GGPLOT2 CONCEPTS

Outline

1. A layered grammar

2. Geometrical layers

3. Statistical layers

4. Facets

5. Storing a ggplot as an object

A layered grammar of graphics

- ggplot2 works on the philosophy of adding layers to the visualization
- there are 7 components of a plot

A layered grammar of graphics

- only 3 of these components are necessary to make a layer:
    - **data** are the subjects & objects of the data visualization
    - **Aesthetic mappings** (aes) substitute visual properties (aesthetics) for the data
    - The **geom** is what relates the data to a visual element.
- these three components allow us maximum flexibility to make subtle changes in each layer to clearly communicate our message.

The grammar of {ggplot2}

Component	Function	Explanation
Data	`ggplot(data)`	*The raw data that you want to visualise.*
Aesthetics	`aes()`	*Aesthetic mappings between variables and visual properties.*
Geometries	`geom_*()`	*The geometric shapes representing the data.*

The grammar of {ggplot2}

Component	Function	Explanation
Data	`ggplot(data)`	*The raw data that you want to visualise.*
Aesthetics	`aes()`	*Aesthetic mappings between variables and visual properties.*
Geometries	`geom_*()`	*The geometric shapes representing the data.*
Statistics	`stat_*()`	*The statistical transformations applied to the data.*
Scales	`scale_*()`	*Maps between the data and the aesthetic dimensions.*
Coordinate System	`coord_*()`	*The positioning of the data in a 2D data visualization.*
Facets	`facet_*()`	*The arrangement of the data into a grid of plots.*
Themes	`theme()` and `theme_*()`	*The overall visual defaults of a plot.*

Example

The Data

OECD's Program for International Student Assessment (PISA)

- aims to inform educational policies and practices
- 540,000 15-year-olds from 72 participating countries and economies
- designed to gauge mastery of key subjects (math, science, reading)
- background questionnaire with questions about themselves, their family and home, and their school and learning experiences

```
## devtools::install_github("haleyjeppson/NCME23data")
library(NCME23data)
## weighted US sample of size 99
data(pisa_usa)
## weighted sample of 1000
data(pisa_small)
```

Data Variables

```
##  [1] "country"                "OECD"
##  [3] "id"                     "weight"
##  [5] "sex"                    "grade"
##  [7] "computer"               "software"
##  [9] "internet"               "addit_time_math"
## [11] "addit_time_science"     "parent_support"
## [13] "parent_status"          "want_best_grades"
## [15] "want_best_student"      "test_anxiety"
## [17] "enjoy_cooperation"      "sense_of_belonging"
## [19] "parent_support_emotional" "HOMESCH"
## [21] "ENTUSE"                 "ICTHOME"
## [23] "ICTSCH"                 "wealth"
## [25] "parent_edu"             "learning_mins"
## [27] "escs_index"             "teacher_support_science"
## [29] "teacher_direct_science" "inquiry_based_science"
## [31] "science_self_efficacy"  "math"
## [33] "reading"                "science"
## [35] "learning_hours"         "region"
```

Data

```
ggplot(data = pisa_usa)
```

Aesthetic Mapping

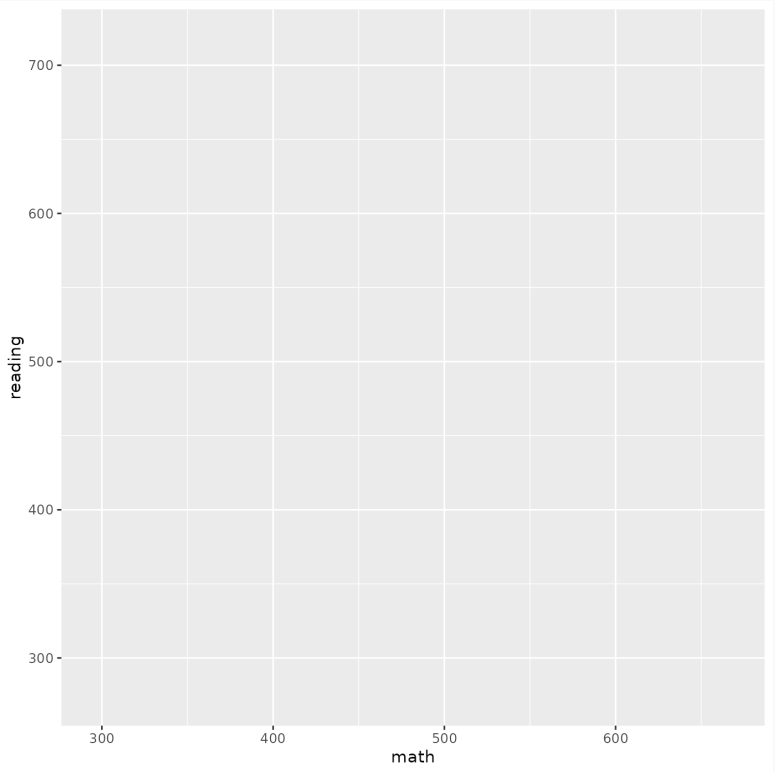
Link variables in data to graphical properties

- **Axes**: `x`, `y`
- **Grouping**: `group`
- **Other visual properties**: `color`, `fill`, `alpha` (transparency), `size`, `shape`, `linetype`
- **Other**: `weight`, `z`, `xmin`, `xmax`, `ymin`, `ymax`, ...

Aesthetic Mapping

Use the `aes()` function inside `ggplot()`

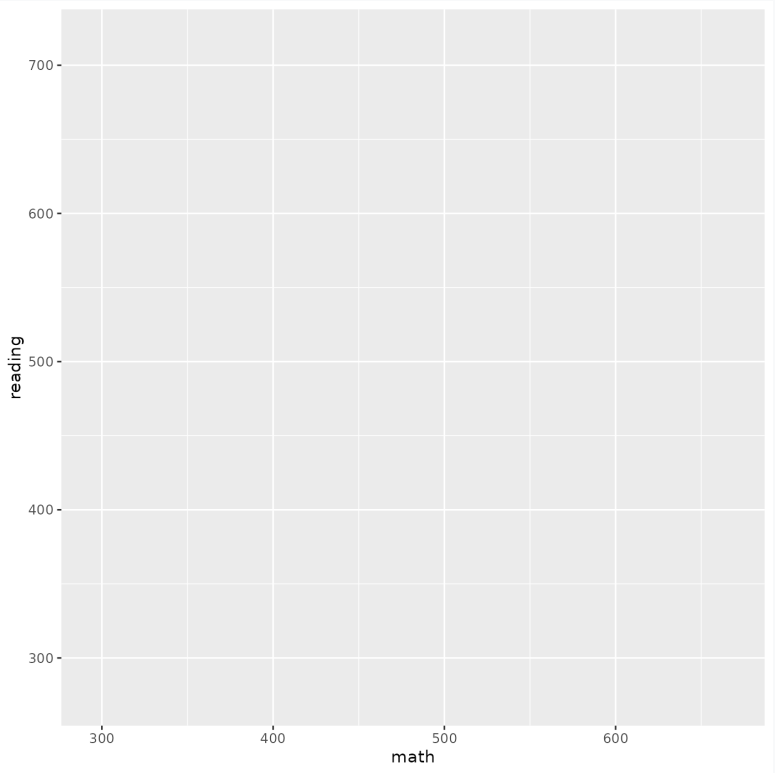
```
ggplot(data = pisa_usa,
  mapping = aes(x = math, y = reading))
```



Aesthetic Mapping

Use implicit matching

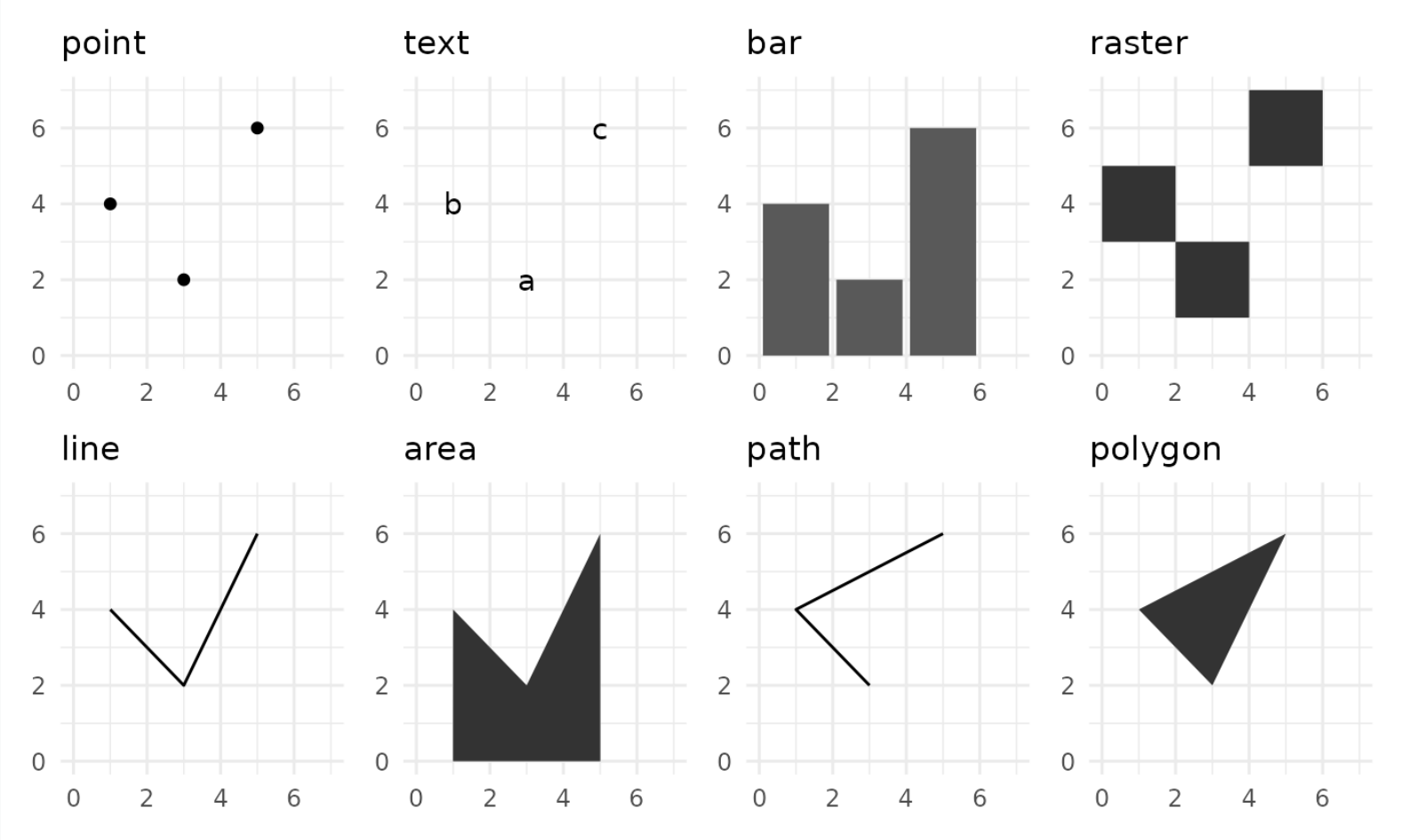
```
ggplot(pisa_usa,
       aes(x = math, y = reading))
```



Geometrical Layers

Geometries

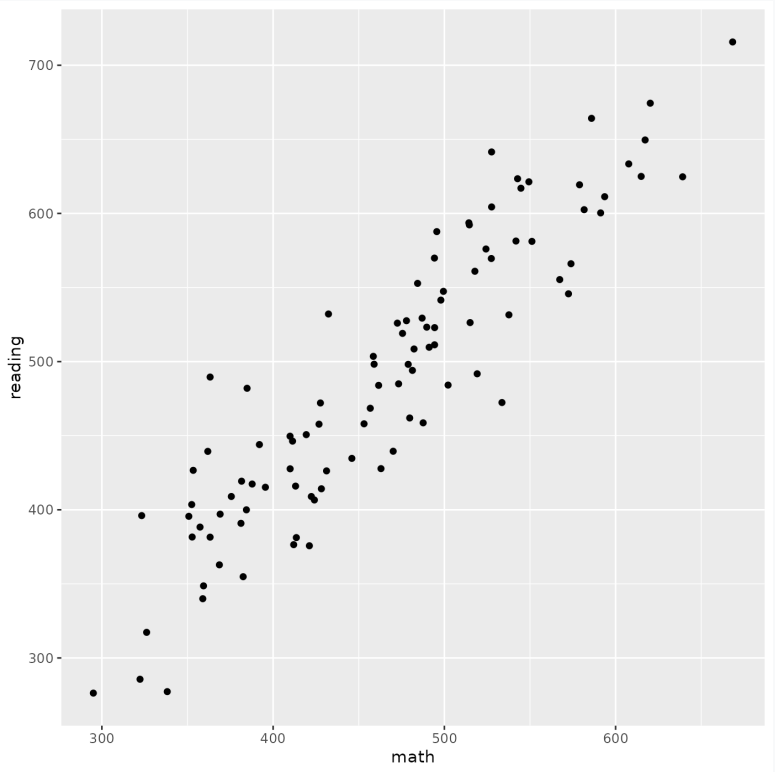
How to interpret aesthetics as graphical representations



Geometries

We build up a data visualization in ggplot2 with the + operator.

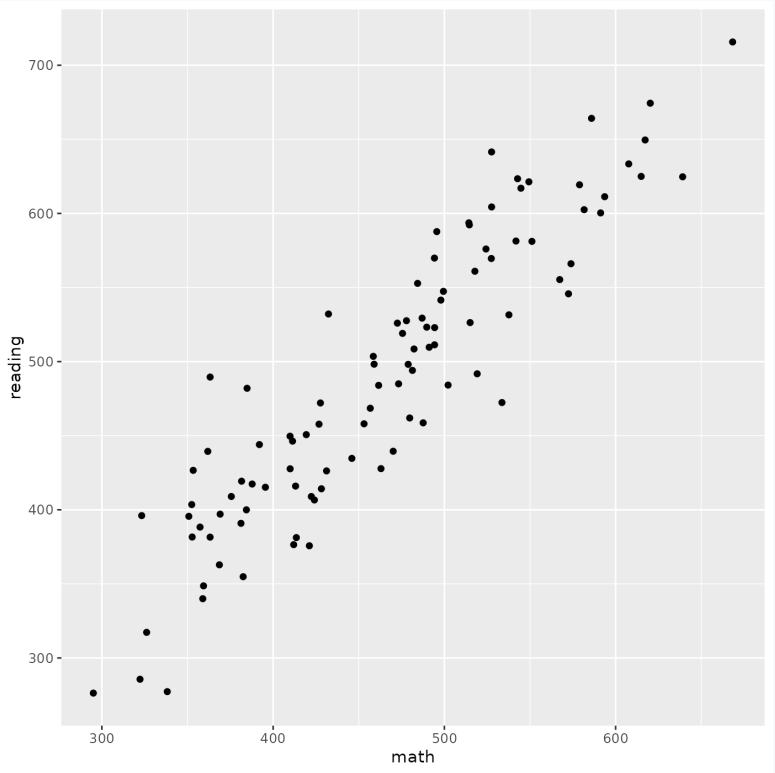
```
ggplot(
  pisa_usa,
  aes(x = math, y = reading)
  ) +
  geom_point()
```



Geometries

Define mappings for a particular geom only

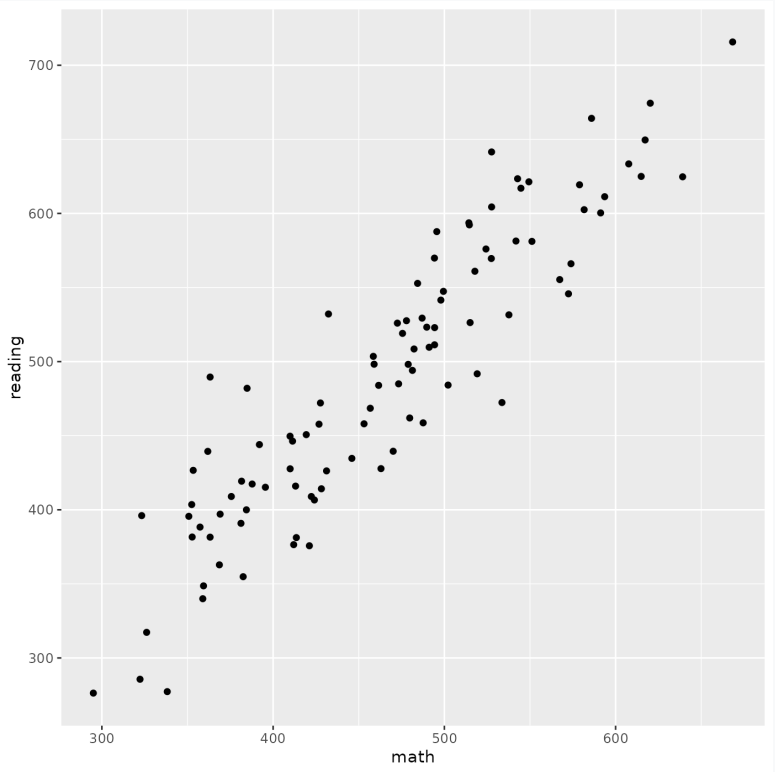
```
ggplot(pisa_usa) +
  geom_point(aes(x = math, y = reading))
```



Geometries

Define data for a particular geom only

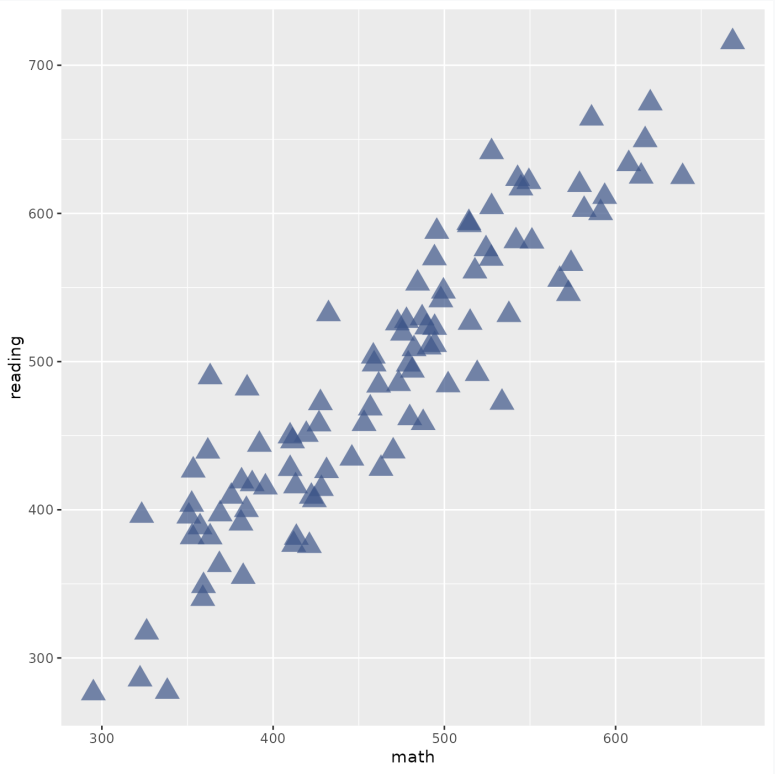
```
ggplot() +
  geom_point(data = pisa_usa,
             aes(x = math, y = reading)
```



Visual Properties of Layers

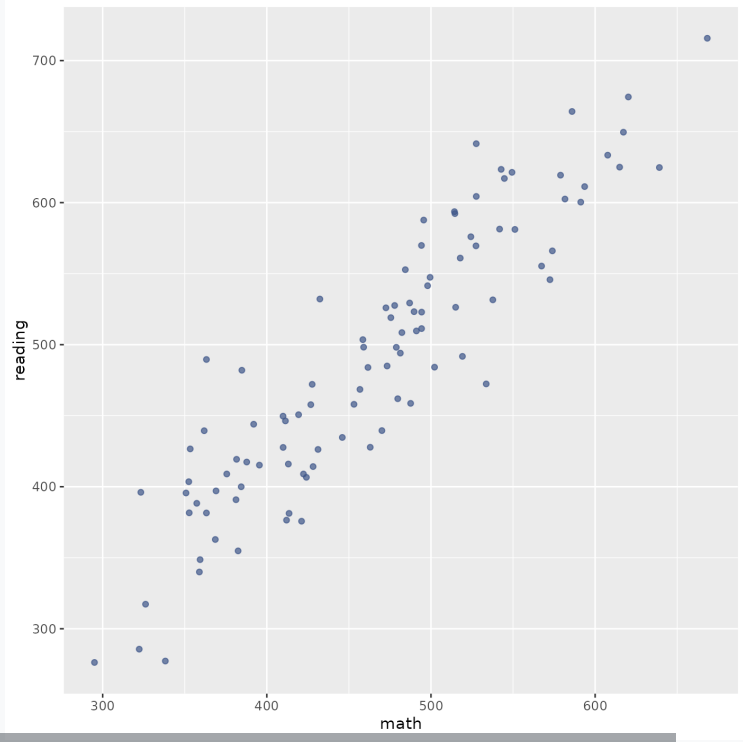
The `geom_*()` suite of functions can take many arguments, which vary by the geom type

```
ggplot(
    pisa_usa,
    aes(x = math, y = reading)
  ) +
  geom_point(
    color = "#3C5488",
    alpha = .7,
    shape = 17,
    stroke = 1,
    size = 5
  )
```

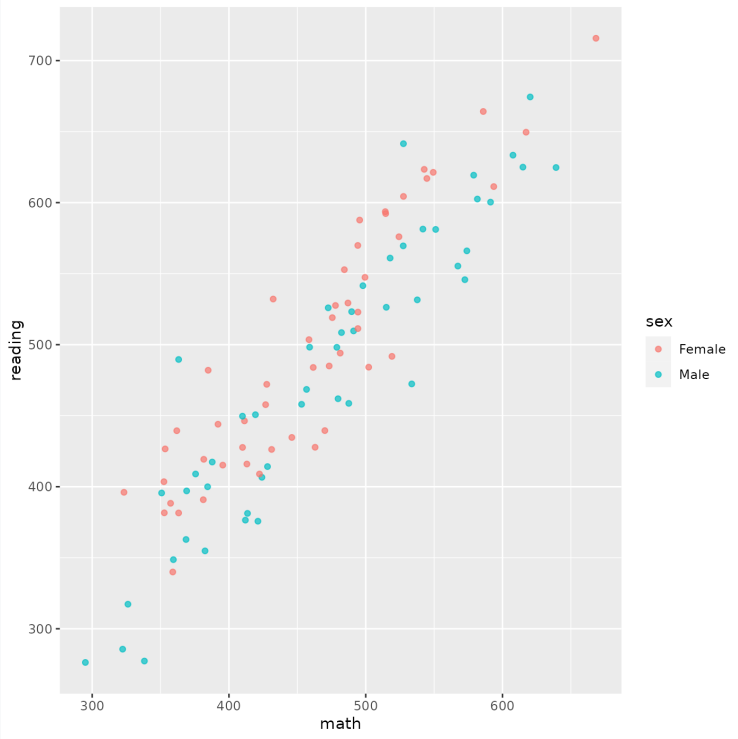


Setting vs Mapping Visual Properties

```
ggplot(
    pisa_usa,
    aes(x = math, y = reading)
  ) +
  geom_point(
    color = "#3C5488",
    alpha = .7
  )
```

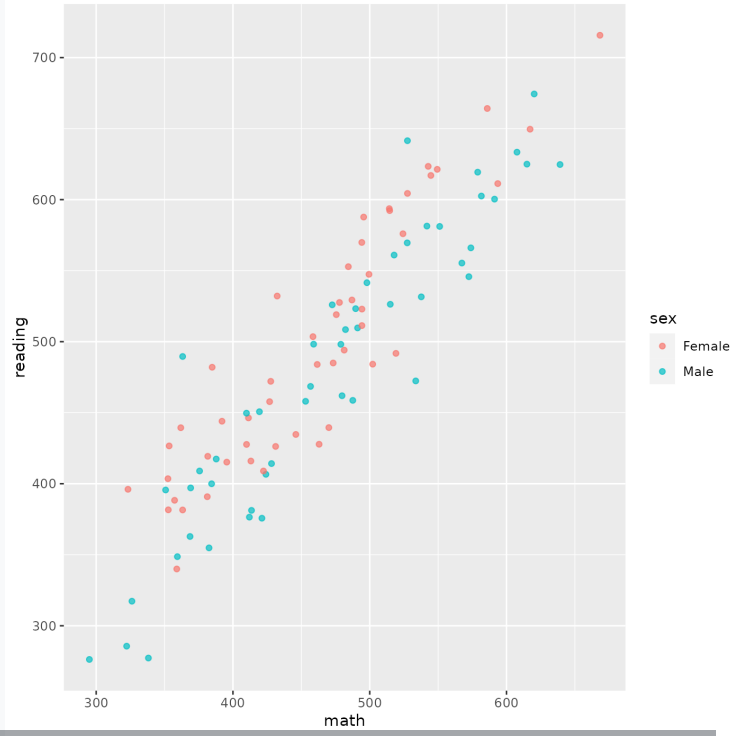



```
ggplot(
    pisa_usa,
    aes(x = math, y = reading)
  ) +
  geom_point(
    aes(color = sex),
    alpha = .7
  )
```

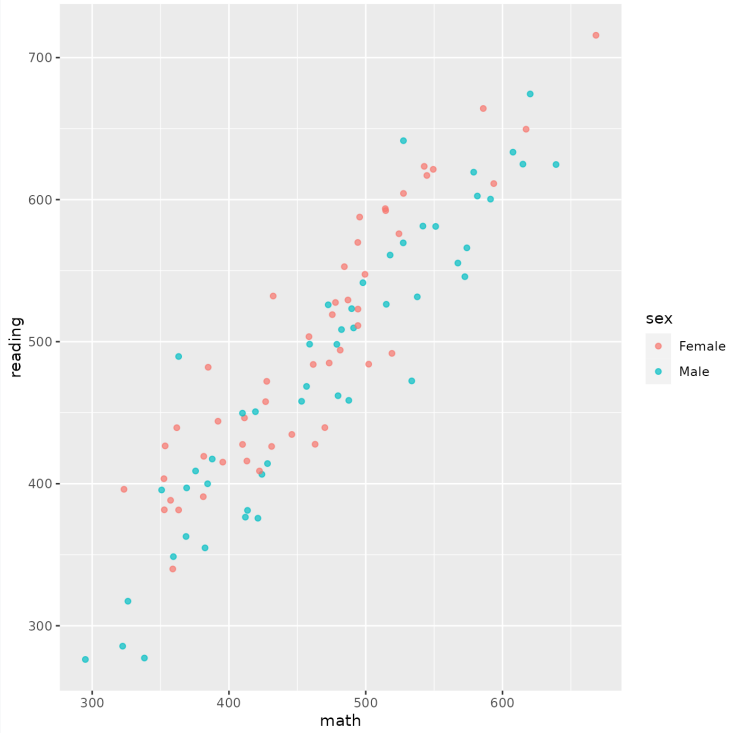


Local vs. Global Encoding

```
ggplot(
    pisa_usa,
    aes(x = math, y = reading)
  ) +
  geom_point(
    aes(color = sex),
    alpha = .7
  )
```



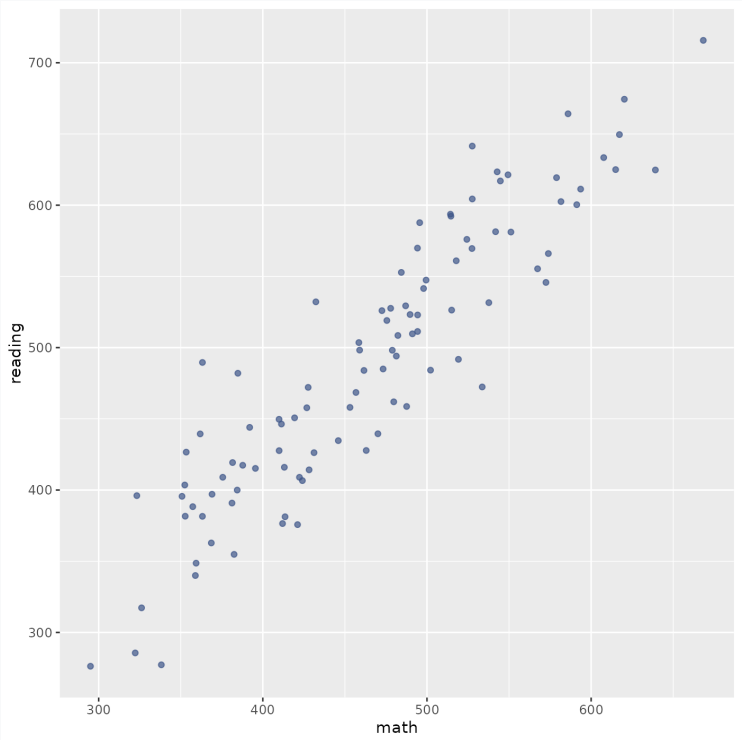

```
ggplot(
    pisa_usa,
    aes(x = math, y = reading,
        color = sex),
  ) +
  geom_point(
    alpha = .7
  )
```



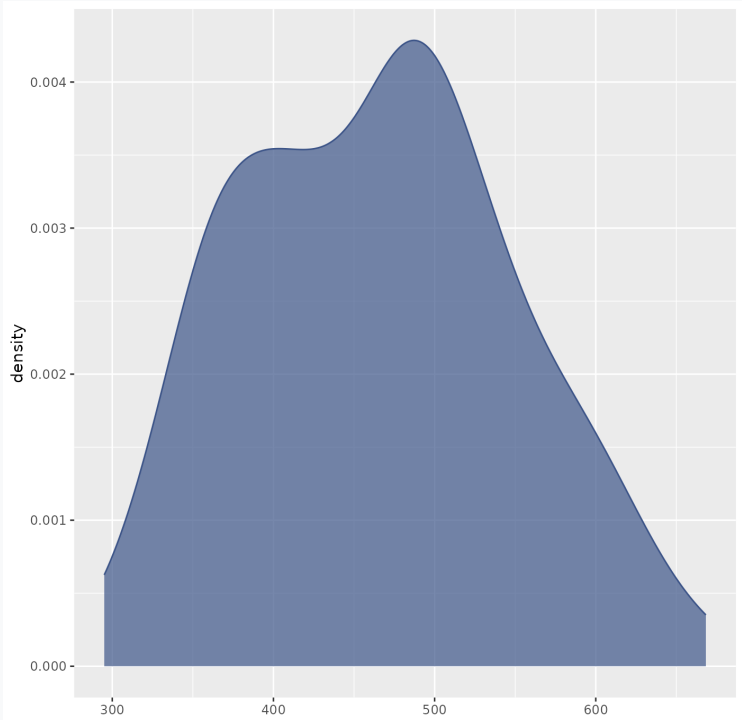
Other geoms

There are many types of geoms and their mapping requirements differ

```
ggplot(pisa_usa) +
  geom_point(
    aes(x = math, y = reading),
    color = "#3C5488",
    alpha = .7)
```



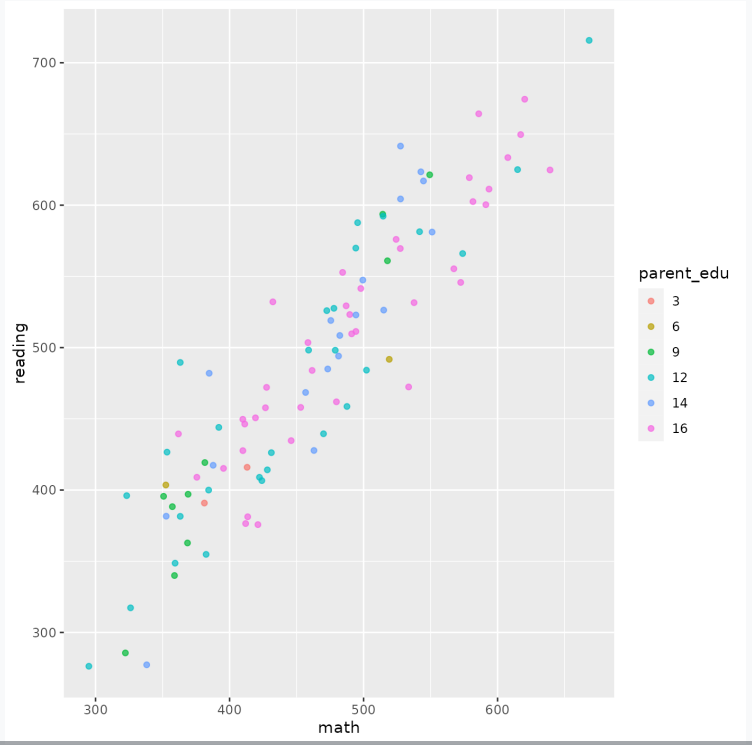

```
ggplot(pisa_usa) +
  geom_density(
    aes(x = math),
    color = "#3C5488",
    fill = "#3C5488",
    alpha = .7)
```



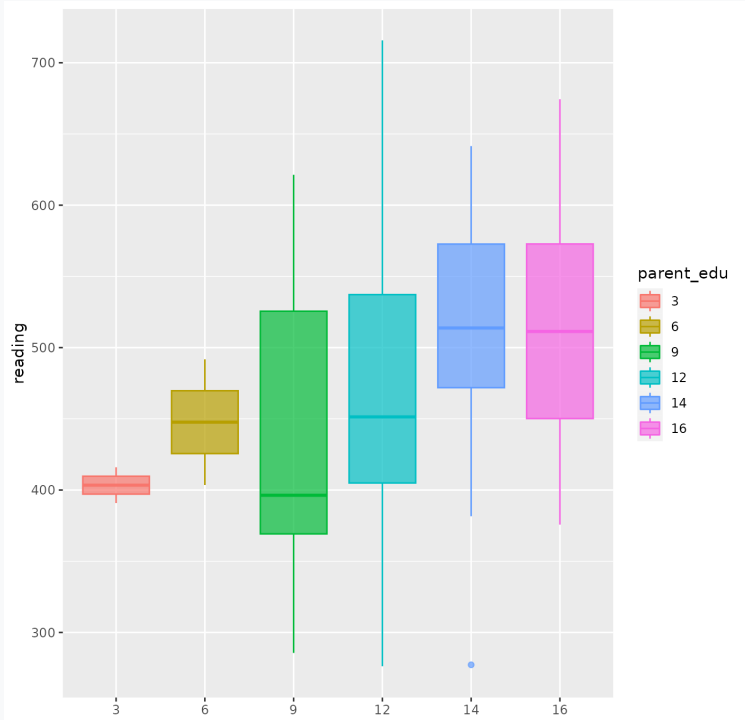
Other geoms

There are many types of geoms and their mapping requirements differ

```
ggplot(pisa_usa) +
  geom_point(
    aes(x = math, y = reading,
        color = parent_edu),
    alpha = .7)
```




```
ggplot(pisa_usa) +
  geom_boxplot(
    aes(x = parent_edu, y = reading,
        color = parent_edu,
        fill = parent_edu),
    alpha = .7)
```



Your Turn

Use code below to create a histogram of the math scores.

- Can you modify the width of the bins?
  (Hint: run `?geom_histogram`)

```
ggplot(pisa_usa, aes(x = math)) +
  ## your code here
```

Use code below to create boxplots of math scores by sex.

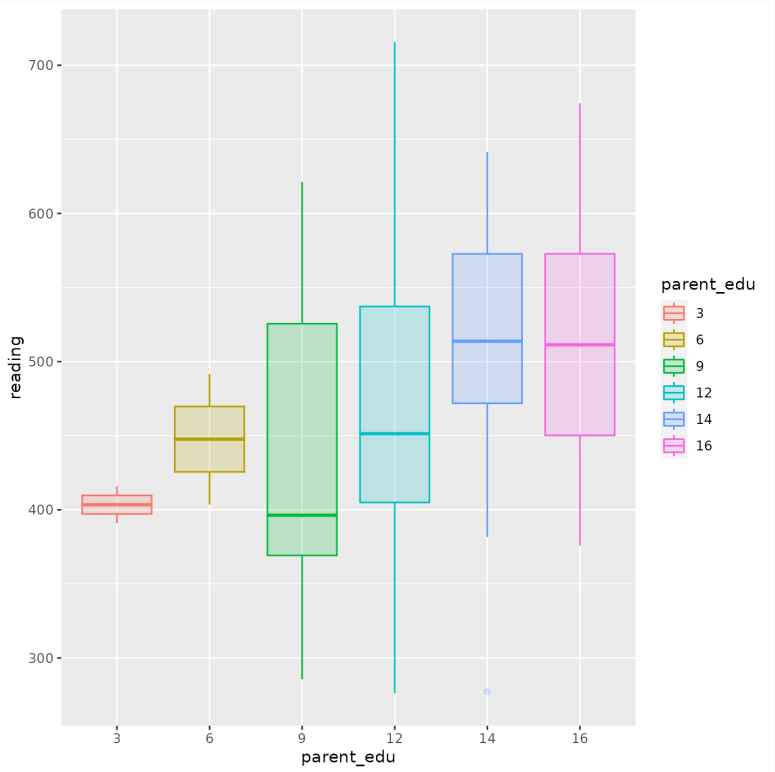
- Can you make a violin plot instead a boxplot?
- Can you add color to the boxplots/violins?
  (Hint: run `?geom_violin`)

```
ggplot(pisa_usa, aes(x = sex, y = math)) +
  ## your code here
```

Adding Layers

Begin with plot with one layer

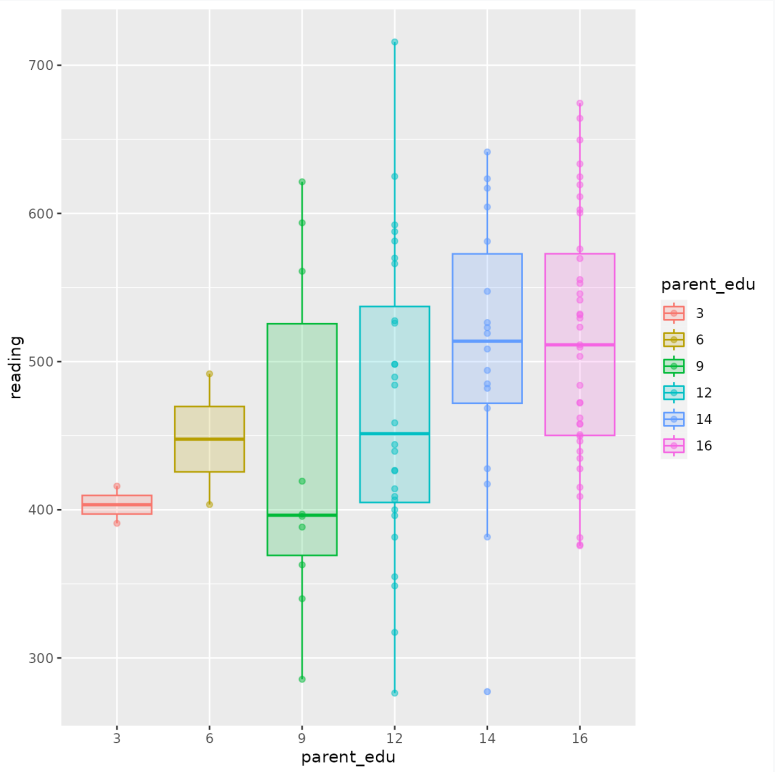
```
ggplot(
    pisa_usa,
    aes(x = parent_edu, y = reading,
        fill = parent_edu,
        color = parent_edu)
  ) +
  geom_boxplot(
    alpha = .2
  )
```



Adding Layers

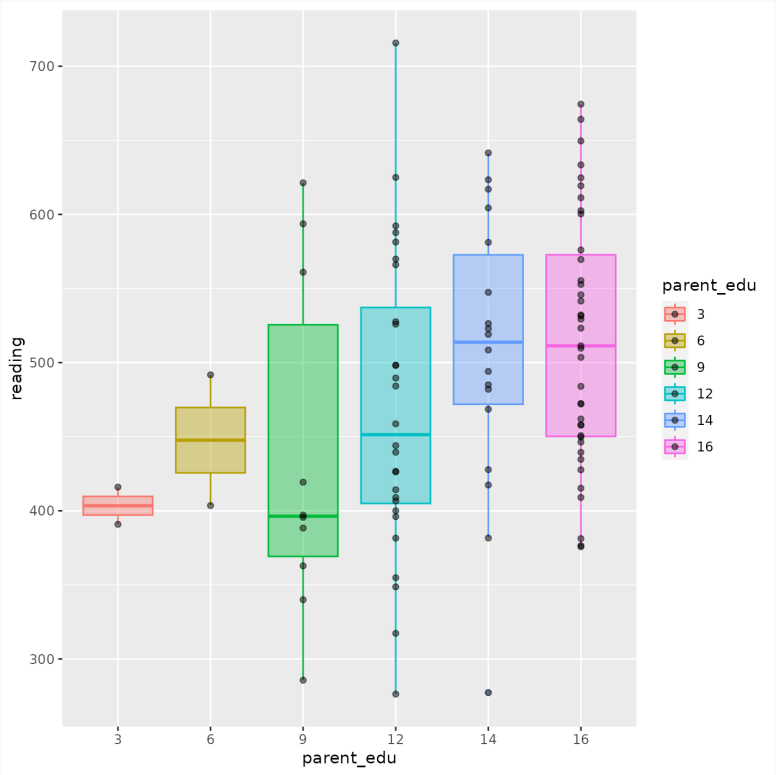
Layers are stacked in the order of code appearance

```
ggplot(
    pisa_usa,
    aes(x = parent_edu, y = reading,
        fill = parent_edu,
        color = parent_edu)
  ) +
  geom_boxplot(
    alpha = .2
  ) +
  geom_point(
    alpha = .5
  )
```



Overwrite Global Aesthetics

```
ggplot(
    pisa_usa,
    aes(x = parent_edu, y = reading,
        fill = parent_edu,
        color = parent_edu)
  ) +
  geom_boxplot(

    alpha = .4
  ) +
  geom_point(
    color = "black",
    alpha = .5
  )
```



What did we need?

Data, Aesthetics, Geometries

Everything else has sensible defaults

Statistical Layers

Statistics

Describes how the data are modified in order to be expressed through the `geom`.

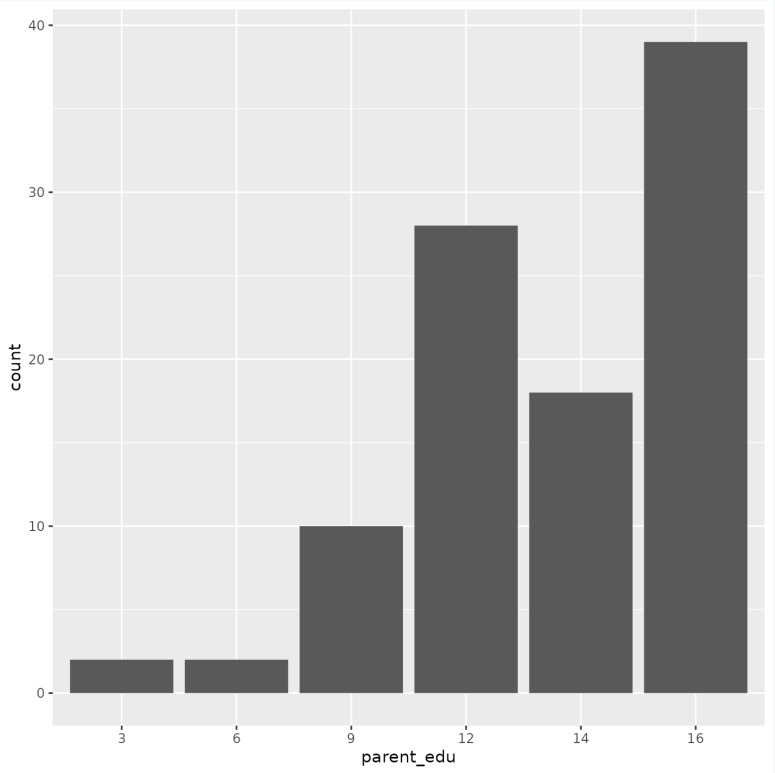
Stats and geoms go together.

- Every `geom` has a default `stat` and vice versa.
    - Count number of observations in each category for a bar chart
    - Calculate summary statistics for a boxplot.
- `stat` can be specified inside of a geom and vice versa.

stat_*() & geom_*()

geom_bar() uses stat_count() by default

```
ggplot(pisa_usa, aes(x = parent_edu)) +
  geom_bar()
```



stat_*() & geom_*()

If you have precomputed data, use identity stat

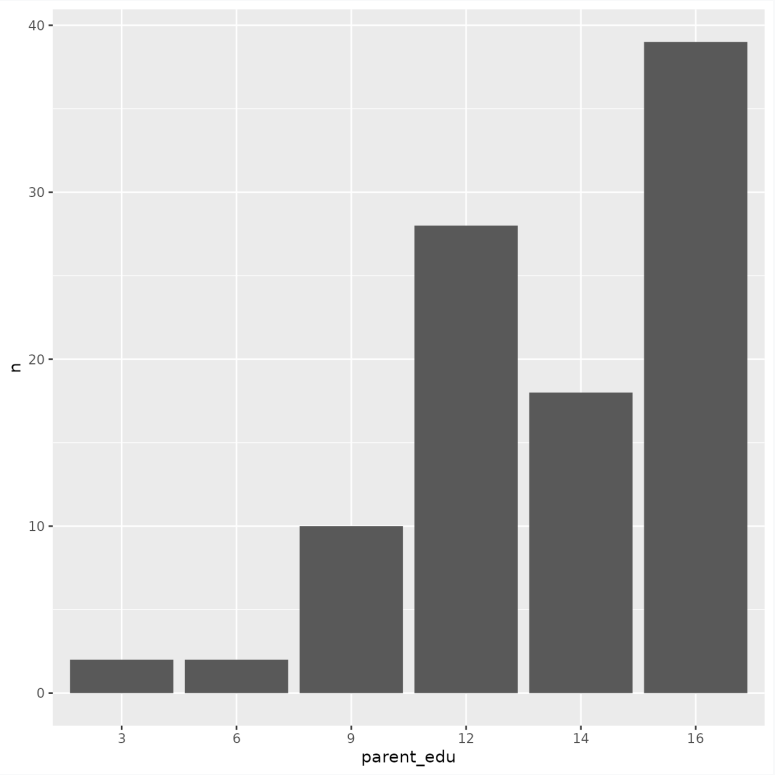
```
pisa_usa_counted <- pisa_usa %>%
  count(parent_edu)

pisa_usa_counted
```

```
## # A tibble: 6 × 2
##   parent_edu     n
##   <fct>      <int>
## 1 3              2
## 2 6              2
## 3 9             10
## 4 12            28
## 5 14            18
## 6 16            39
```

stat_*() & geom_*()

If you have precomputed data, use identity stat

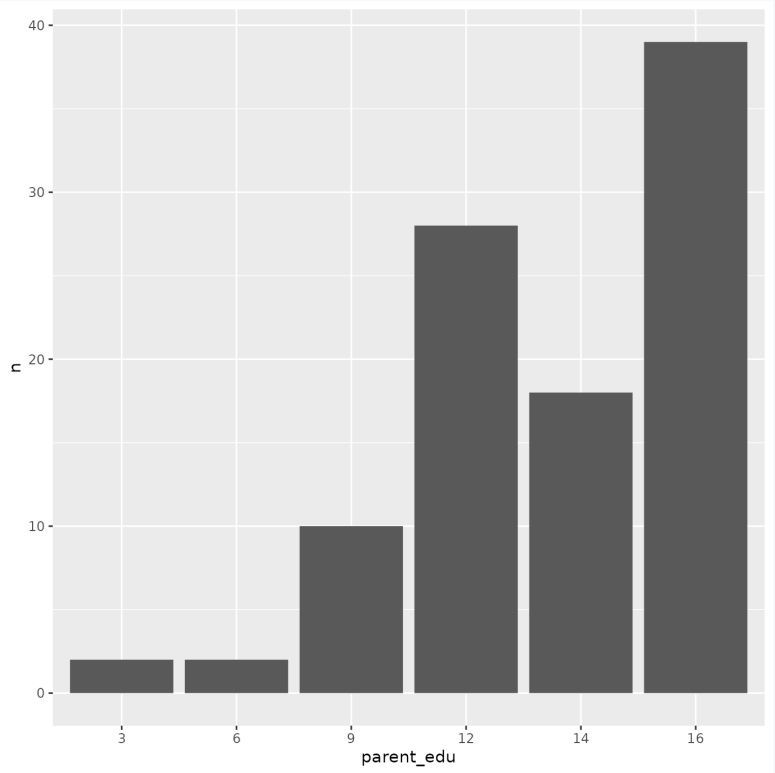
```
pisa_usa_counted <- pisa_usa %>%
  count(parent_edu)

ggplot(pisa_usa_counted,
       aes(x = parent_edu)) +
  geom_bar(aes(y = n),
           stat = 'identity')
```



stat_*() & geom_*()

... or use the geom_col() shortcut

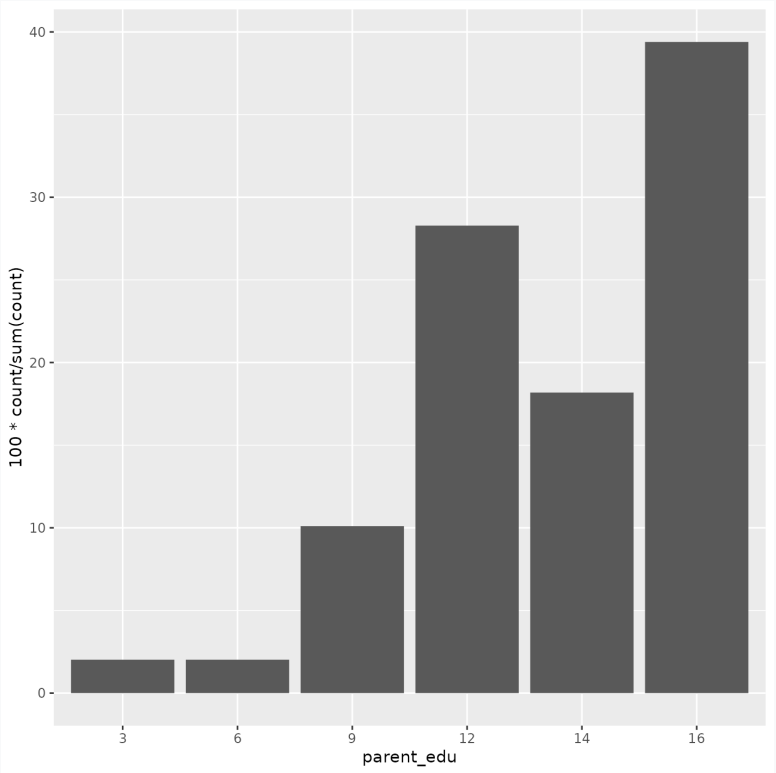
```
pisa_usa_counted <- pisa_usa %>%
  count(parent_edu)

ggplot(pisa_usa_counted,
       aes(x = parent_edu)) +
  geom_col(aes(y = n))
```



stat_*() & geom_*()

Use `after_stat()` to modify mapping from stats

```
ggplot(pisa_usa) +
  geom_bar(
    aes(
      x = parent_edu,
      y = after_stat(
        100 * count / sum(count)
        )
    )
  )
```



Facets

Facets

Split data into multiple panels by categories

- shows the same visualization for different subsets of the data
  - aka conditioning
- a way to avoid overplotting

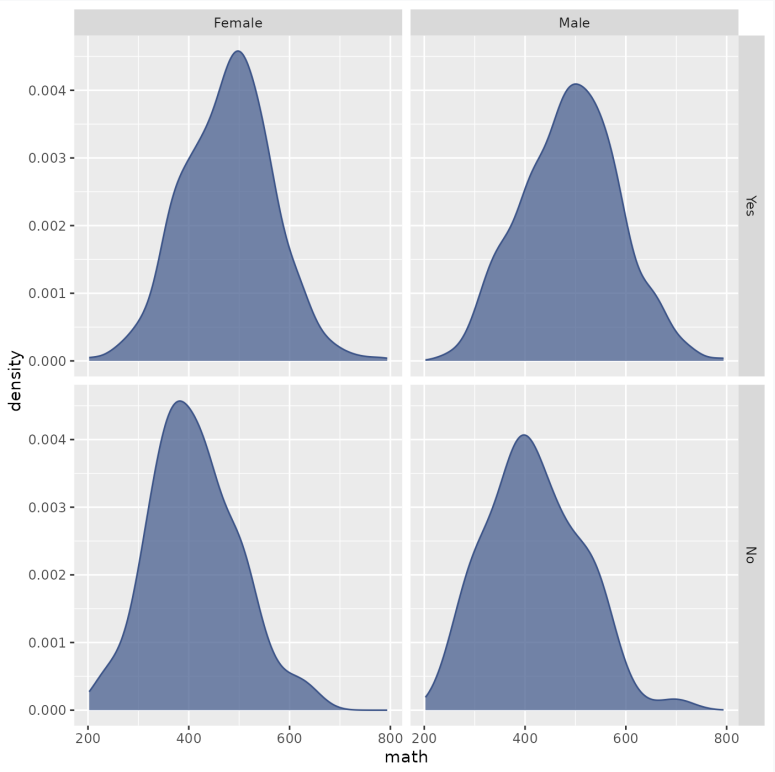
Two faceting functions:

- `facet_grid()`
  - create a grid of graphs, by rows and columns
- `facet_wrap()`
  - create small multiples by "wrapping" a series of plots

facet_grid()

- use `vars()` to call on the variables

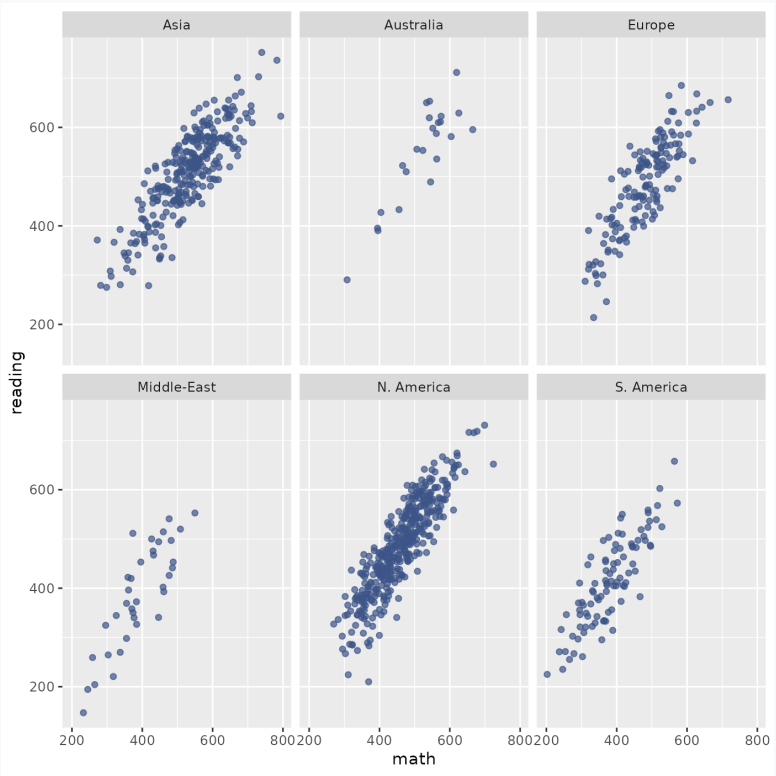
```
ggplot(pisa_small,
       aes(x = math)) +
  geom_density(
    color = "#3C5488",
    fill = "#3C5488",
    alpha = .7
  ) +
  facet_grid(
    cols = vars(sex),
    rows = vars(OECD)
  )
```



facet_wrap()

- use `vars()` to call on the variables
- `nrow` and `ncol` arguments for dictating shape of grid

```
ggplot(pisa_small,
       aes(math, reading)) +
  geom_point(
    color = "#3C5488",
    alpha = .7
  ) +
  facet_wrap(vars(region))
```



Your Turn

Use `nrow` or `ncol` to alter the shape of the grid in the `facet_wrap()` example to have two columns. Then again with one row.

Use the labeller parameter to modify the panel labels in the `facet_grid()` example such that the row labels read 'OCED: Yes' and 'OCED: No'. (Hint: run `?labeller`)

ggplots as objects

Save & inspect a ggplot object

```
pisa_plot <- ggplot(pisa_usa, aes(x = math, y = reading, color = sex)) +
  geom_point(alpha = .7)

class(pisa_plot)
```

```
## [1] "gg"     "ggplot"
```

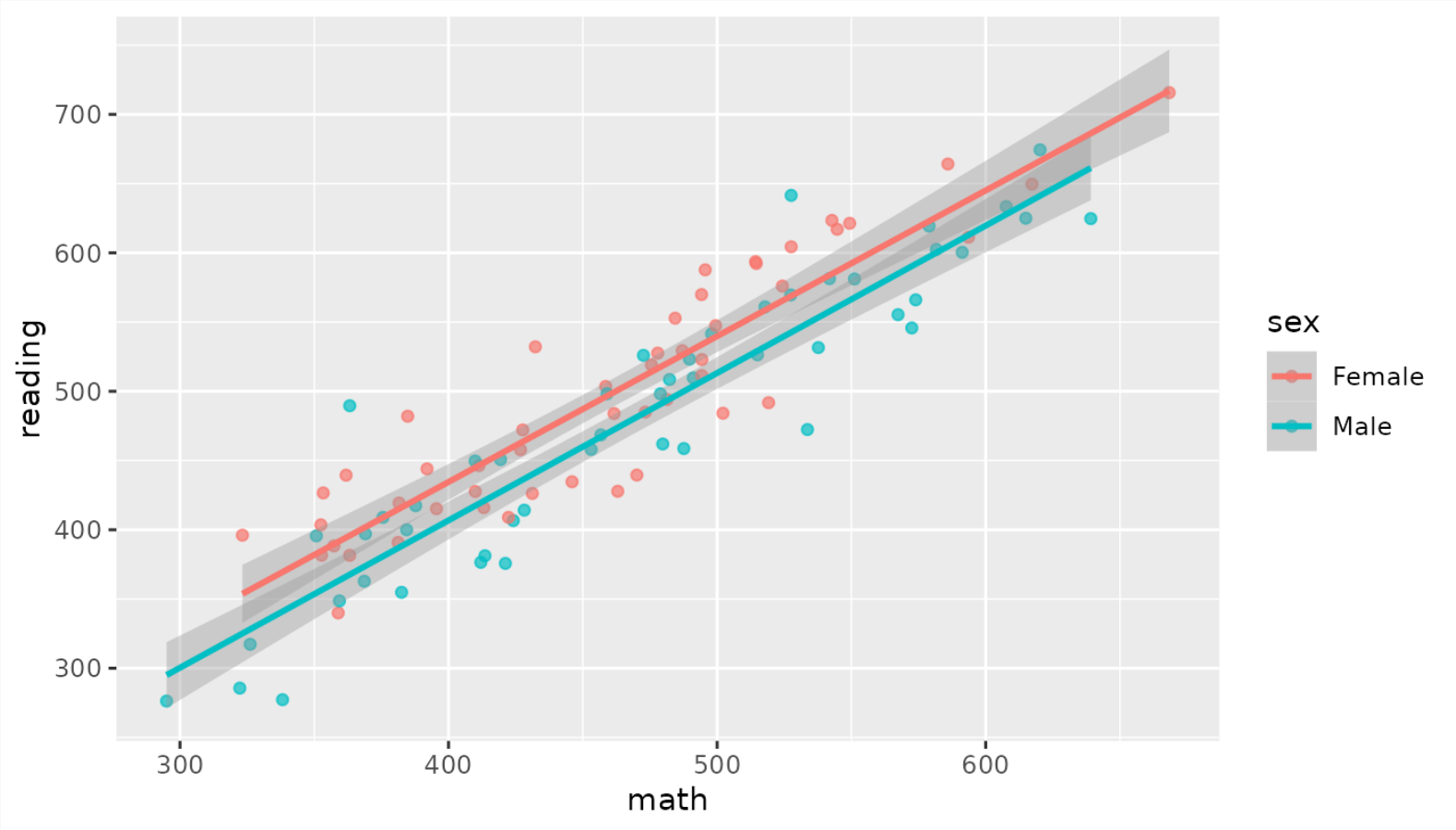
Inspect a ggplot object

```
str(pisa_plot)
```

```
## List of 9
##  $ data       : tibble [99 × 36] (S3: tbl_df/tbl/data.frame)
##   ..$ country                : chr [1:99] "United States" "United States" "U
##   ..$ OECD                   : chr [1:99] "Yes" "Yes" "Yes" "Yes" ...
##   ..$ id                     : int [1:99] 84010767 84010299 84002440 8400490
##   ..$ weight                 : num [1:99] 759 688 610 462 823 ...
##   ..$ sex                    : chr [1:99] "Female" "Male" "Female" "Female"
##   ..$ grade                  : num [1:99] 10 10 10 10 10 10 11 10 10 10 ...
##   ..$ computer               : chr [1:99] "Yes" "No" "Yes" "Yes" ...
##   ..$ software               : chr [1:99] "Yes" "Yes" "Yes" "Yes" ...
##   ..$ internet               : Factor w/ 2 levels "Yes","No": 1 1 2 1 1 1 1
##   ..$ addit_time_math        : int [1:99] 1 9 6 2 14 4 7 10 19 2 ...
##   ..$ addit_time_science     : int [1:99] 2 11 6 1 14 1 7 10 19 5 ...
##   ..$ parent_support         : chr [1:99] "Strongly agree" "Strongly agree"
##   ..$ parent_status          : chr [1:99] "Strongly agree" "Agree" "Strongly
##   ..$ want_best_grades       : Factor w/ 4 levels "Strongly agree",..: 1 1 1
##   ..$ want_best_student      : chr [1:99] "Strongly agree" "Strongly agree"
##   ..$ test_anxiety           : num [1:99] 0.857 -0.475 -0.539 1.724 -0.308 .
##   ..$ enjoy_cooperation      : num [1:99] 0.946 1.042 0.576 2.288 -0.288 ...
##   ..$ sense_of_belonging     : num [1:99] 0.445 -1.196 -0.988 -0.862 -0.338
##   ..$ parent_support_emotional: num [1:99] 1.099 -0.75 0.566 1.099 1.099 ...
##   ..$ HOMESCH                : num [1:99] NA NA NA NA NA NA NA NA NA NA ...
##   ..$ ENTUSE                 : num [1:99] NA NA NA NA NA NA NA NA NA NA ...
##   ..$ ICTHOME                : int [1:99] NA NA NA NA NA NA NA NA NA NA ...
##   ..$ ICTSCH                 : int [1:99] NA NA NA NA NA NA NA NA NA NA ...
##   ..$ wealth                 : num [1:99] 2.1451 -0.7076 -0.7871 0.0963 0.09
```

Add to a ggplot object

```
pisa_plot +
  geom_smooth(method = "lm")
```



Recap

- `{ggplot2}` is a powerful library for reproducible graphic design
- the components follow a consistent syntax
- each ggplot needs at least data, some aesthetics, and a layer
- we set constant propeties outside `aes()`
- ... and map data-related properties inside `aes()`
- local settings and mappings override global properties
- grouping allows applying layers for subsets
- we can store a ggplot object and add to it afterwards

Resources

- Documentation: http://ggplot2.tidyverse.org/reference/
- RStudio cheat sheet for ggplot2
- Sam Tyner's ggplot2 workshop
- Thomas Lin Pedersen's ggplot2 webinar: part 1 and part 2
- Cedric Scherer's "A ggplot2 tutorial for beautiful plotting in R"