

Data Visualization - 4. Show the Right Numbers

Kieran Healy

Code Horizons

December 10, 2023



Show the Right Numbers



Load the packages we need

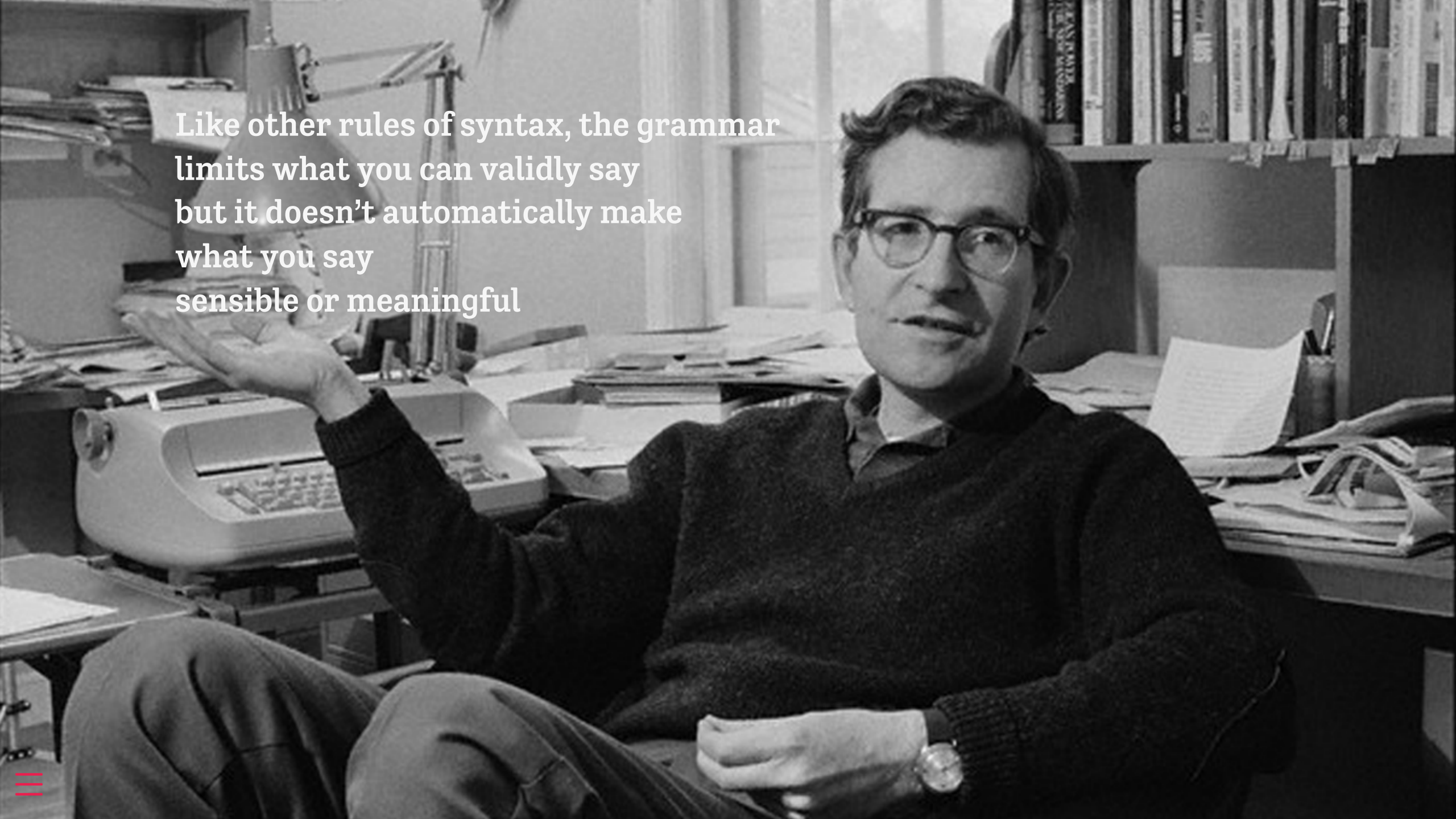
```
library(tidyverse)      # Your friend and mine  
library(gapminder)      # Gapminder data  
library(here)           # Portable file paths  
library(socviz)         # Handy socviz functions
```

ggplot implements a
grammar of graphics

A grammar of graphics

The grammar is a set of rules for how to produce graphics from data, by *mapping* data to or *representing* it by geometric **objects** (like points and lines) that have aesthetic **attributes** (like position, color, size, and shape), together with further rules for transforming data if needed, for adjusting scales and their guides, and for projecting results onto some coordinate system.

Like other rules of syntax, the grammar
limits what you can validly say
but it doesn't automatically make
what you say
sensible or meaningful



Grouped data and the group aesthetic

Try to make a lineplot

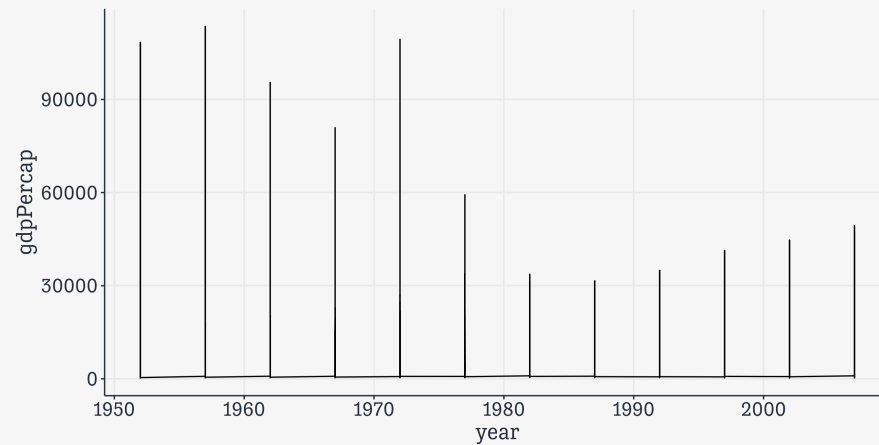
```
1 p ← ggplot(data = gapminder,  
2           mapping = aes(x = year,  
3                         y = gdpPercap))
```


Try to make a lineplot

```
1 p <- ggplot(data = gapminder,  
2             mapping = aes(x = year,  
3                           y = gdpPercap)) +  
4   geom_line()
```

Try to make a lineplot

```
1 p <- ggplot(data = gapminder,  
2             mapping = aes(x = year,  
3                           y = gdpPercap)) +  
4   geom_line()  
5  
6 p
```



Try to make a lineplot

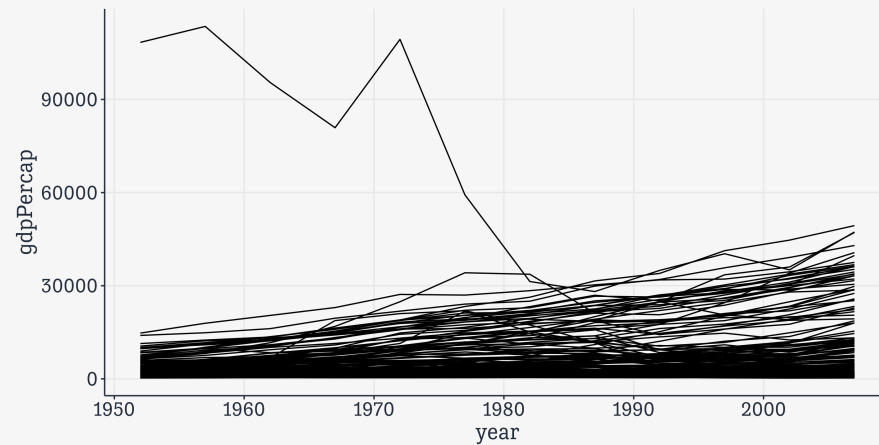
```
1 p ← ggplot(data = gapminder,  
2           mapping = aes(x = year,  
3                         y = gdpPercap))
```

Try to make a lineplot

```
1 p <- ggplot(data = gapminder,  
2             mapping = aes(x = year,  
3                           y = gdpPercap)) +  
4   geom_line(mapping = aes(group = country))
```

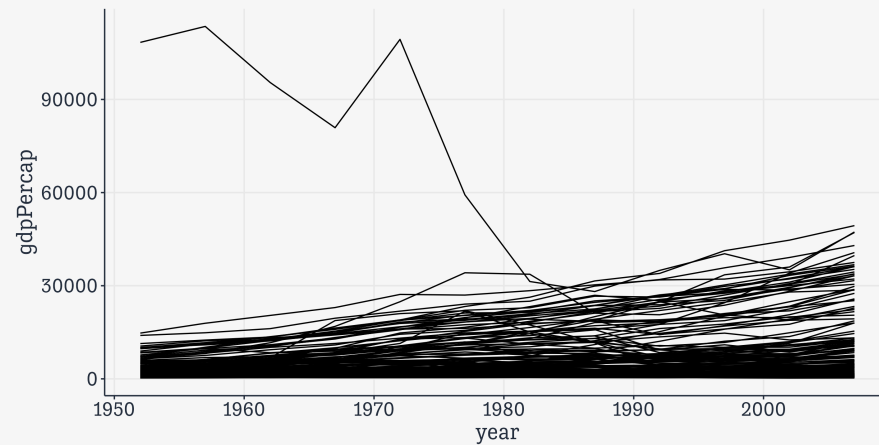
Try to make a lineplot

```
1 p <- ggplot(data = gapminder,  
2             mapping = aes(x = year,  
3                           y = gdpPercap)) +  
4   geom_line(mapping = aes(group = country))  
5  
6 p
```



Try to make a lineplot

```
1 p <- ggplot(data = gapminder,  
2             mapping = aes(x = year,  
3                           y = gdpPercap)) +  
4   geom_line(mapping = aes(group = country))  
5  
6 p
```



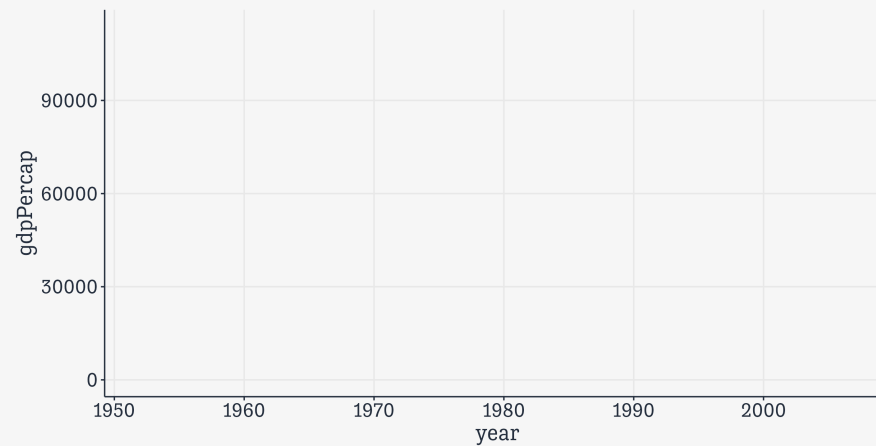
Facet the plot

```
1 gapminder
```

```
# A tibble: 1,704 × 6
  country      continent  year lifeExp      pop
  <fct>        <fct>    <int>   <dbl>   <int>
1 Afghanistan Asia      1952    28.8  8425333
779.
2 Afghanistan Asia      1957    30.3  9240934
821.
3 Afghanistan Asia      1962    32.0 10267083
853.
4 Afghanistan Asia      1967    34.0 11537966
836.
5 Afghanistan Asia      1972    36.1 13079460
740.
6 Afghanistan Asia      1977    38.4 14880372
786.
7 Afghanistan Asia      1982    39.0 12881816
```

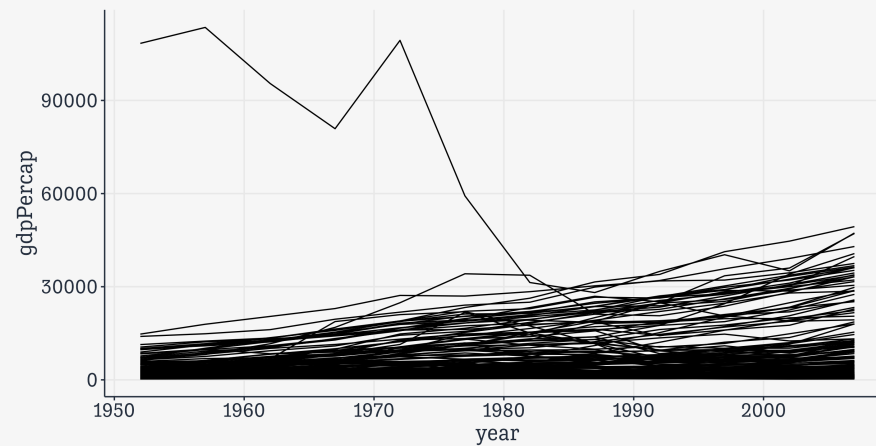
Facet the plot

```
1 gapminder ►  
2   ggplot(mapping =  
3     aes(x = year,  
4     y = gdpPercap))
```



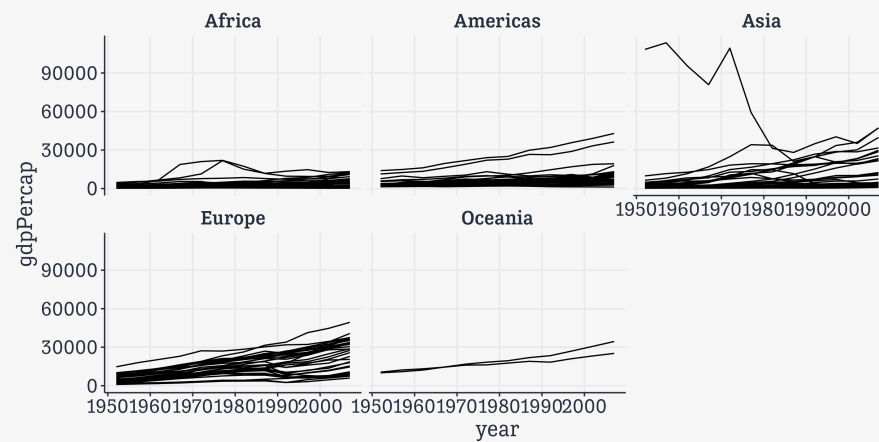
Facet the plot

```
1 gapminder ►  
2   ggplot(mapping =  
3     aes(x = year,  
4         y = gdpPercap)) +  
5   geom_line(mapping = aes(group = country))
```



Facet the plot

```
1 gapminder ►  
2   ggplot(mapping =  
3     aes(x = year,  
4         y = gdpPercap)) +  
5   geom_line(mapping = aes(group = country))  
6   facet_wrap(~ continent)
```



Faceting is very powerful

Faceting

A facet is not a geom; it's a way of arranging repeated geoms by some additional variable

Facets use R's "formula" syntax: `facet_wrap(~ continent)`

Read the `~` as "on" or "by"

Faceting

You can also use this syntax: `facet_wrap(vars(continent))`

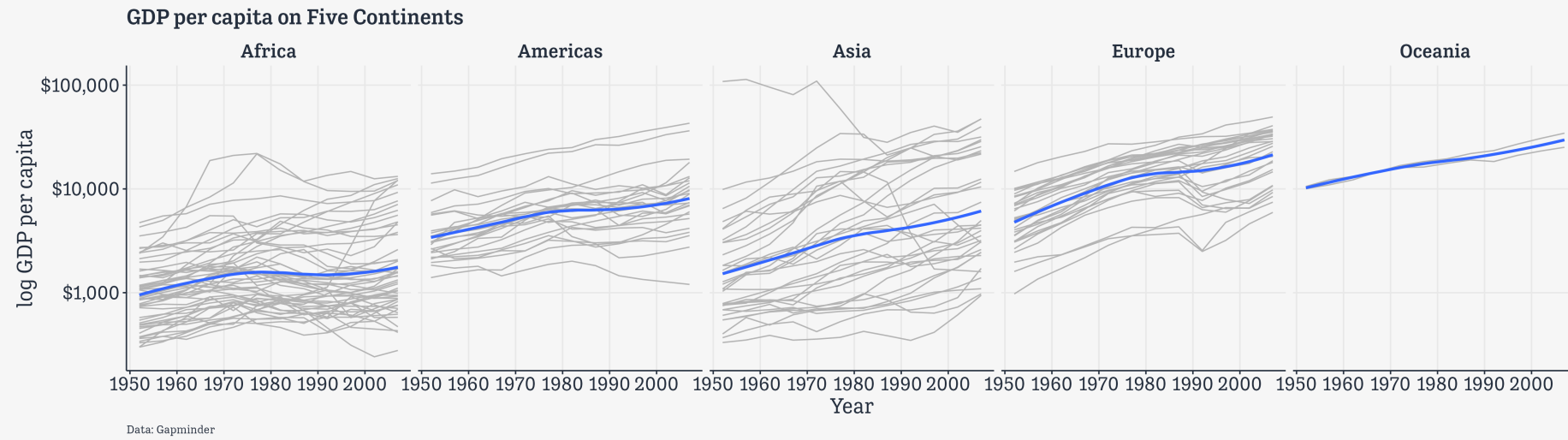
This is newer, and consistent with other ways of referring to variables within tidyverse functions.

Facets in action

```
p ← ggplot(data = gapminder,
           mapping = aes(x = year,
                        y = gdpPercap))

p_out ← p + geom_line(color="gray70",
                    mapping=aes(group = country)) +
  geom_smooth(size = 1.1,
             method = "loess",
             se = FALSE) +

  scale_y_log10(labels=scales::label_dollar()) +
  facet_wrap(~ continent, ncol = 5) + #<<
  labs(x = "Year",
       y = "log GDP per capita",
       title = "GDP per capita on Five Continents",
       caption = "Data: Gapminder")
```



A more polished faceted plot.



One-variable summaries

The **midwest** dataset

County-level census data for Midwestern U.S. Counties

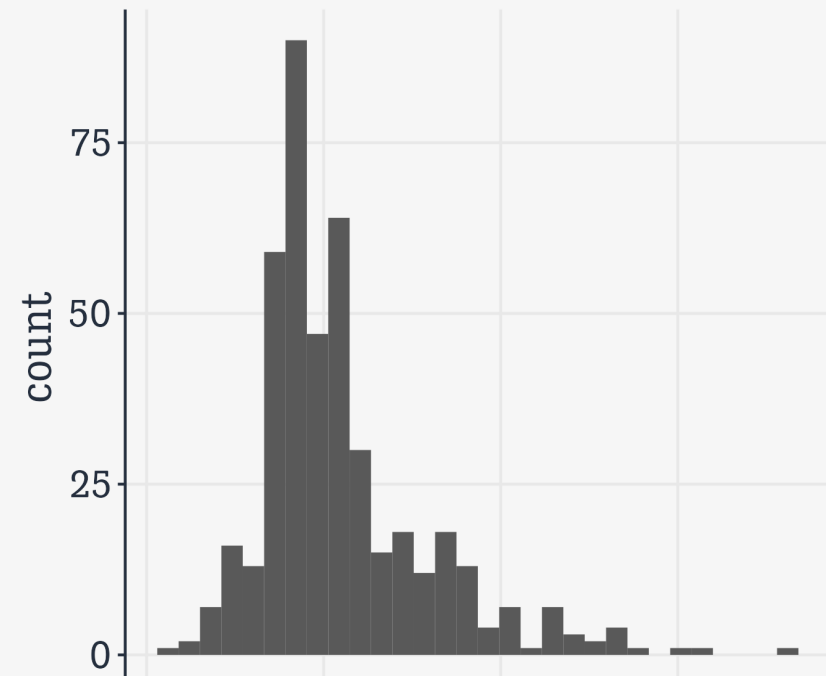
```
midwest
```

```
# A tibble: 437 × 28
  PID county state area poptotal popdensity popwhite popblack popamerindian
  <int> <chr> <chr> <dbl> <int> <dbl> <int> <int> <int>
1 561 ADAMS IL 0.052 66090 1271. 63917 1702 98
2 562 ALEXAN... IL 0.014 10626 759 7054 3496 19
3 563 BOND IL 0.022 14991 681. 14477 429 35
4 564 BOONE IL 0.017 30806 1812. 29344 127 46
5 565 BROWN IL 0.018 5836 324. 5264 547 14
6 566 BUREAU IL 0.05 35688 714. 35157 50 65
7 567 CALHOUN IL 0.017 5322 313. 5298 1 8
8 568 CARROLL IL 0.027 16805 622. 16519 111 30
9 569 CASS IL 0.024 13437 560. 13384 16 8
10 570 CHAMPA... IL 0.058 173025 2983. 146506 16559 331
# i 427 more rows
# i 19 more variables: popasian <int>, popother <int>, percwhite <dbl>,
# percblack <dbl>, percamerindian <dbl>, percasian <dbl>, percother <dbl>,
# popadults <int>, perchsd <dbl>, percollege <dbl>, percprof <dbl>,
# poppovertyknown <int>, percpovertyknown <dbl>, percbelowpoverty <dbl>,
# percchildbelowpovert <dbl>, percadultpoverty <dbl>,
```

stat_ functions behind the scenes

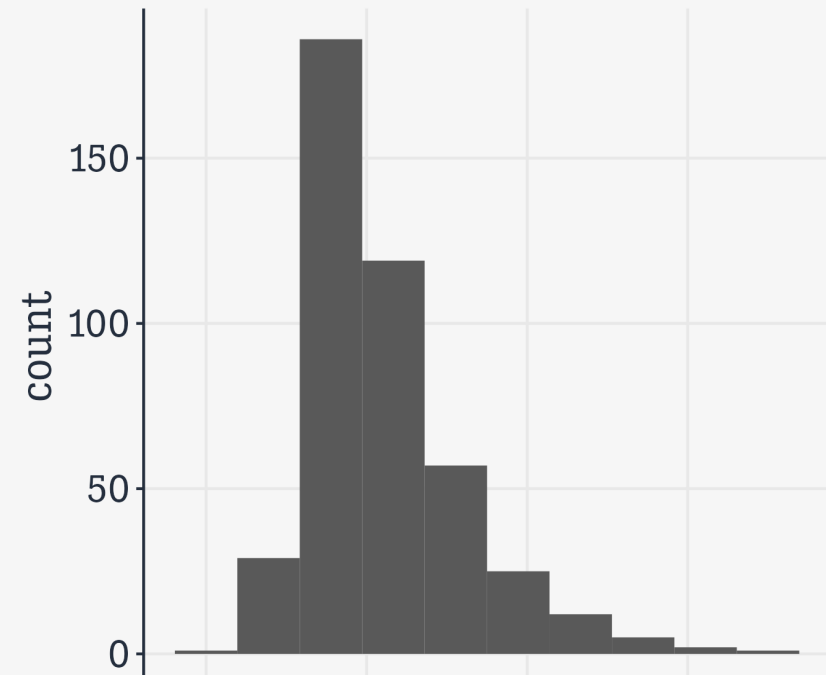
```
p ← ggplot(data = midwest,  
            mapping = aes(x = area))  
  
p + geom_histogram()
```

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.



stat_ functions behind the scenes

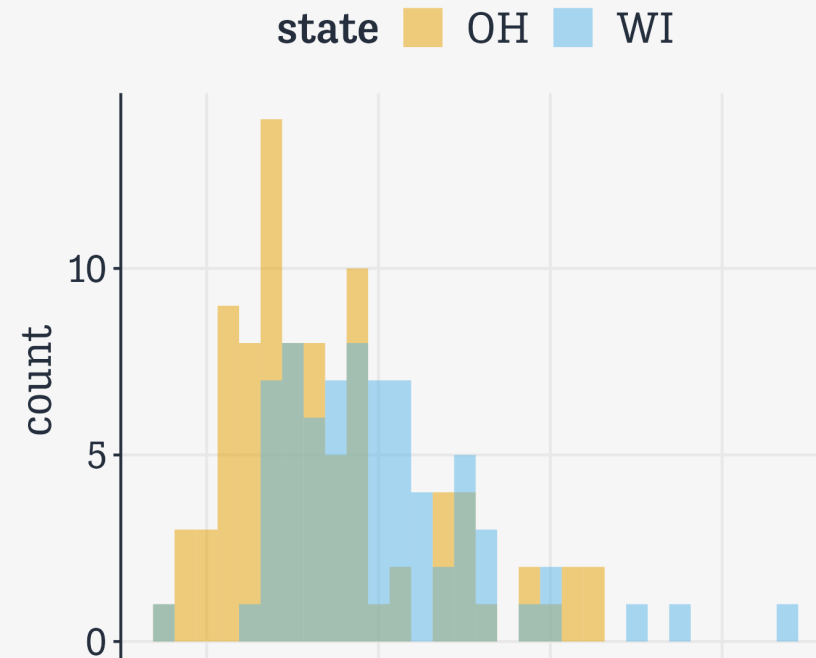
```
p ← ggplot(data = midwest,  
            mapping = aes(x = area))  
  
p + geom_histogram(bins = 10)
```



Compare two distributions

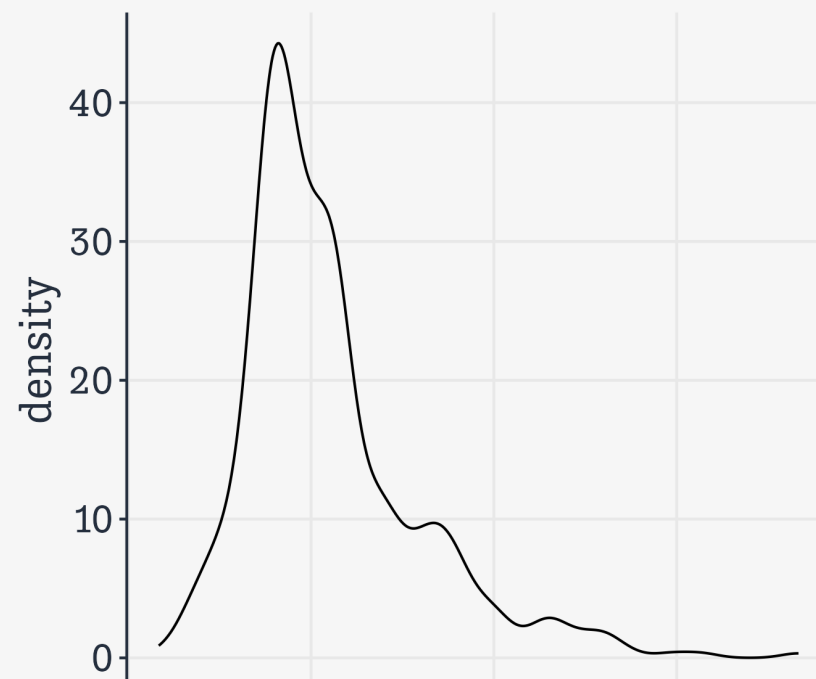
```
## Two state codes
oh_wi ← c("OH", "WI")

midwest ▷
  filter(state %in% oh_wi) ▷
  ggplot(mapping = aes(x = percollege,
                        fill = state)) +
  geom_histogram(alpha = 0.5,
                 position = "identity")
```



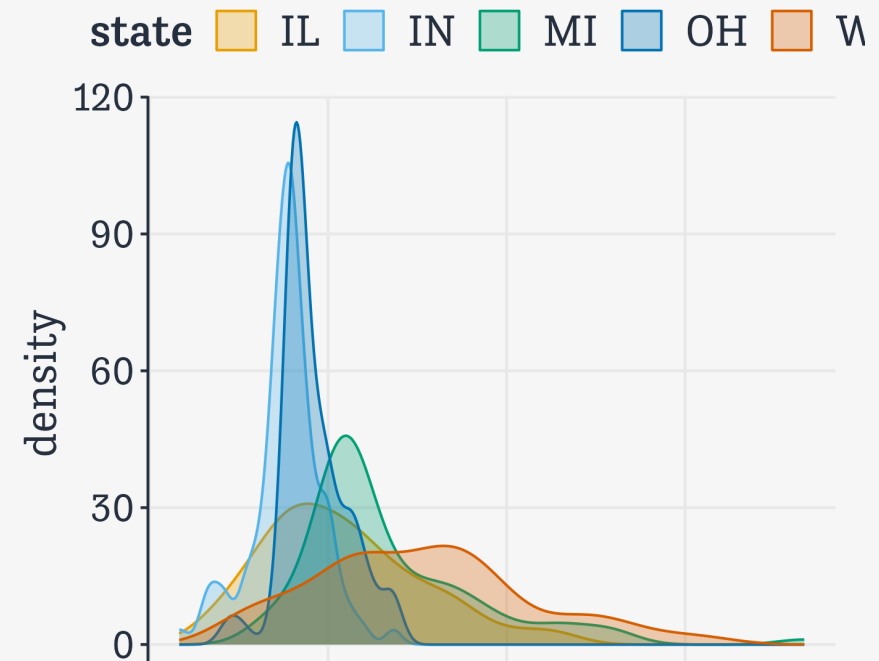
geom_density()

```
p ← ggplot(data = midwest,  
            mapping = aes(x = area))  
  
p + geom_density()
```



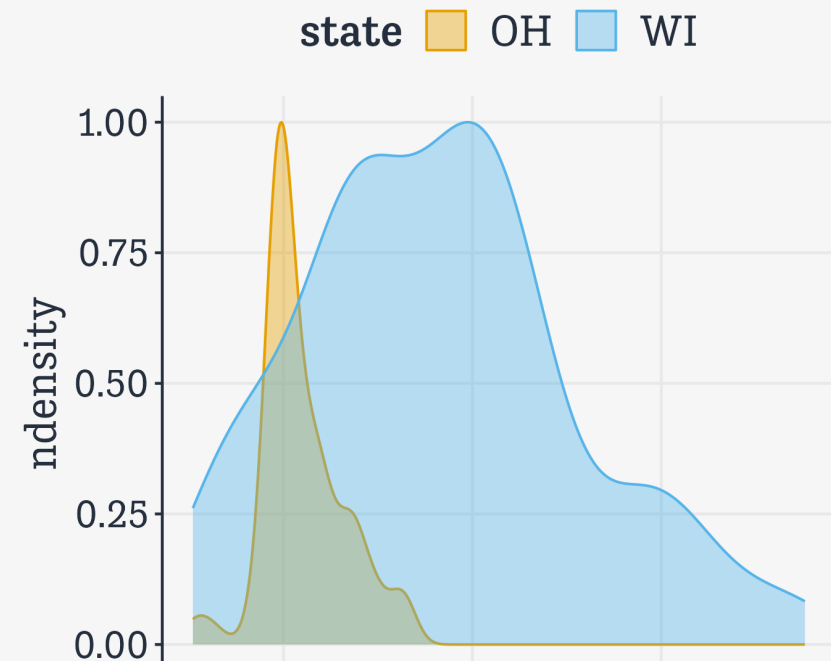
geom_density()

```
p ← ggplot(data = midwest,  
            mapping = aes(x = area,  
                          fill = state,  
                          color = state))  
p + geom_density(alpha = 0.3)
```



geom_density()

```
midwest >
  filter(state %in% oh_wi) >
  ggplot(mapping = aes(x = area,
                       fill = state,
                       color = state)) +
  geom_density(mapping = aes(y = after_stat(nde
                                alpha = 0.4)
```



**Compare subgroups to a
reference distribution**

Some made-up data

Consider 3,000 observations of some unit (e.g., a county) with summary measures for each group, and the population average.

df

```
# A tibble: 3,000 × 5
  unit  pop_a pop_b  pop_c pop_total
  <int> <dbl> <dbl>   <dbl>   <dbl>
1     1  1.29  1.93  -0.0869  1.09
2     2  0.522  0.536 -0.762   0.190
3     3  2.14  1.47  -0.616   1.15
4     4  1.13  0.673 -0.242   0.575
5     5  1.04  1.30   1.18   1.12
6     6  1.80  0.140  2.05   1.33
7     7  0.186  1.30  -0.709   0.476
8     8 -0.953  0.520 -2.44  -0.767
9     9  0.700  1.66  -1.09   0.749
10    10  0.0416 0.484 -0.180   0.177
# i 2,990 more rows
```

Get the data into long format!

```
1 df
```

```
# A tibble: 3,000 × 5
  unit  pop_a pop_b  pop_c pop_total
  <int> <dbl> <dbl>   <dbl>   <dbl>
1     1  1.29  1.93 -0.0869    1.09
2     2  0.522 0.536 -0.762    0.190
3     3  2.14  1.47 -0.616    1.15
4     4  1.13  0.673 -0.242    0.575
5     5  1.04  1.30  1.18    1.12
6     6  1.80  0.140  2.05    1.33
7     7  0.186  1.30 -0.709    0.476
8     8 -0.953  0.520 -2.44   -0.767
9     9  0.700  1.66 -1.09    0.749
10    10  0.0416 0.484 -0.180    0.177
# i 2,990 more rows
```

Get the data into long format!

```
1 df ▶  
2 pivot_longer(cols = pop_a:p
```

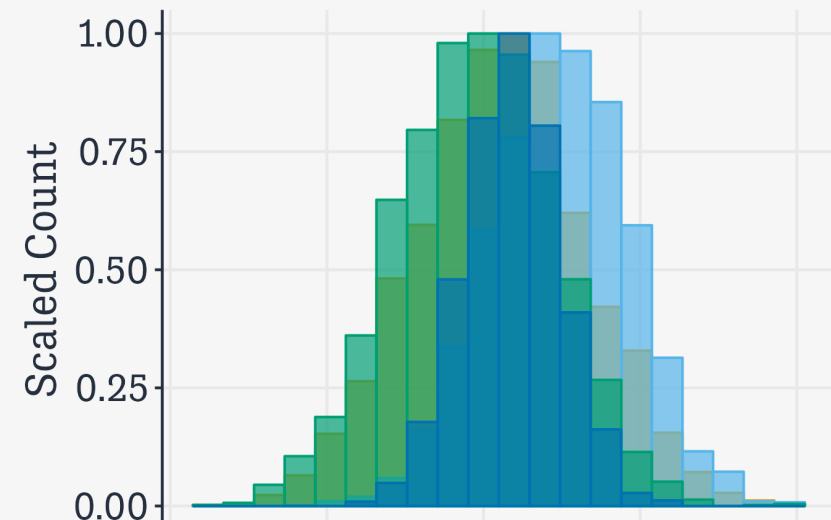
```
# A tibble: 12,000 × 3  
  unit name      value  
  <int> <chr>      <dbl>  
1     1 pop_a      1.29  
2     1 pop_b      1.93  
3     1 pop_c     -0.0869  
4     1 pop_total  1.09  
5     2 pop_a      0.522  
6     2 pop_b      0.536  
7     2 pop_c     -0.762  
8     2 pop_total  0.190  
9     3 pop_a      2.14  
10    3 pop_b      1.47  
# i 11,990 more rows
```

First effort: Hard to read

```
df >
  pivot_longer(cols = pop_a:pop_total) >
  ggplot() +
  geom_histogram(mapping = aes(x = value,
                              y = after_stat(n),
                              color = name, fill =
                                stat = "bin", bins = 20,
                                linewidth = 0.5, alpha = 0.7,
                                position = "identity")) +
  labs(x = "Measure", y = "Scaled Count", color
        fill = "Group",
        title = "Comparing Subgroups: Histograms")
```

Comparing Subgroups: Histograms

Group pop_a pop_b pop_c po



A little pivot trick

```
1 # Treat pop_a to pop_total as a single
2 df
```

```
# A tibble: 3,000 × 5
  unit  pop_a pop_b  pop_c pop_total
<int> <dbl> <dbl>   <dbl>   <dbl>
1     1  1.29  1.93 -0.0869    1.09
2     2  0.522 0.536 -0.762    0.190
3     3  2.14  1.47 -0.616    1.15
4     4  1.13  0.673 -0.242    0.575
5     5  1.04  1.30  1.18    1.12
6     6  1.80  0.140  2.05    1.33
7     7  0.186 1.30 -0.709    0.476
8     8 -0.953 0.520 -2.44   -0.767
9     9  0.700 1.66 -1.09    0.749
10    10  0.0416 0.484 -0.180    0.177
# i 2,990 more rows
```

A little pivot trick

```
1 # Treat pop_a to pop_total as a single
2 df ▶
3   pivot_longer(cols = pop_a:pop_total)
```

```
# A tibble: 12,000 × 3
  unit name      value
  <int> <chr>      <dbl>
1     1 pop_a      1.29
2     1 pop_b      1.93
3     1 pop_c    -0.0869
4     1 pop_total  1.09
5     2 pop_a      0.522
6     2 pop_b      0.536
7     2 pop_c    -0.762
8     2 pop_total  0.190
9     3 pop_a      2.14
10    3 pop_b      1.47
# i 11,990 more rows
```

A little pivot trick

```
1 # Just treat pop_a to pop_c as the single  
2 # Notice that pop_total just gets repeated  
3 df
```

```
# A tibble: 3,000 × 5  
  unit  pop_a pop_b  pop_c pop_total  
  <int> <dbl> <dbl>   <dbl>   <dbl>  
1     1  1.29  1.93 -0.0869    1.09  
2     2  0.522  0.536 -0.762    0.190  
3     3  2.14  1.47 -0.616    1.15  
4     4  1.13  0.673 -0.242    0.575  
5     5  1.04  1.30  1.18    1.12  
6     6  1.80  0.140  2.05    1.33  
7     7  0.186  1.30 -0.709    0.476  
8     8 -0.953  0.520 -2.44   -0.767  
9     9  0.700  1.66 -1.09    0.749  
10    10  0.0416 0.484 -0.180    0.177  
# i 2,990 more rows
```

A little pivot trick

```
1 # Just treat pop_a to pop_c as the single population
2 # Notice that pop_total just gets repeated
3 df ▶
4 pivot_longer(cols = pop_a:pop_c)
```

```
# A tibble: 9,000 × 4
  unit pop_total name    value
<int>   <dbl> <chr>   <dbl>
1     1     1.09 pop_a    1.29
2     1     1.09 pop_b    1.93
3     1     1.09 pop_c   -0.0869
4     2     0.190 pop_a    0.522
5     2     0.190 pop_b    0.536
6     2     0.190 pop_c   -0.762
7     3     1.15 pop_a    2.14
8     3     1.15 pop_b    1.47
9     3     1.15 pop_c   -0.616
10    4     0.575 pop_a    1.13
# i 8,990 more rows
```


Now facet with that data

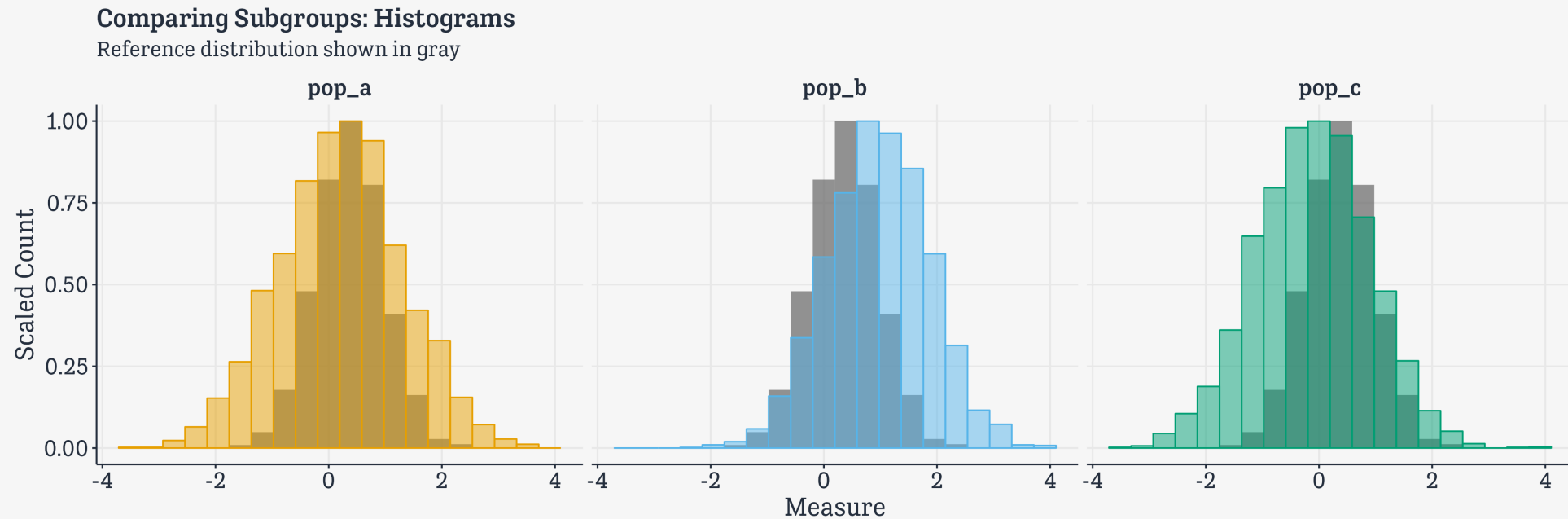
```
p_out <- df >
pivot_longer(cols = pop_a:pop_c) >
ggplot() +
  geom_histogram(mapping = aes(x = pop_total, #<<
                                y = after_stat(ncount)),
                 bins = 20, alpha = 0.7,
                 fill = "gray40", linewidth = 0.5) +
  geom_histogram(mapping = aes(x = value, #<<
                                y = after_stat(ncount),
                                color = name, fill = name),
                 stat = "bin", bins = 20, linewidth = 0.5,
                 alpha = 0.5) +
  guides(color = "none", fill = "none") + #<<
  labs(x = "Measure", y = "Scaled Count",
       title = "Comparing Subgroups: Histograms",
       subtitle = "Reference distribution shown in gray")
```

Remember, we can layer geoms one on top of the other. Here we call `geom_histogram()` twice.

What happens if you comment one or other of them out?

The call to `guides()` turns off the legend for the color and fill, because we don't need them.

Now facet with that data



**Avoid counting up,
when necessary**

Sometimes no counting is needed

```
titanic
```

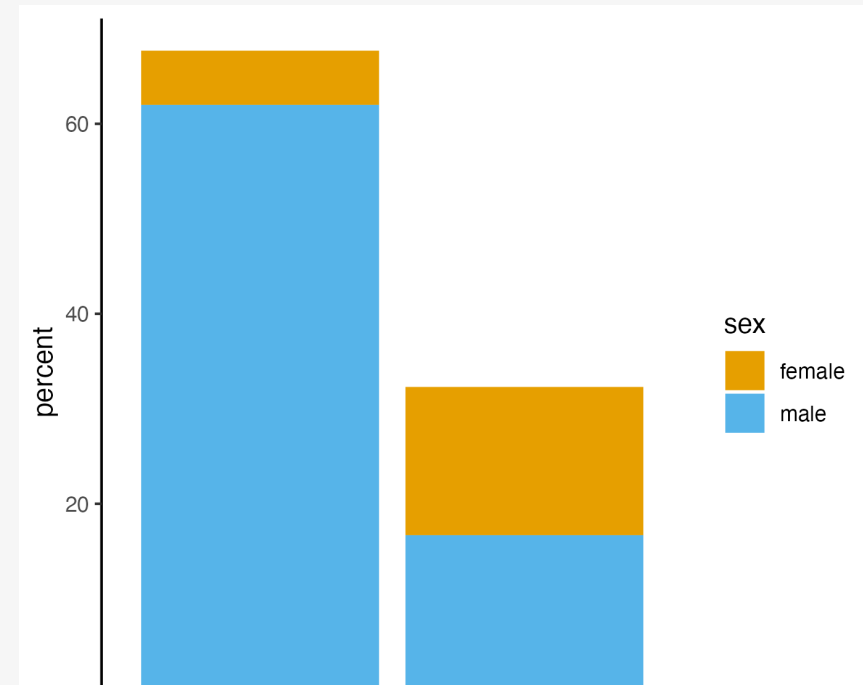
| | fate | sex | n | percent |
|---|----------|--------|------|---------|
| 1 | perished | male | 1364 | 62.0 |
| 2 | perished | female | 126 | 5.7 |
| 3 | survived | male | 367 | 16.7 |
| 4 | survived | female | 344 | 15.6 |

Here we just have a summary table and want to plot a few numbers directly in a bar chart.

geom_bar() wants to count up

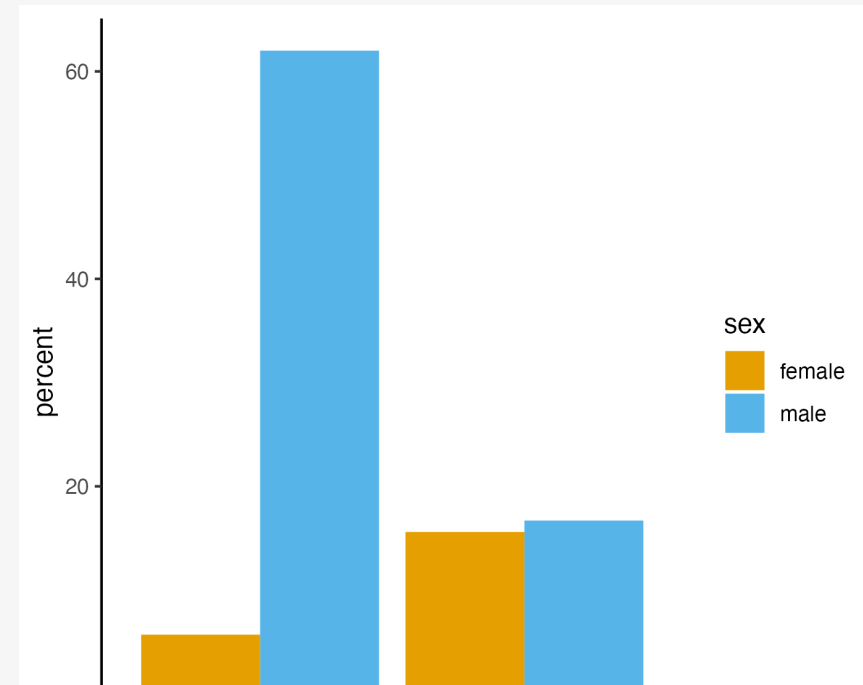
By default `geom_bar()` tries to count up data by category. (Really it's the `stat_count()` function that does this behind the scenes.) By saying `stat="identity"` we explicitly tell it not to do that. This also allows us to use a `y` mapping. Normally this would be the result of the counting up.

```
p ← ggplot(data = titanic,  
           mapping = aes(x = fate,  
                         y = percent,  
                         fill = sex))  
p + geom_bar(stat = "identity") #<<
```



geom_bar() stacks by default

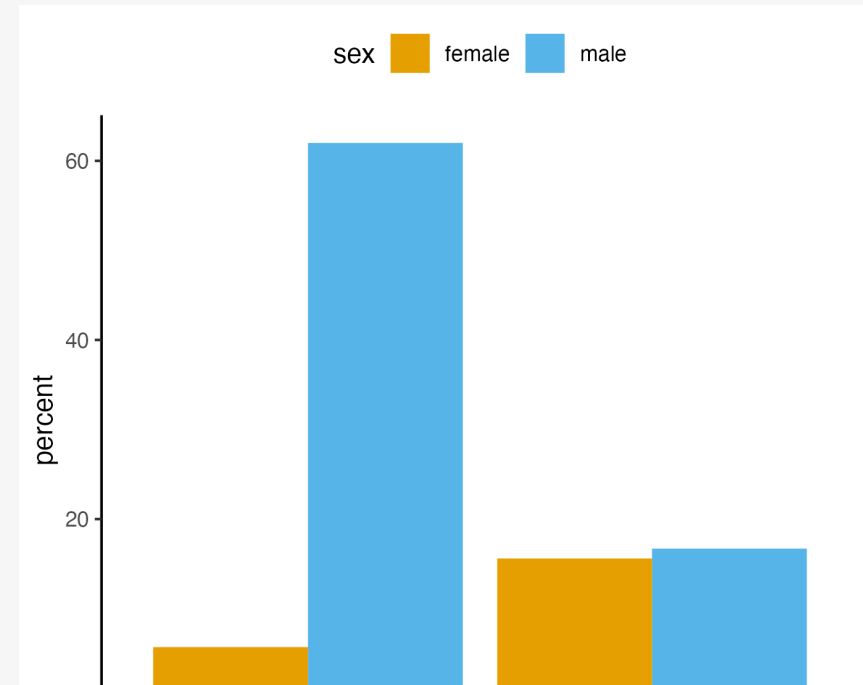
```
p ← ggplot(data = titanic,  
           mapping = aes(x = fate,  
                         y = percent,  
                         fill = sex))  
p + geom_bar(stat = "identity",  
            position = "dodge") #<<
```



A quick `theme()` adjustment

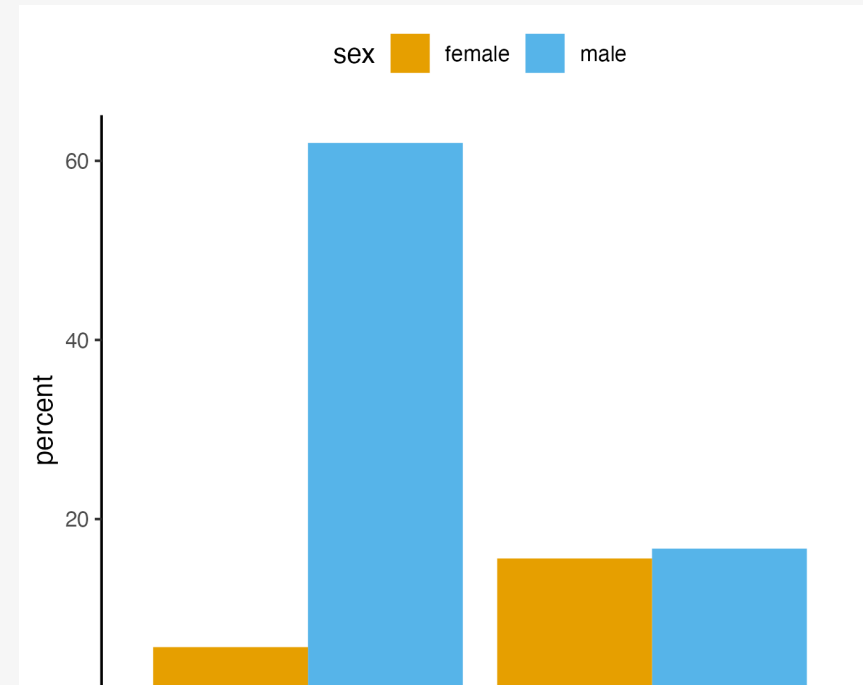
The `theme()` function controls the styling of parts of the plot that don't belong to its “grammatical” structure. That is, that are not contributing to directly representing data.

```
p ← ggplot(data = titanic,
           mapping = aes(x = fate,
                        y = percent,
                        fill = sex))
p + geom_bar(stat = "identity",
            position = "dodge") +
  theme(legend.position = "top") #<<
```



For convenience, use `geom_col()`

```
p ← ggplot(data = titanic,  
           mapping = aes(x = fate,  
                         y = percent,  
                         fill = sex))  
p + geom_col(position = "dodge") + #<<  
  theme(legend.position = "top")
```



Using `geom_col()` for thresholds

```
oecd_sum
```

```
# A tibble: 57 × 5
# Groups:   year [57]
  year other    usa diff hi_lo
  <int> <dbl> <dbl> <dbl> <chr>
1  1960  68.6  69.9  1.30 Below
2  1961  69.2  70.4  1.20 Below
3  1962  68.9  70.2  1.30 Below
4  1963  69.1  70    0.900 Below
5  1964  69.5  70.3  0.800 Below
6  1965  69.6  70.3  0.700 Below
7  1966  69.9  70.3  0.400 Below
8  1967  70.1  70.7  0.600 Below
9  1968  70.1  70.4  0.300 Below
10 1969  70.1  70.6  0.5    Below
# i 47 more rows
```

Data comparing U.S. average life expectancy
to the rest of the OECD average.

`diff` is difference in years with respect to
the U.S.

`hi_lo` is a flag saying whether the OECD is
above or below the U.S.

Using `geom_col()` for thresholds

```
p ← ggplot(data = oecd_sum,
            mapping = aes(x = year,
                          y = diff,
                          fill = hi_lo))

p_out ← p + geom_col() +
  geom_hline(yintercept = 0, linewidth = 1) +
  guides(fill = "none") +
  labs(x = NULL, #<<
       y = "Difference in Years",
       title = "The U.S. Life Expectancy Gap",
       subtitle = "Difference between U.S.
                  OECD average life expectancies, 1960-2014",
       caption = "Data: OECD.")
```

`geom_hline()` doesn't take any data argument. It just draws a horizontal line with a given y-intercept.

`x = NULL` means "Don't label the x-axis (not even with the default value, the variable name).

Using `geom_col()` for thresholds

