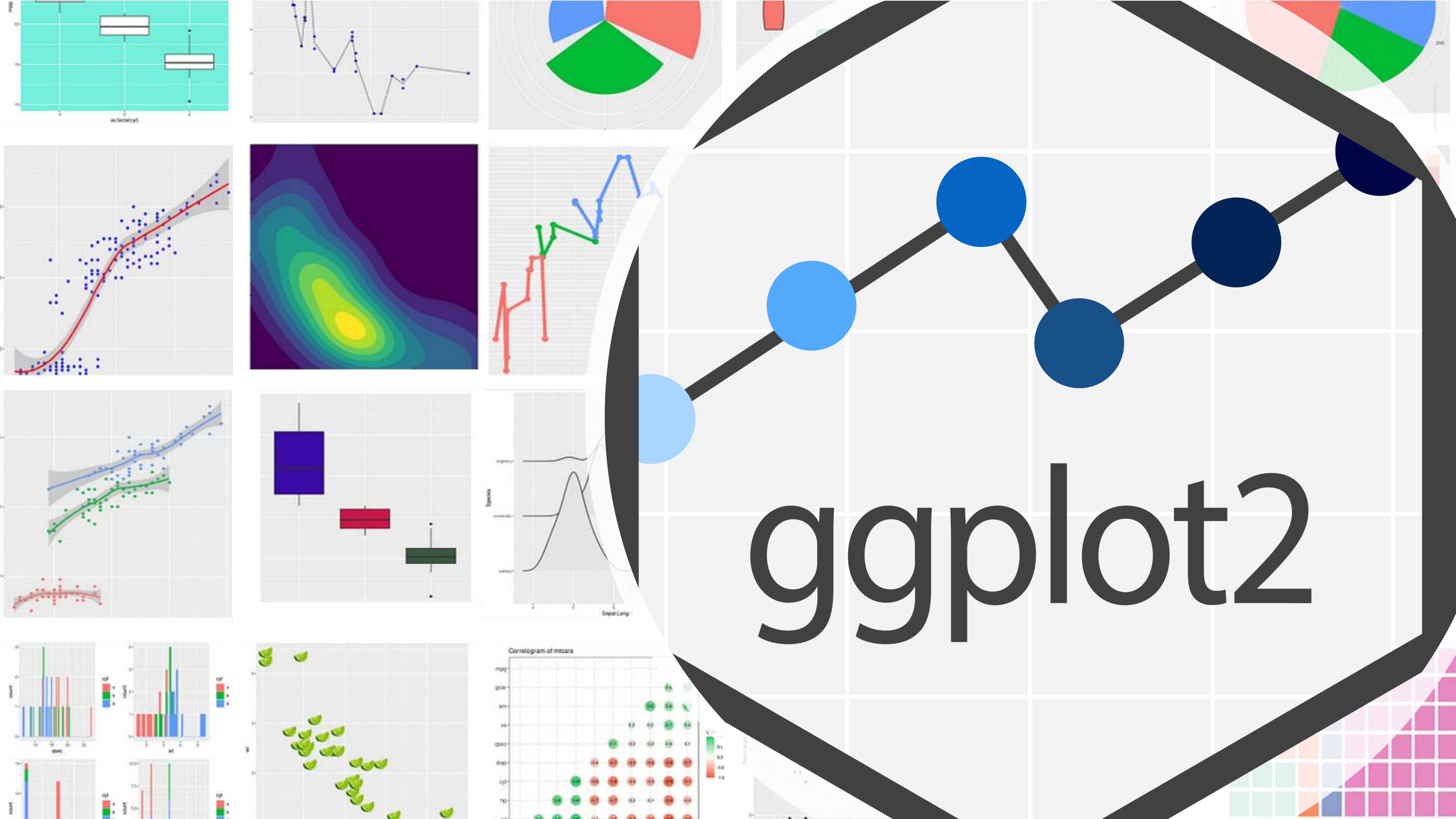

ggplot2
Correlogram of mtcars
Species
Sepal.Length

First Things First: Course Evaluation

https://uzk-evaluation.uni-koeln.de/evasys/online.php?p=W4L6C

Recap: Data Visualization

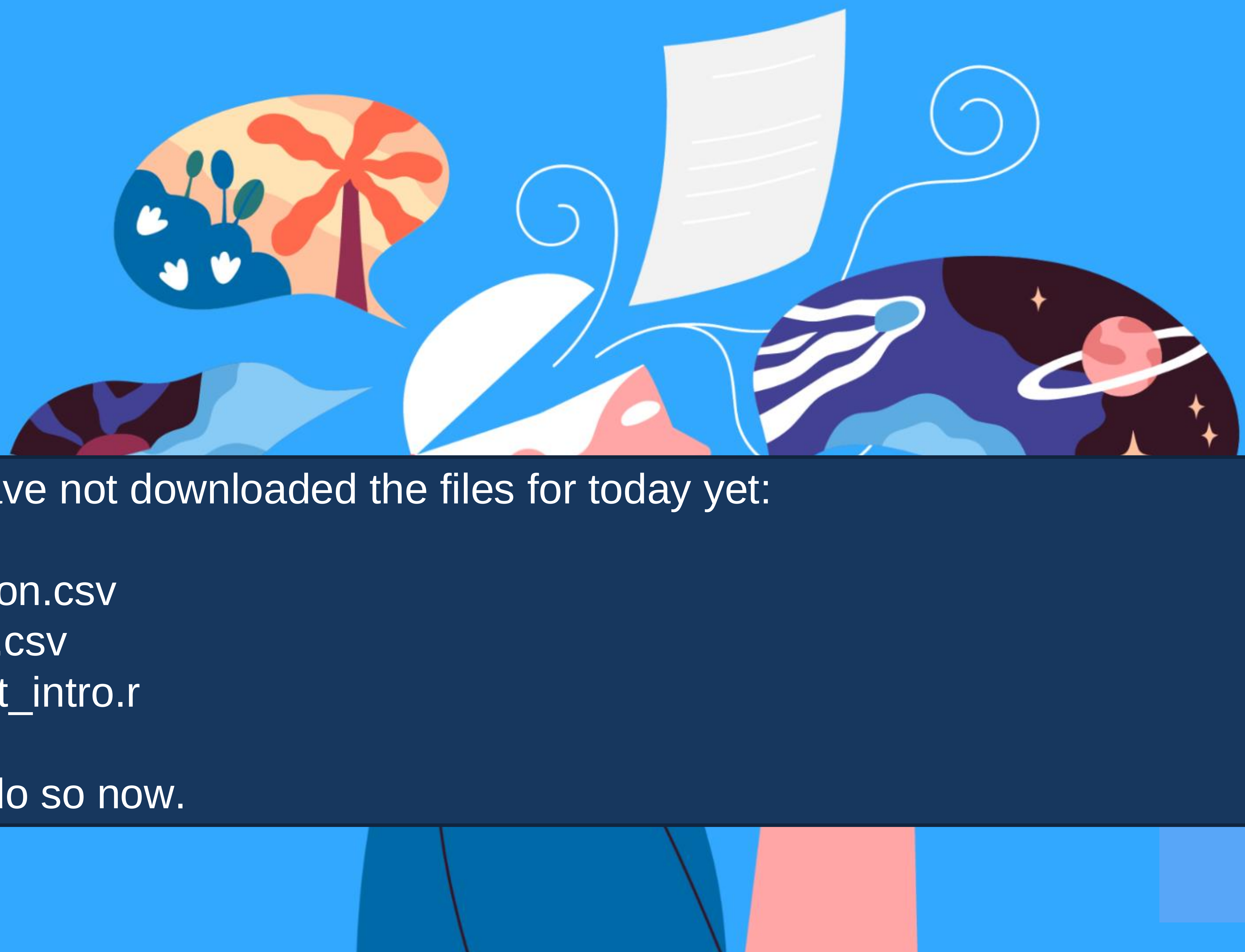
1) Why use data visualization?

2) What makes a good data visualization?

3) Think about your audience – Who?

4) Tables, charts and colors

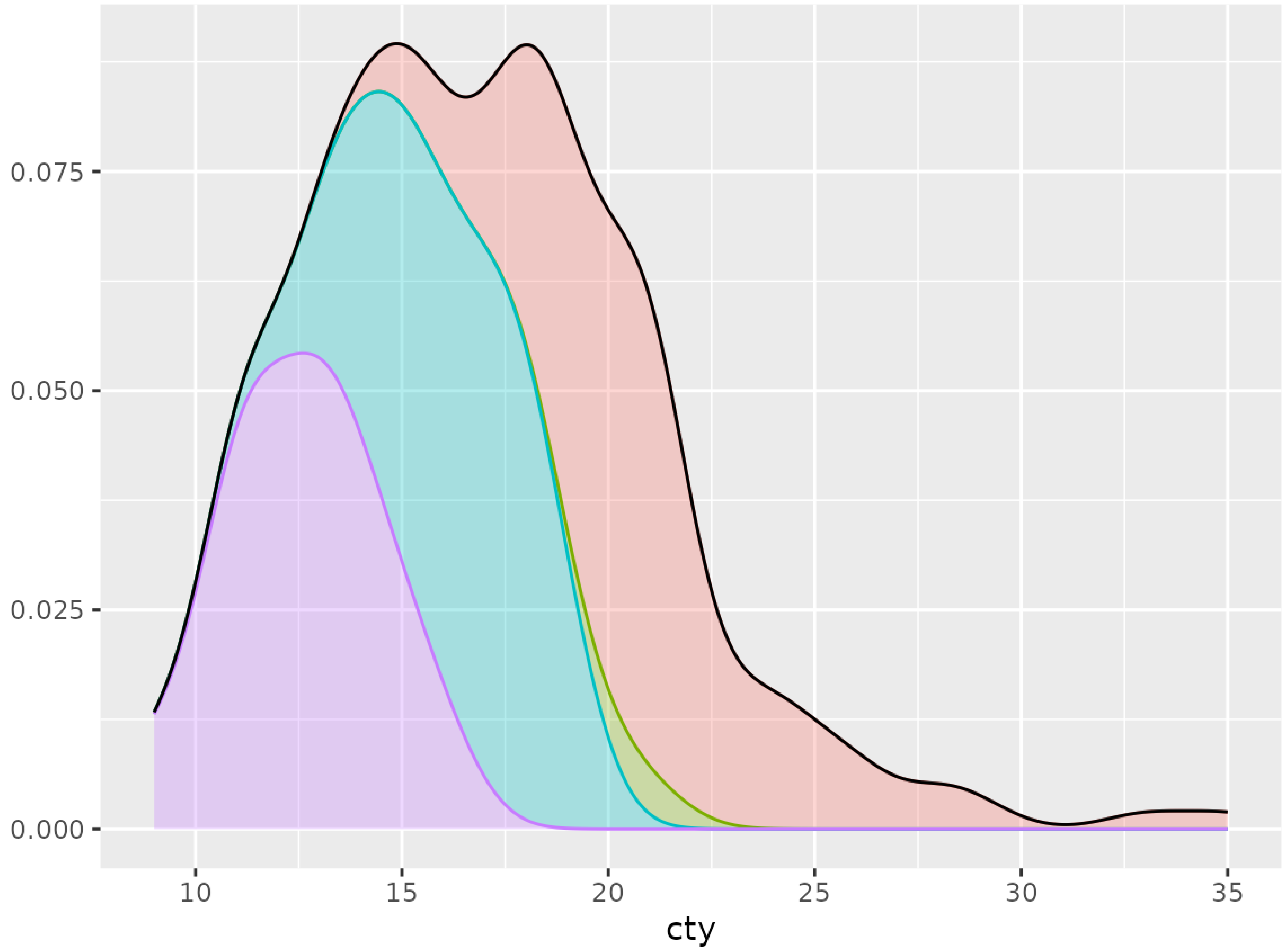
5) Common “zero-code” tools for data visualization



If you have not downloaded the files for today yet:

- taxation.csv
- sales.csv
- ggplot_intro.r

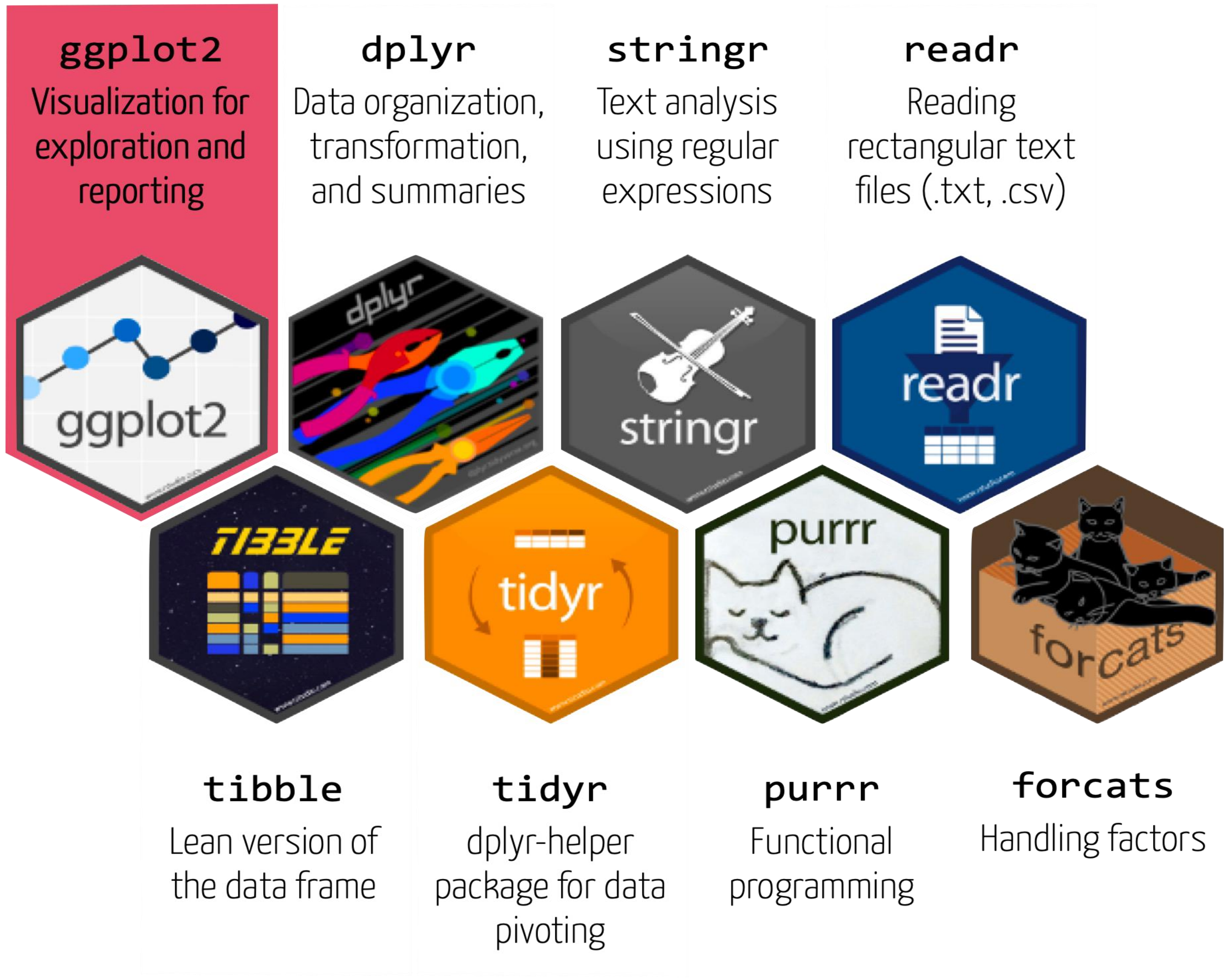
Please do so now.



Agenda

- ggplot
- ggplot practice

Intro to ggplot



ggplot2
Visualization for exploration and reporting

dplyr
Data organization, transformation, and summaries

stringr
Text analysis using regular expressions

readr
Reading rectangular text files (.txt, .csv)

tibble
Lean version of the data frame

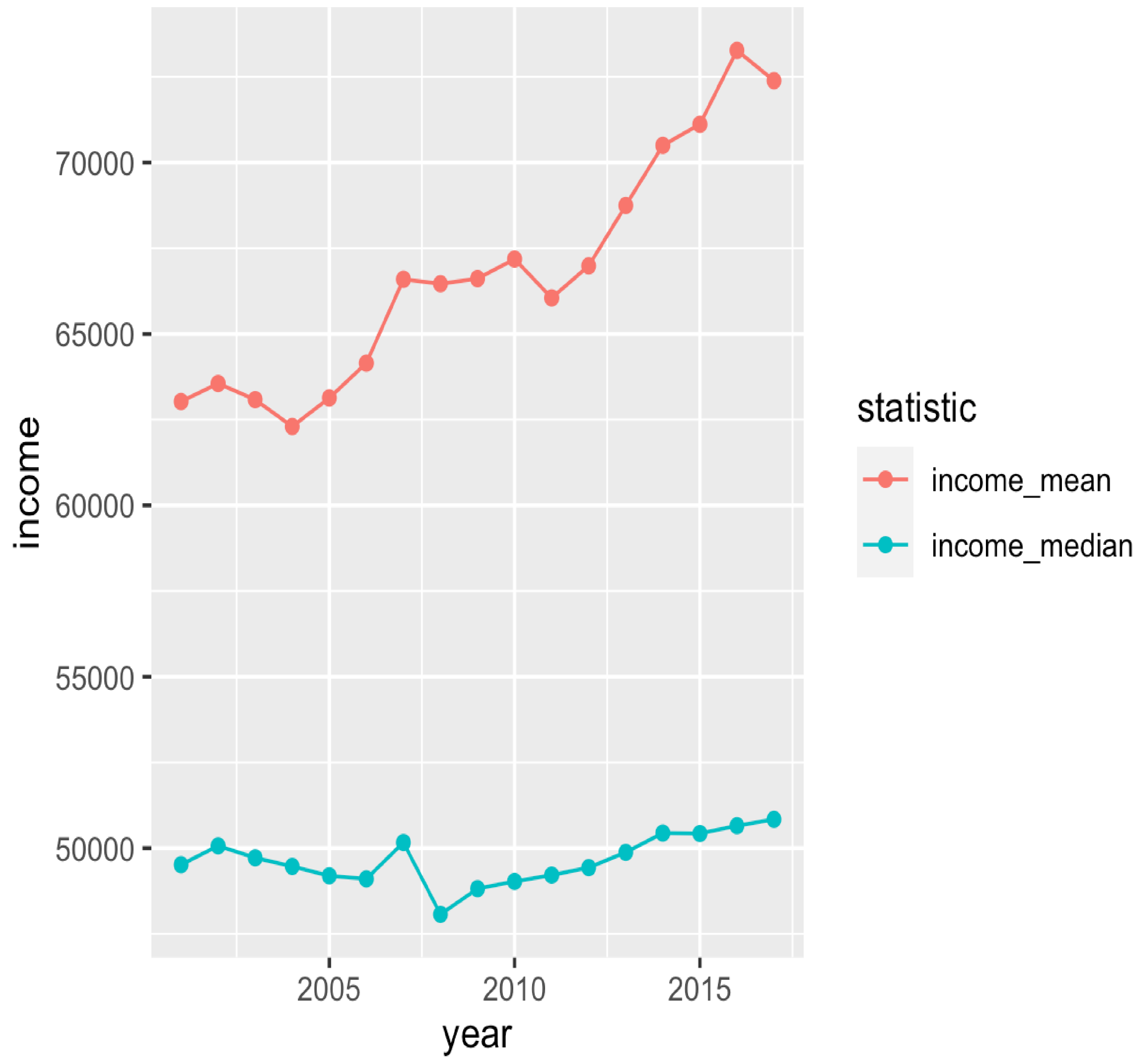
tidyr
dplyr-helper package for data pivoting

purrr
Functional programming

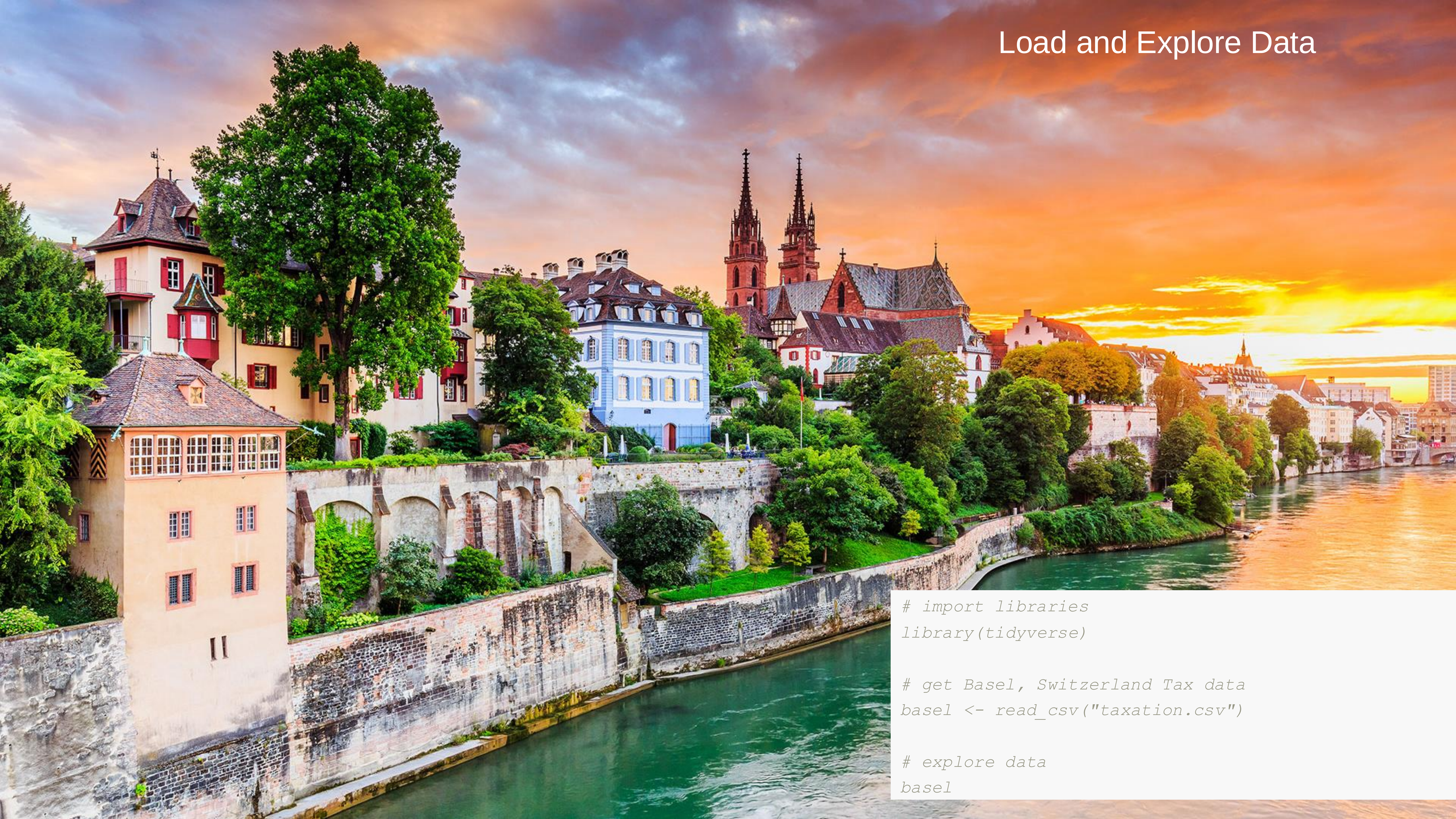
forcats
Handling factors

ggplot2 uses layers to create plots

1) `data`: the data set
2) `mapping`: the plot's structure
   - What do the axes represent?
   - Map variables to colors, shapes, sizes
3) `geoms`: geometric shapes illustrating data
4) `facets`: stratify the plot according to variable
5) `labs`: plot annotation
6) `themes`: aesthetic details
7) `scales`: scaling of dimensions

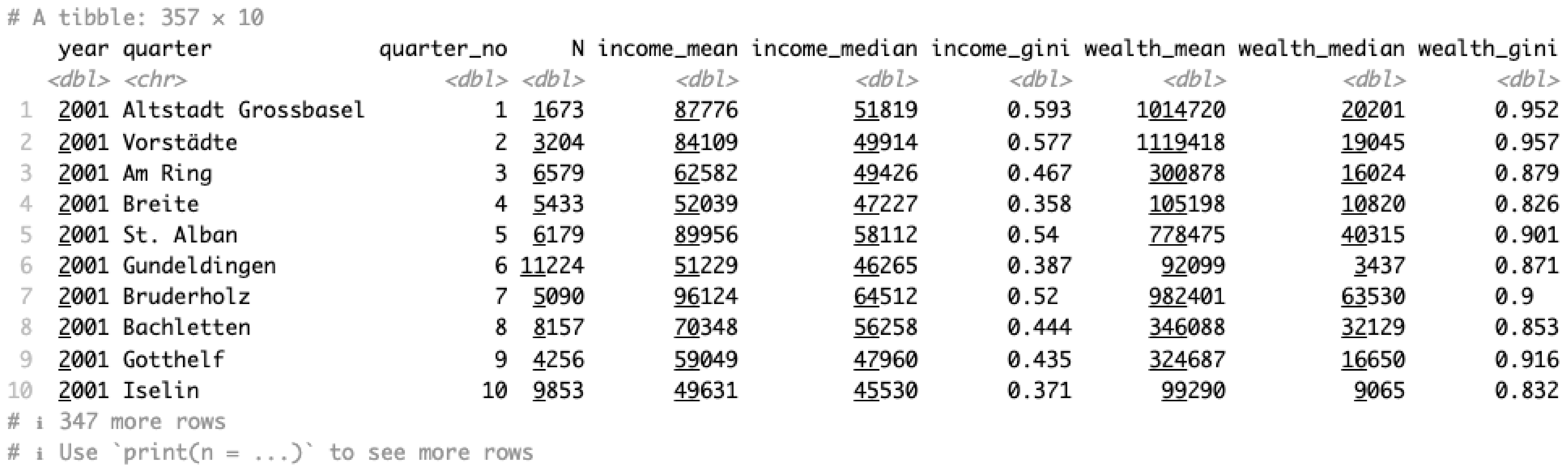


Load and Explore Data



```
# import libraries
library(tidyverse)

# get Basel, Switzerland Tax data
basel <- read_csv("taxation.csv")

# explore data
basel
```

Data:taxation.csv



```
# A tibble: 357 × 10
    year quarter              quarter_no     N income_mean income_median income_gini wealth_mean wealth_median wealth_gini
   <dbl> <chr>                     <dbl> <dbl>       <dbl>         <dbl>       <dbl>       <dbl>         <dbl>       <dbl>
 1  2001 Altstadt Grossbasel           1  1673       87776         51819       0.593     1014720         20201       0.952
 2  2001 Vorstädte                     2  3204       84109         49914       0.577     1119418         19045       0.957
 3  2001 Am Ring                       3  6579       62582         49426       0.467      300878         16024       0.879
 4  2001 Breite                        4  5433       52039         47227       0.358      105198         10820       0.826
 5  2001 St. Alban                     5  6179       89956         58112       0.54       778475         40315       0.901
 6  2001 Gundeldingen                  6 11224       51229         46265       0.387       92099          3437       0.871
 7  2001 Bruderholz                    7  5090       96124         64512       0.52       982401         63530       0.9
 8  2001 Bachletten                    8  8157       70348         56258       0.444      346088         32129       0.853
 9  2001 Gotthelf                      9  4256       59049         47960       0.435      324687         16650       0.916
10  2001 Iselin                       10  9853       49631         45530       0.371       99290          9065       0.832
# ℹ 347 more rows
# ℹ Use `print(n = ...)` to see more rows
```

Prepare dataset

Calculate `mean` + `median` income for each year

```
basel_avg <- basel %>%
  group_by(year) %>%
  summarize(income_mean = mean(income_mean),
            income_median = mean(income_median))
```

```
basel_avg
```

```
# A tibble: 17 × 3
 year         income_mean     income_median
<dbl>               <dbl>             <dbl>
 1     2001        63027.            49516.
 2     2002        63555.            50066.
 3     2003        63083.            49717.
 4     2004        62298.            49467.
 5     2005        63133.            49192.
 6     2006        64148.            49102.
 7     2007        66594.            50164.
 8     2008        66463.            48068.
 9     2009        66614.            48818.
10     2010        67185.            49028.
11     2011        66050.            49213.
12     2012        66987.            49433.
13     2013        68748.            49878.
14     2014        70499.            50440.
15     2015        71115.            50426.
16     2016        73272.            50653.
17     2017        72388.            50840.
```

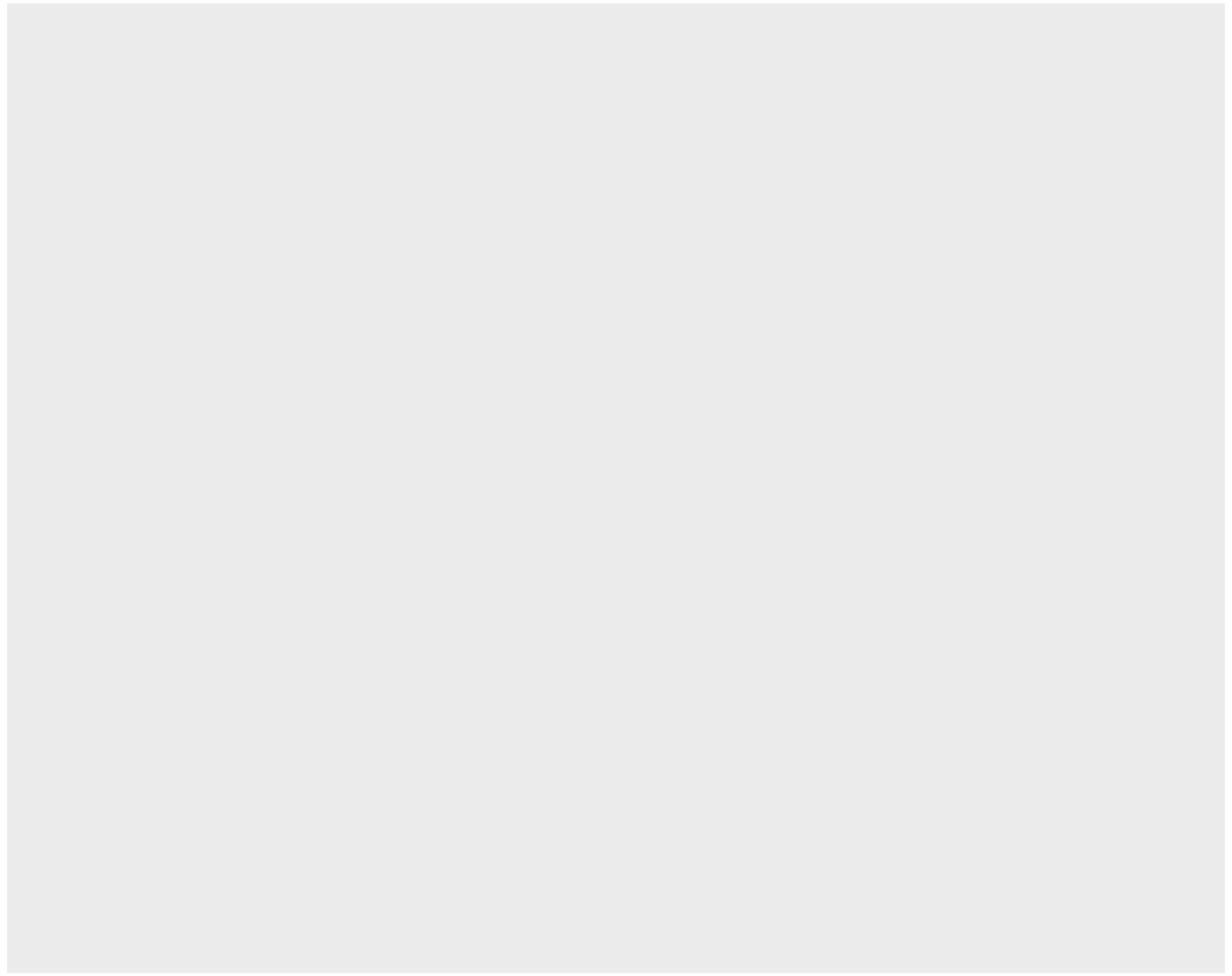
Data

All plots start with `ggplot()`

Two arguments:

- `data` | The data set (`tibble`)
- `mapping` | The plot structure. Defined using `aes()`

```
ggplot(data = basel_avg)
```



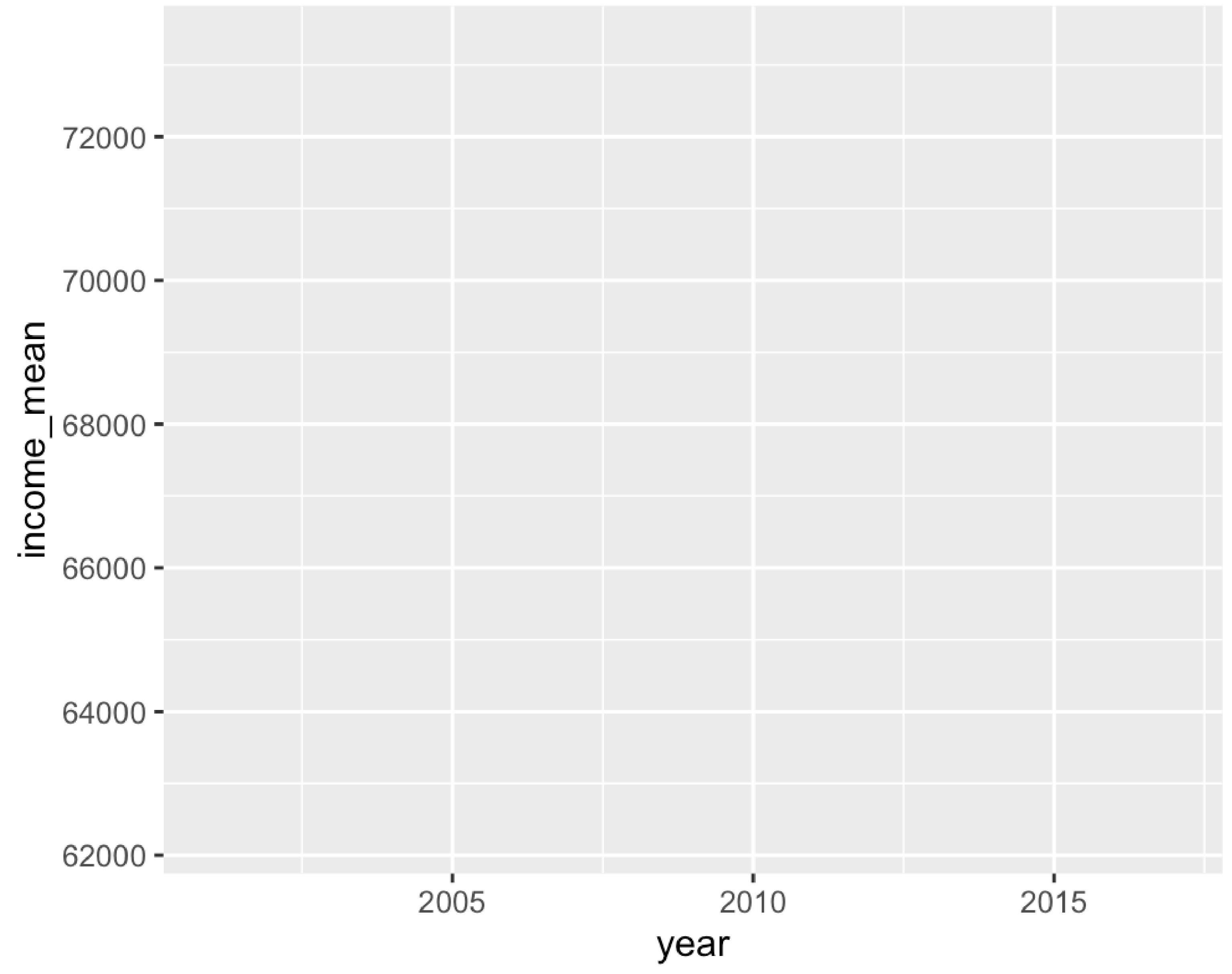
aes()

`aes()` helps define the structure of the **mapping** argument.

Key arguments:

- `x,y` | defines axes
- `color,fill` | defines colors
- `alpha` | defines opacity
- `size` | defines sizes
- `shape` |defines shapes (e.g., circles or squares)

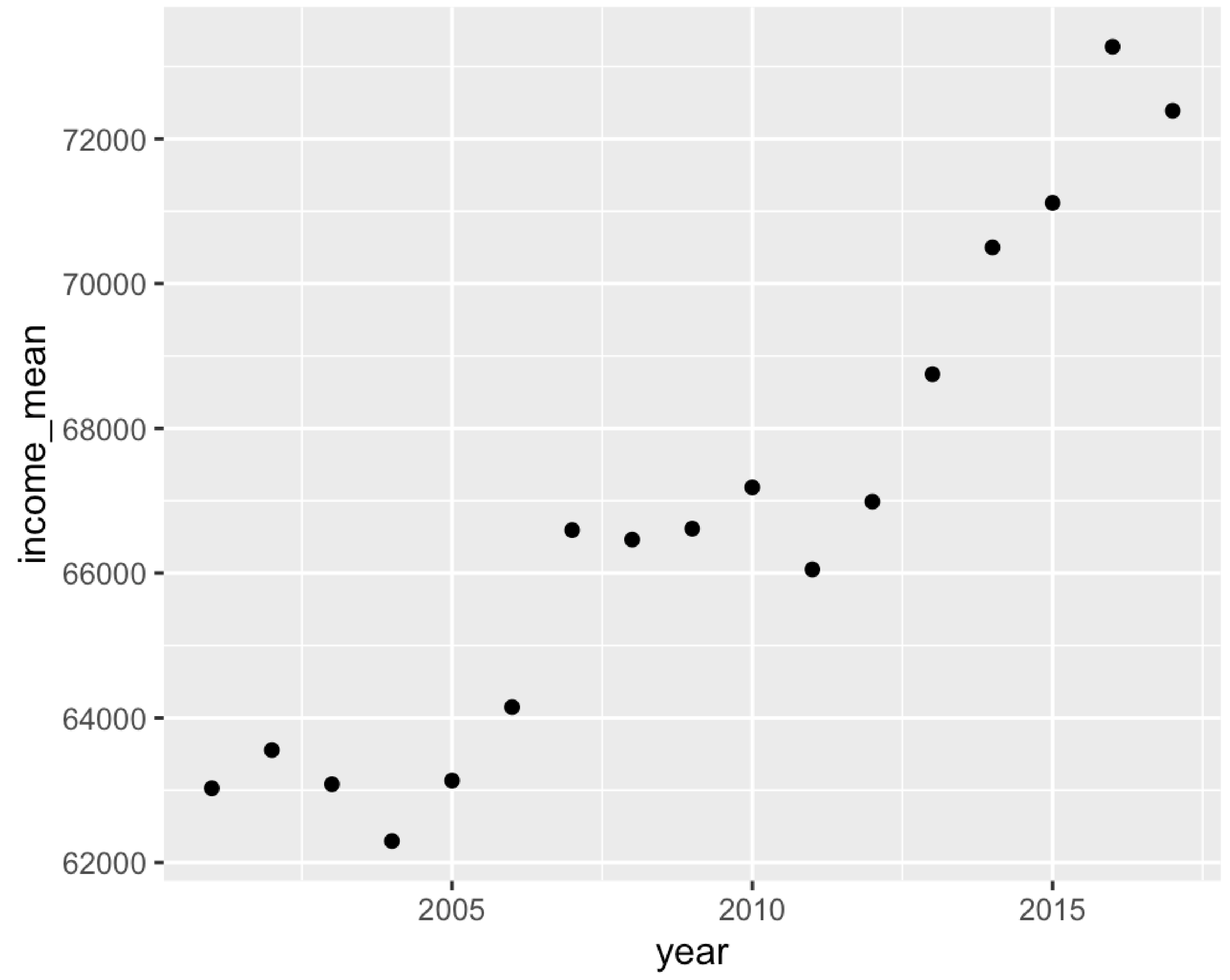
```
ggplot(data = basel_avg,
       mapping = aes(x = year,
                     y = income_mean))
```



geom_*()

The + operator "adds" **additional elements** to the plot.

```
ggplot(data = basel_avg,
       mapping = aes(x = year,
                     y = income_mean)) +

  # Show as points
  geom_point()
```

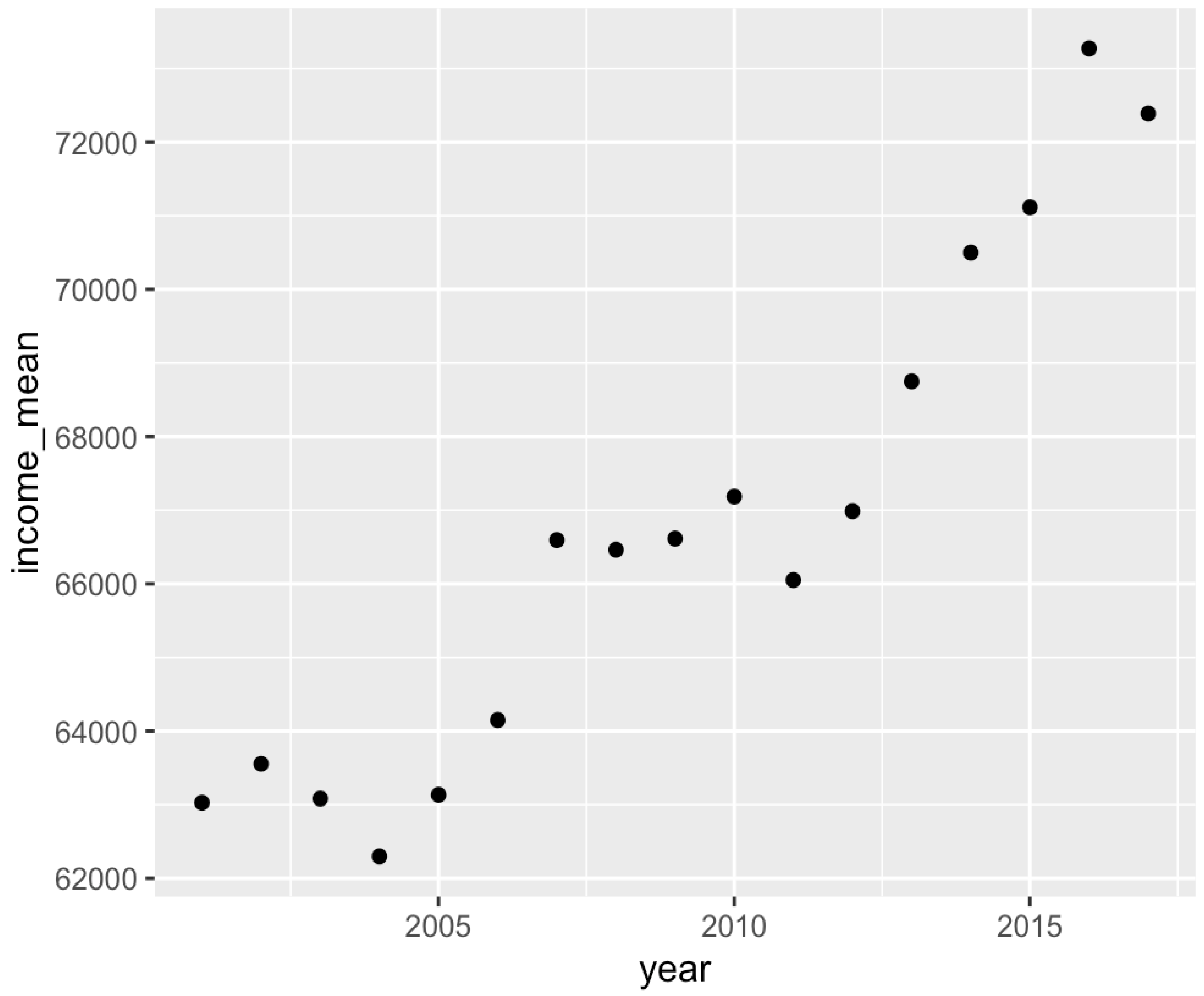


`geom_*()`

`geom_*()` functions define which geometric objects are used to illustrate the data.

A few examples `geoms`:

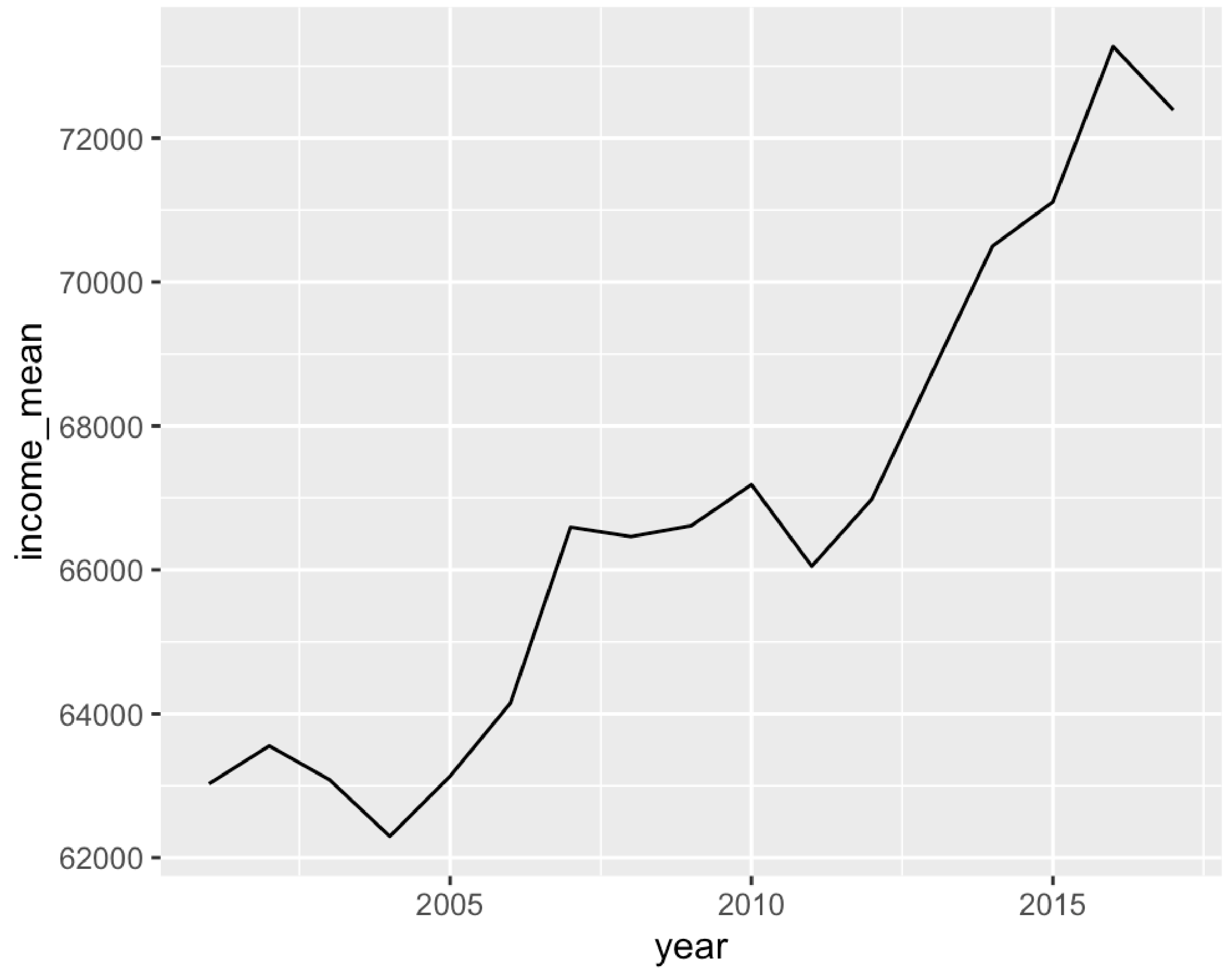
- `geom_point()` | for points
- `geom_line()` | for lines
- `geom_smooth()` | for smooth curves
- `geom_bar()` | for bars
- `geom_boxplot()` | for box-plots
- `geom_violin()` | for violin-plots



geom_*()

`geom_*()` functions define which geometric objects are used to illustrate the data.

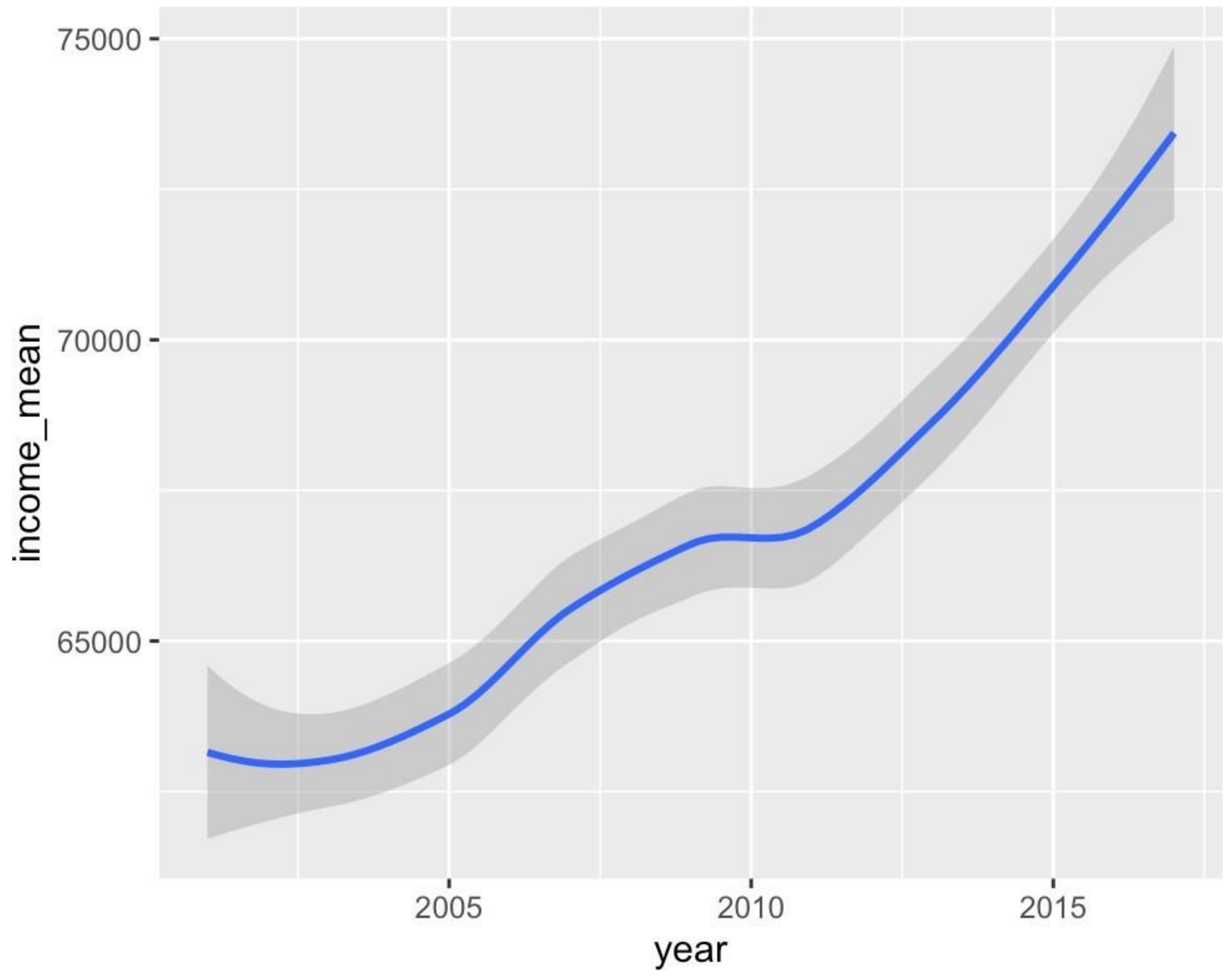
```
ggplot(data = basel_avg,
       mapping = aes(x = year,
                     y = income_mean)) +

  # Show as lines
  geom_line()
```



geom_*()

`geom_*()` functions define which geometric objects are used to illustrate the data.

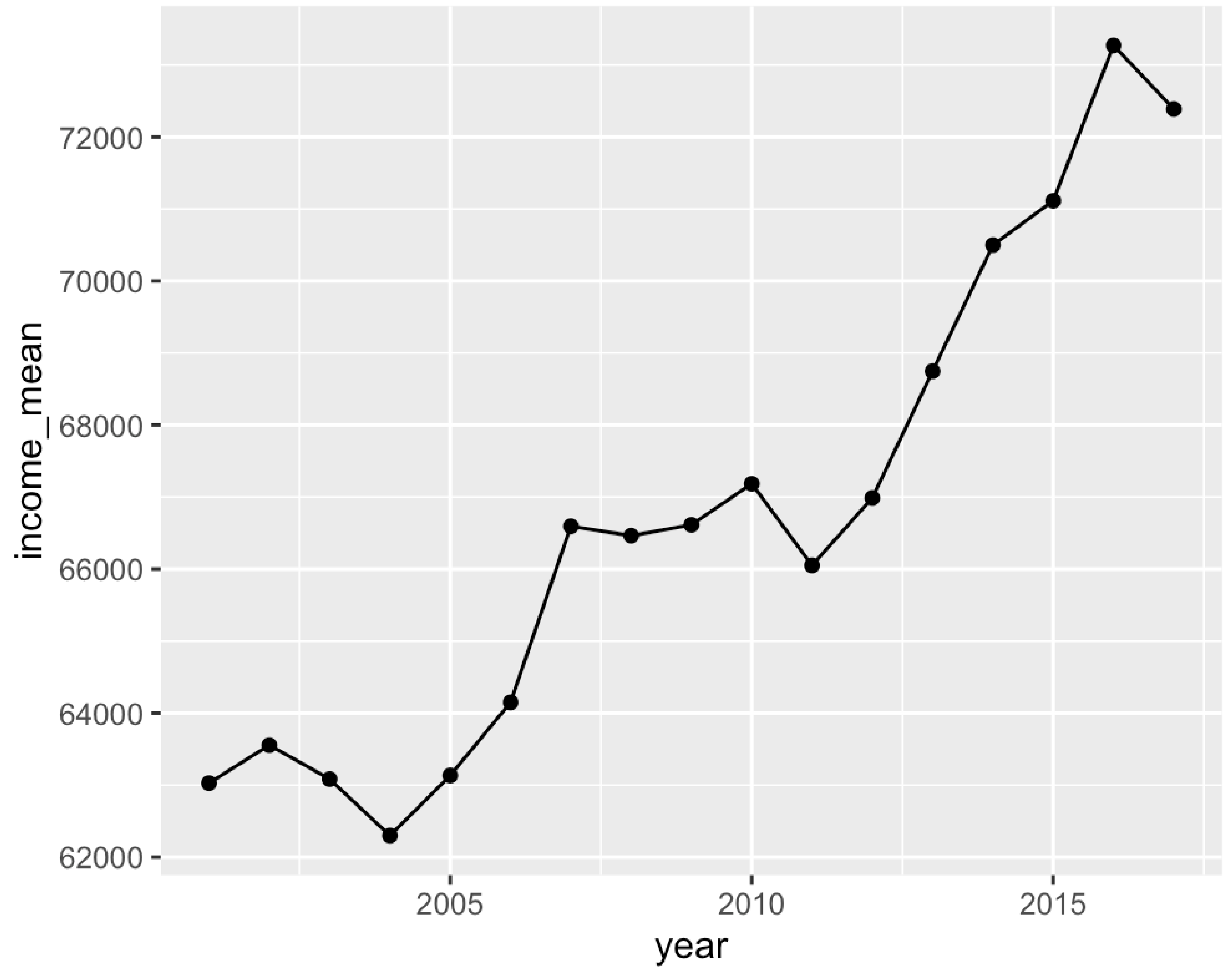
```
ggplot(data = basel_avg,
       mapping = aes(x = year,
                     y = income_mean)) +

  # Show as smoothed curve
  geom_smooth()
```



geom_*()

`geom_*()` functions define which geometric objects are used to illustrate the data.

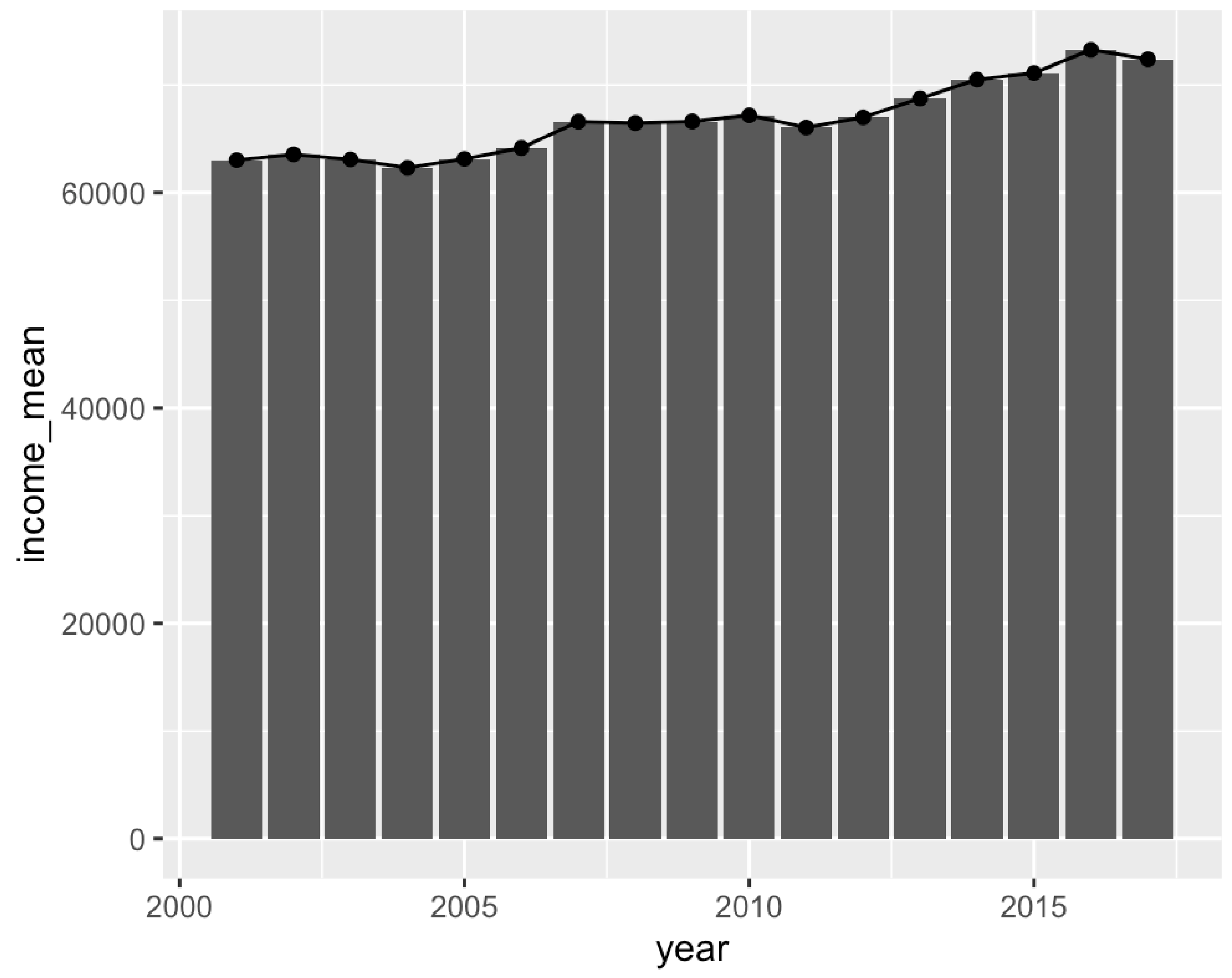
```
ggplot(data = basel_avg,
       mapping = aes(x = year,
                     y = income_mean)) +

  # Show as points and lines
  geom_point()   +
  geom_line()
```



`geom_*()`

`geom_*()` functions define which geometric objects are used to illustrate the data.

```
ggplot(data = basel_avg,
       mapping = aes(x = year,
                     y = income_mean)) +

  # Add bars (not necessarily recommended)
  geom_bar(stat = "identity") +

  # Show as points and lines
  geom_point()   +
  geom_line()
```

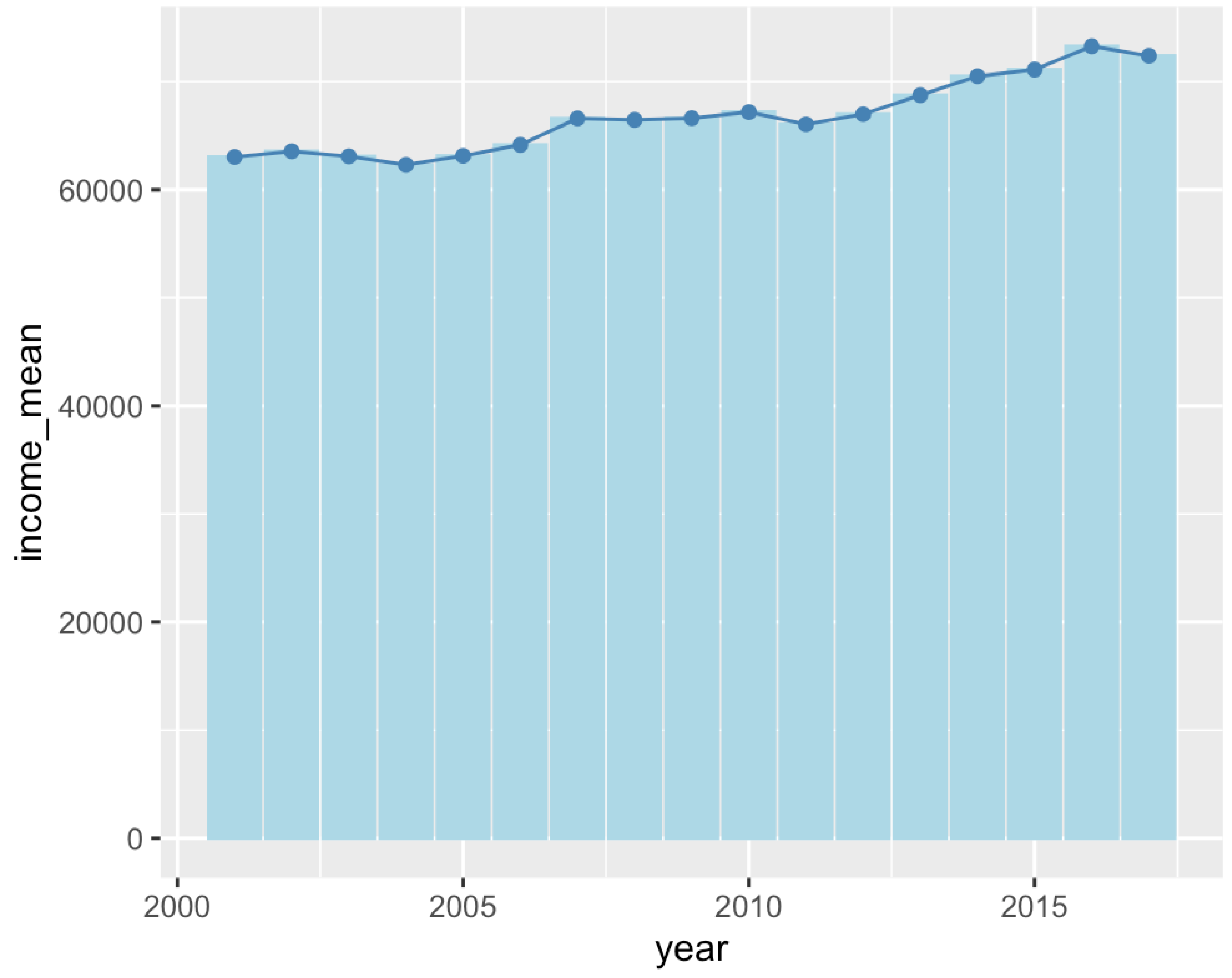


Colors

R understands a large number of **color names** (see `colors()` for the whole set).

Additionally colors can be specified using **hex codes** or the `rgb()` function.

```
ggplot(data = basel_avg,
       mapping = aes(x = year,
                     y = income_mean)) +

  # Add bars (not necessarily recommended)
  geom_bar(stat = "identity",
           col = "lightblue",
           fill = "lightblue") +

  # Show as points and lines
  geom_point(col = "#4682B4") +
  geom_line(col = "#4682B4")
```

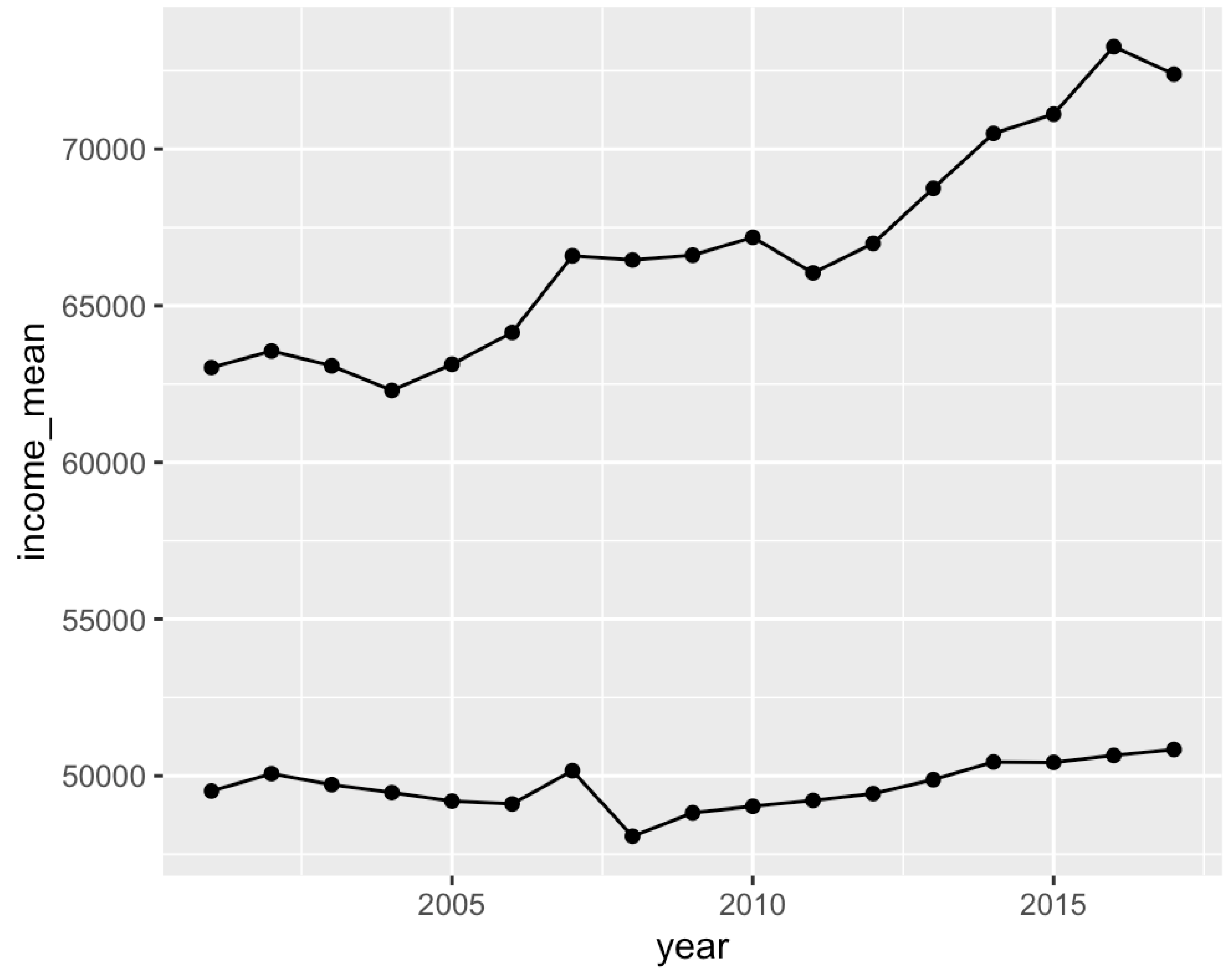


`geom_*()`

Most `geom_*()` functions allow the independent specification of **`data`** and **`mapping`**.

Can be used to add geoms for other cases or variables in the data.

```
ggplot(data = basel_avg,
       mapping = aes(x = year,
                     y = income_mean)) +
  geom_point() +
  geom_line() +

  # Add points and lines for median
  geom_point(aes(y = income_median)) +
  geom_line(aes(y = income_median))
```



Wrangling

Oftentimes, creating the desired plot requires appropriate data wrangling.

`ggplot` works best with **long data formats**.

```r
# pivot to long format
basel_avg_long <- basel_avg  %>%
  pivot_longer(-year,
               names_to = "statistic",
               values_to = "income")
```

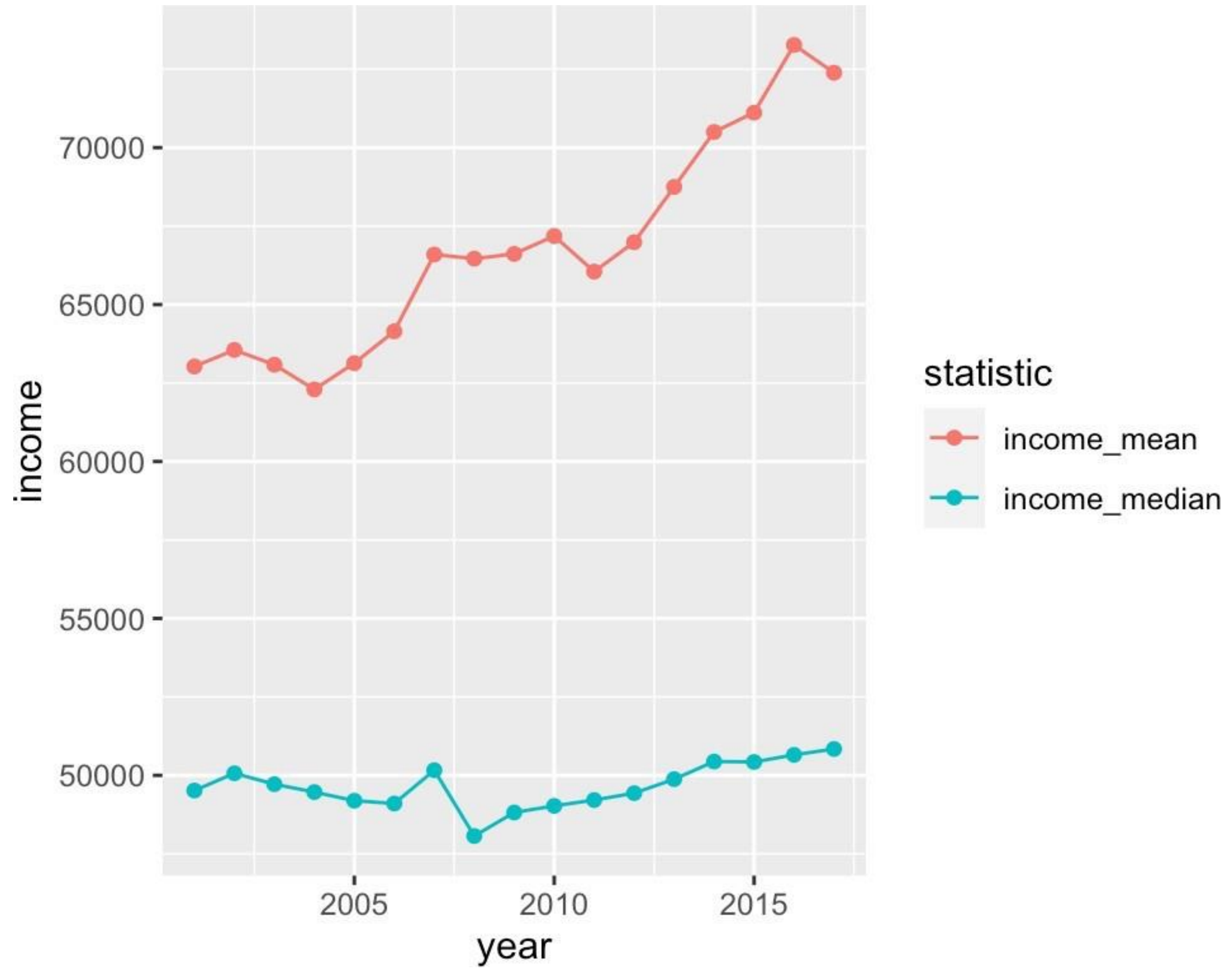
```r
basel_avg_long
```

```
# A tibble: 34 × 3
     year statistic     income
    <dbl> <chr>          <dbl>
1    2001 income_mean   63027.
2    2001 income_median 49516.
3    2002 income_mean   63555.
4    2002 income_median 50066.
5    2003 income_mean   63083.
6    2003 income_median 49717.
  7  2004 income_mean   62298.
  8  2004 income_median 49467.
  9  2005 income_mean   63133.
10   2005 income_median 49192.
# …  with
```

aes()

aes() helps define the structure of the **mapping** argument

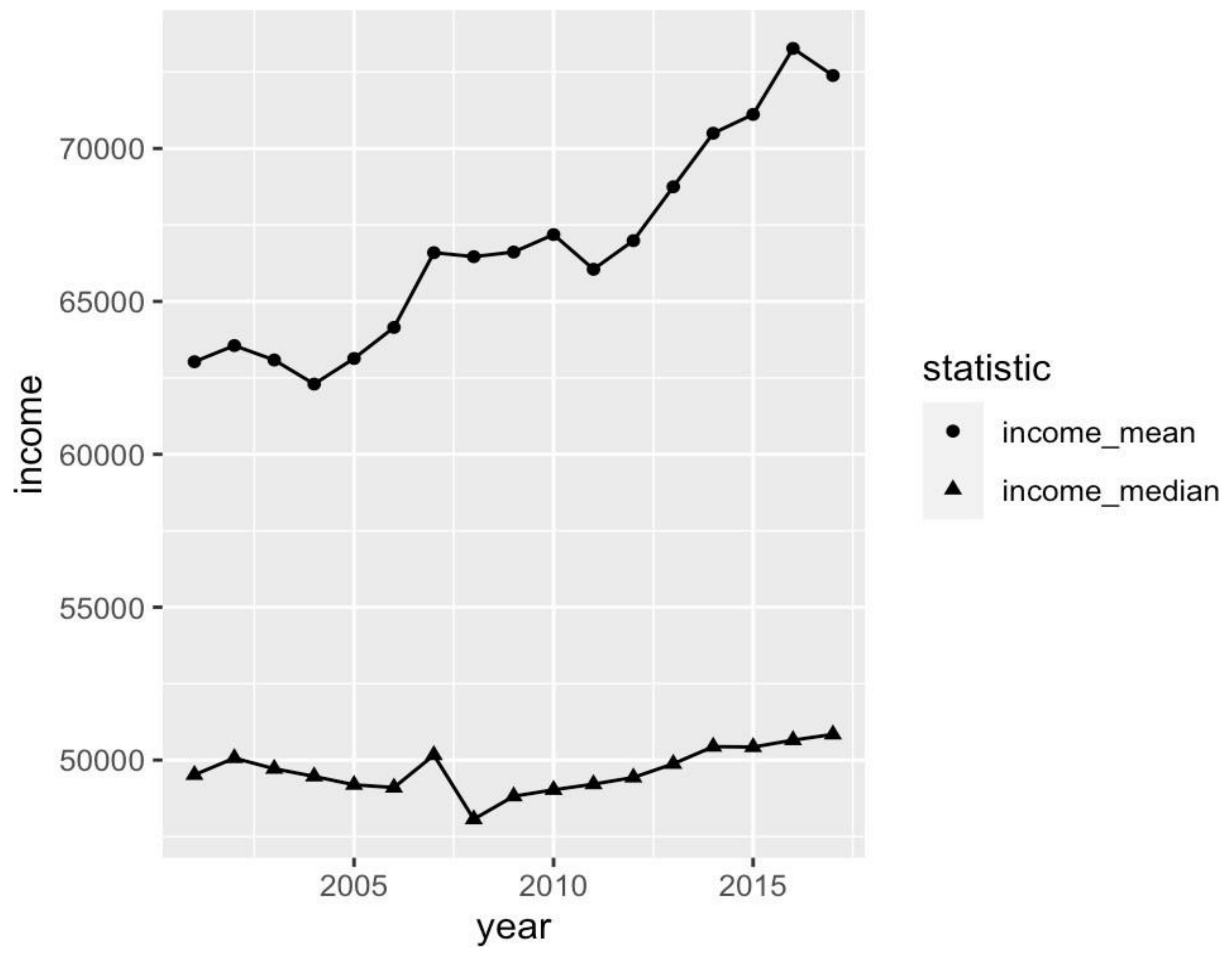
```
# use basel_avg_long
ggplot(data = basel_avg_long,
       mapping = aes(
         x = year,
         y = income,

         # add color dimension
         col = statistic)) +
  geom_point() +
  geom_line()
```



aes()

aes() helps define the structure of the **mapping** argument

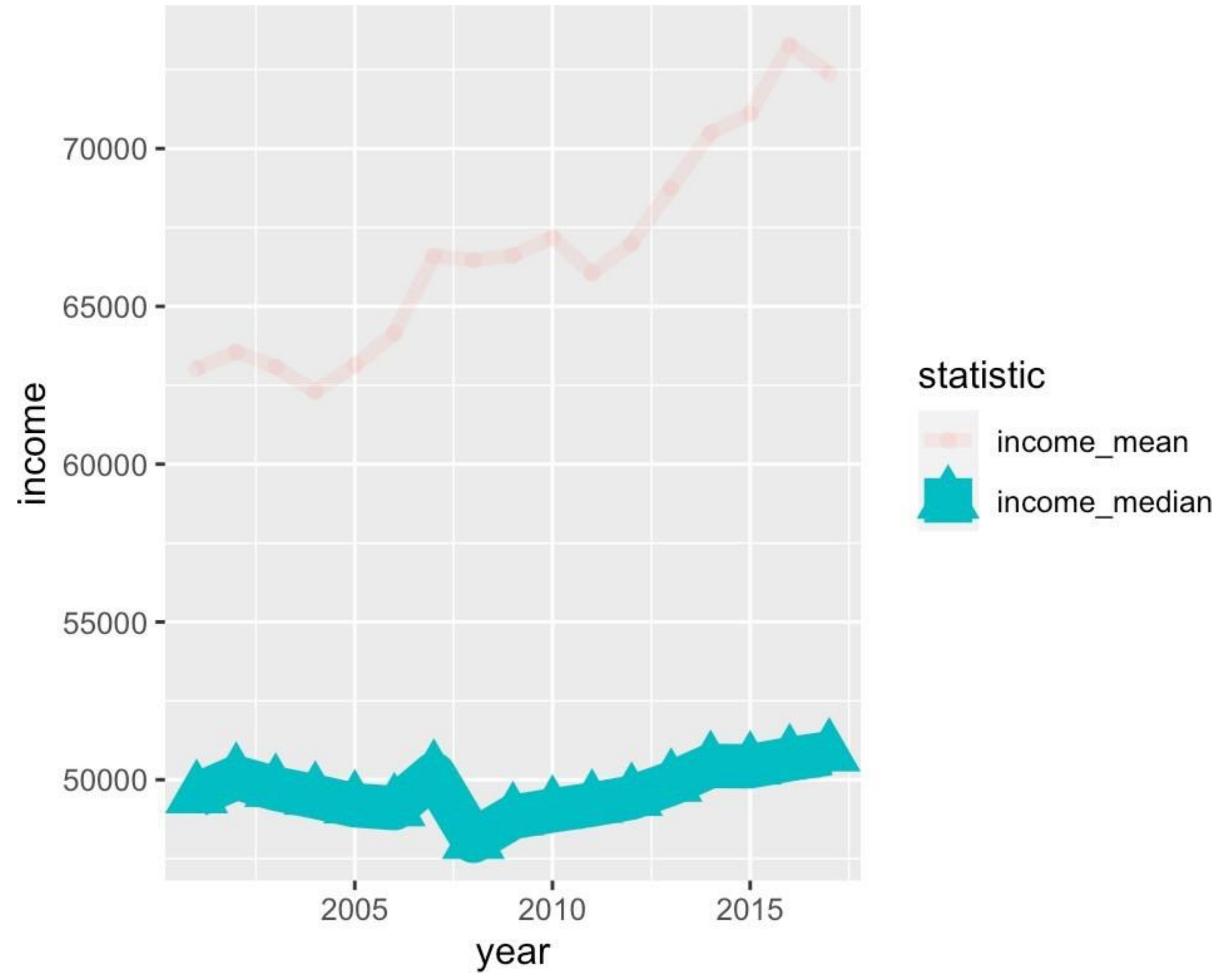
```
# use basel_avg_long
ggplot(data = basel_avg_long,
       mapping = aes(
         x = year,
         y = income,

         # add shape dimension
         shape = statistic)) +
  geom_point() +
  geom_line()
```



aes()

aes() helps define the structure of the **mapping** argument

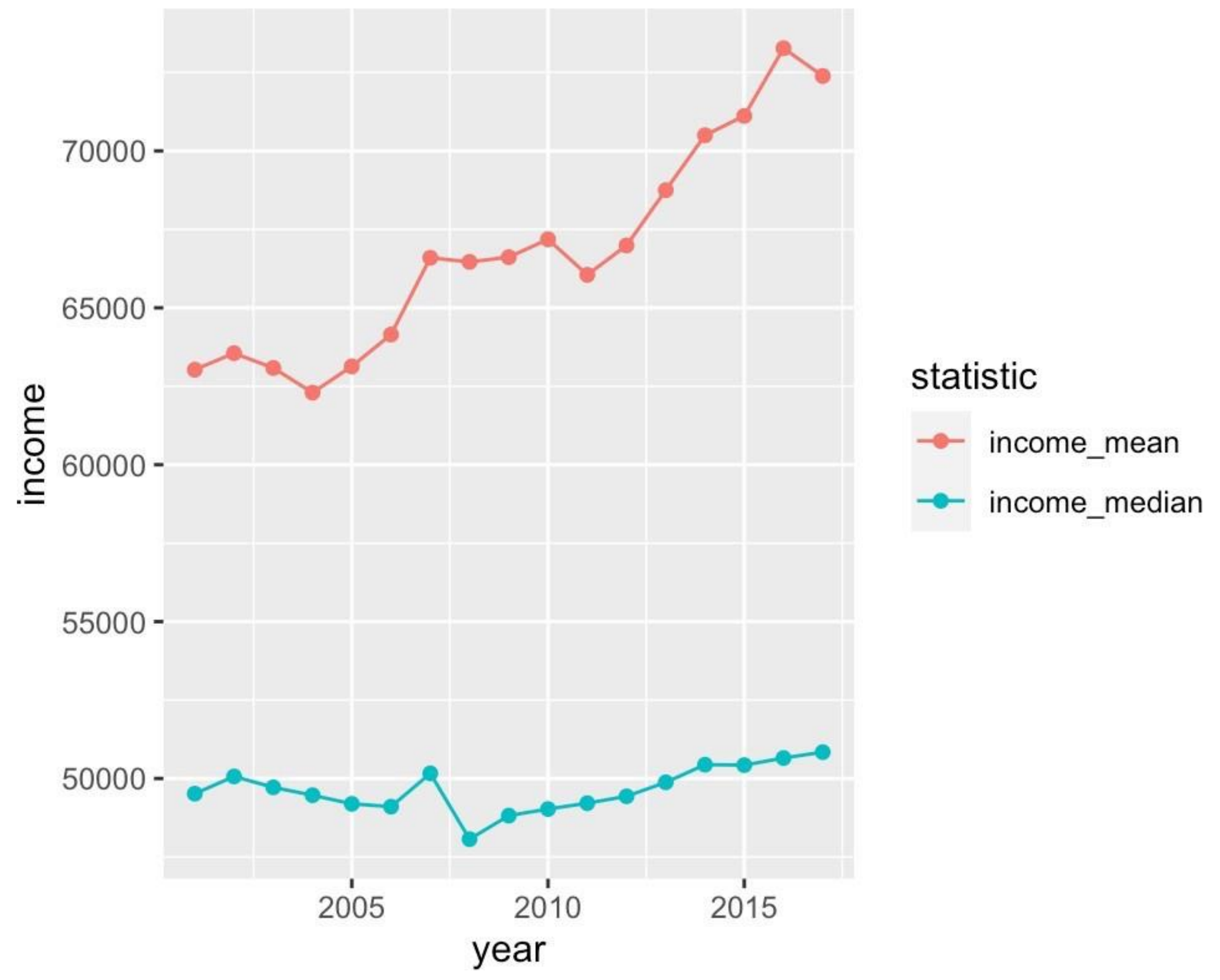
```
# use basel_avg_long
ggplot(data = basel_avg_long,
       mapping = aes(
         x = year,
         y = income,

         # add many dimensions
         # (not recommended)
         col = statistic,
         shape = statistic,
         size = statistic,
         alpha = statistic)) +
  geom_point() +
  geom_line()
```



aes()

`aes()` helps define the structure of the **`mapping`** `argument`

```
# use basel_avg_long
ggplot(data = basel_avg_long,
       mapping = aes(
         x = year,
         y = income,
         # add many dimensions
         col = statistic))+
  geom_point() +
  geom_line()
```



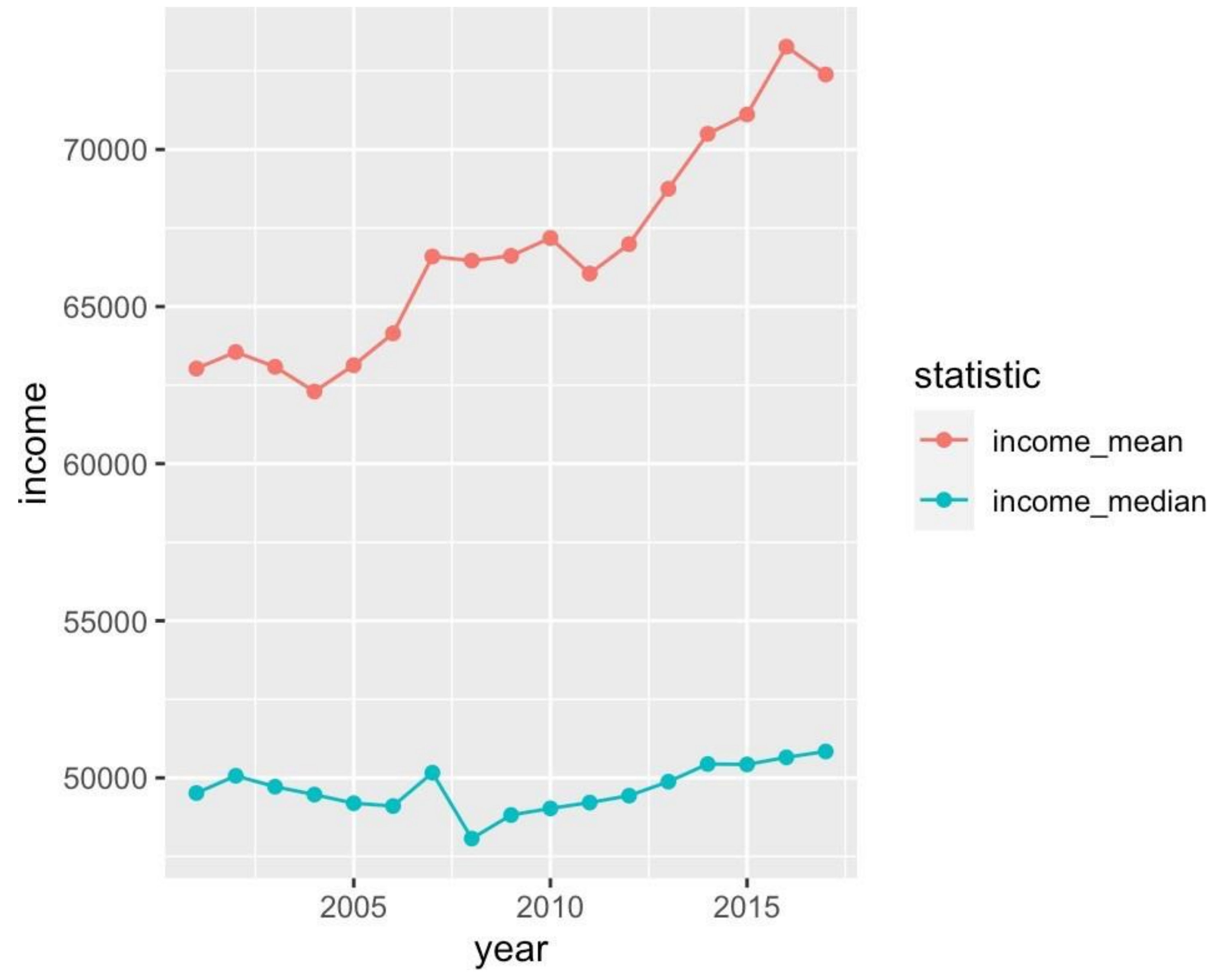
facet_*()

Facetting creates the **same plot for groups** defined by another variable.

Key functions:

```
facet_wrap()
facet_grid()
```

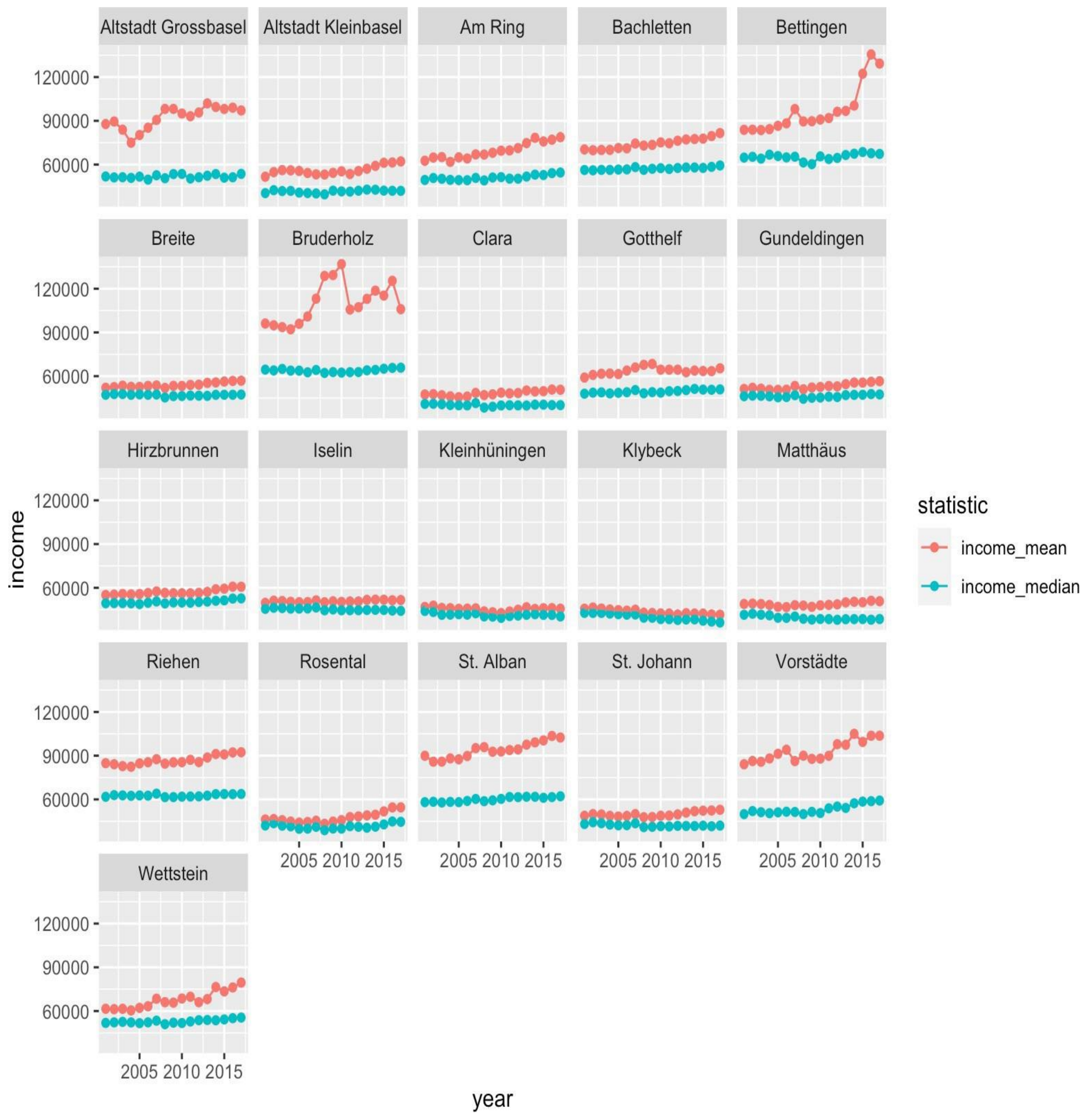
```
basel_long <- basel %>%
  pivot_longer(c(income_mean, income_median),
               names_to = 'statistic',
               values_to = 'income')
```



facet_*()

Facetting creates the **same plot for groups** defined by another variable.

```
# use basel_long
ggplot(data = basel_long,
       mapping = aes(
         x = year,
         y = income,
         col = statistic)) +
  geom_point() +
  geom_line() +

  # facet by quarter
  facet_wrap(~quarter)
```



patchwork

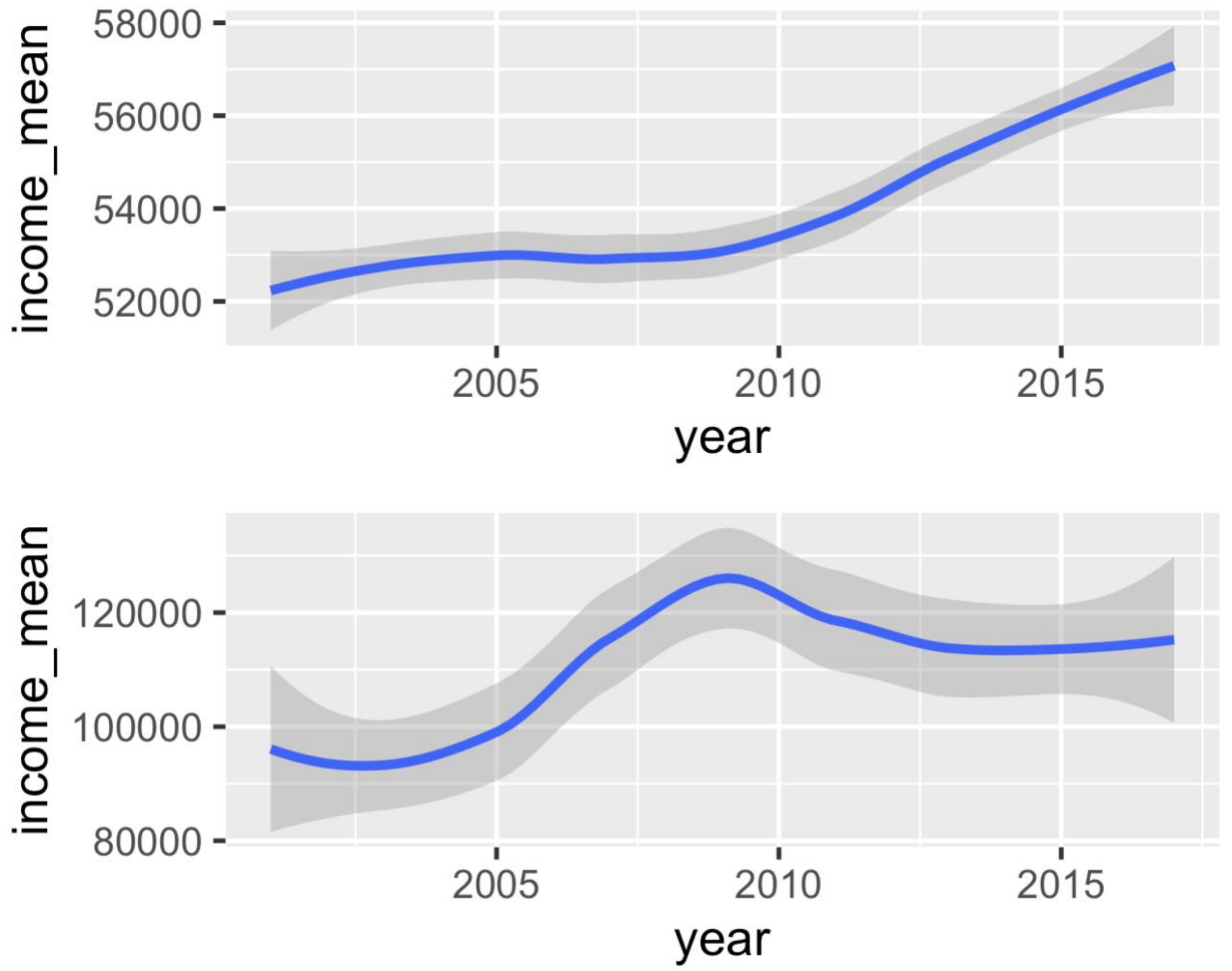
patchwork provides a simple syntax to combine plots.

patchwork syntax:

+ | combine horizontally
/ | combine vertically
| | spacer
() | grouper
& | apply to all
plot_layout | control layout

```
# use patchwork
  graph1 <- basel %>%
    filter(quarter == "Breite") %>%
    ggplot(aes(x = year,
               y = income_mean
               )) +
    geom_smooth()

  graph2 <- basel %>%
    filter(quarter == "Bruderholz") %>%
    ggplot(aes(x = year,
               y = income_mean
    )) +
      geom_smooth()
```

```
graph1/graph2
```



patchwork

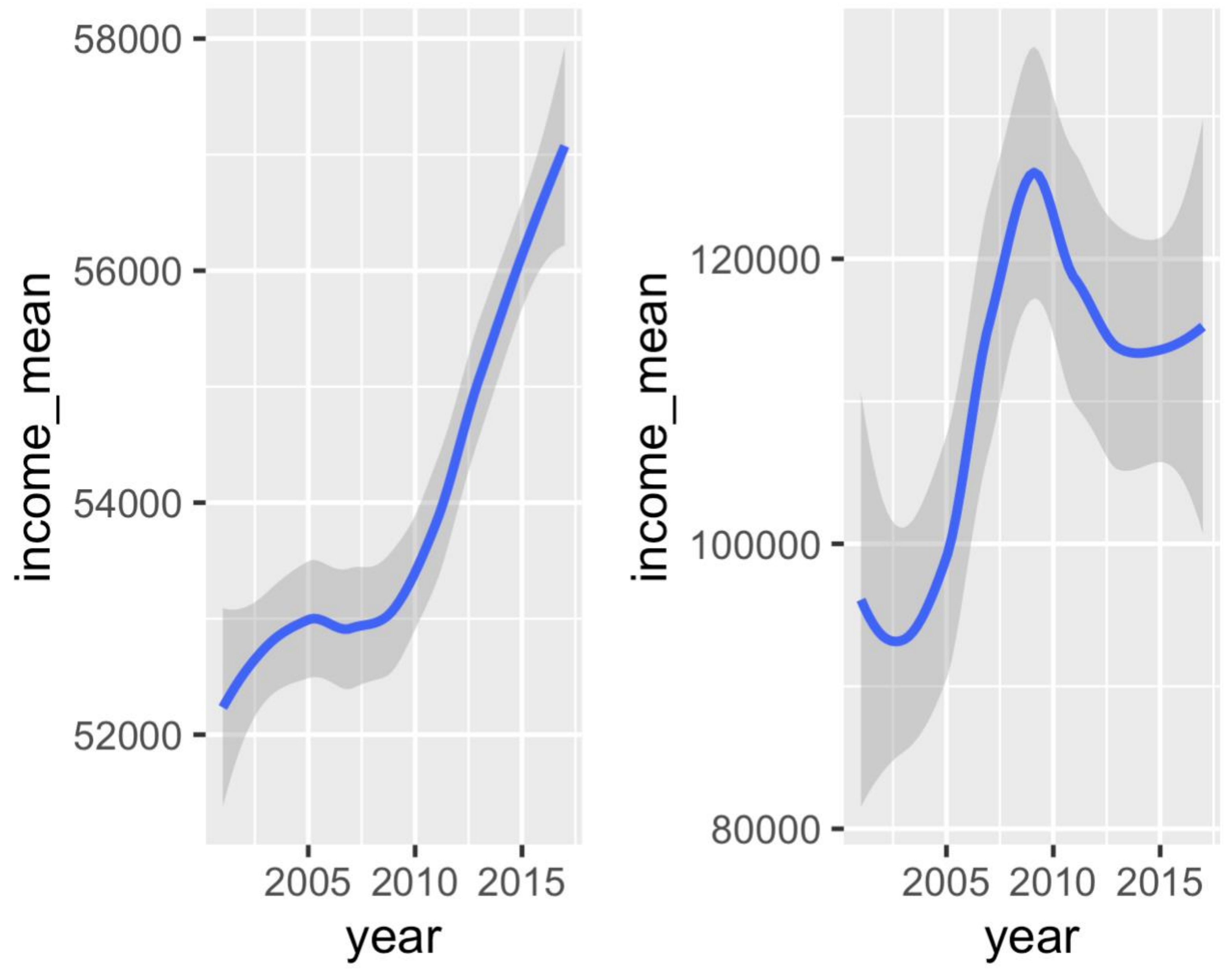
patchwork provides a simple syntax to combine plots.

patchwork syntax:

+	combine horizontally
/	combine vertically
|	spacer
()	grouper
&	apply to all
plot_layout	control layout

```
# use patchwork
  graph1 <- base1 %>%
    filter(quarter == "Breite") %>%
    ggplot(aes(x = year,
               y = income_mean
               )) +
    geom_smooth()

  graph2 <- base1 %>%
    filter(quarter == "Bruderholz") %>%
    ggplot(aes(x = year,
               y = income_mean
    )) +
      geom_smooth()
```

```
graph1+graph2
```



patchwork

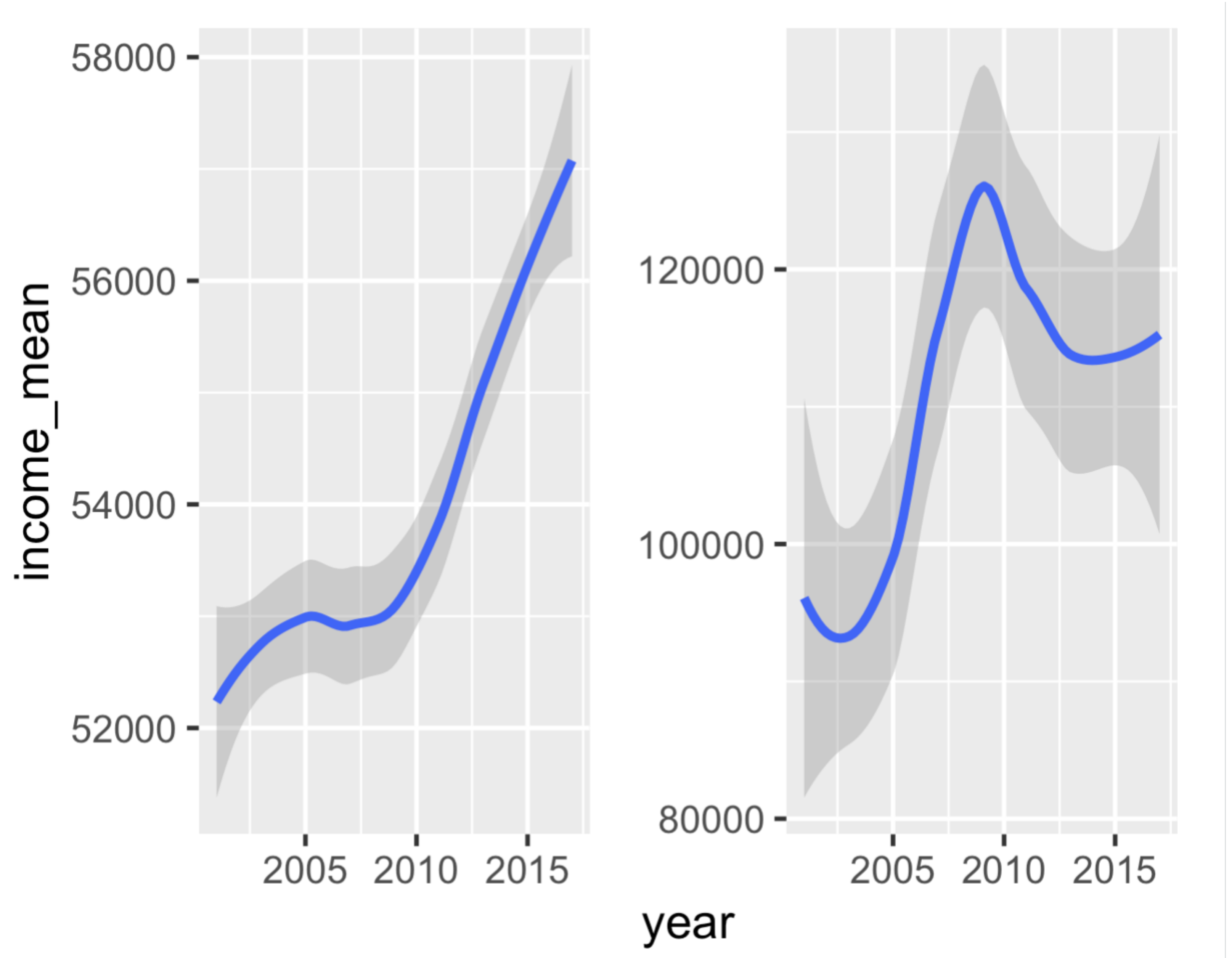
patchwork provides a simple syntax to combine plots.

patchwork syntax:

+ | combine horizontally
/ | combine vertically
| | spacer
() | grouper
& | apply to all
plot_layout | control layout

```
# use patchwork
  graph1 <- base1 %>%
    filter(quarter == "Breite") %>%
    ggplot(aes(x = year,
               y = income_mean
               )) +
    geom_smooth()

  graph2 <- base1 %>%
    filter(quarter == "Bruderholz") %>%
    ggplot(aes(x = year,
               y = income_mean
    )) +
      geom_smooth()
```

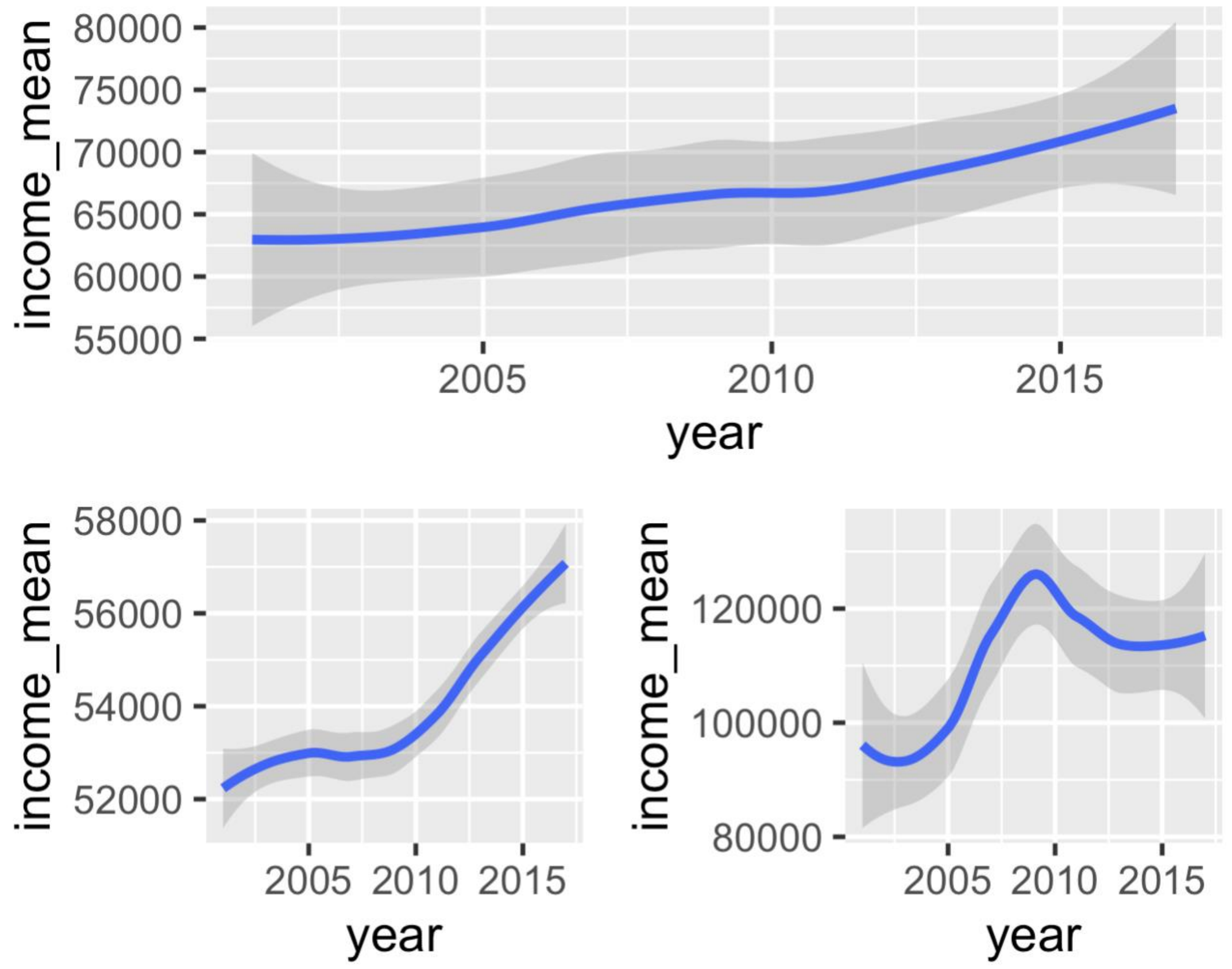
```
graph1+graph2+plot_layout(axes="collect")
```



patchwork

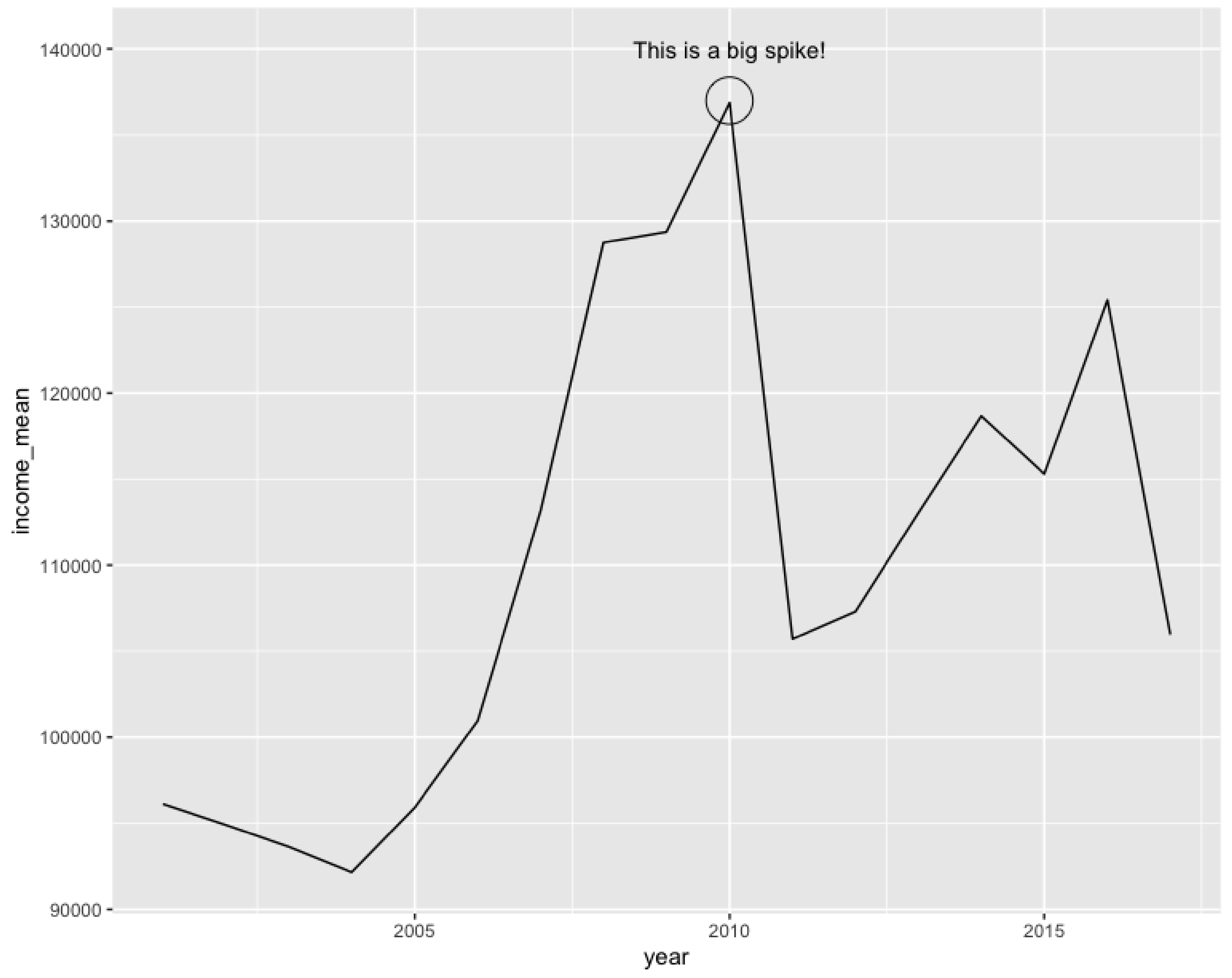
```
# use patchwork
  graph1 <- basel %>%
    filter(quarter == "Breite") %>%
    ggplot(aes(x = year,
                 y = income_mean
                 )) +
    geom_smooth()

  graph2 <- basel %>%
    filter(quarter == "Bruderholz") %>%
    ggplot(aes(x = year,
                 y = income_mean
    )) +
      geom_smooth()

   all <- basel %>%
    ggplot(aes(x = year,
                 y = income_mean
    )) +
      geom_smooth()
```

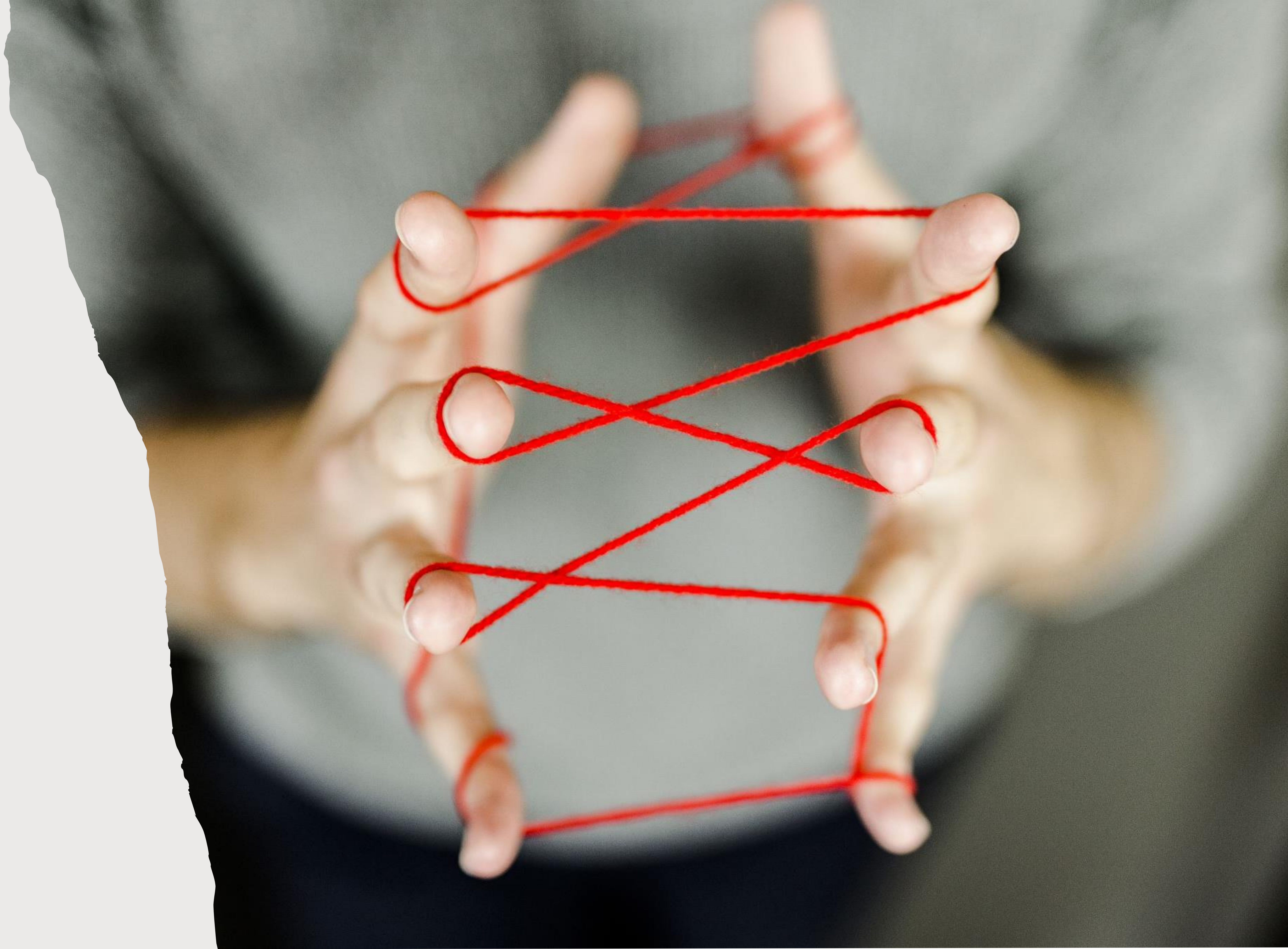
```
all/(graph1+graph2)+
  plot_layout(guides="collect")
```



annotation

```
# Annotation example

 basel %>%
    filter(quarter == "Bruderholz") %>%
    ggplot(aes(x = year,
               y = income_mean
    )) +
    annotate(geom="text", x=2010, y=140000,
             label="This is a big spike!") +
    annotate(geom="point",
             x=2010, y=137000, size=10,
             shape=21, fill="transparent") +
    geom_line()
```

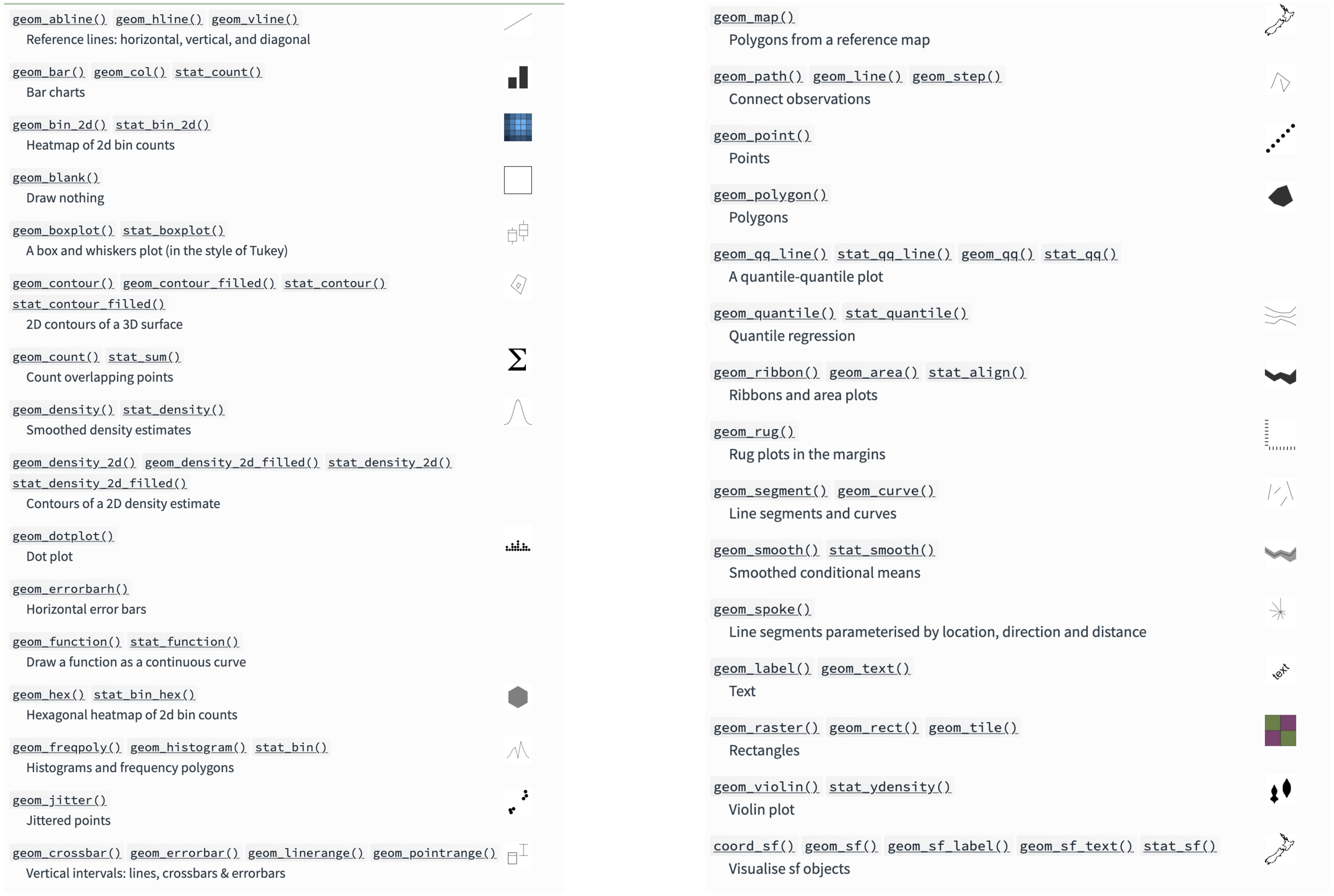




Practice

```
# clean it up a bit
# change format of 'Order Date' to "%m/%d/%y"
# Use "%Y" to create a new variable "Year"
# Define "Year" as numeric
# Create a graph that shows a point for the total profit for each year
# Connect the dots using a line.
# Add the total sales as well to the graph
# make the profit darkgreen
# change the shape of the profit to 0 (square)
# create a bar chart summarizing total sales by segment
# assign the color of the "fill" by the total sales
# create a bar chart that shows the sales sum by state instead
# of segment. Color the fill again by total sales. Flip the coordinates
# using coord_flip()
# let's reorders state by Sales
# let's only take the top 10
# create a plot of the total profit for each region.
  Use facet_wrap()
```

```
# create a line chart of total profit of Furniture, Office Supplies,
# and Everything (no filter) by year. Show the overall summary up top and the
# furniture and office supplies side-by-side underneath
# so a line graph of mean sales across years. On the value for 2013
# provide an annotation that we had a "Successful Social Media Campaign"
# that year
# Make a graph using something other than lines and dots (see slides)
```



`geom_abline()` `geom_hline()` `geom_vline()`
Reference lines: horizontal, vertical, and diagonal

`geom_bar()` `geom_col()` `stat_count()`
Bar charts

`geom_bin_2d()` `stat_bin_2d()`
Heatmap of 2d bin counts

`geom_blank()`
Draw nothing

`geom_boxplot()` `stat_boxplot()`
A box and whiskers plot (in the style of Tukey)

`geom_contour()` `geom_contour_filled()` `stat_contour()` `stat_contour_filled()`
2D contours of a 3D surface

`geom_count()` `stat_sum()`
Count overlapping points

`geom_density()` `stat_density()`
Smoothed density estimates

`geom_density_2d()` `geom_density_2d_filled()` `stat_density_2d()` `stat_density_2d_filled()`
Contours of a 2D density estimate

`geom_dotplot()`
Dot plot

`geom_errorbarh()`
Horizontal error bars

`geom_function()` `stat_function()`
Draw a function as a continuous curve

`geom_hex()` `stat_bin_hex()`
Hexagonal heatmap of 2d bin counts

`geom_freqpoly()` `geom_histogram()` `stat_bin()`
Histograms and frequency polygons

`geom_jitter()`
Jittered points

`geom_crossbar()` `geom_errorbar()` `geom_linerange()` `geom_pointrange()`
Vertical intervals: lines, crossbars & errorbars

`geom_map()`
Polygons from a reference map

`geom_path()` `geom_line()` `geom_step()`
Connect observations

`geom_point()`
Points

`geom_polygon()`
Polygons

`geom_qq_line()` `stat_qq_line()` `geom_qq()` `stat_qq()`
A quantile-quantile plot

`geom_quantile()` `stat_quantile()`
Quantile regression

`geom_ribbon()` `geom_area()` `stat_align()`
Ribbons and area plots

`geom_rug()`
Rug plots in the margins

`geom_segment()` `geom_curve()`
Line segments and curves

`geom_smooth()` `stat_smooth()`
Smoothed conditional means

`geom_spoke()`
Line segments parameterised by location, direction and distance

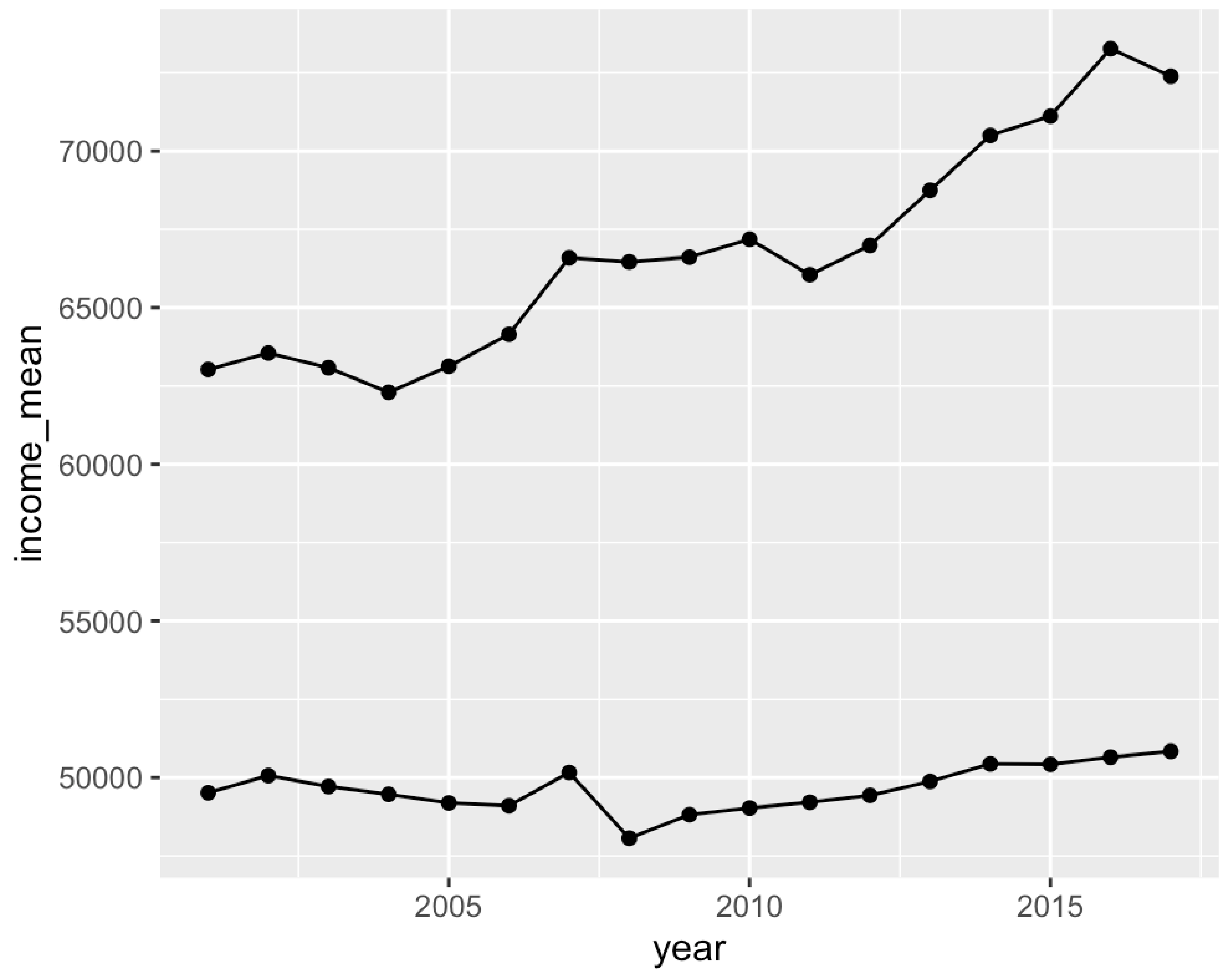
`geom_label()` `geom_text()`
Text

`geom_raster()` `geom_rect()` `geom_tile()`
Rectangles

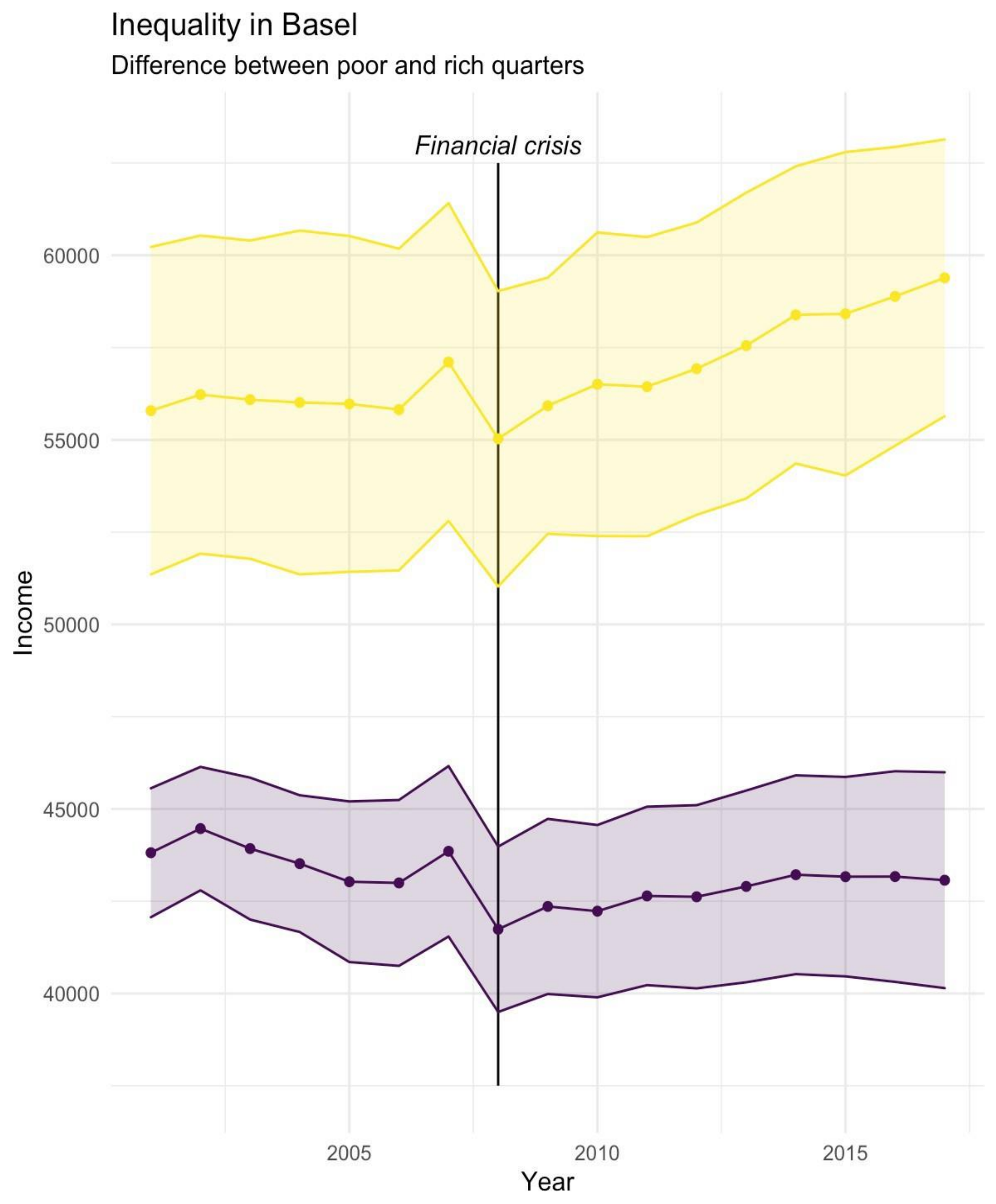
`geom_violin()` `stat_ydensity()`
Violin plot

`coord_sf()` `geom_sf()` `geom_sf_label()` `geom_sf_text()` `stat_sf()`
Visualise sf objects

Next time: Styling



vs


Inequality in Basel
Difference between poor and rich quarters
Financial crisis
Income
Year