

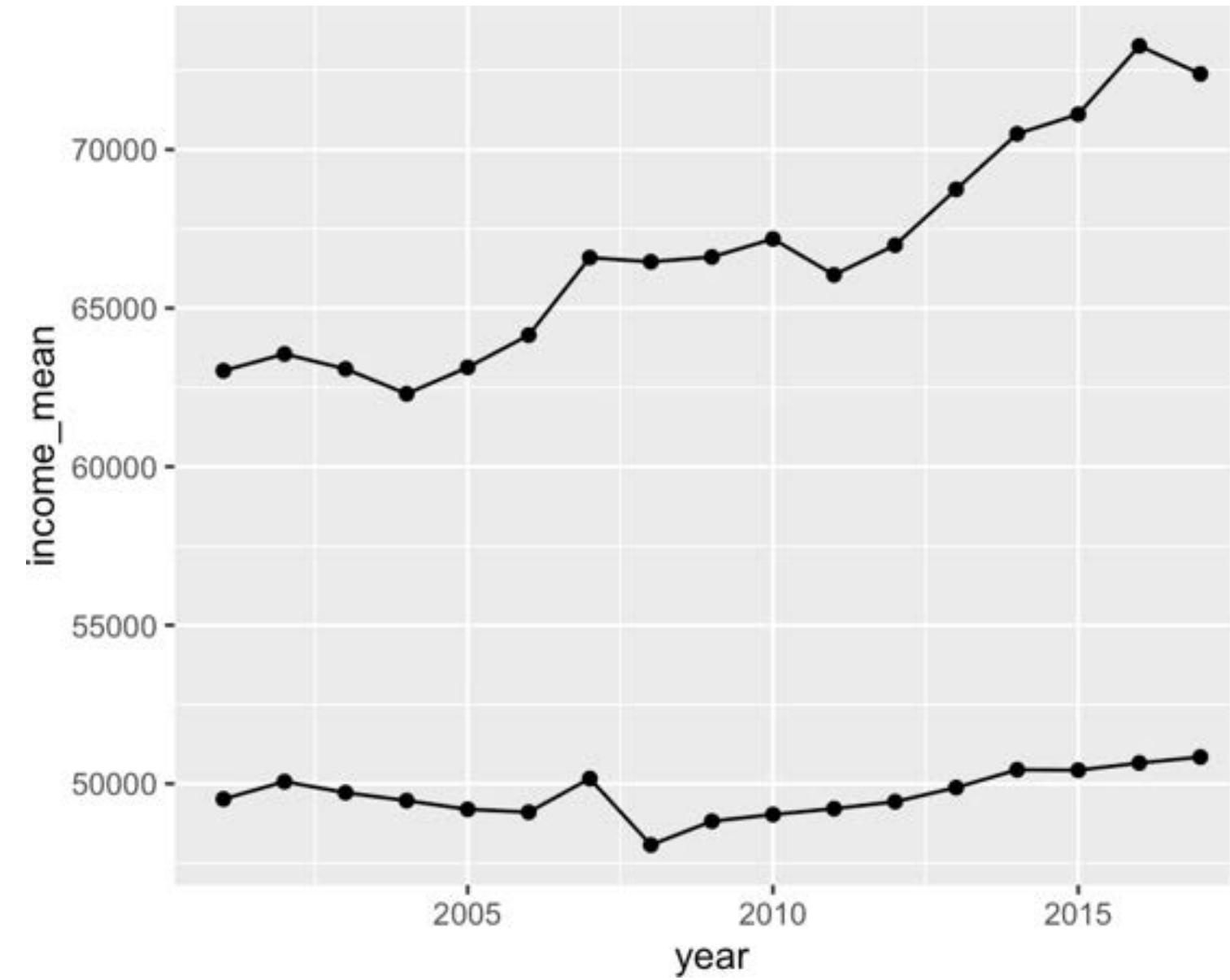


# Data Visualization Styling Plots

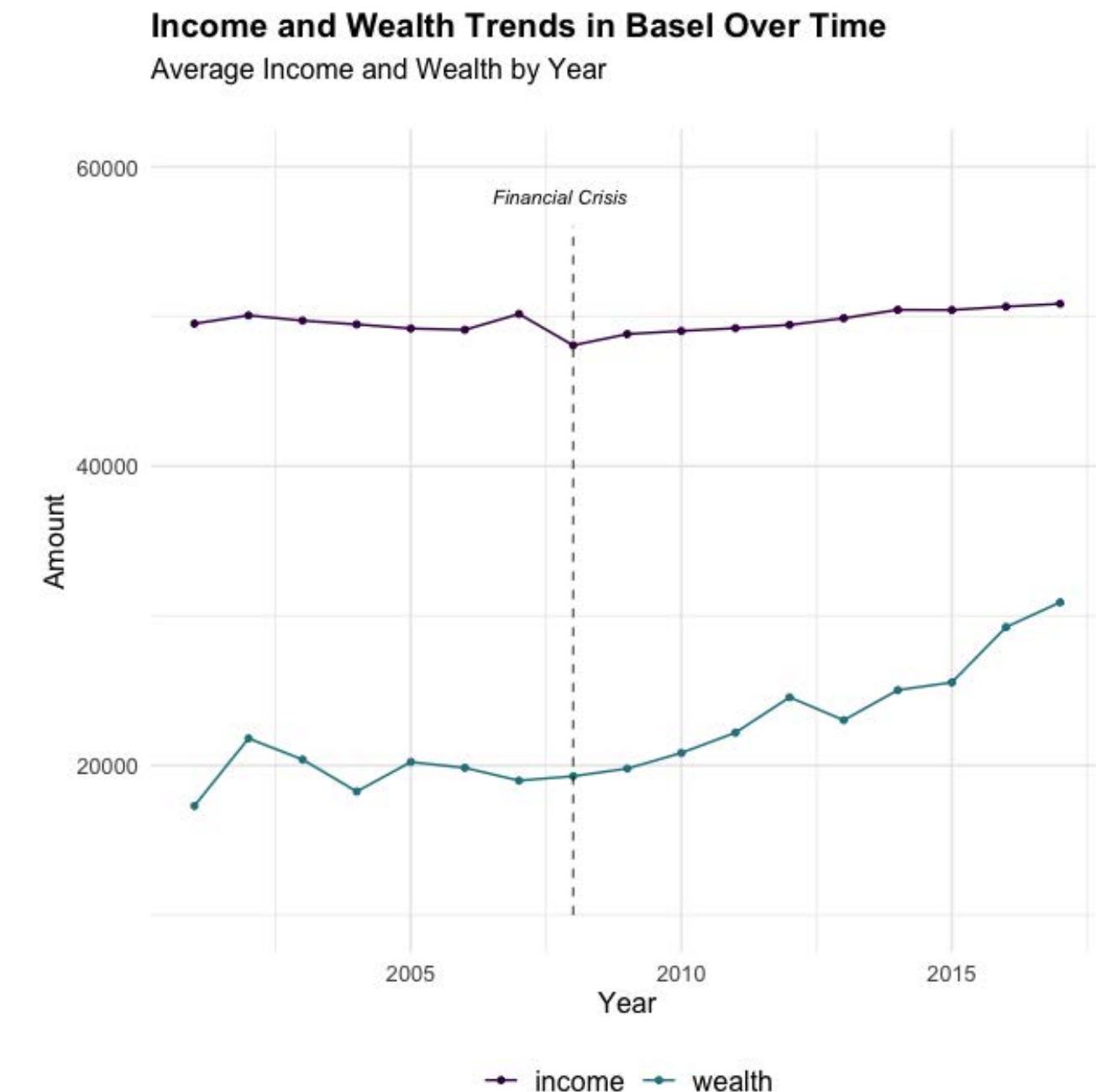
University  
of Cologne



# Why is styling important?

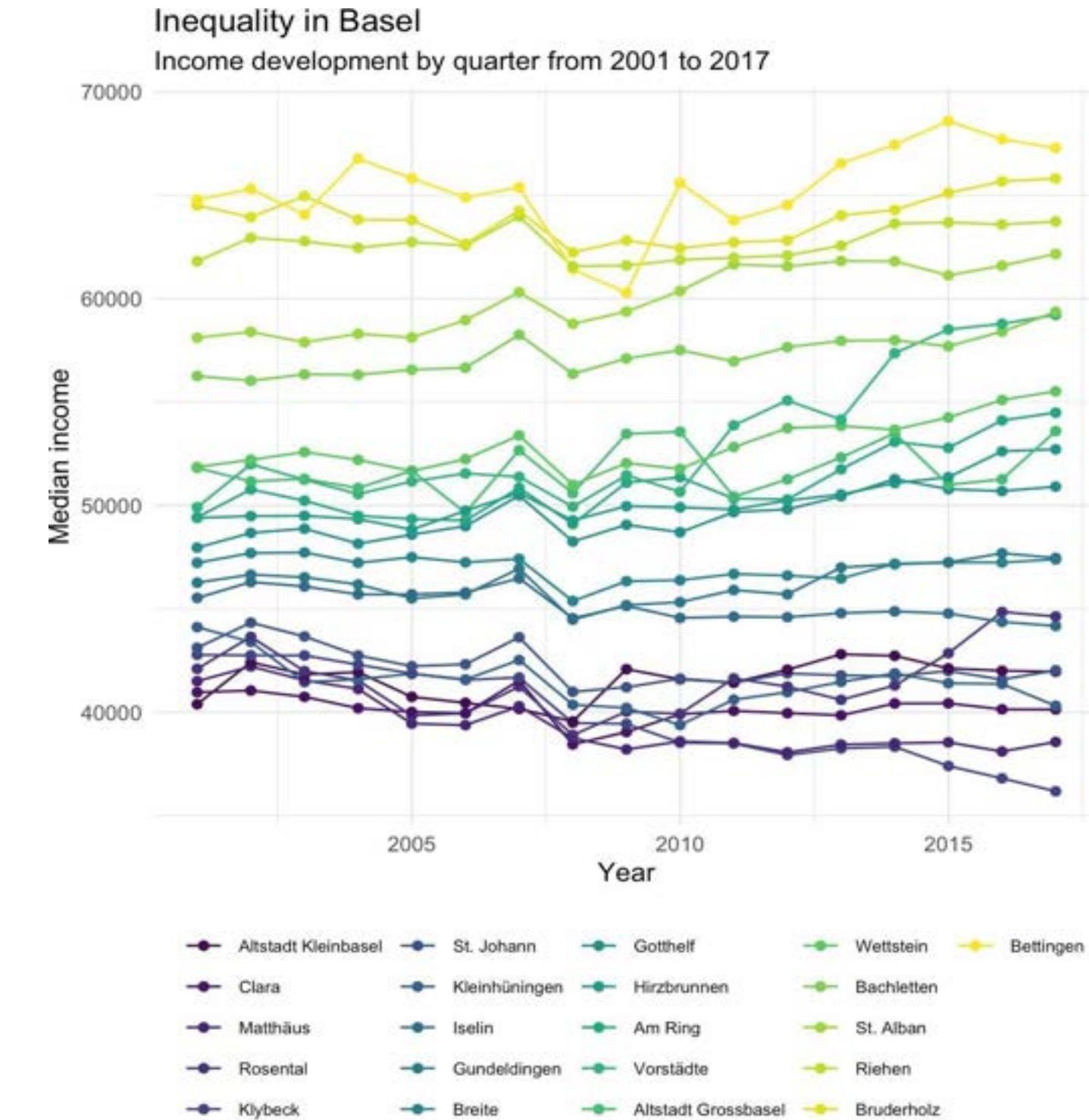


vs

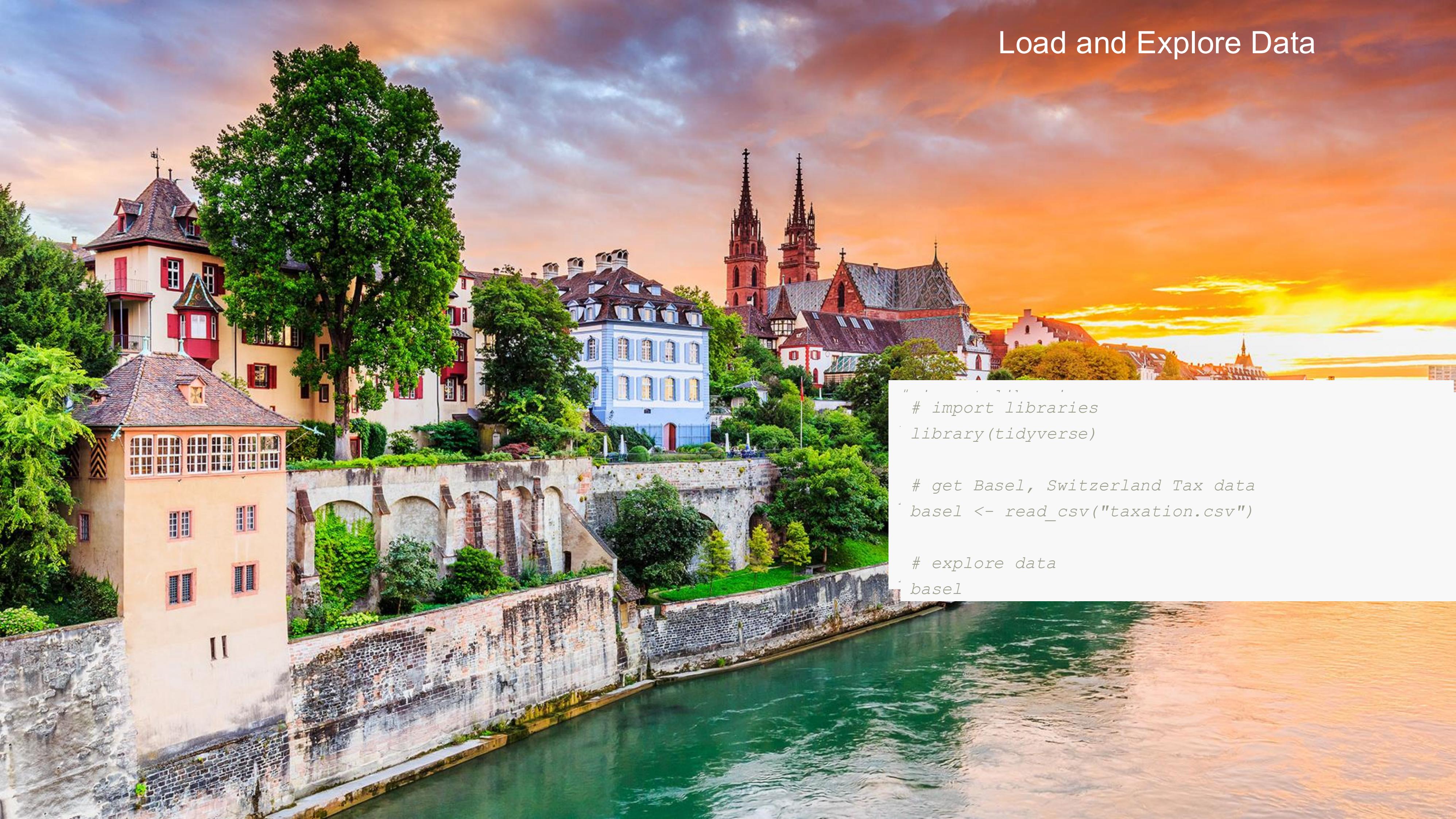


# Styling Plots

- (1) Use existing `theme_*`() presets.
- (2) Customize details using `theme()`
- (3) Adjust dimensions using `scale_*`().
- (4) Add an notion using `labs()`.



# Load and Explore Data



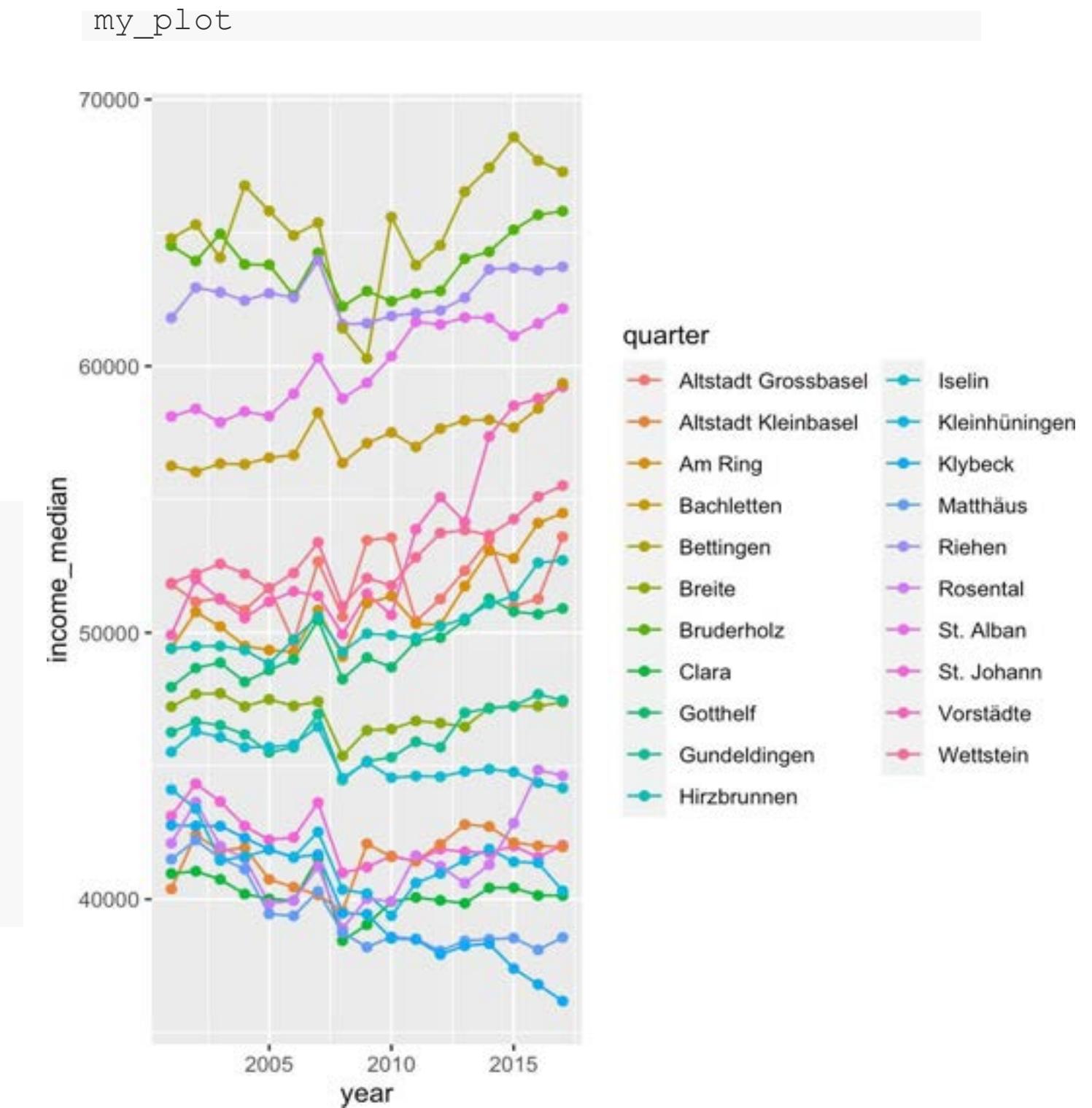
```
"# import libraries  
library(tidyverse)  
  
# get Basel, Switzerland Tax data  
basel <- read_csv("taxation.csv")  
  
# explore data  
basel
```

# The gg object

The output of `ggplot()` can be stored in an gg object.

The gg object can be expanded using `+` and the plot can be generated by a simple print.

```
# store plot as object  
  
myplot <- data %>%  
  ggplot(aes(x=year, y=income_median, col=quarter)) +  
  geom_line() +  
  geom_point()
```



# theme\_\*

Using `theme_*` the plot can be styled according to various presets.

A few themes (go through each theme):

`theme_grey()`

`theme_gray()`

`theme_bw()`

`theme_light()`

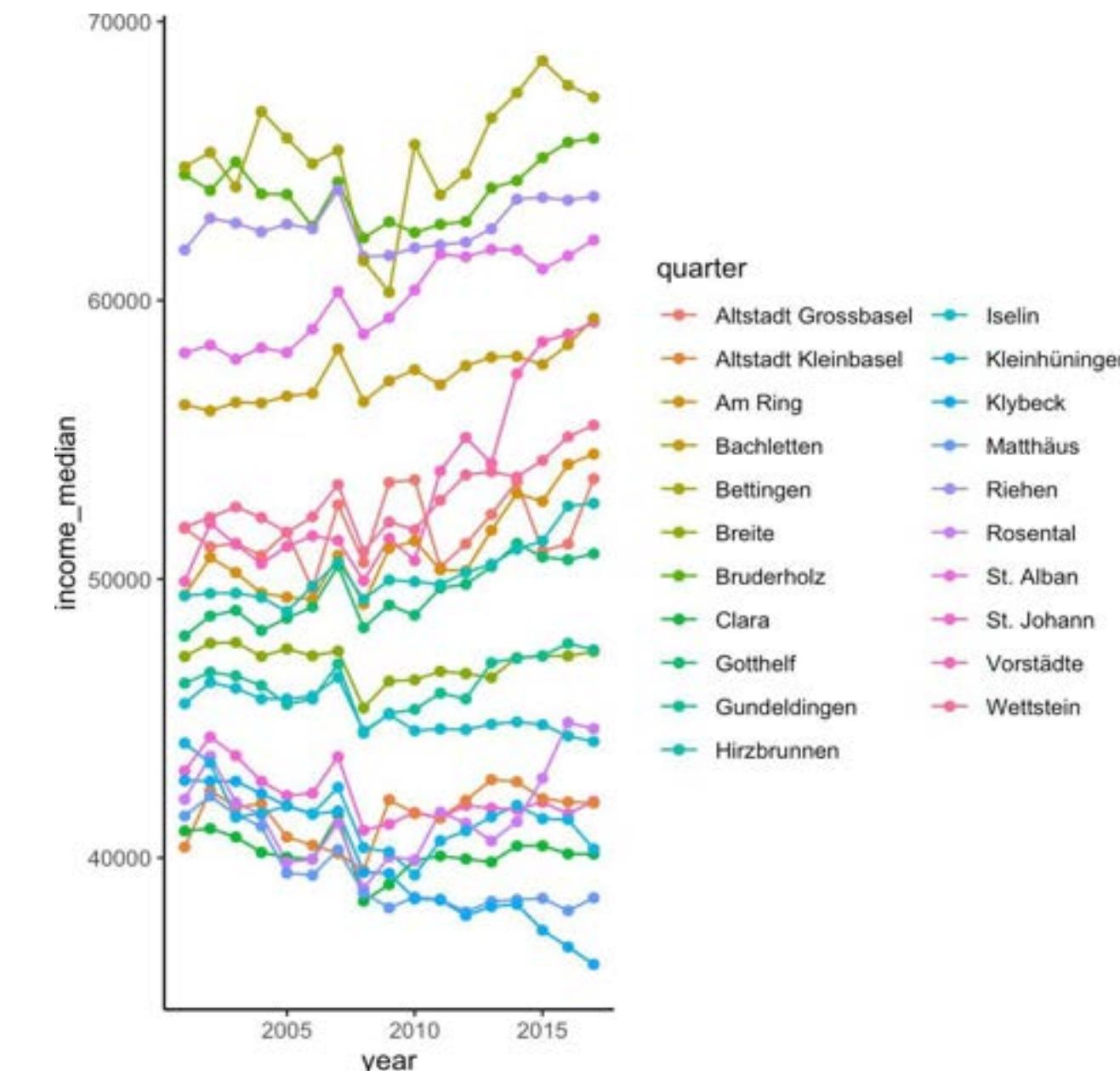
`theme_dark()`

`theme_minimal()`

`theme_classic()`

`theme_void()`

`my_plot + theme_classic()`



# theme\_\*

With **96 arguments** theme() permits specification of all aesthetic details.

Makes use of helper functions:

element\_rect() | for rectangles

element\_line() | for lines

element\_text() | for text

element\_blank() | for removals

```
# Using theme
my_plot +
  theme(argument =
    element_*(),
    argument =
    element_*(),
    ...)
```

theme {ggplot2}

R Documentation

## Modify components of a theme

### Description

Use theme() to modify individual components of a theme, allowing you to control the appearance of all non-data components of the plot. theme() only affects a single plot: see [theme\\_update\(\)](#) if you want modify the active theme, to affect all subsequent plots.

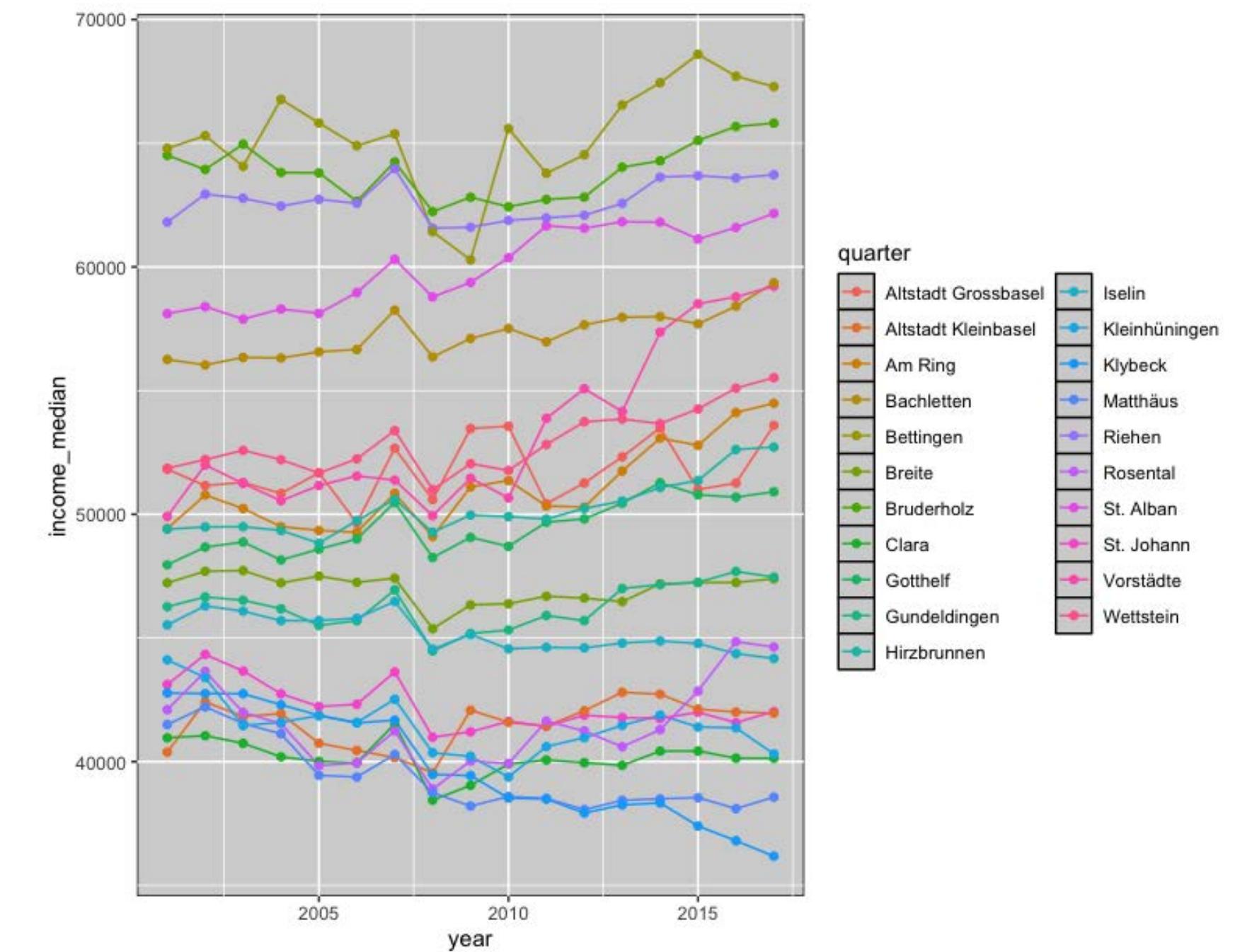
### Usage

```
theme(line, rect, text, title, aspect.ratio, axis.title, axis.title.x,
      axis.title.x.top, axis.title.x.bottom, axis.title.y, axis.title.y.left,
      axis.title.y.right, axis.text, axis.text.x, axis.text.x.top,
      axis.text.x.bottom, axis.text.y, axis.text.y.left, axis.text.y.right,
      axis.ticks, axis.ticks.x, axis.ticks.x.top, axis.ticks.x.bottom, axis.ticks.y,
      axis.ticks.y.left, axis.ticks.y.right, axis.ticks.length, axis.line,
      axis.line.x, axis.line.x.top, axis.line.x.bottom, axis.line.y,
      axis.line.y.left, axis.line.y.right, legend.background, legend.margin,
      legend.spacing, legend.spacing.x, legend.spacing.y, legend.key,
      legend.key.size, legend.key.height, legend.key.width, legend.text,
      legend.text.align, legend.title, legend.title.align, legend.position,
      legend.direction, legend.justification, legend.box, legend.box.just,
      legend.box.margin, legend.box.background, legend.box.spacing,
      panel.background, panel.border, panel.spacing, panel.spacing.x,
      panel.spacing.y, panel.grid, panel.grid.major, panel.grid.minor,
      panel.grid.major.x, panel.grid.major.y, panel.grid.minor.x,
      panel.grid.minor.y, panel.on top, plot.background, plot.title, plot.subtitle,
      plot.caption, plot.tag, plot.tag.position, plot.margin, strip.background,
      strip.background.x, strip.background.y, strip.placement, strip.text,
      strip.text.x, strip.text.y, strip.switch.pad.grid, strip.switch.pad.wrap, ...,
      complete = FALSE, validate = TRUE)
```

# theme\_\*

The `element_rect()` function is used to define the appearance of rectangular elements in a plot

```
# using element_rect ()  
myplot +  
  theme (panel.background = element_rect  
        (fill = "lightgray", color = "black"))
```

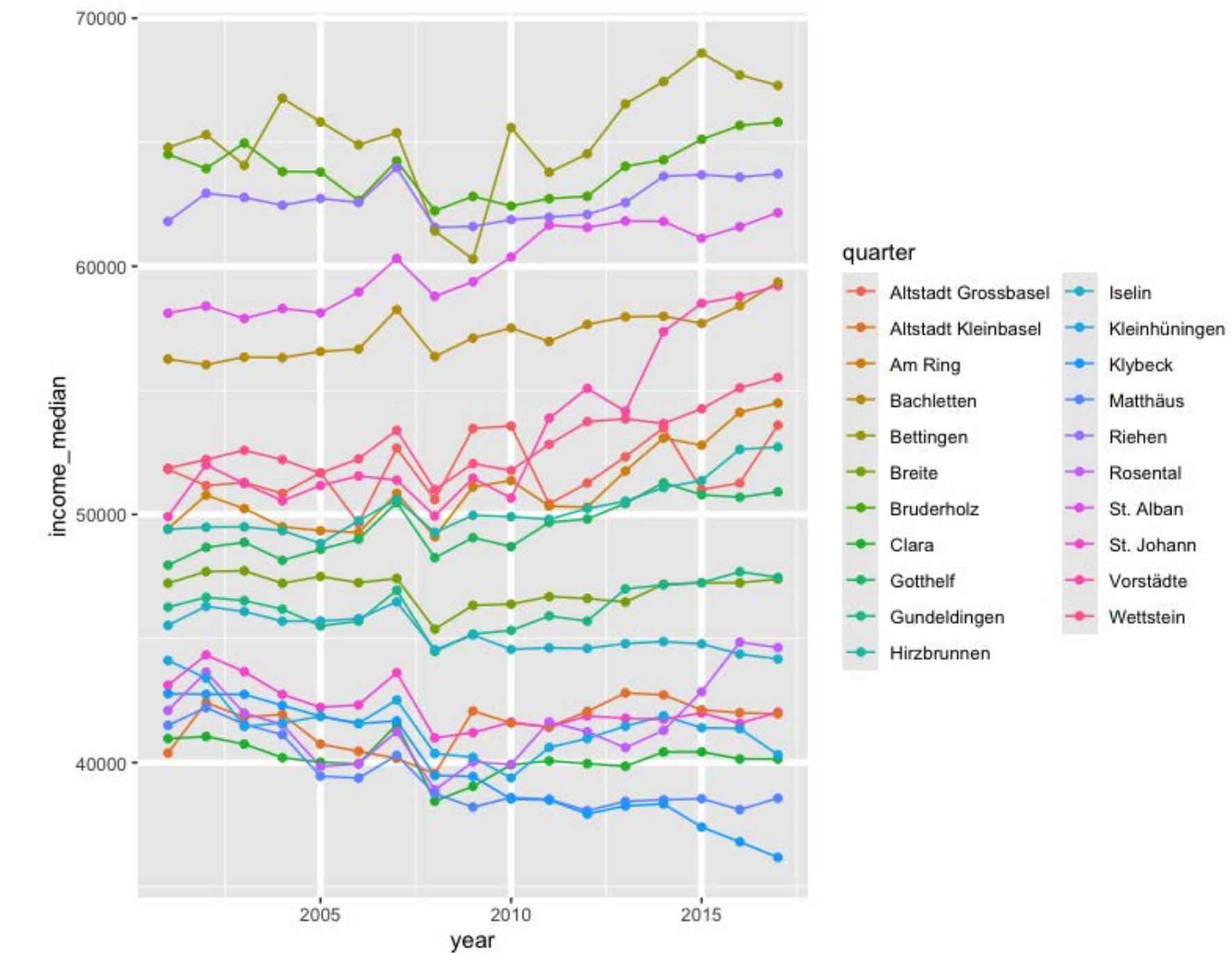


# theme\_\*

The `element_line()` function is used to define the appearance of **line-based elements** in a plot.

It allows you to customize elements such as grid lines, axis lines, and tick marks.

```
# linewidth  
myplot +  
  theme(panel.grid.major = element_line  
        (color = "white", linewidth= 1.5))
```

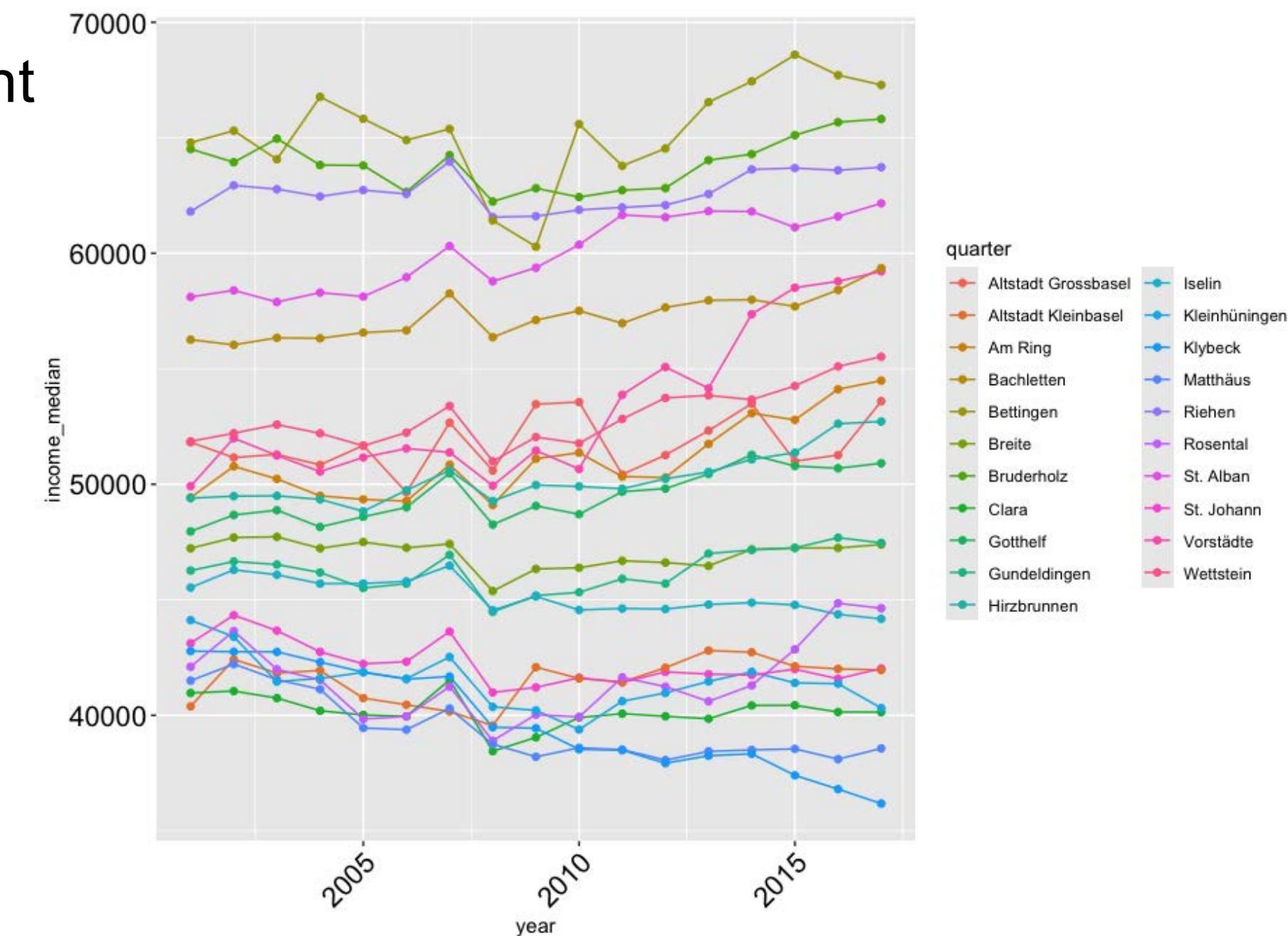


# theme\_\*

The `element_text()` function is used to customize the appearance of **text-based elements**

It provides control over the size, color, alignment, font style, and more.

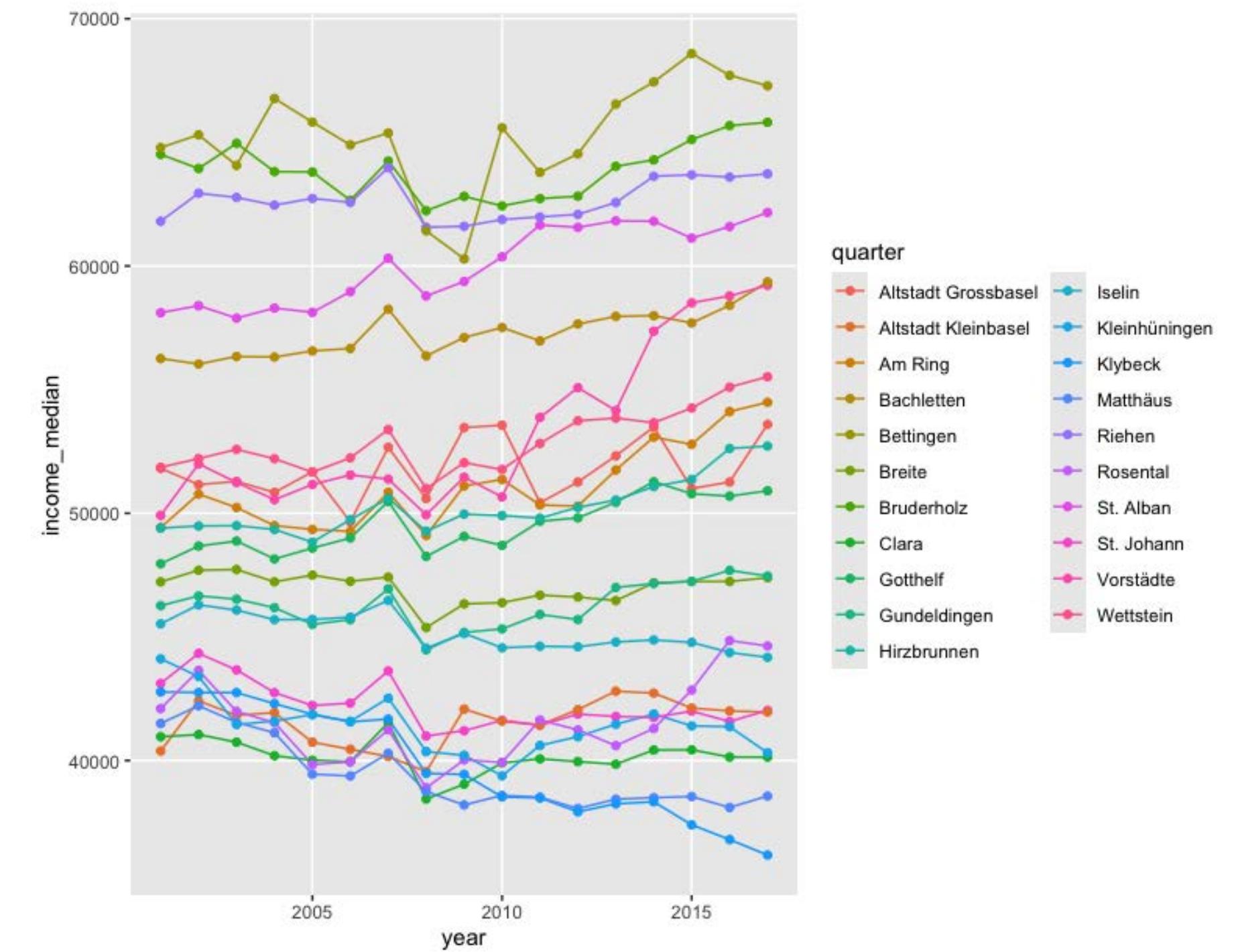
```
# element_text  
myplot +  
  theme (axis.text.x = element_text  
         (size = 15, angle = 45, hjust = 1,  
          color = "black"),  
        axis.text.y = element_text  
         (size = 15, color = "black"))
```



# theme\_\*

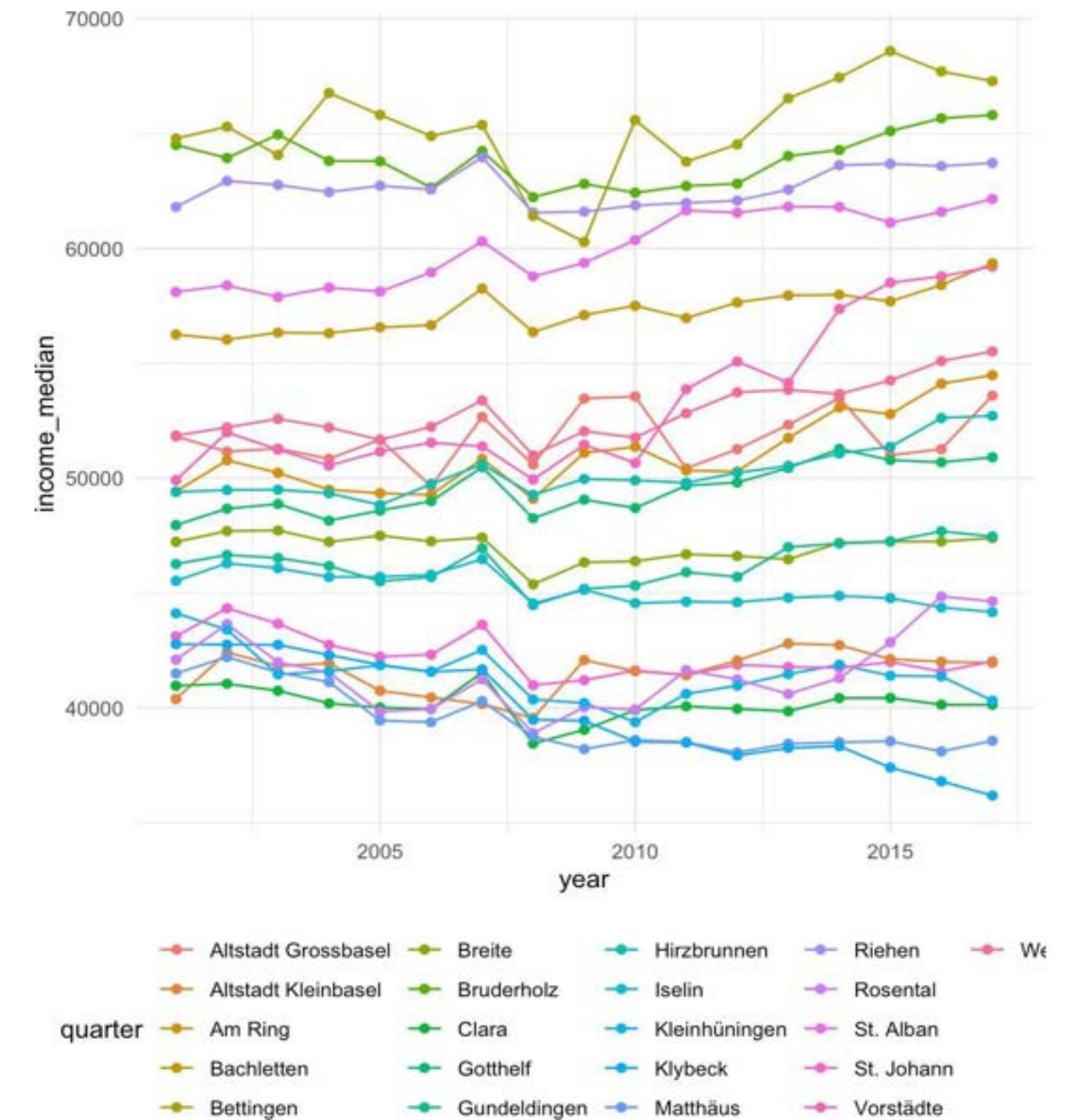
The `element_blank()` function is used to completely **remove or hide graphical elements** from a plot.

```
# element_blank  
myplot +  
  theme (panel.grid.minor =  
        element_blank() )
```



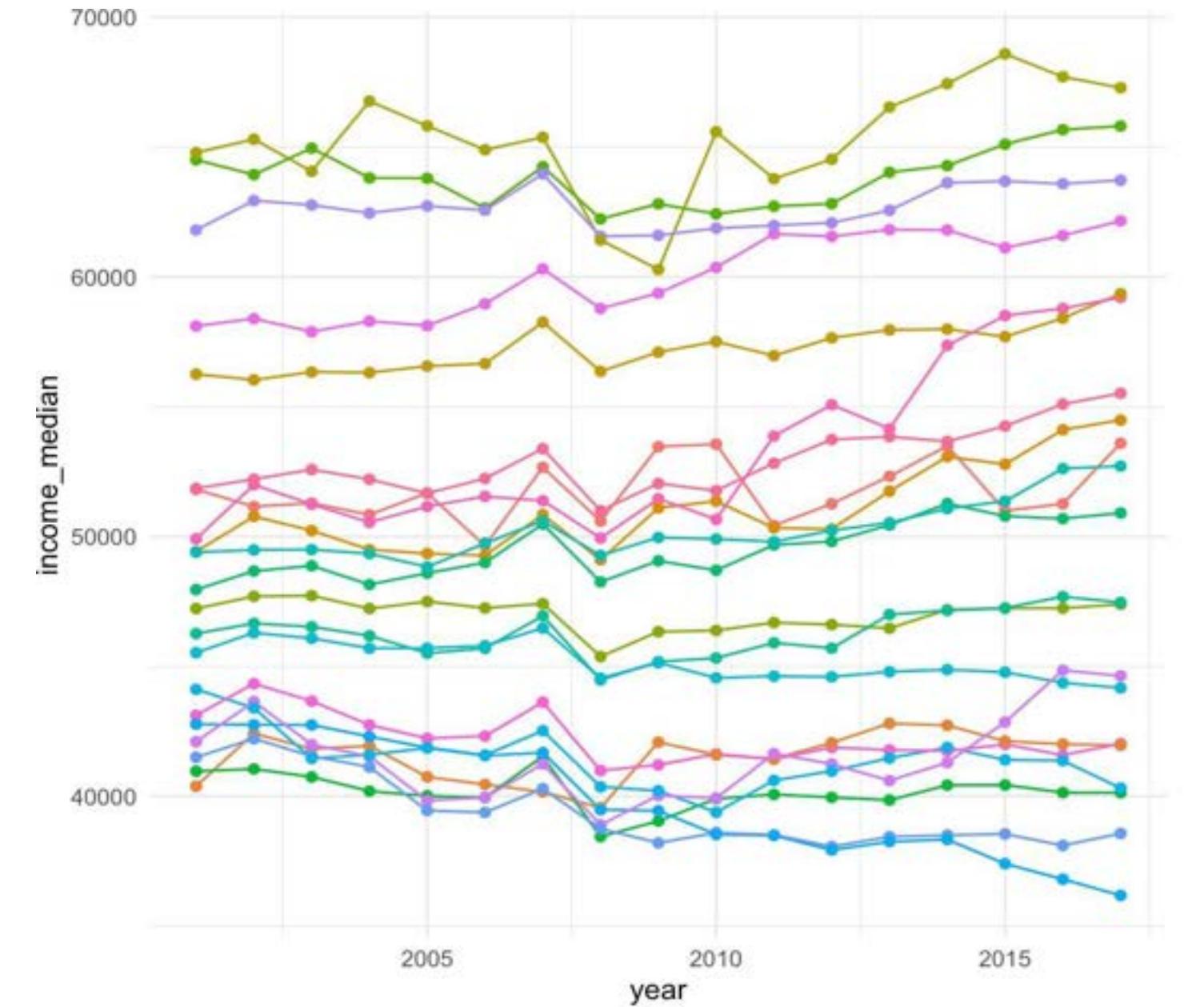
# theme\_\*

```
# Fixing the legend  
myplot +  
  theme_minimal() +  
  
# move legend to bottom  
theme(legend.position = "bottom")
```



# theme\_\*

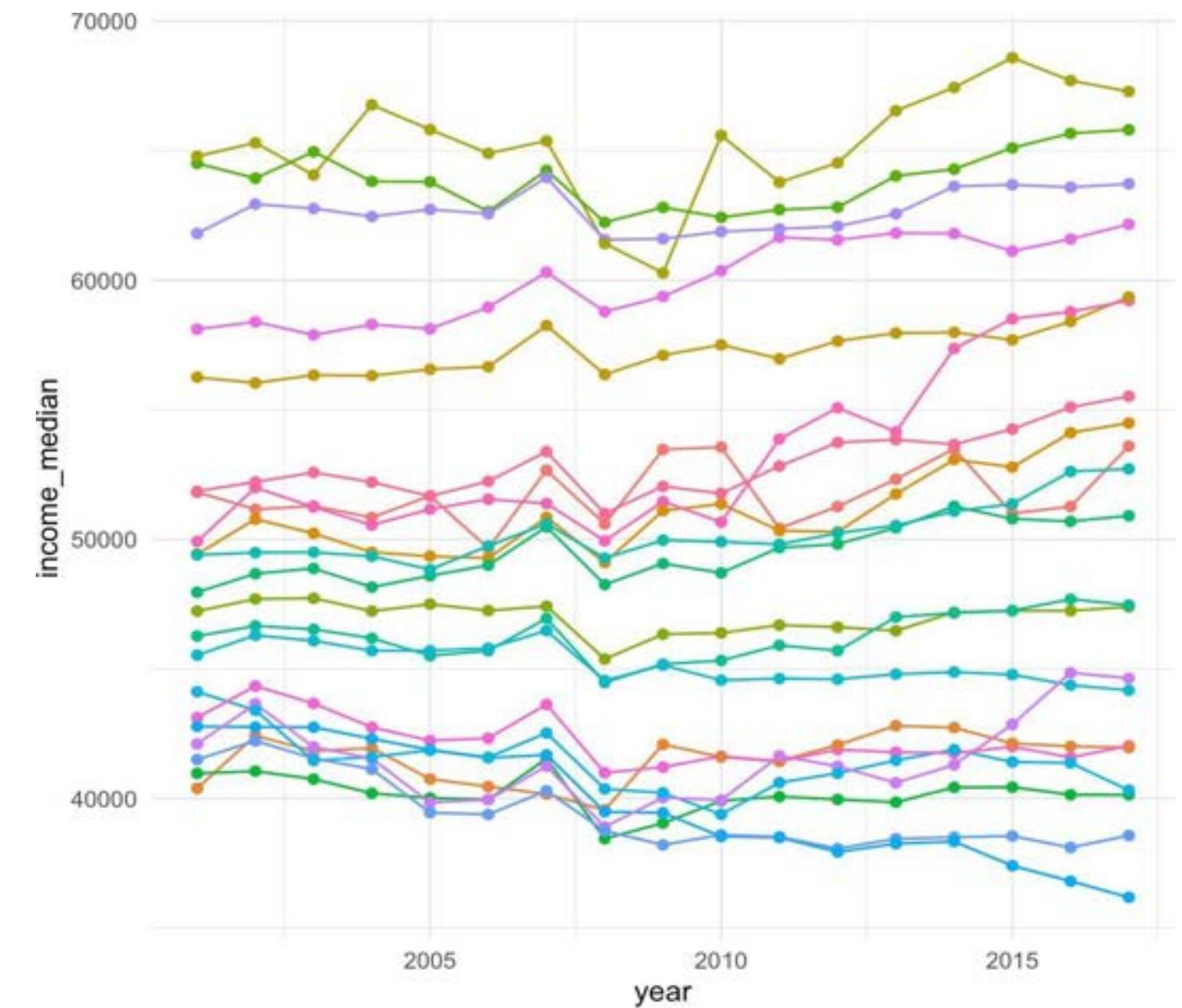
```
# Fixing the legend  
myplot +  
  theme_minimal() +  
  theme(legend.position = "bottom",  
  
    # remove legend title  
    legend.title = element_blank())
```



Altstadt Grossbasel	Breite	Hirzbrunnen	Riehen	Wettste
Altstadt Kleinbasel	Bruderholz	Iselin	Rosental	
Am Ring	Clara	Kleinhüningen	St. Alban	
Bachletten	Gotthelf	Klybeck	St. Johann	
Bettigen	Gundeldingen	Klybeck	Matthäus	Vorstädte

# theme\_\*

```
# Fixing the legend
my_plot +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    legend.title =
      element_blank(),
    # reduce legend text size
    legend.text = element_text(size = 7))
```



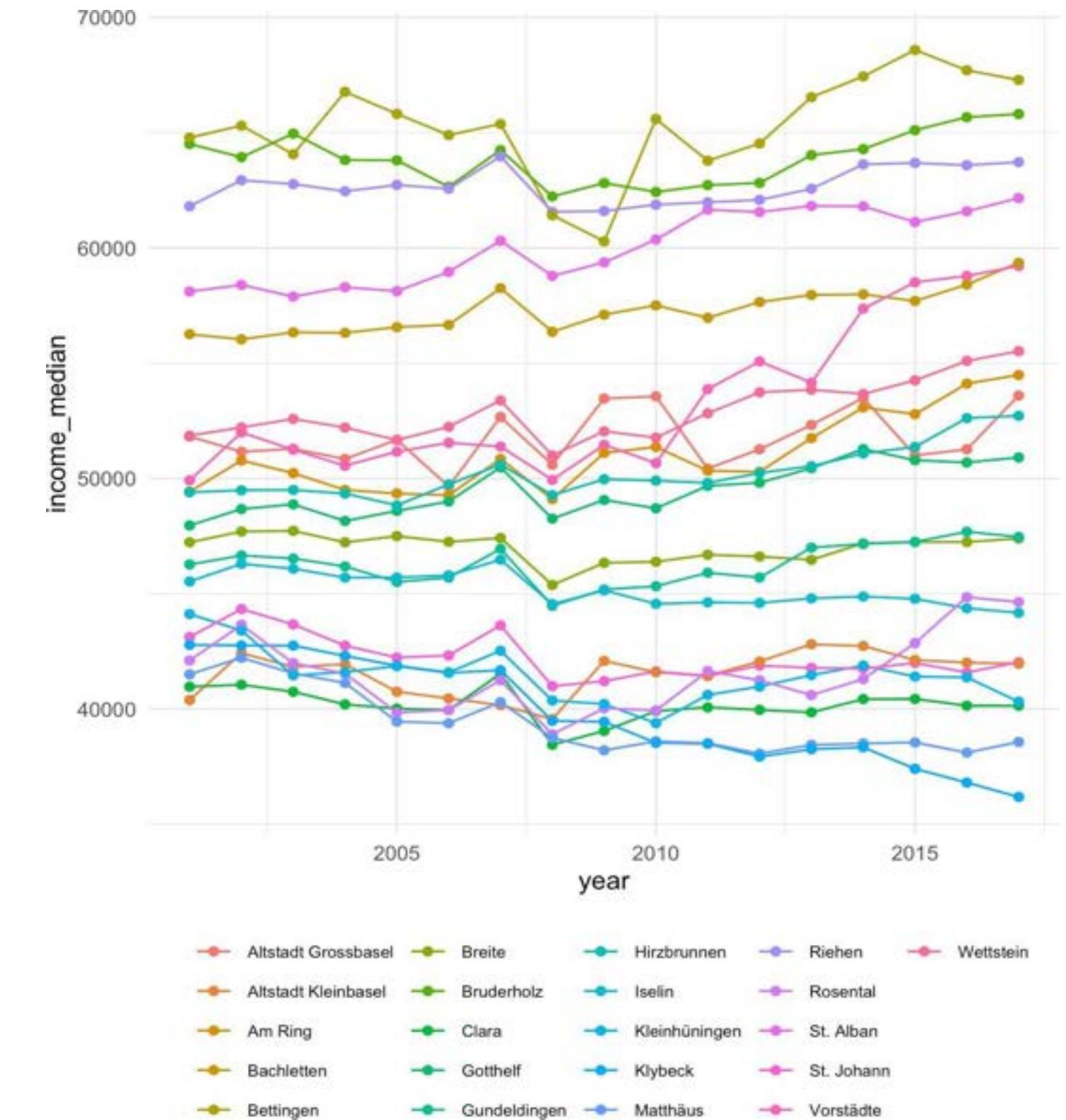
Altstadt Grossbasel	Breite	Hirzbrunnen	Riehen	Wettstein
41000	45000	42000	43000	44000
42000	46000	43000	44000	45000
43000	47000	44000	45000	46000
44000	48000	45000	46000	47000
45000	49000	46000	47000	48000
46000	50000	47000	48000	49000
47000	51000	48000	49000	50000
48000	52000	49000	50000	51000
49000	53000	50000	51000	52000
50000	54000	51000	52000	53000
51000	55000	52000	53000	54000
52000	56000	53000	54000	55000
53000	57000	54000	55000	56000
54000	58000	55000	56000	57000
55000	59000	56000	57000	58000
56000	60000	57000	58000	59000
57000	61000	58000	59000	60000
58000	62000	59000	60000	61000
59000	63000	60000	61000	62000
60000	64000	61000	62000	63000
61000	65000	62000	63000	64000
62000	66000	63000	64000	65000
63000	67000	64000	65000	66000
64000	68000	65000	66000	67000
65000	69000	66000	67000	68000
66000	70000	67000	68000	69000
67000	69000	68000	69000	70000
68000	68000	69000	70000	71000
69000	67000	70000	71000	72000
70000	66000	71000	72000	73000
71000	65000	72000	73000	74000
72000	64000	73000	74000	75000
73000	63000	74000	75000	76000
74000	62000	75000	76000	77000
75000	61000	76000	77000	78000
76000	60000	77000	78000	79000
77000	59000	78000	79000	80000
78000	58000	79000	80000	81000
79000	57000	80000	81000	82000
80000	56000	81000	82000	83000
81000	55000	82000	83000	84000
82000	54000	83000	84000	85000
83000	53000	84000	85000	86000
84000	52000	85000	86000	87000
85000	51000	86000	87000	88000
86000	50000	87000	88000	89000
87000	49000	88000	89000	90000
88000	48000	89000	90000	91000
89000	47000	90000	91000	92000
90000	46000	91000	92000	93000
91000	45000	92000	93000	94000
92000	44000	93000	94000	95000
93000	43000	94000	95000	96000
94000	42000	95000	96000	97000
95000	41000	96000	97000	98000
96000	40000	97000	98000	99000
97000	39000	98000	99000	100000
98000	38000	99000	100000	101000
99000	37000	100000	101000	102000
100000	36000	101000	102000	103000
101000	35000	102000	103000	104000
102000	34000	103000	104000	105000
103000	33000	104000	105000	106000
104000	32000	105000	106000	107000
105000	31000	106000	107000	108000
106000	30000	107000	108000	109000
107000	29000	108000	109000	110000
108000	28000	109000	110000	111000
109000	27000	110000	111000	112000
110000	26000	111000	112000	113000
111000	25000	112000	113000	114000
112000	24000	113000	114000	115000
113000	23000	114000	115000	116000
114000	22000	115000	116000	117000
115000	21000	116000	117000	118000
116000	20000	117000	118000	119000
117000	19000	118000	119000	120000
118000	18000	119000	120000	121000
119000	17000	120000	121000	122000
120000	16000	121000	122000	123000
121000	15000	122000	123000	124000
122000	14000	123000	124000	125000
123000	13000	124000	125000	126000
124000	12000	125000	126000	127000
125000	11000	126000	127000	128000
126000	10000	127000	128000	129000
127000	9000	128000	129000	130000
128000	8000	129000	130000	131000
129000	7000	130000	131000	132000
130000	6000	131000	132000	133000
131000	5000	132000	133000	134000
132000	4000	133000	134000	135000
133000	3000	134000	135000	136000
134000	2000	135000	136000	137000
135000	1000	136000	137000	138000
136000	0	137000	138000	139000

# theme\_\*

We can also save our adjustments as an object  
and combine it with a preexisting theme.

```
# save my theme
my_theme <-
  theme(legend.position =
    "bottom", legend.title =
    element_blank(),
    legend.text = element_text(size = 7))

# Add my theme
my_plot +
  theme_minimal() +
  my_theme
```



# scale\_\*( )

Various `scale_*` functions permit specification of all **dimensions**, including axes, colors, sizes, etc.

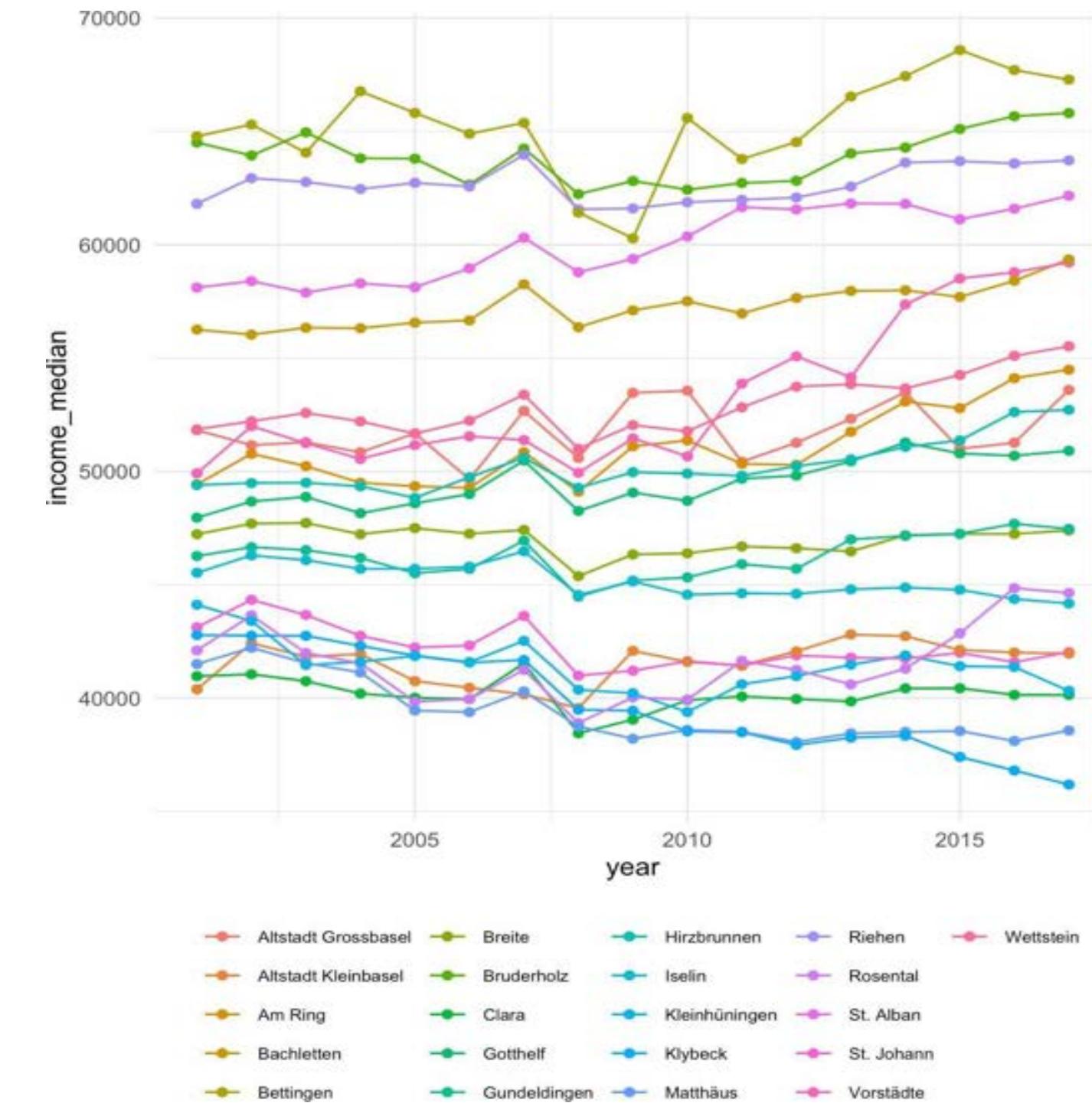
# Groups of scale \*() functions:

scale xy \* | scales axes

scale size \* | scales sizes

scale alpha \* | scales opacity

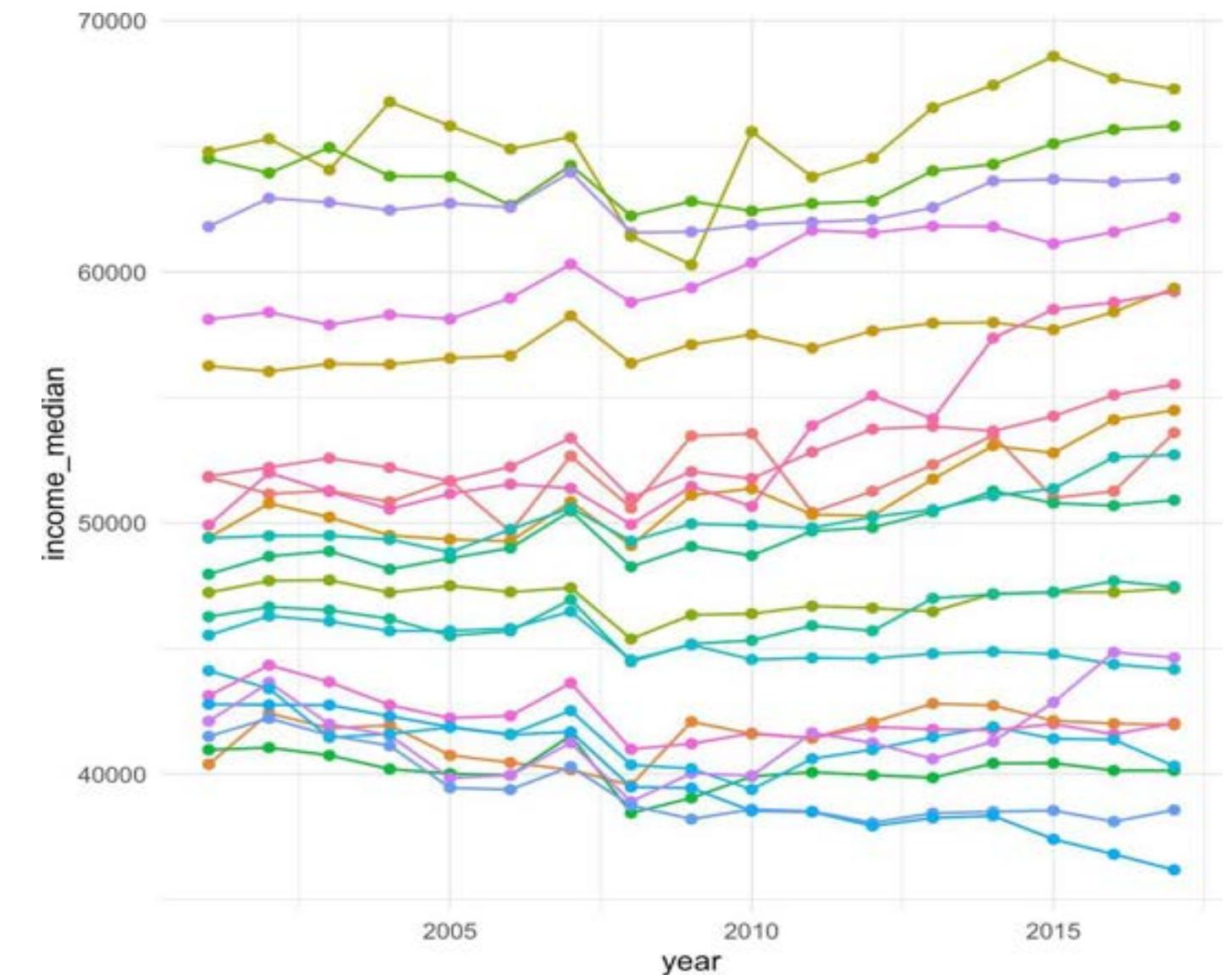
scale\_color \* | scales\_colors



# scale\_\*

scale\_xy\_\* can be used to set costum scales

```
# scale_xy  
myplot + scale_x_continuous(  
  limits = c(2001, 2017),  
  breaks = seq (2001, 2017, by = 3))
```

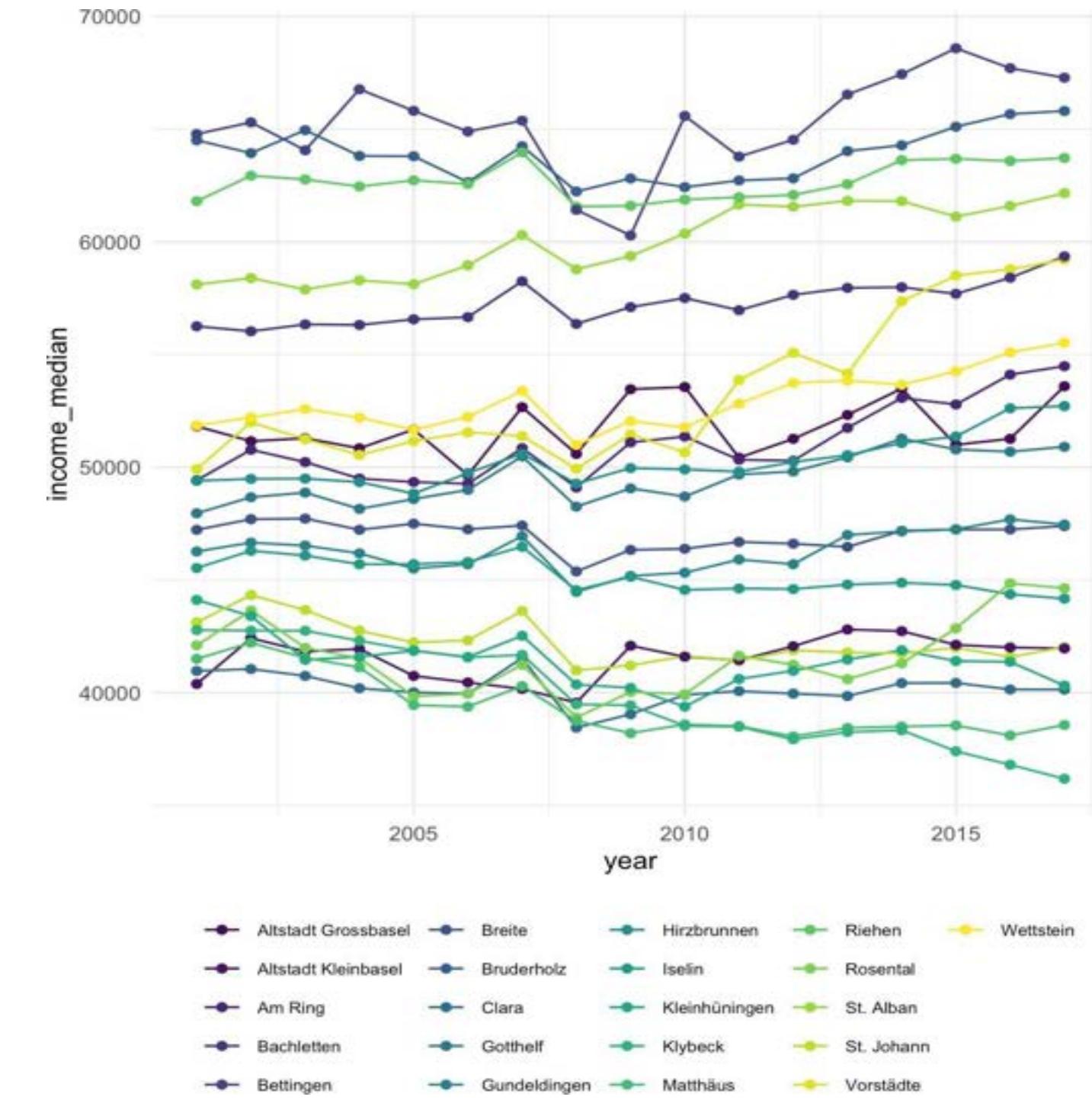


Altstadt Grossbasel	Breite	Hirzbrunnen	Riehen	Wettstein
Altstadt Kleinbasel	Bruderholz	Iselin	Rosental	
Am Ring	Clara	Kleinhüningen	St. Alban	
Bachletten	Gotthelf	Klybeck	St. Johann	
Bettingen	Gundeldingen	Mathäus	Vorstädte	

# scale\_\*

Various scale\_\*() functions permit specification of all **dimensions**, including axes, colors, sizes, etc.

```
# Fixing the legend  
my_plot +  
  theme_minimal() +  
  theme(  
    legend.position = "bottom",  
    legend.title = element_blank(),  
    legend.text = element_text(size=7) +  
  
# color using viridis  
  scale_color_viridis_d()
```



# Wrangling

Again, wrangling can help with plotting. The order of discrete variables can be controlled using **factors**.

```
# change data
basel_order <- data %>%
  # factor ordered by income in 2001
  arrange(year, income_median) %>%
  mutate(quarter = as_factor(quarter))

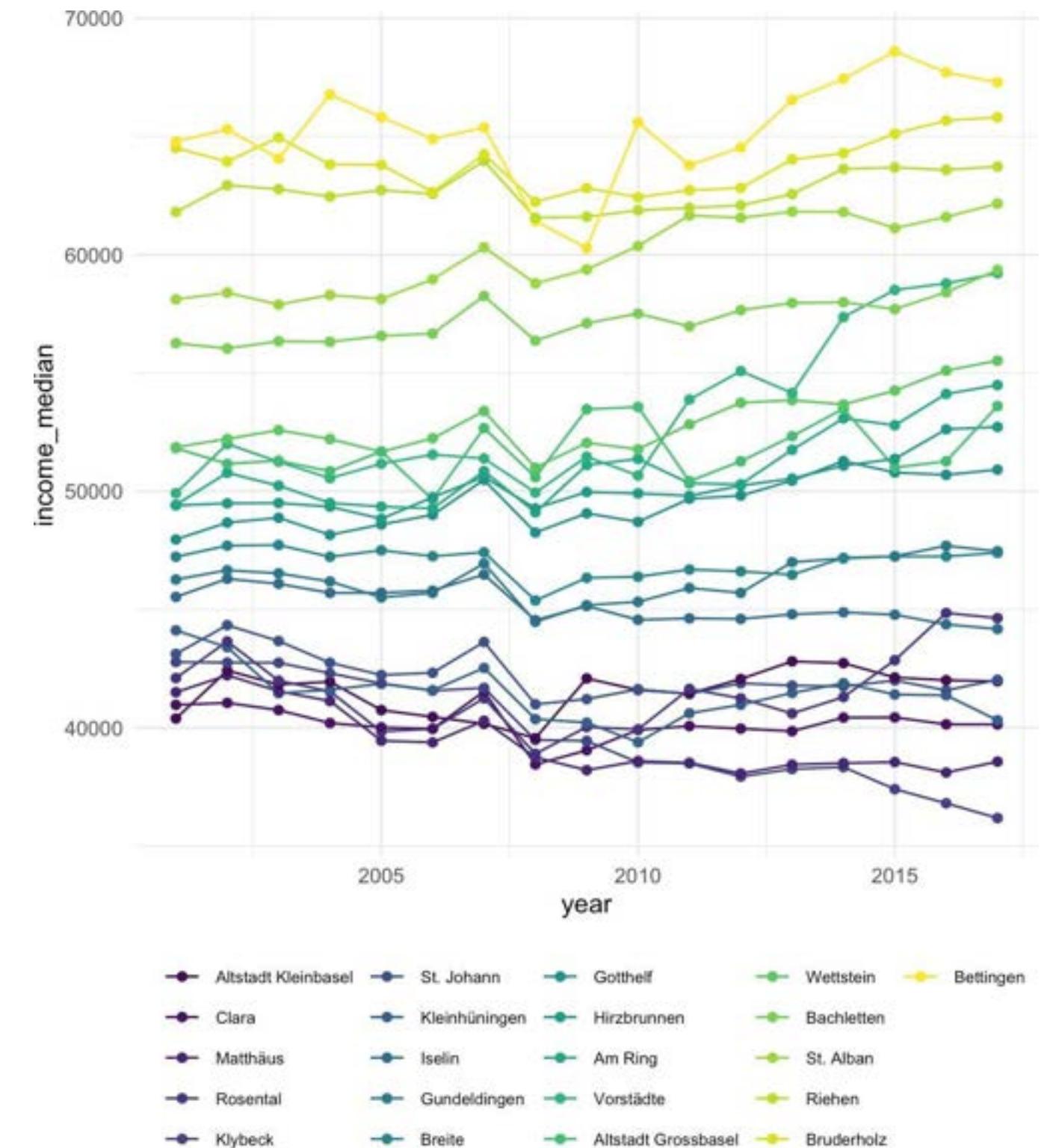
# show data
basel_order
```

	year	quarter	quarter_no	N	income_mean	income_median
	<dbl>	<fct>	<dbl>	<dbl>	<dbl>	<dbl>
1	2001	Altstadt Kleinbasel	12	1659	51648	40387
2	2001	Clara	13	2416	47435	40964
3	2001	Matthäus	17	9089	48892	41500
4	2001	Rosental	16	2499	46221	42100
5	2001	Klybeck	18	4053	45651	42777
6	2001	St. Johann	11	10493	48766	43118
7	2001	Kleinhüningen	19	1363	46859	44115
8	2001	Iselin	10	9853	49631	45530
9	2001	Gundeldingen	6	11224	51229	46265
10	2001	Breite	4	5433	52039	47227

# Wrangling

Colors are assigned according to the order of factor levels

```
basel_order %>%  
  
# original code  
ggplot(aes(x = year, y = income_median,  
col = quarter)) +  
  geom_line() + geom_point() +  
  theme_minimal() +  
  theme(legend.position = "bottom",  
    legend.title = element_blank(),  
    legend.text = element_text(size=7)) +  
  
# ordered implicitly  
  scale_color_viridis_d(option = "viridis")
```

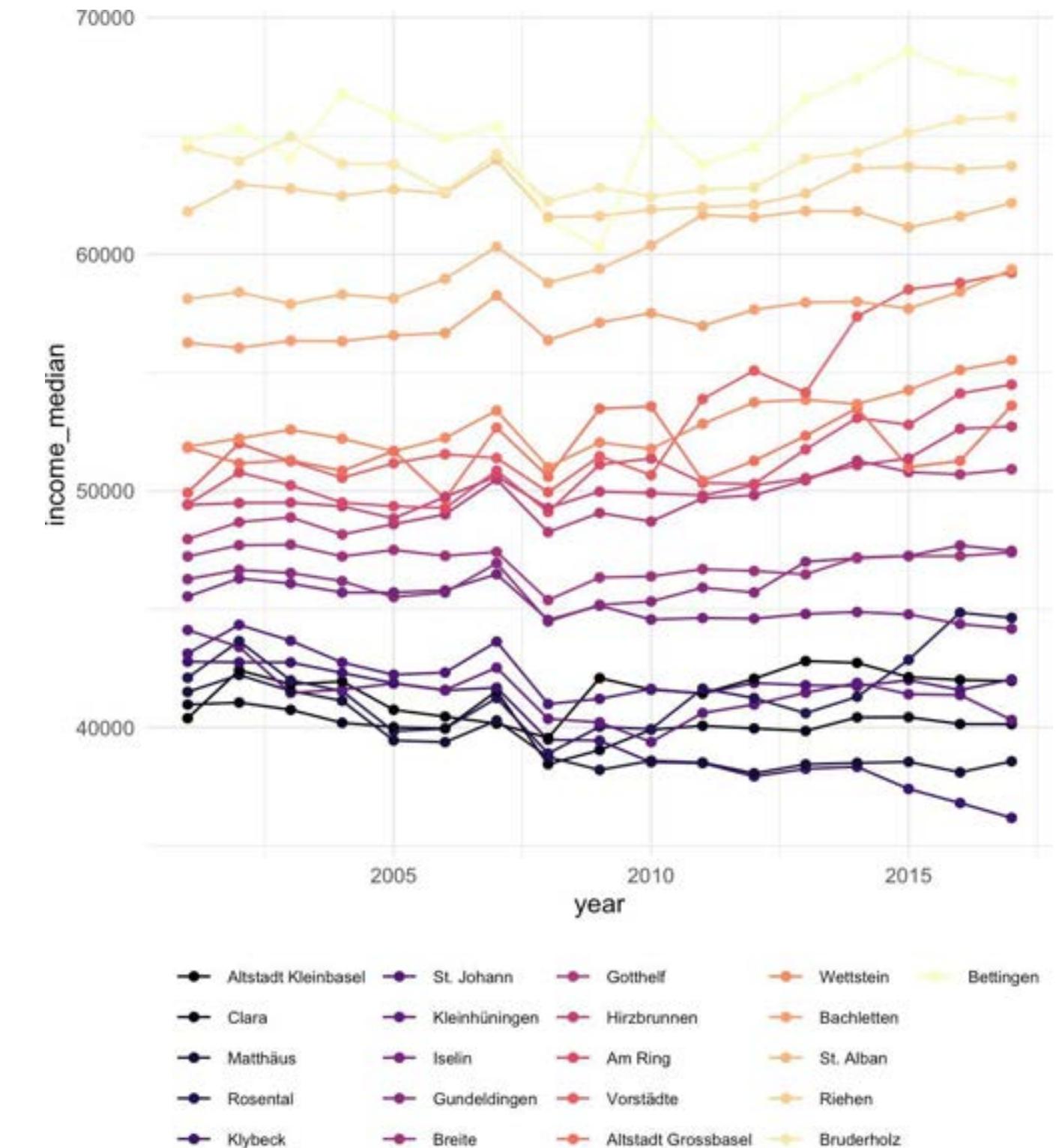




# Viridis

viridis includes several preset color sets that can **useful for different scenarios.**

```
basel_order %>%  
  
# original code  
ggplot(aes(x = year, y = income_median,  
col = quarter)) +  
  geom_line() + geom_point() +  
  theme_minimal() +  
  theme(legend.position = "bottom",  
legend.title = element_blank(),  
legend.text = element_text(size=7)) +  
  
# ordered implicitly  
  scale_color_viridis_d(option = "plasma")
```



# Labs ()

Various annotations can be added using `labs()`.

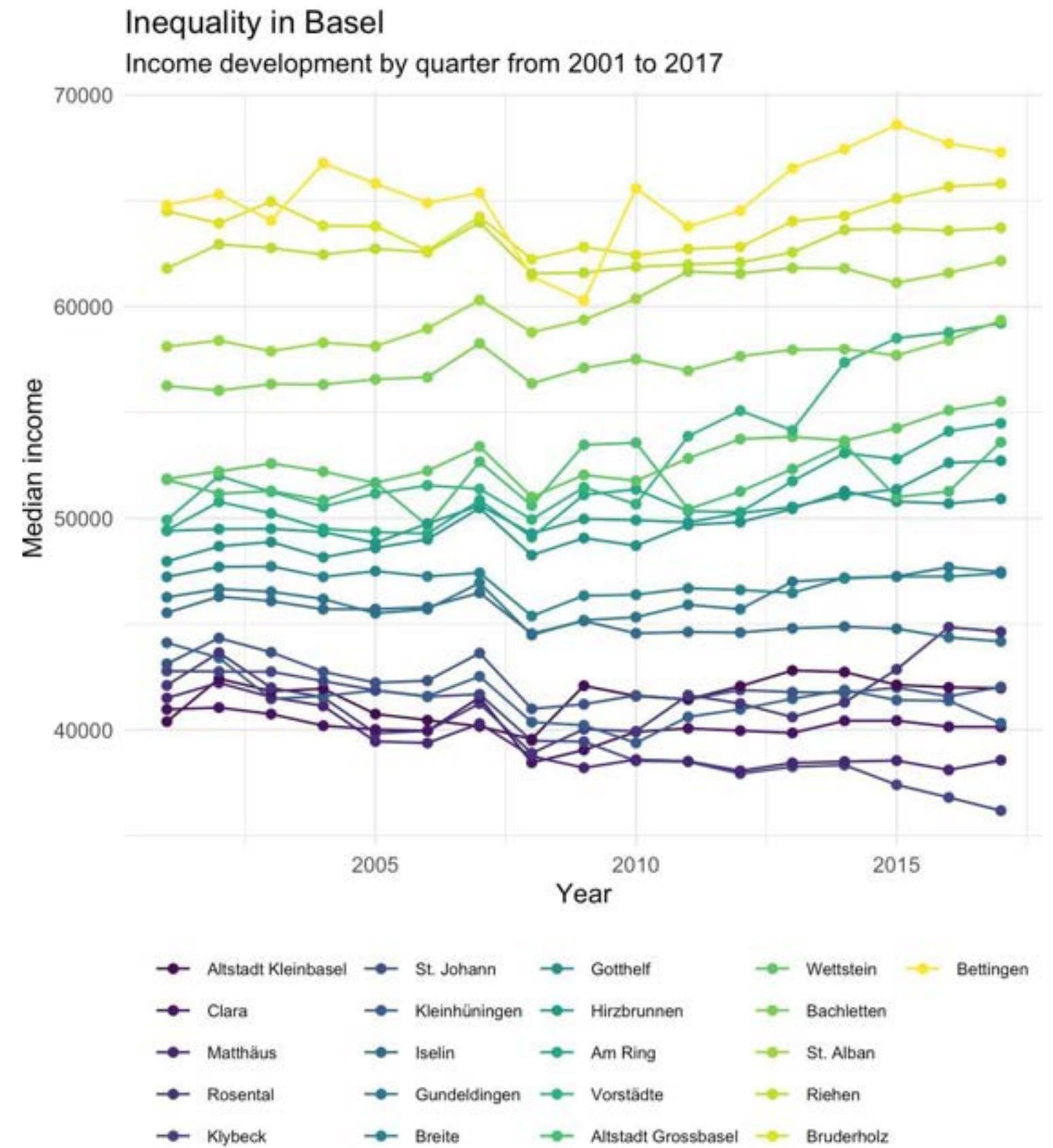
Key arguments:

`x, y | axes`

`title, subtitle | title and subtitle`

`caption | caption`

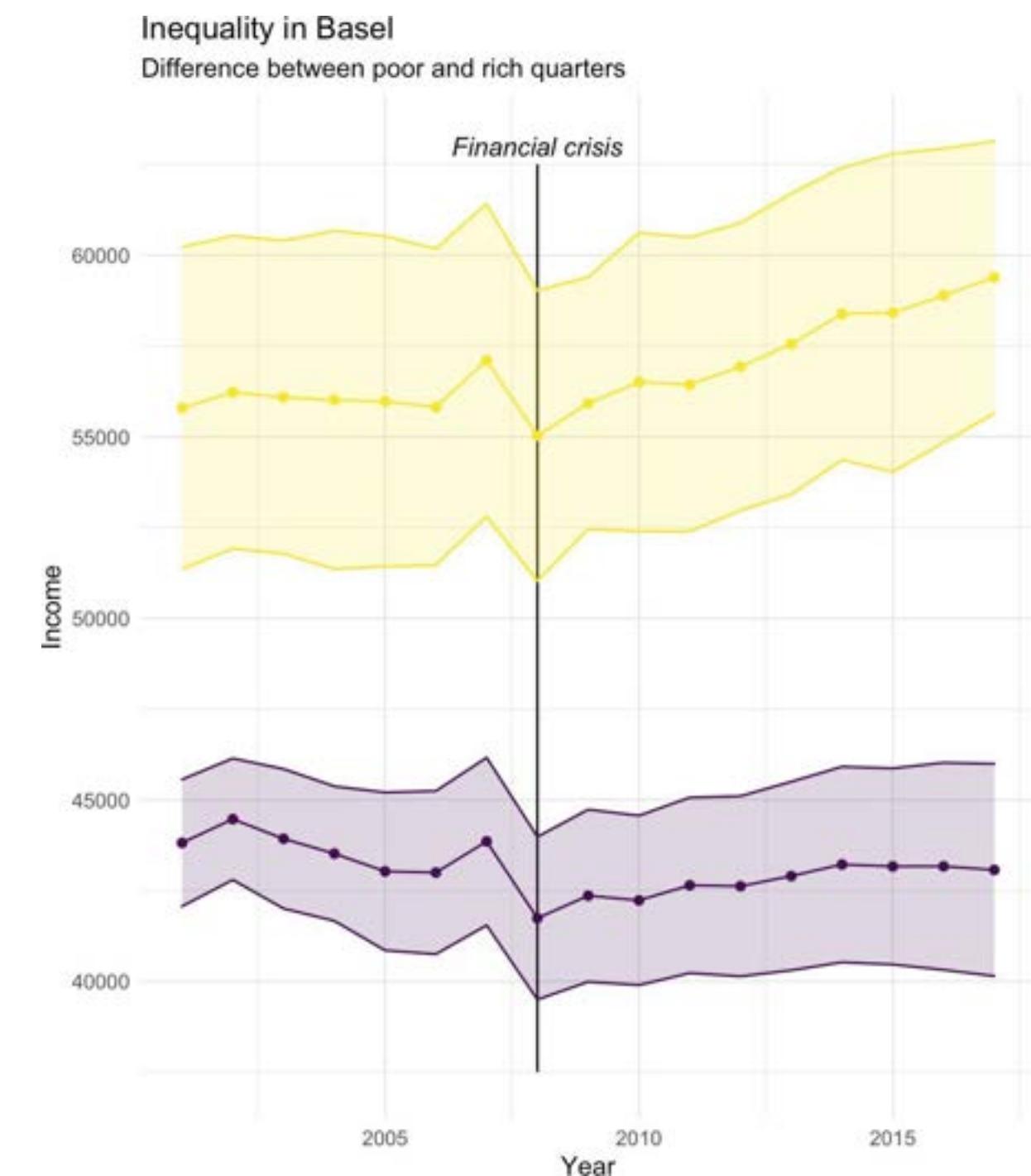
```
my_plot +  
  labs(x = "Year",  
       y = "Median income",  
       title = "Inequality in Basel",  
       subtitle = "Income development...",  
       caption = "Source: Open Data...")
```



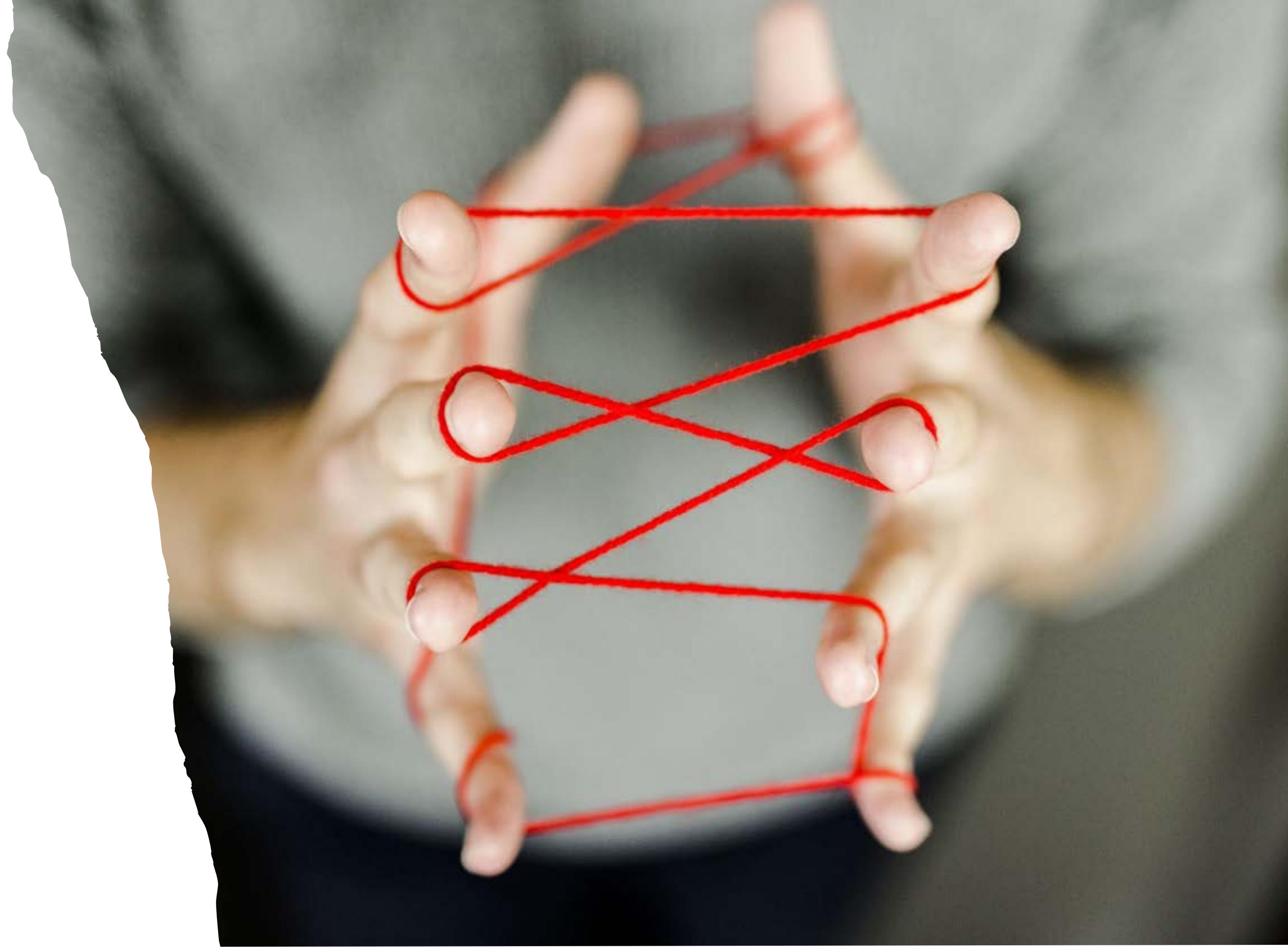
# annotate ()

annotate() can be used to **add any type of plot elements** (layers) using vector specification.

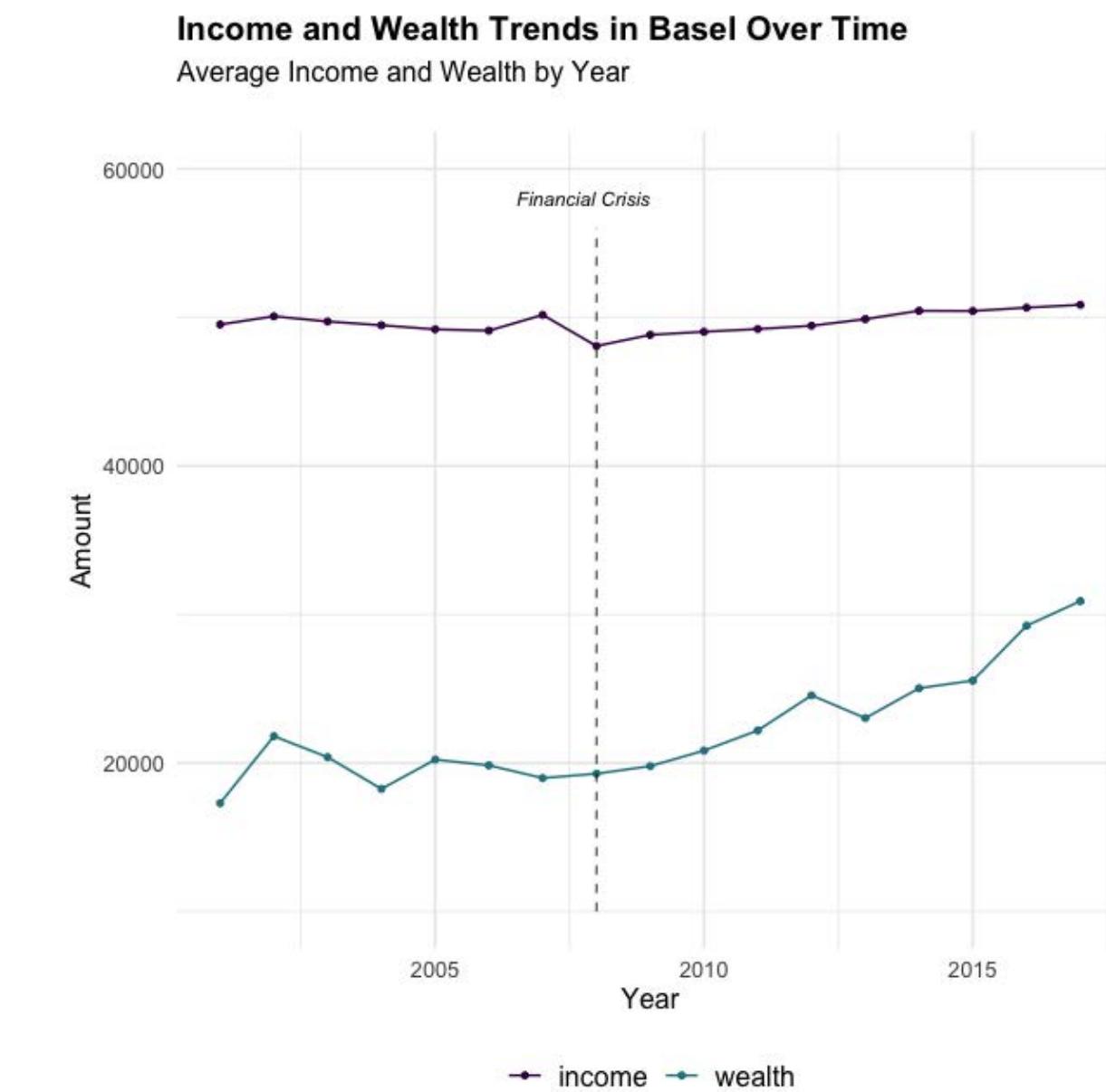
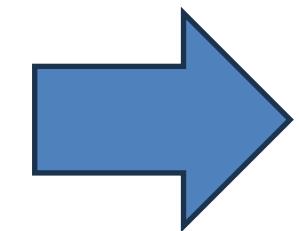
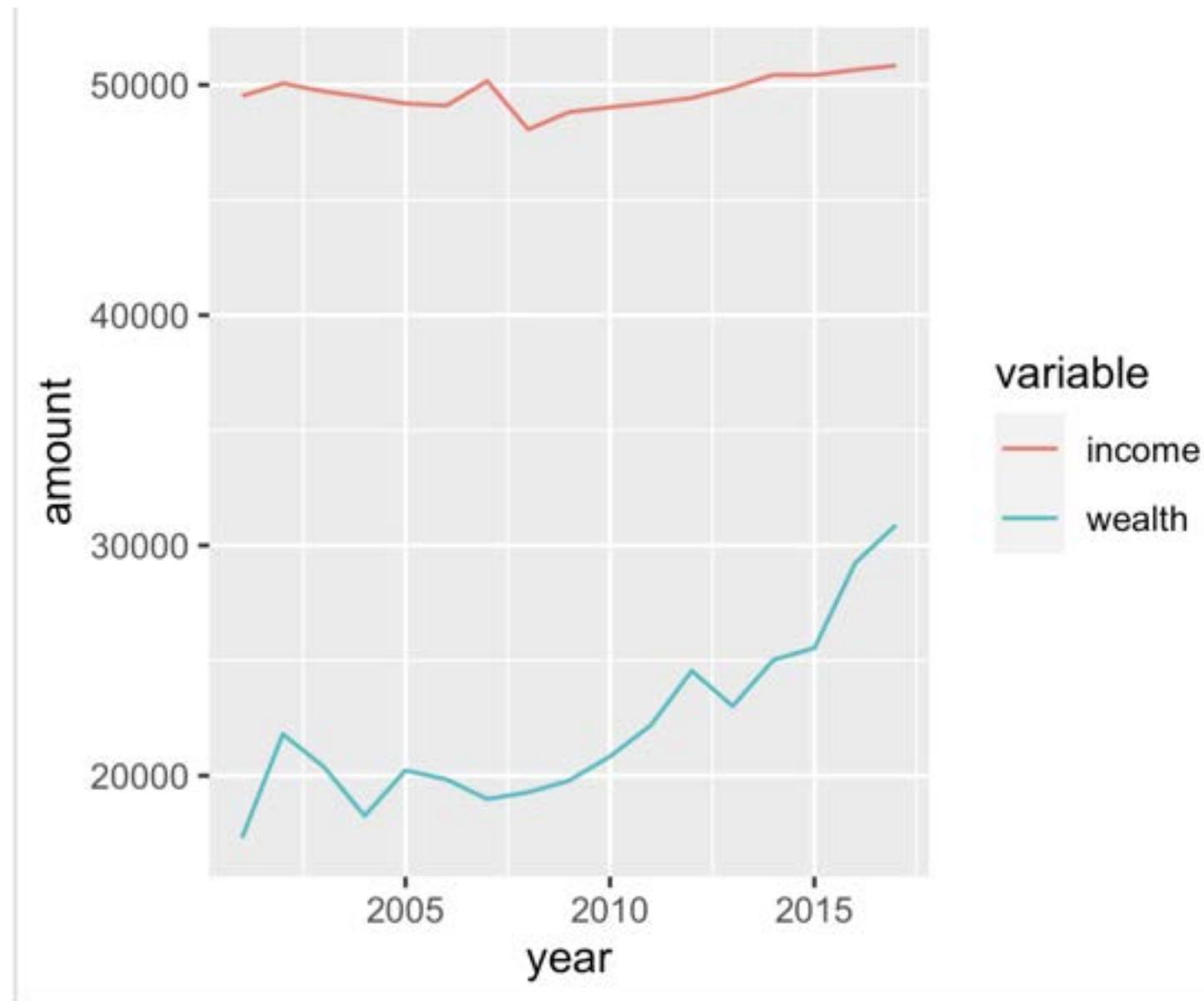
```
... +
# annotation
  labs(title = "Inequality in Basel", subtitle =
"Difference between...", x = "Year",
       y = "Income") +
  annotate('line',
           x = 2009, y = c(40000, 92000)) +
  annotate('text',
           x = 2005, y = 95000,
           label = "Financial crisis", fontface =
"italic")
```



Practice

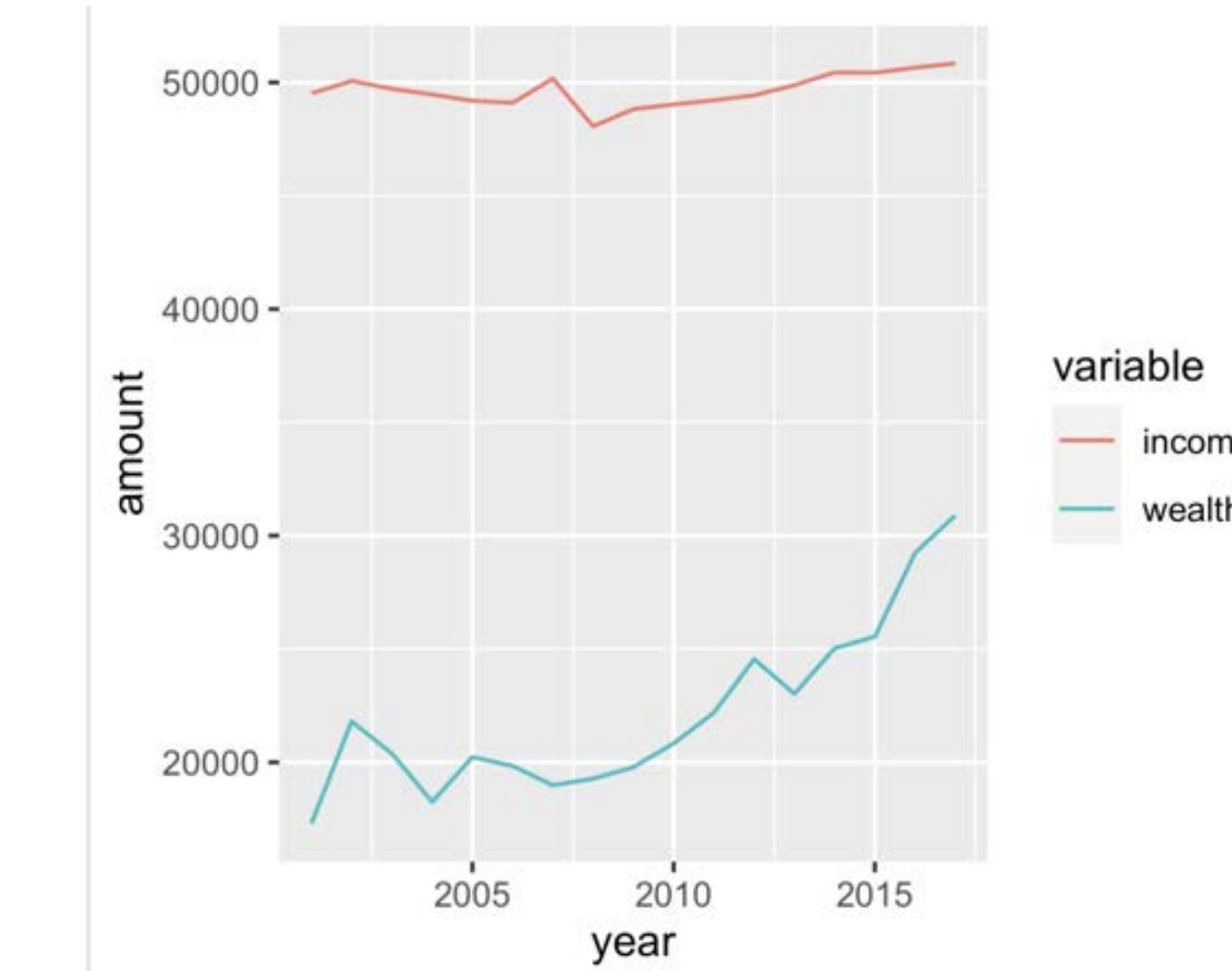


# Let's get from the left plot the right one



Practice:  
<https://ggplot2.tidyverse.org/reference/theme.html>

```
base %>%
  group_by(year) %>%
  summarize(income = mean(income_median),
            wealth = mean(wealth_median)) %>%
  pivot_longer(c(income, wealth),
               names_to = "variable",
               values_to = "amount") %>%
  ggplot(aes(x = year, y = amount, color = variable)) +
  geom_line()
```



```
# add points to the lines. Change line size to 0.5 and point size to 1
```

```
# assign color "#440154" to the data for income and "#26828E" to wealth
```

```
# Scale the y axes from 10000 to 60000
```

```
# add an annotation at x = 2009 to mark the financial crisis
```

```
# add a title, a subtitle and insert meaningful axis titles
```

```
# attach theme_minimal to the plot
```

```
# create margins on each side of the plot
```

```
# make the title of the plot bold and create a margin between the  
subtitle and the plot
```

```
# place the legend at the bottom of the plot, remove the legend title  
and set the font size to 12
```

```
# Add standard errors to the lines
```

# Exam Questions

- If I were to give you a line of code, could you select the correct answer explaining what the code does?