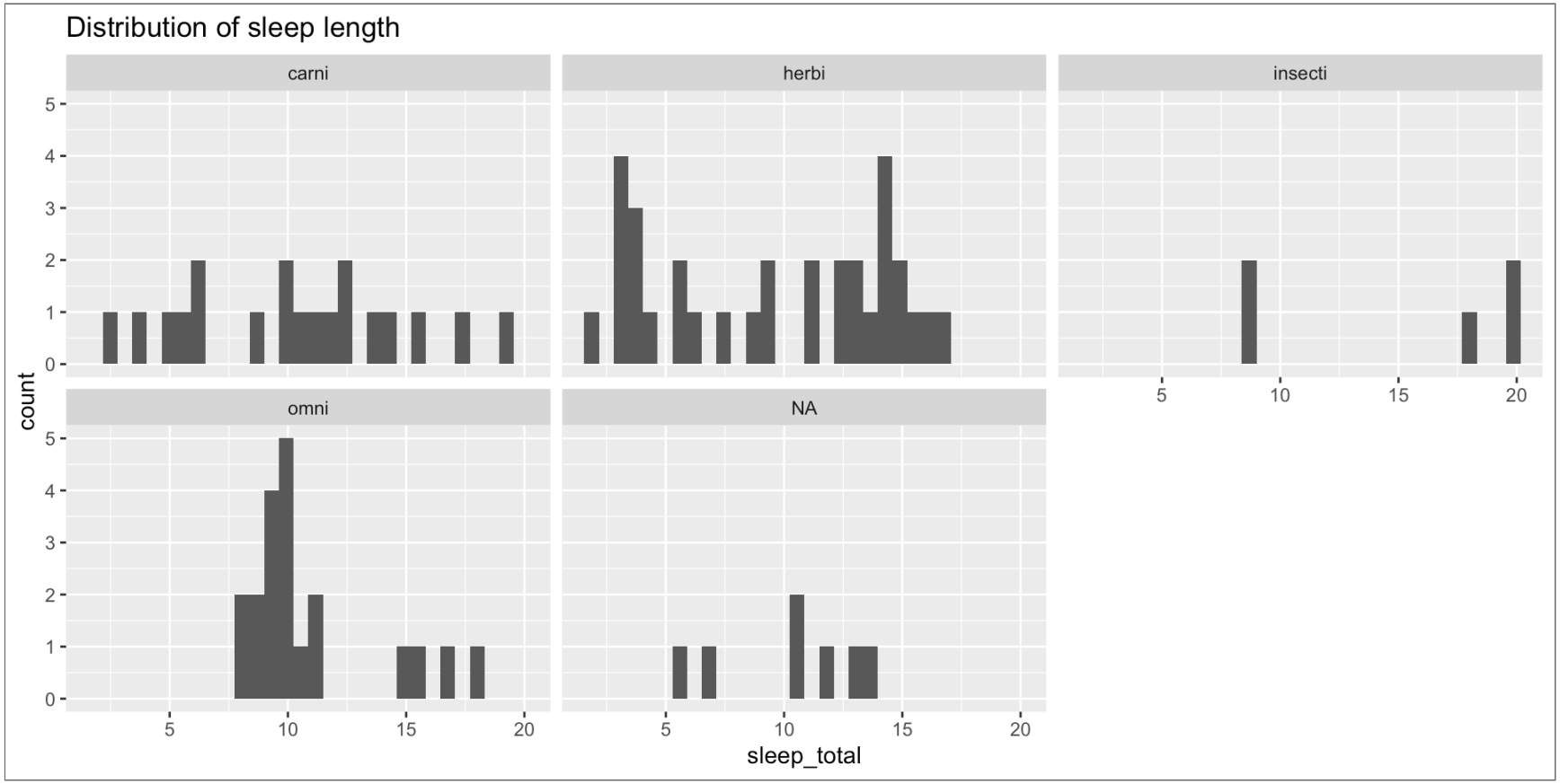
`facet_wrap` vs `facet_grid`



Now, let's add facetting

```
sleep_hist +
  facet_wrap(~vore)
```

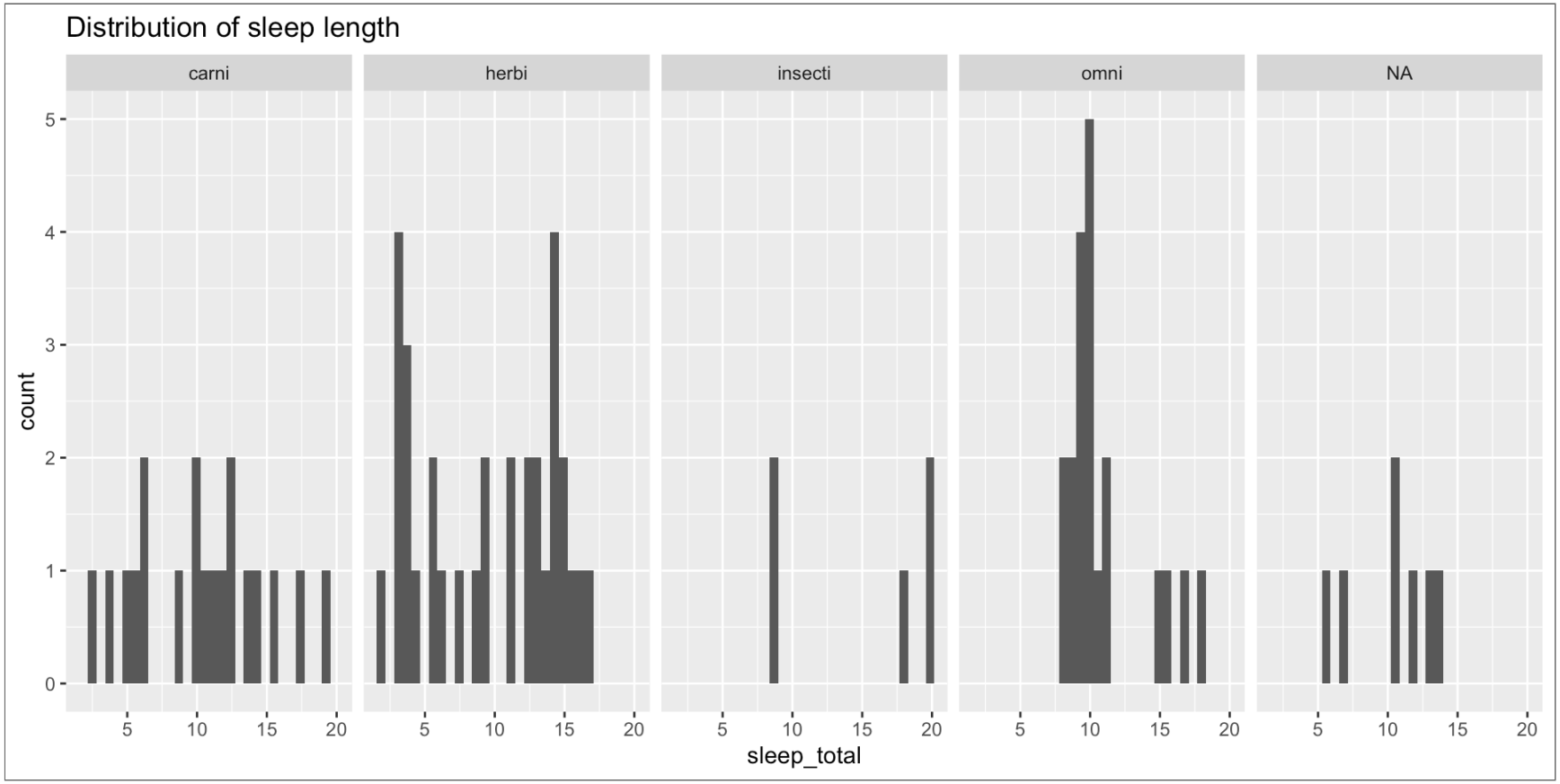
Now, let's add facetting



And with facet_grid

```
sleep_hist +
  facet_grid(~vore)
```

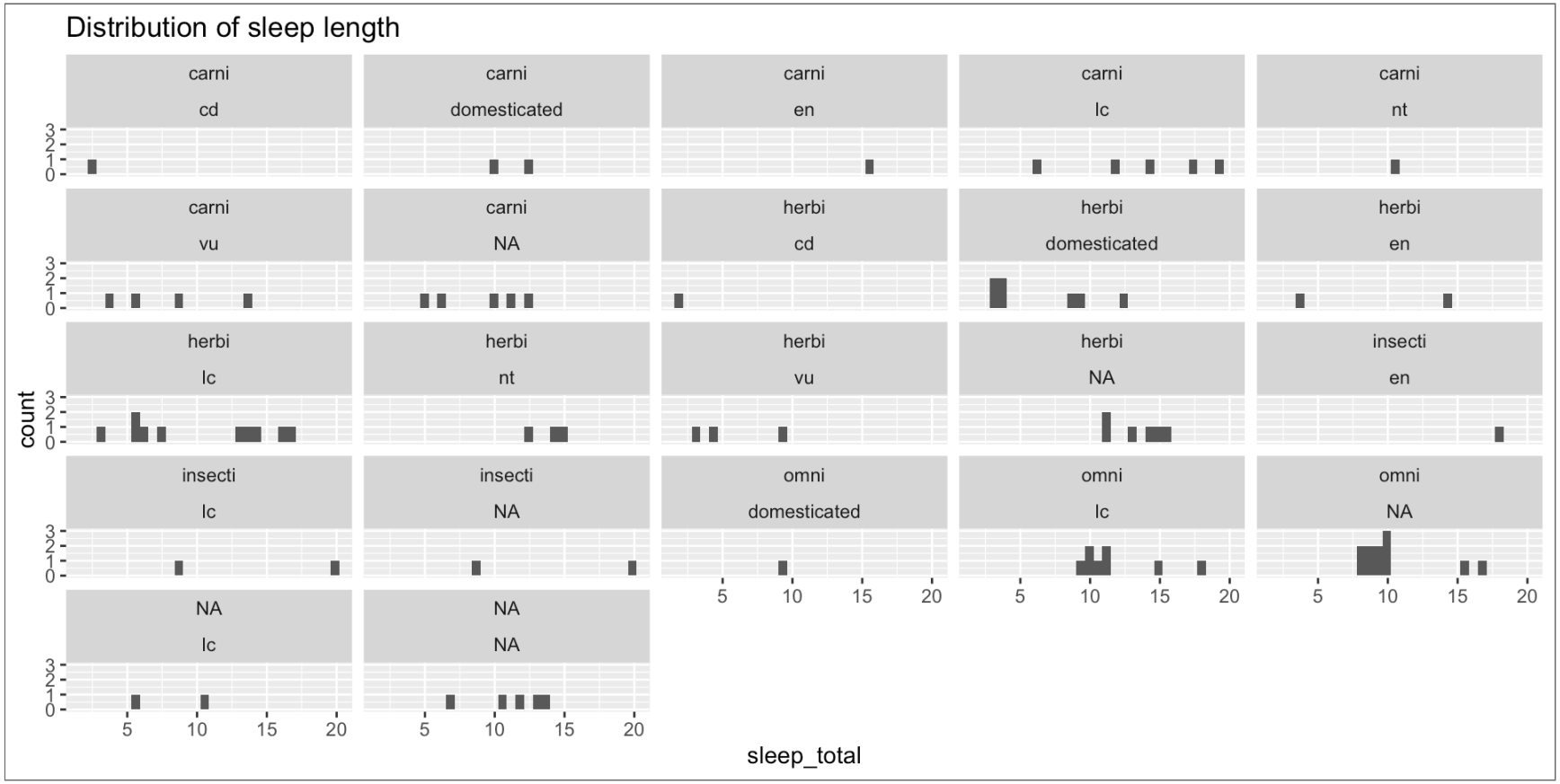
And with facet_grid



Two variables in facetting

```
sleep_hist +
  facet_wrap(vore~conservation)
```

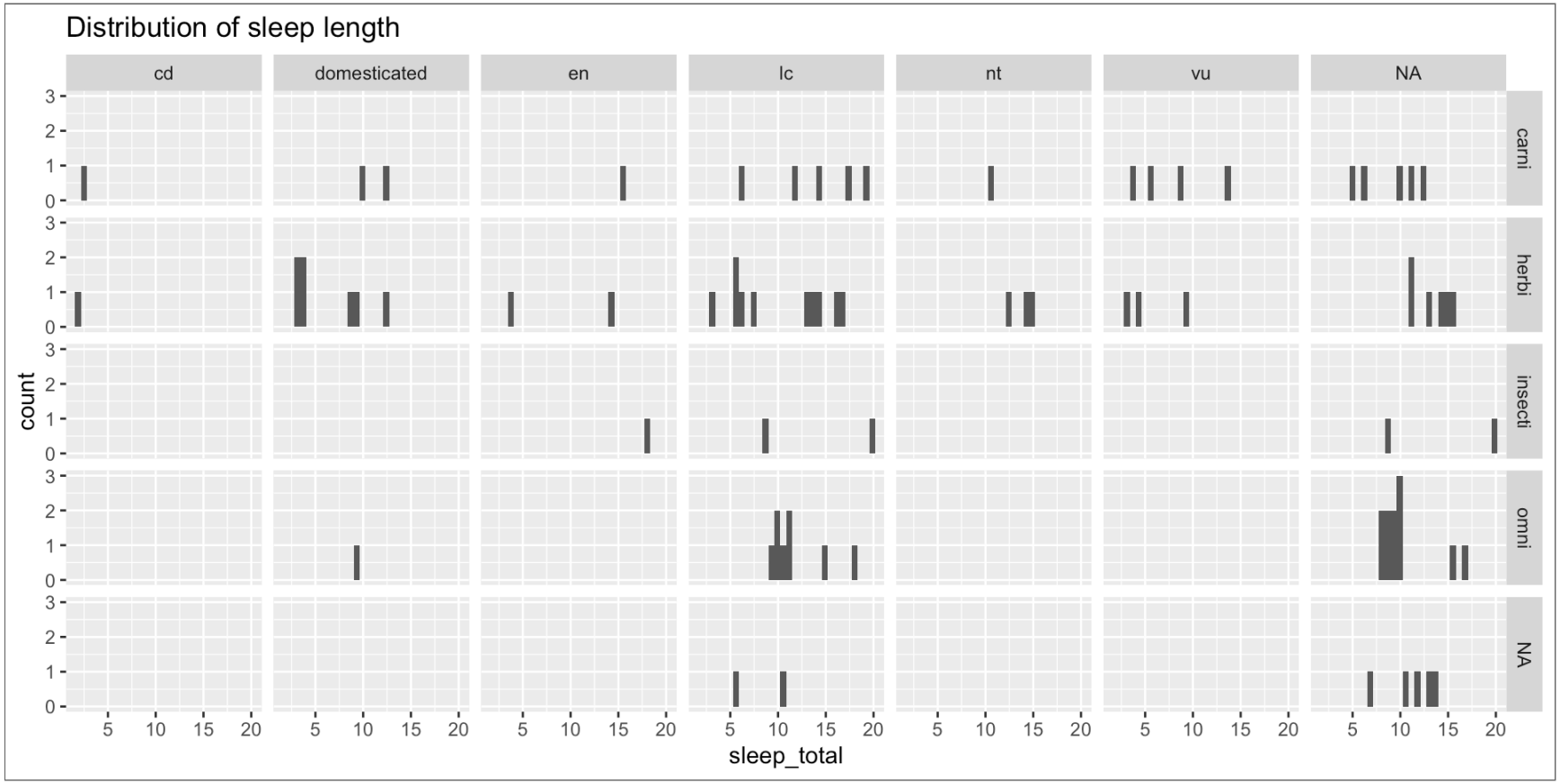
Two variables in facetting



And with facet_grid

```
sleep_hist +
  facet_grid(vore~conservation)
```

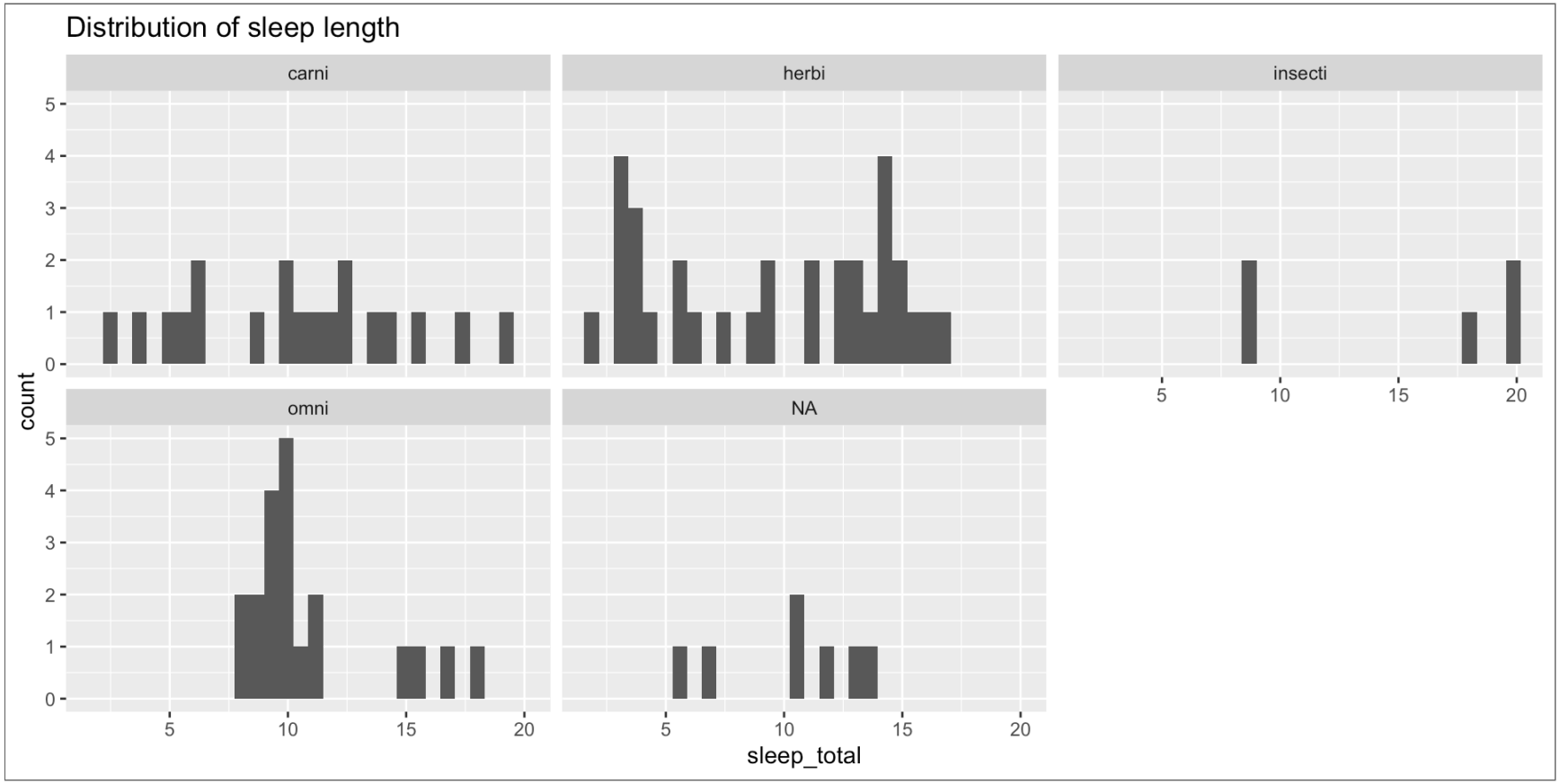
And with facet_grid



Final note: scales = "fixed"

```
1 sleep_hist +
2   facet_wrap(~vore)
```

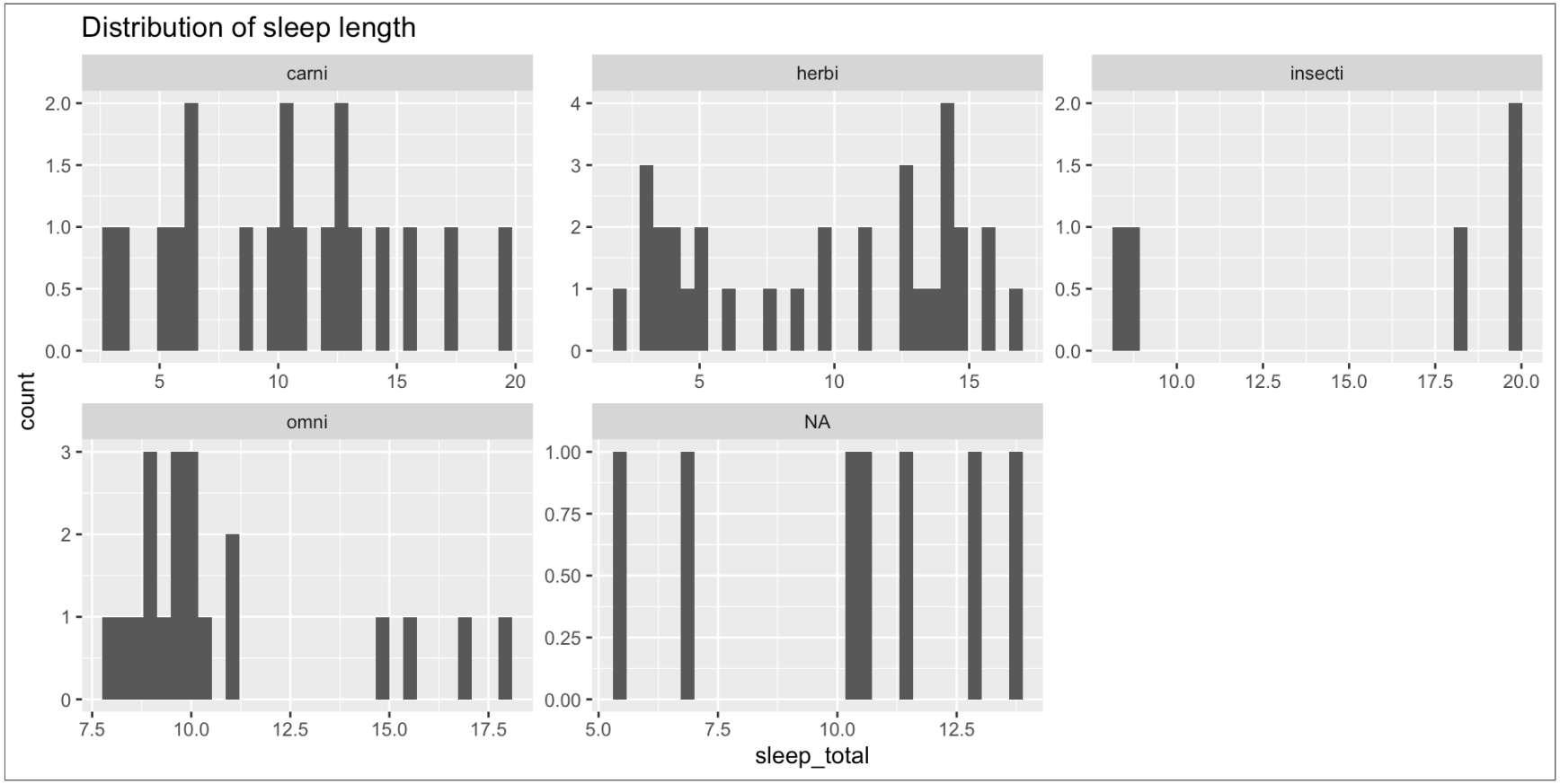
Final note: scales = "fixed"



And if scales = "free"?

```
sleep_hist +
  facet_wrap(~vore, scales = "free")
```

And if scales = "free"?



The end