Show the Right Numbers

Data Visualization: Session 4

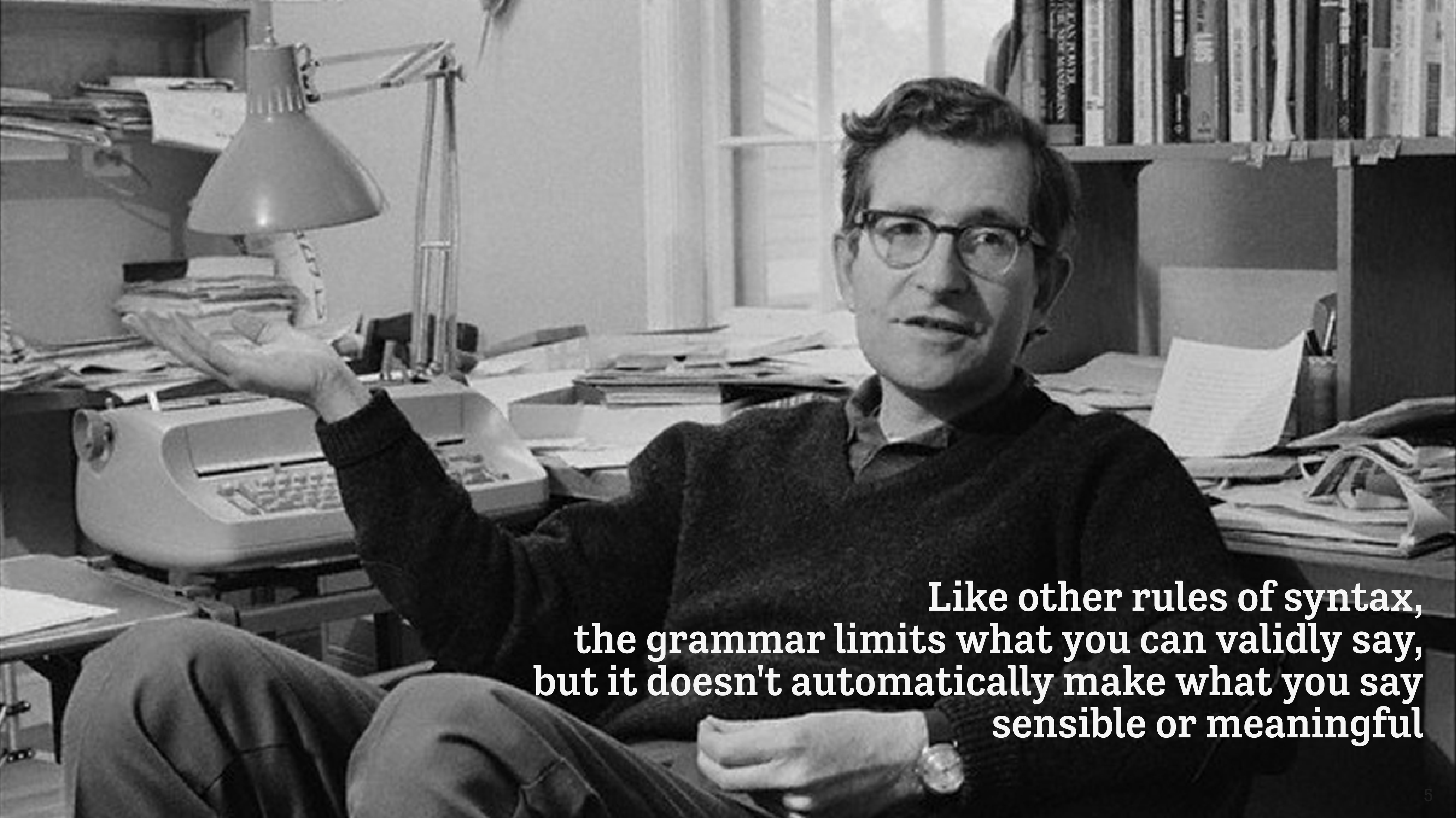
Kieran Healy
Code Horizons, May 2022

Set up our workspace

```r
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

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

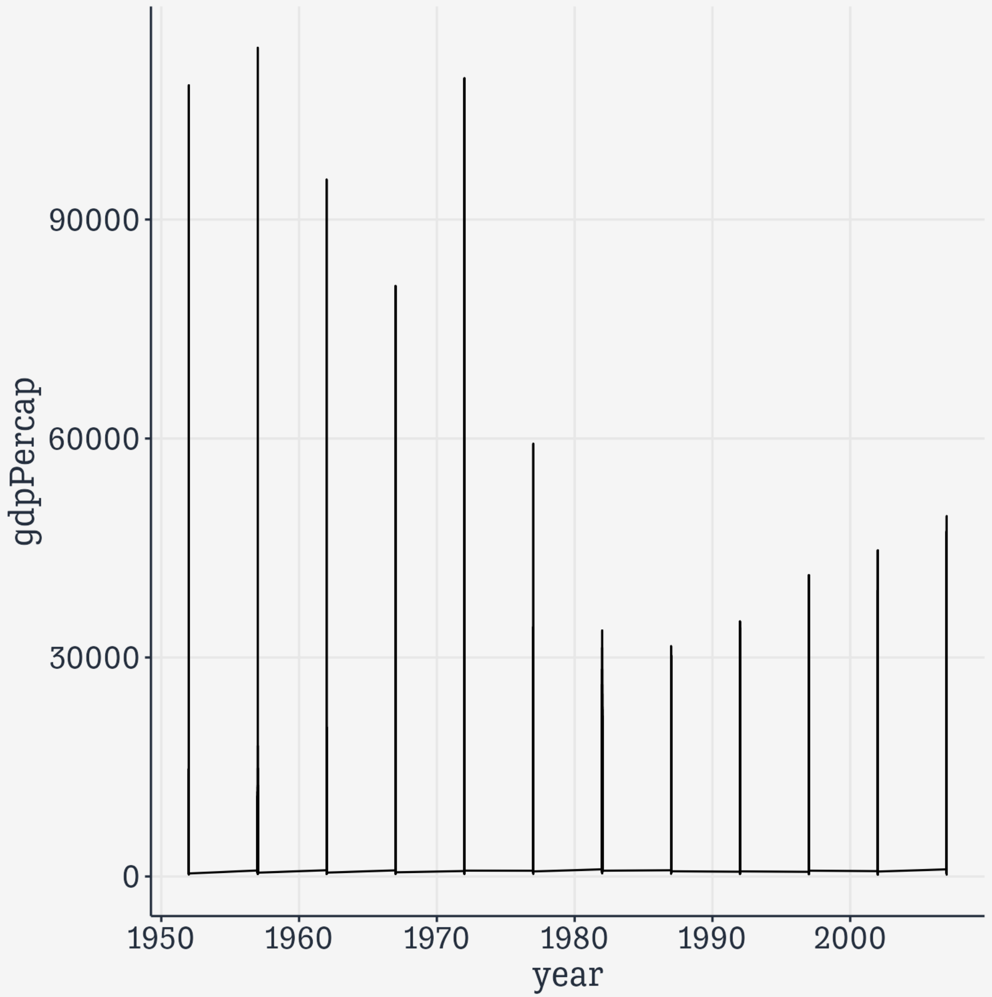
Grouped data and the `group` aesthetic

Try to make a lineplot

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

Try to make a lineplot

```r
p <- ggplot(data = gapminder,
            mapping = aes(x = year,
                          y = gdpPercap))
p + geom_line()
```

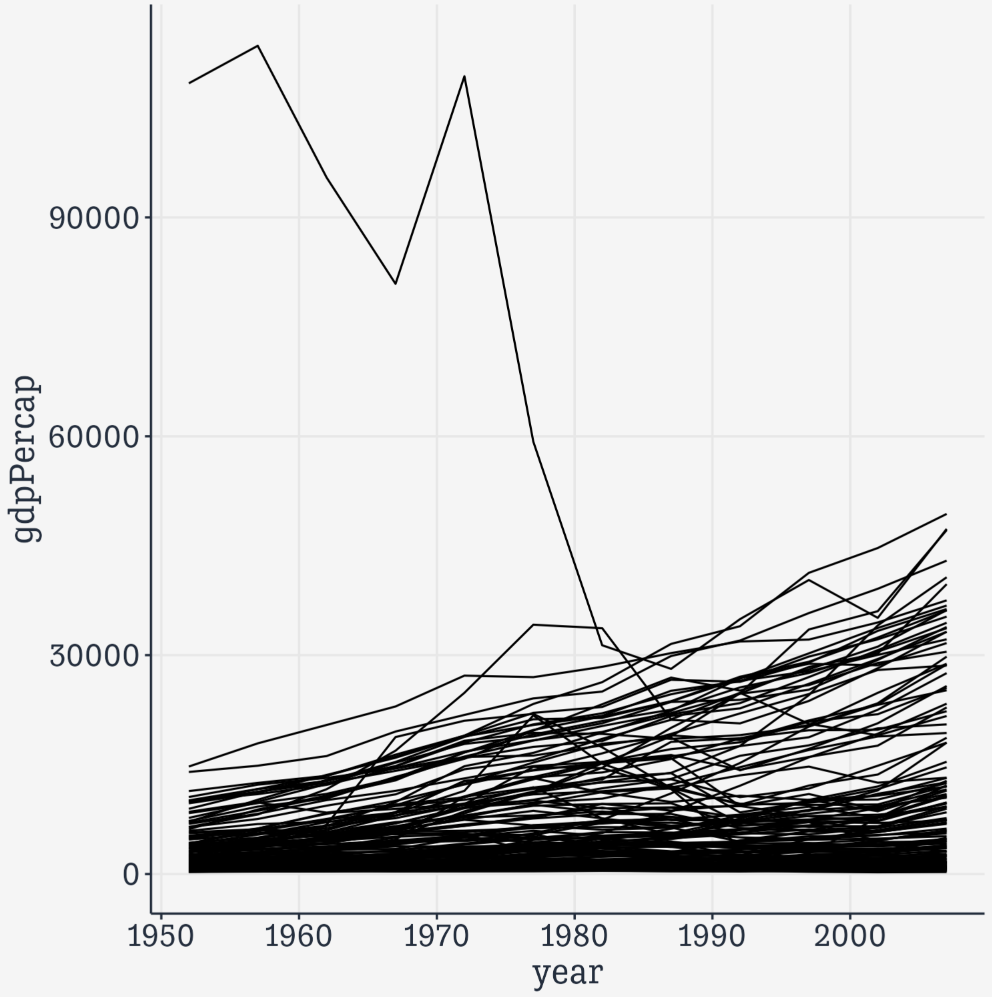


Try to make a lineplot

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

Try to make a lineplot

```r
p <- ggplot(data = gapminder,
            mapping = aes(x = year,
                          y = gdpPercap))

## Geoms can take their own mappings, remember
p + geom_line(mapping = aes(group = country))
```

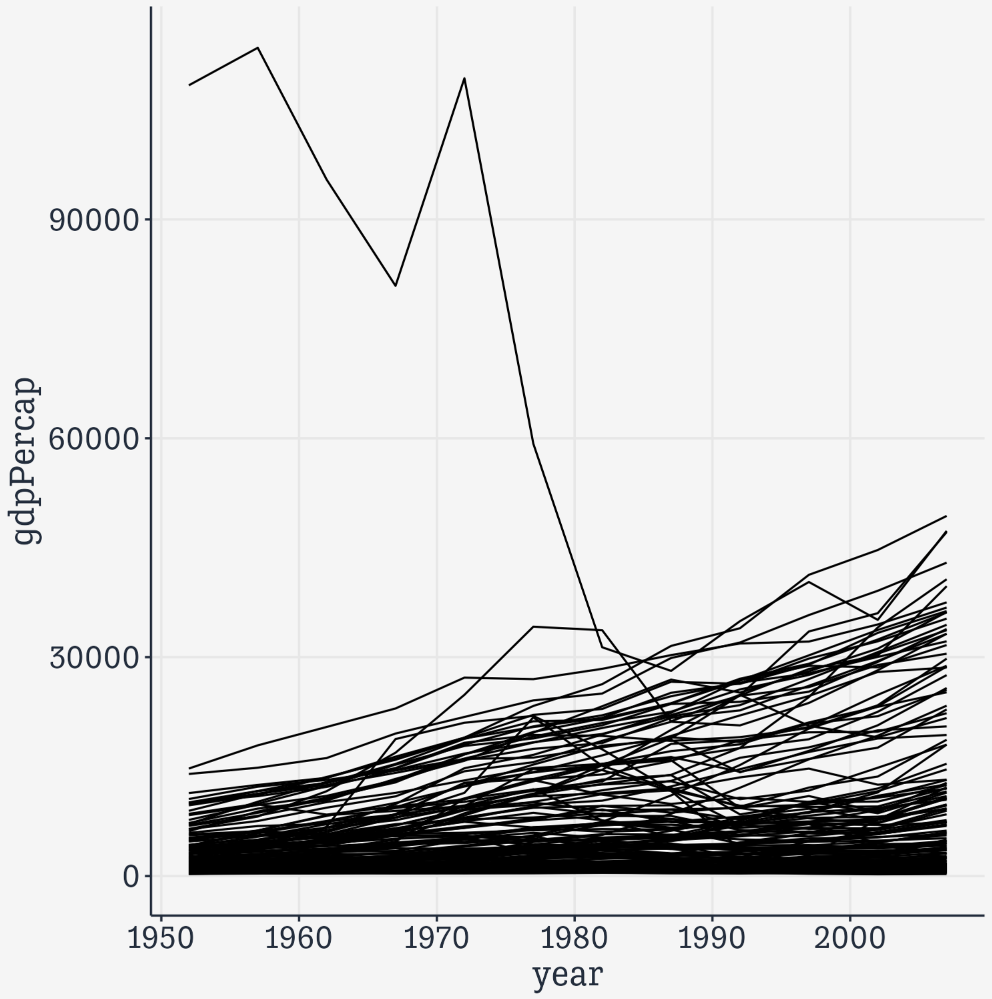


Facet the plot

```r
p <- ggplot(data = gapminder,
            mapping = aes(x = year,
                          y = gdpPercap))
```

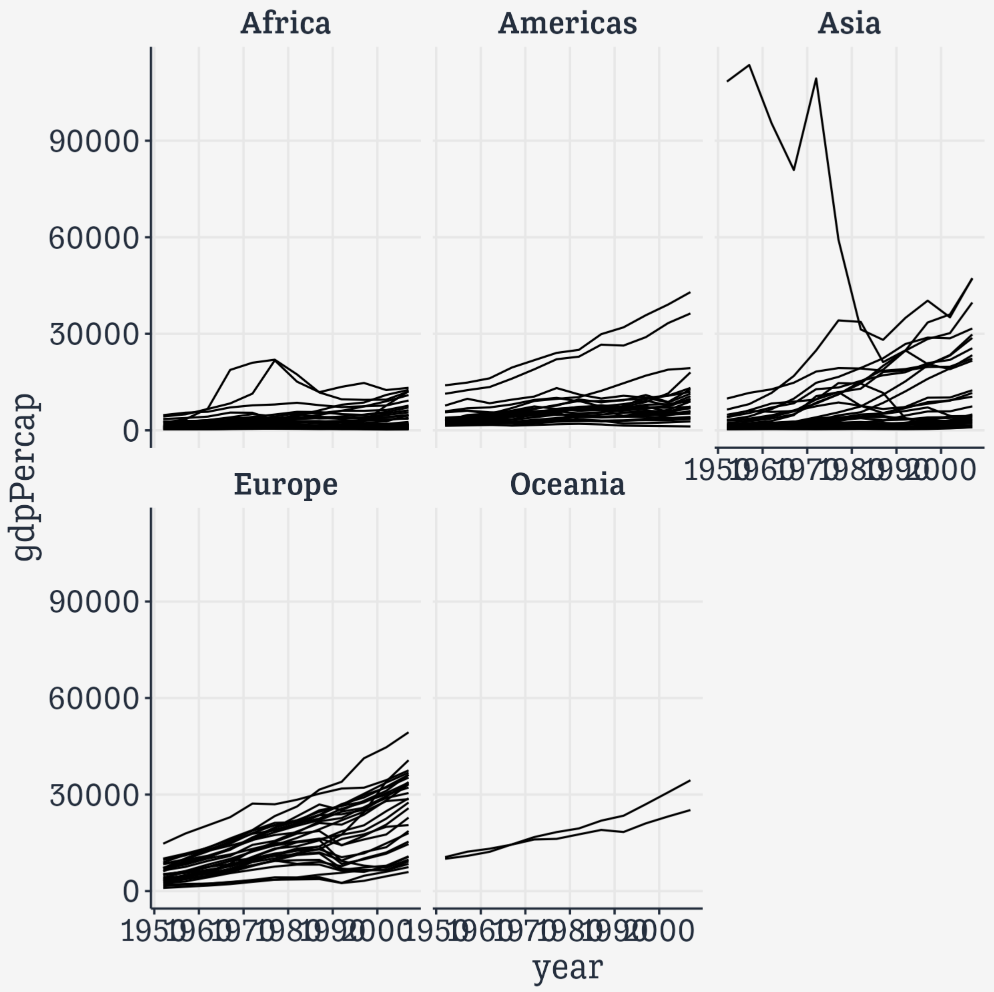
Facet the plot

```r
p <- ggplot(data = gapminder,
            mapping = aes(x = year,
                          y = gdpPercap))

## Geoms can take their own mappings, remember
p + geom_line(mapping = aes(group = country))
```



Facet the plot

```r
p <- ggplot(data = gapminder,
            mapping = aes(x = year,
                          y = gdpPercap))

## Geoms can take their own mappings, remember
p + geom_line(mapping = aes(group = country)) +
  facet_wrap(~ continent)
```



Faceting is a very powerful tool

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

Faceting is a very powerful tool

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)`

Faceting is a very powerful tool

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 is a very powerful tool

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"

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

Faceting is a very powerful tool

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)`

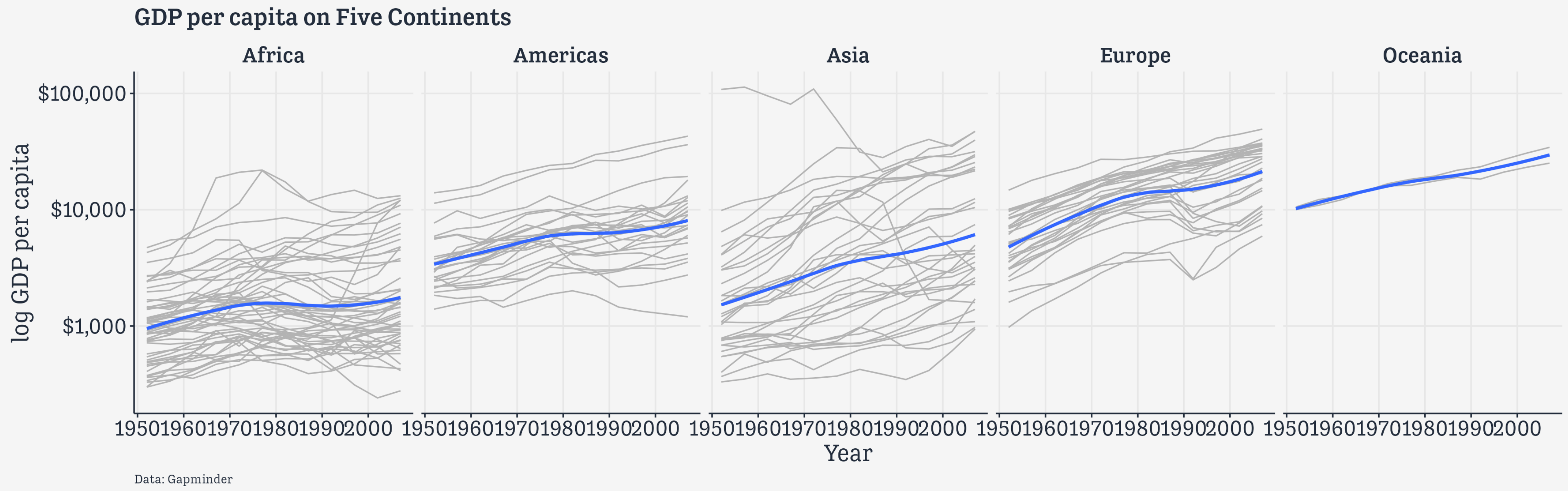
Read the `~` as "on" or "by"

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

```r
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

midwest

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

```r
midwest
```

```
## # A tibble: 437 × 28
##      PID county    state  area poptotal popdensity popwhite popblack popamerindian popasian popother percwhite percblack percame
##    <int> <chr>     <chr> <dbl>    <int>      <dbl>    <int>    <int>         <int>    <int>    <int>     <dbl>     <dbl>
##  1   561 ADAMS     IL    0.052    66090      1271.    63917     1702            98      249      124      96.7    2.58
##  2   562 ALEXANDER IL    0.014    10626       759      7054     3496            19       48        9      66.4   32.9
##  3   563 BOND      IL    0.022    14991       681.    14477      429            35       16       34      96.6    2.86
##  4   564 BOONE     IL    0.017    30806      1812.    29344      127            46      150     1139      95.3    0.412
##  5   565 BROWN     IL    0.018     5836       324.     5264      547            14        5        6      90.2    9.37
##  6   566 BUREAU    IL    0.05     35688       714.    35157       50            65      195      221      98.5    0.140
##  7   567 CALHOUN   IL    0.017     5322       313.     5298        1             8       15        0      99.5    0.0188
##  8   568 CARROLL   IL    0.027    16805       622.    16519      111            30       61       84      98.3    0.661
##  9   569 CASS      IL    0.024    13437       560.    13384       16             8       23        6      99.6    0.119
## 10   570 CHAMPAIGN IL    0.058   173025      2983.   146506    16559           331     8033     1596      84.7    9.57
## # … with 427 more rows, and 13 more variables: percother <dbl>, popadults <int>, perchsd <dbl>, percollege <dbl>, percprof <dbl>
## #   poppovertyknown <int>, percpovertyknown <dbl>, percbelowpoverty <dbl>, percchildbelowpovert <dbl>, percadultpoverty <dbl>,
## #   percelderlypoverty <dbl>, inmetro <int>, category <chr>
```

stat_ functions work behind the scenes

```
p <- ggplot(data = midwest,
            mapping = aes(x = area))

p + geom_histogram()
```

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

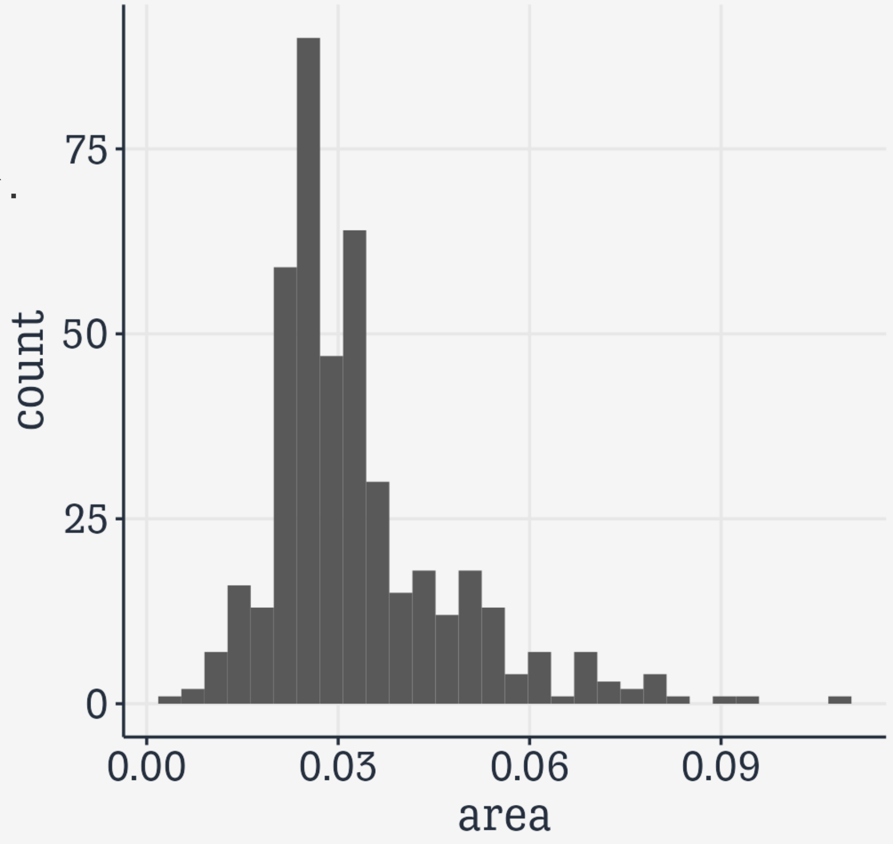
Here the default `stat_` function for this geom has to make a choice. It is letting us know we might want to override it.

stat_ functions work behind the scenes

```r
p <- ggplot(data = midwest,
            mapping = aes(x = area))

p + geom_histogram()
```

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

Here the default stat_ function for this geom has to make a choice. It is letting us know we might want to override it.



`stat_` functions work behind the scenes

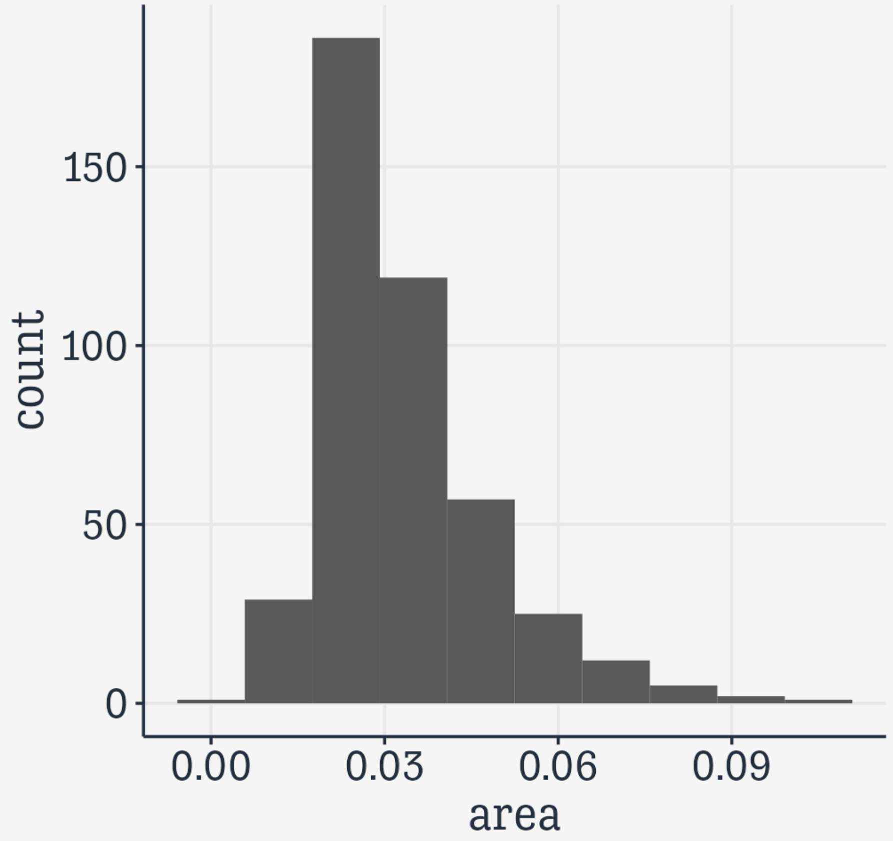
```
p <- ggplot(data = midwest,
            mapping = aes(x = area))

p + geom_histogram(bins = 10)
```

We can choose *either* the number of bins *or* the `binwidth`

`stat_` functions work behind the scenes

```r
p <- ggplot(data = midwest,
            mapping = aes(x = area))

p + geom_histogram(bins = 10)
```

We can choose *either* the number of bins *or* the `binwidth`



Compare two distributions

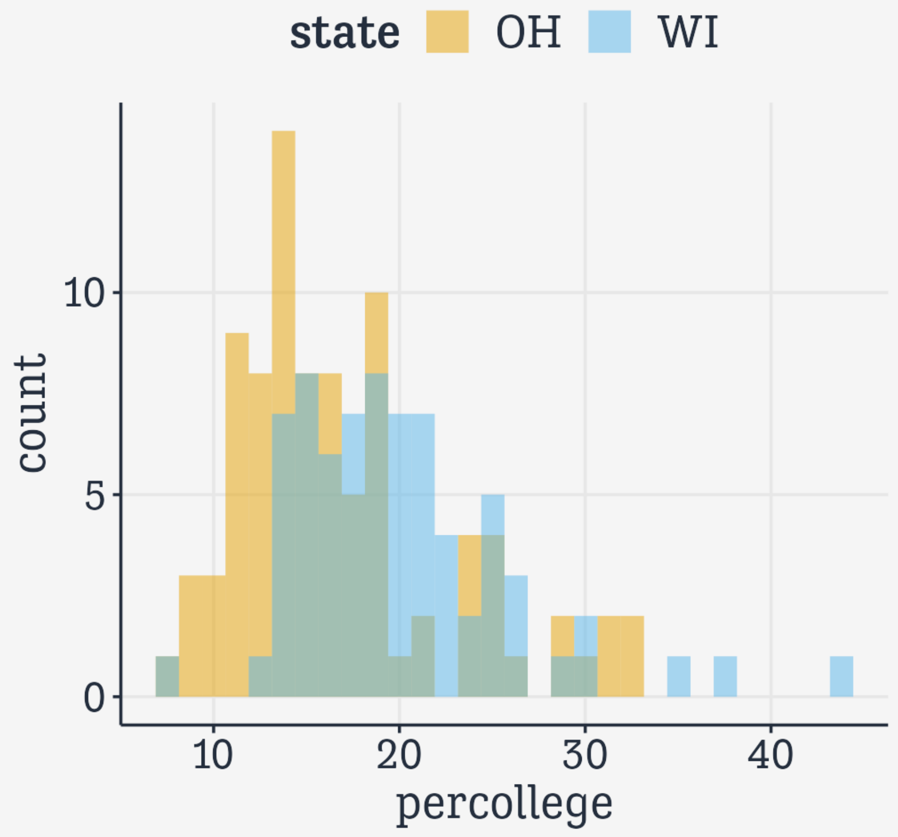
```r
## Two state codes
oh_wi <- c("OH", "WI")

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

Here we do the whole thing in a pipeline using the pipe and the `dplyr` verb `filter()` to subset rows of the data by some condition. Experiment with leaving the `position` argument out, or changing it to `"dodge"`.

Compare two distributions

```r
## 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")
```

Here we do the whole thing in a pipeline using the pipe and the `dplyr` verb `filter()` to subset rows of the data by some condition. Experiment with leaving the `position` argument out, or changing it to `"dodge"`.

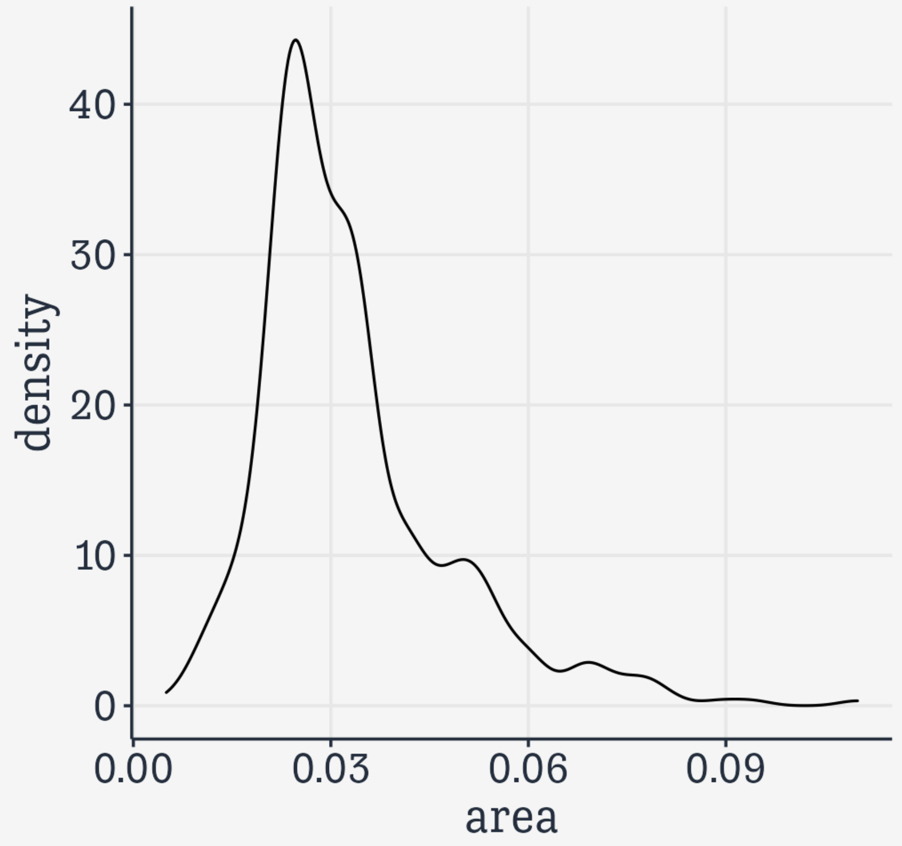


geom_hist()'s counterpart, geom_density()

```r
p <- ggplot(data = midwest,
            mapping = aes(x = area))

p + geom_density()
```

geom_hist()'s counterpart, geom_density()

```
p <- ggplot(data = midwest,
            mapping = aes(x = area))

p + geom_density()
```

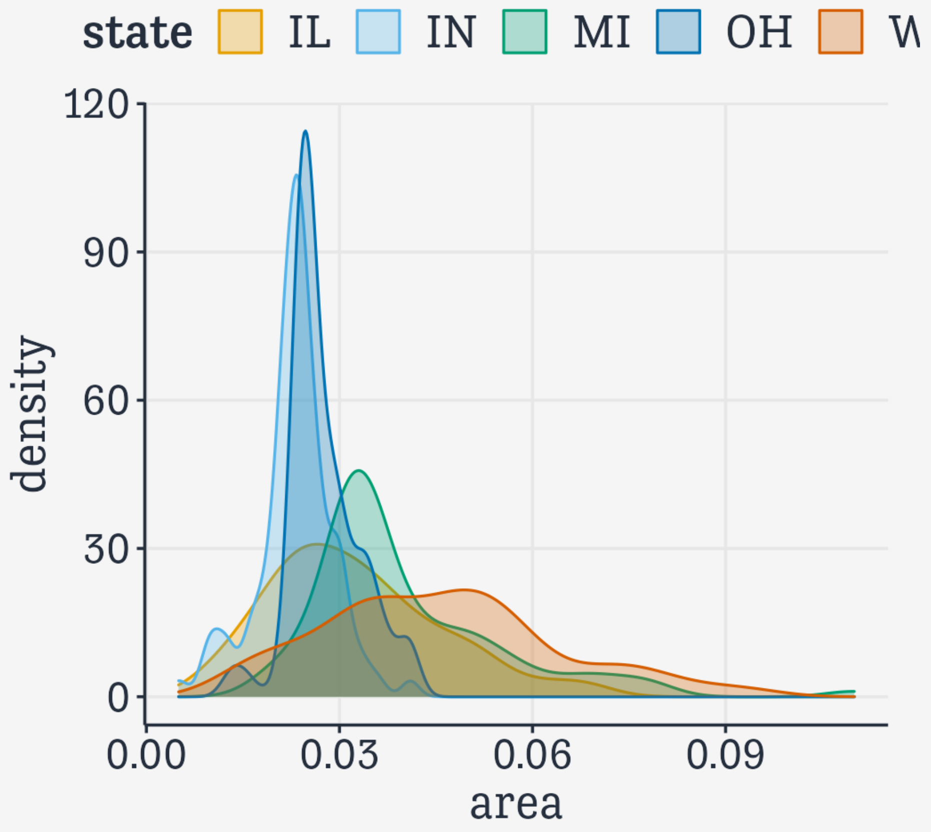


geom_hist()'s counterpart, geom_density()

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

geom_hist()'s counterpart, geom_density()

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



geom_hist()'s counterpart, geom_density()

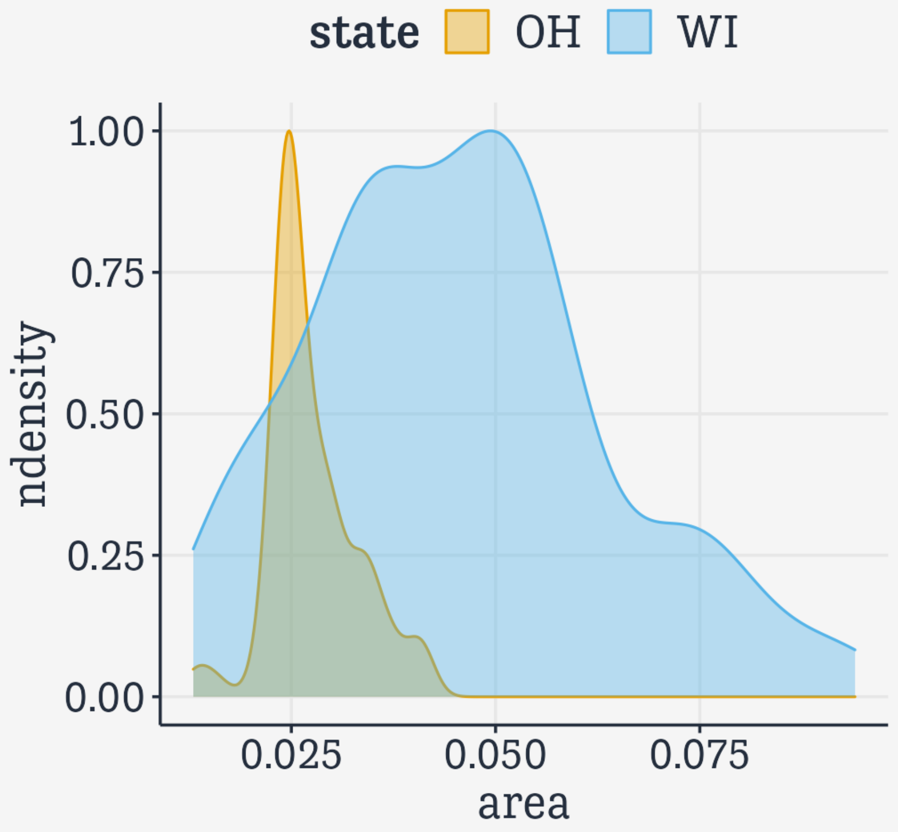
```r
midwest |>
  filter(state %in% oh_wi) |>
  ggplot(mapping = aes(x = area,
                       fill = state,
                       color = state)) +
  geom_density(mapping = aes(y = ..ndensity..),
               alpha = 0.4)
```

`..ndensity..` here is not in our data! It's *computed*. Histogram and density geoms have default statistics, but you can ask them to do more. The `stat_` functions associated with each `geom_` do this work behind the scenes.

geom_hist()'s counterpart, geom_density()

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

`..ndensity..` here is not in our data! It's *computed*. Histogram and density geoms have default statistics, but you can ask them to do more. The `stat_` functions associated with each `geom_` do this work behind the scenes.



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
## # … with 2,990 more rows
```

First effort: Hard to read

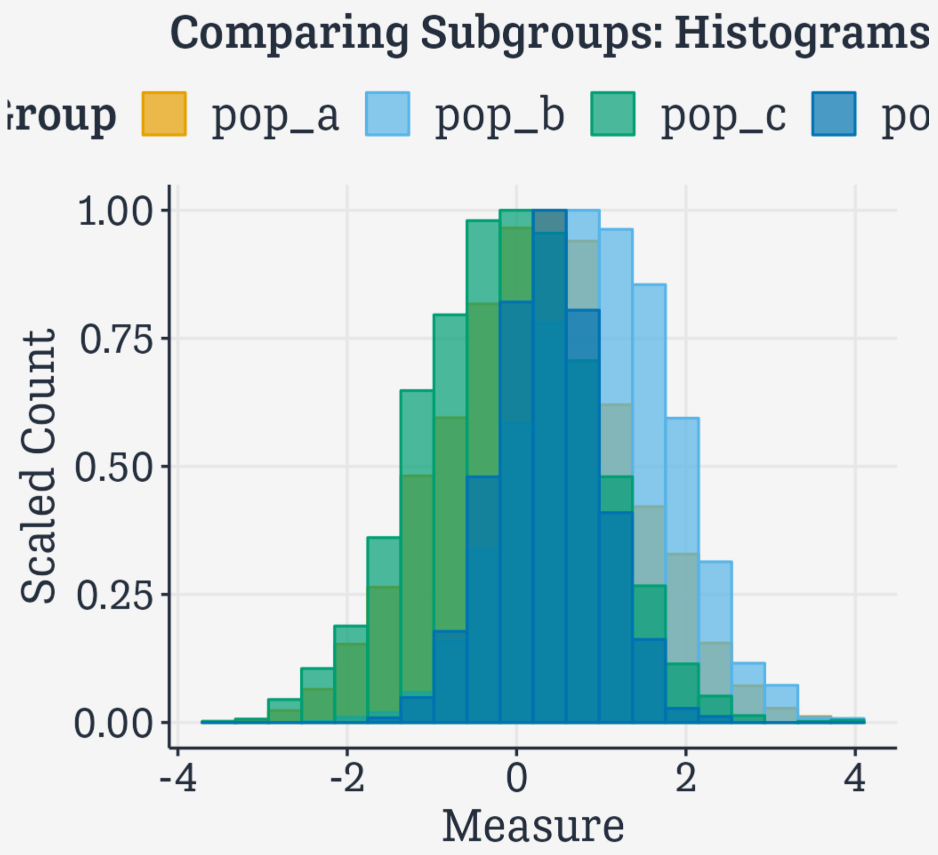
```
df |>
  pivot_longer(cols = pop_a:pop_total)  |>
  ggplot() +
  geom_histogram(mapping = aes(x = value,
                               y = ..ncount..,
                          color = name, fill = name),
            stat = "bin", bins = 20, size = 0.5,
            alpha = 0.7,
            position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

Again, `..ncount..` is computed. The periods on either side are just a naming convention to show that the measure is computed by the `stat_` function (and to make sure it doesn't clash with any actual names in your data.)

First effort: Hard to read

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

Again, `..ncount..` is computed. The periods on either side are just a naming convention to show that the measure is computed by the `stat_` function (and to make sure it doesn't clash with any actual names in your data.)



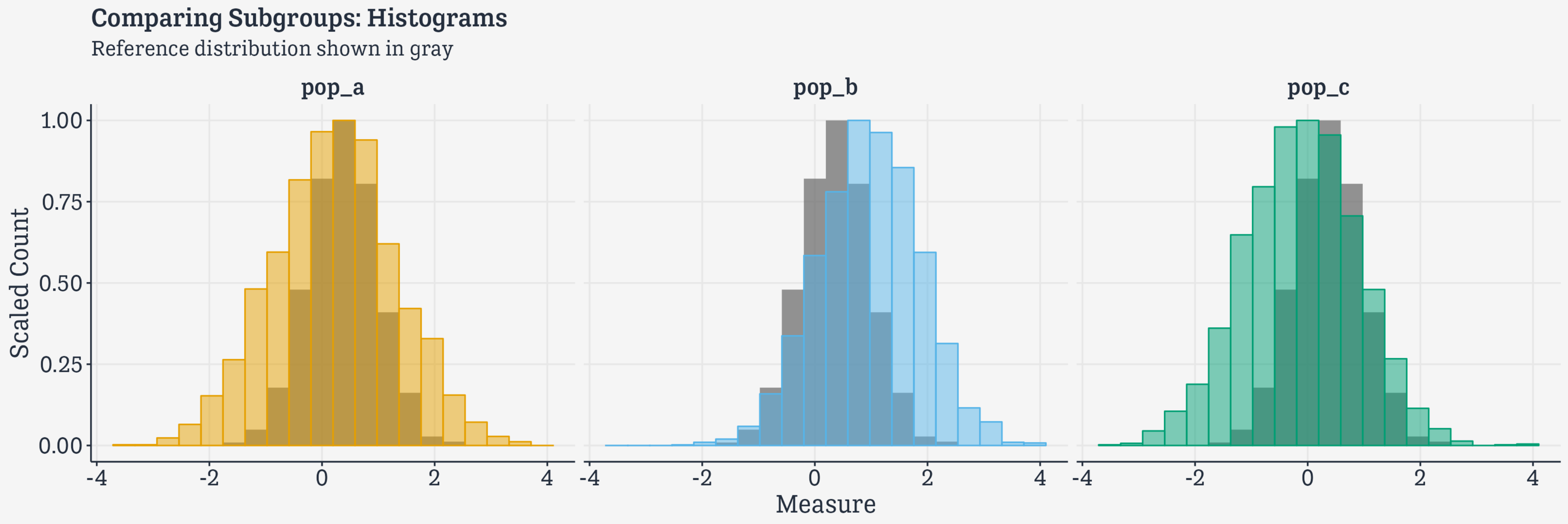
Try faceting instead

```r
p_out <- df |>
  pivot_longer(cols = pop_a:pop_c) |>
  ggplot() +
  geom_histogram(mapping = aes(x = pop_total,
                               y = ..ncount..),
              bins = 20, alpha = 0.7,
              fill = "gray40", size = 0.5) +
  geom_histogram(mapping = aes(x = value,
                               y = ..ncount..,
                          color = name, fill = name),
            stat = "bin", bins = 20, size = 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") +
  facet_wrap(~ name, nrow = 1)
```

Something we haven't seen before, but will be using a lot: 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.

Try faceting instead



Avoid counting up, if necessary

Sometimes no counting is required

```
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

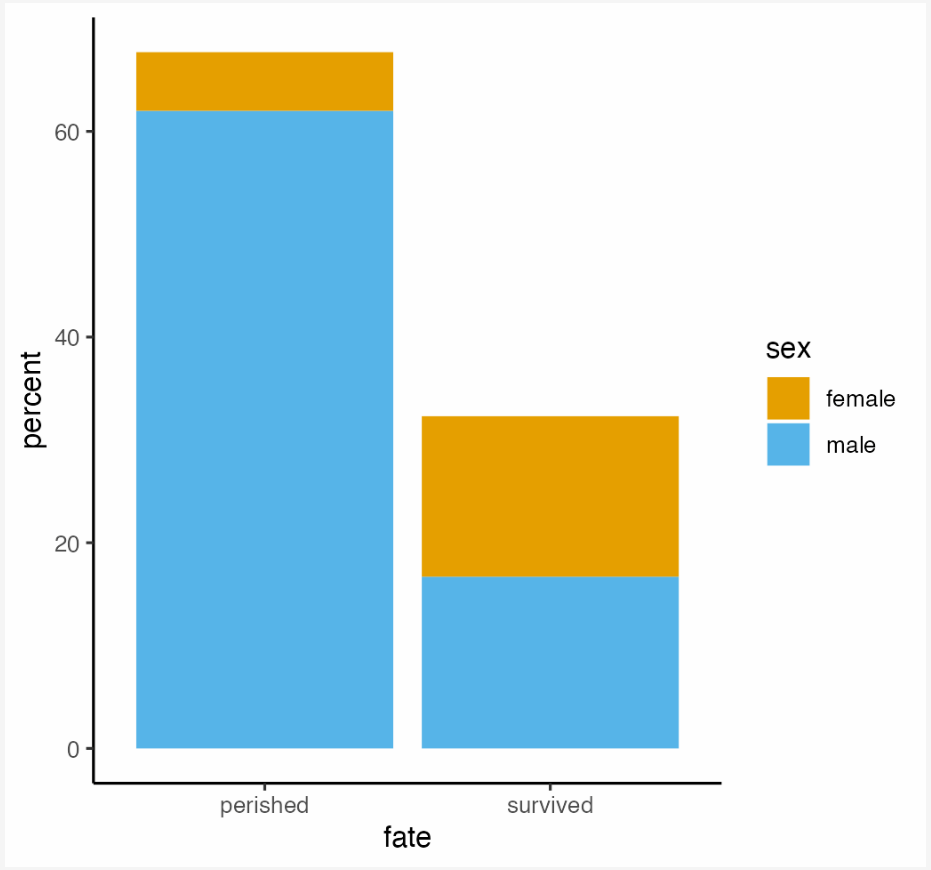
```
p <- ggplot(data = titanic,
            mapping = aes(x = fate,
                          y = percent,
                          fill = sex))
p + geom_bar(stat = "identity")
```

By default `geom_bar()` tries to count up data by category. By saying `stat="identity"` we explicitly tell it not to do that. This also allows us to use a y mapping, because normally this would be determined by the counting up.

geom_bar() wants to count up

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

By default `geom_bar()` tries to count up data by category. By saying `stat="identity"` we explicitly tell it not to do that. This also allows us to use a y mapping, because normally this would be determined by the counting up.



geom_bar() stacks bars by default

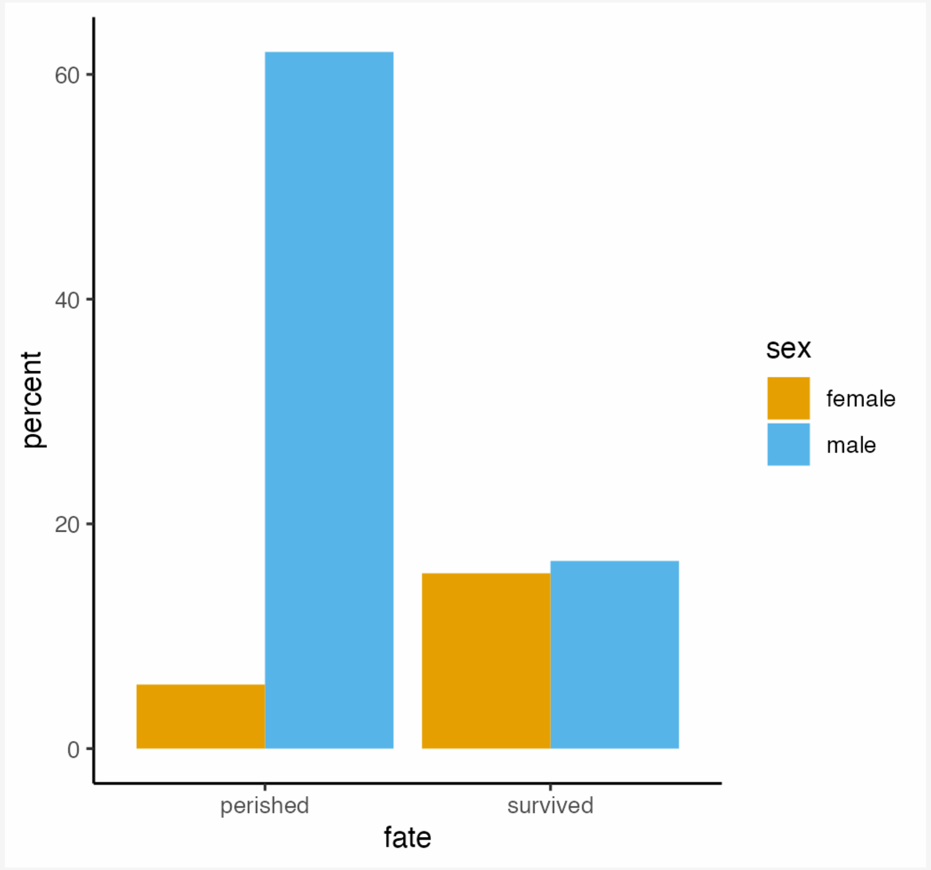
```r
p <- ggplot(data = titanic,
            mapping = aes(x = fate,
                          y = percent,
                          fill = sex))
p + geom_bar(stat = "identity",
             position = "dodge")
```

Position arguments adjust whether the things drawn are placed on top of one another (`"stack"`), side-by-side (`"dodge"`), or taken as-is (`"identity"`).

geom_bar() stacks bars by default

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

Position arguments adjust whether the things drawn are placed on top of one another (`"stack"`), side-by-side (`"dodge"`), or taken as-is (`"identity"`).



A quick theme() adjustment

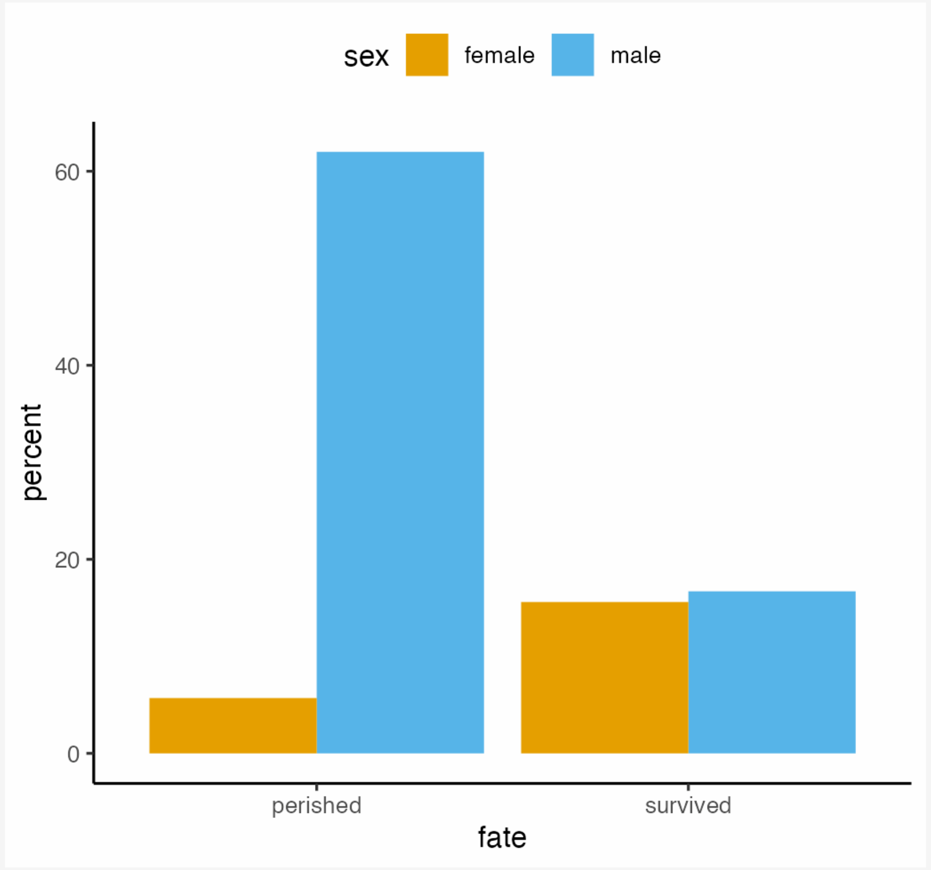
```r
p <- ggplot(data = titanic,
            mapping = aes(x = fate,
                          y = percent,
                          fill = sex))
p + geom_bar(stat = "identity",
             position = "dodge") +
  theme(legend.position = "top")
```

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.

A quick theme() adjustment

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

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.



For convenience, use geom_col()

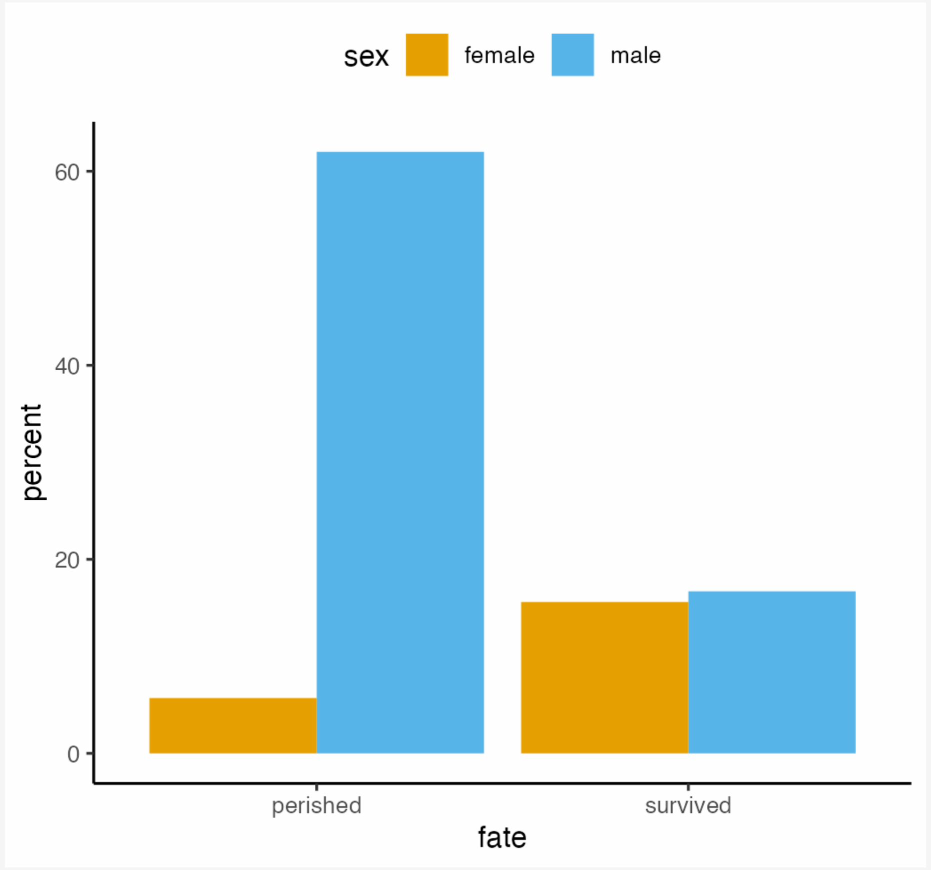
```
p <- ggplot(data = titanic,
            mapping = aes(x = fate,
                          y = percent,
                          fill = sex))
p + geom_col(position = "dodge") +
  theme(legend.position = "top")
```

`geom_col()` assumes `stat = "identity` by default. It's for when you want to directly plot a table of values.

For convenience, use geom_col()

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

`geom_col()` assumes `stat = "identity` by default. It's for when you want to directly plot a table of values.



Using geom_col() for thresholds

```r
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
## # … with 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.

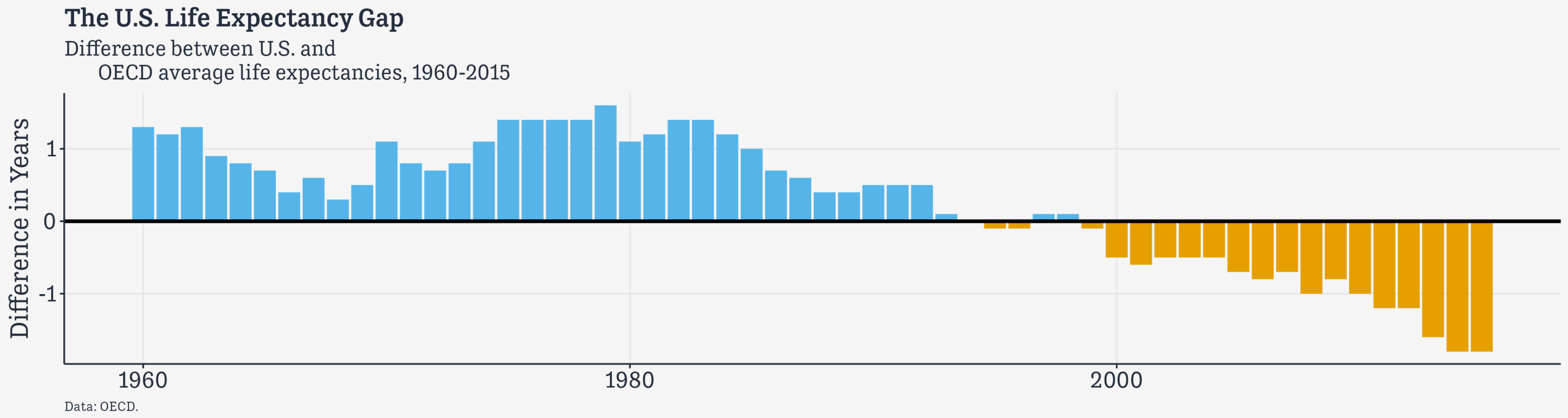
Using geom_col() for thresholds

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

p_out <- p + geom_col() +
  geom_hline(yintercept = 0, size = 1.2) +
  guides(fill = "none") +
  labs(x = NULL,
       y = "Difference in Years",
       title = "The U.S. Life Expectancy Gap",
       subtitle = "Difference between U.S. and
       OECD average life expectancies, 1960-2015",
       caption = "Data: OECD.")
```

`geom_hline()` 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


The U.S. Life Expectancy Gap
Difference between U.S. and
OECD average life expectancies, 1960-2015
Difference in Years
Data: OECD.