Styling with ggplot2

Data Visualization for Social Good

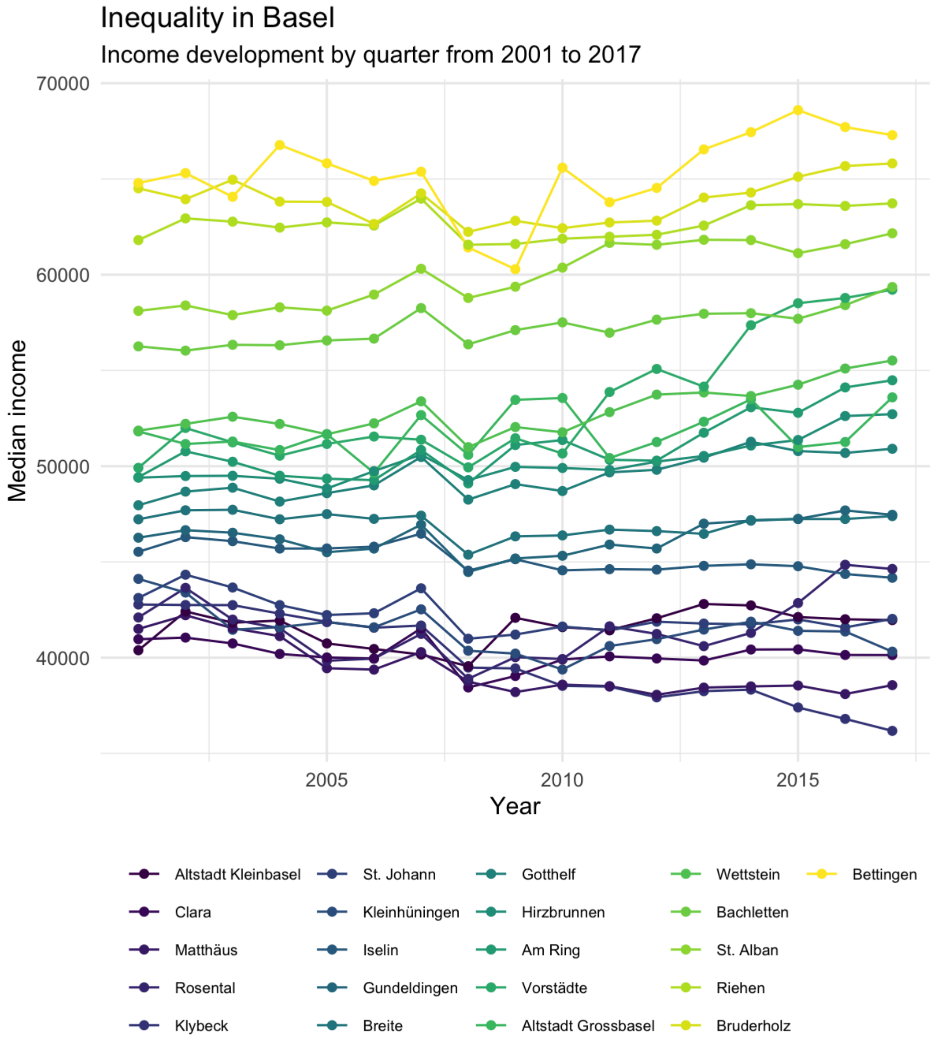
CorrelAid Switzerland

🕒 🏠 ✉ in

February 2021

Styling plots

1. Save plot as `gg` object.
2. Use existing `theme_*()` presets.
3. Customize details using `theme()`.
4. Adjust dimensions using `scale_*()`.
5. Add annotion using `labs()`.
6. Write your plot as a `.pdf` or `.png`.


Inequality in Basel

Income development by quarter from 2001 to 2017

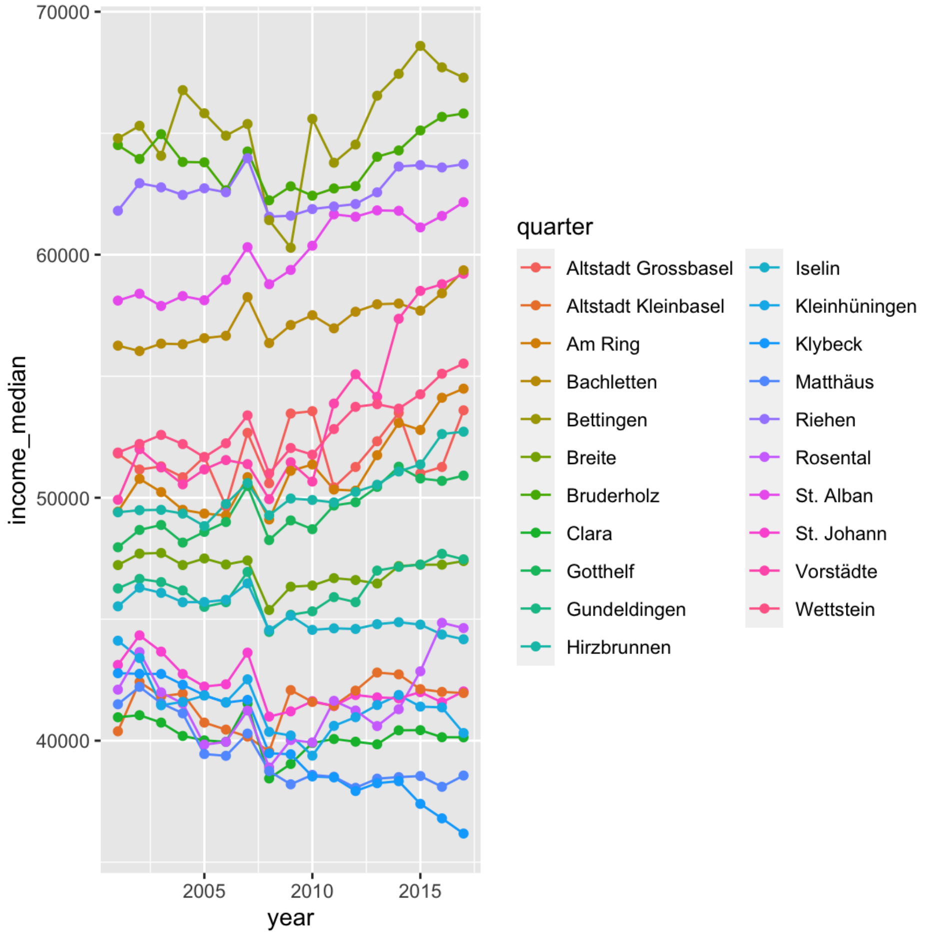
Source: Open Data Basel Stadt

the gg object

1. The output of `ggplot()` can be stored in an `gg` object.
2. The `gg` object can be expanded using + and the plot can be generated by a simple print.

```r
# store plot as object
my_plot <- ggplot(basel,
                  aes(x = year,
                  y = income_median,
                  col = quarter)) +
  geom_line() +
  geom_point()
```

```r
my_plot
```



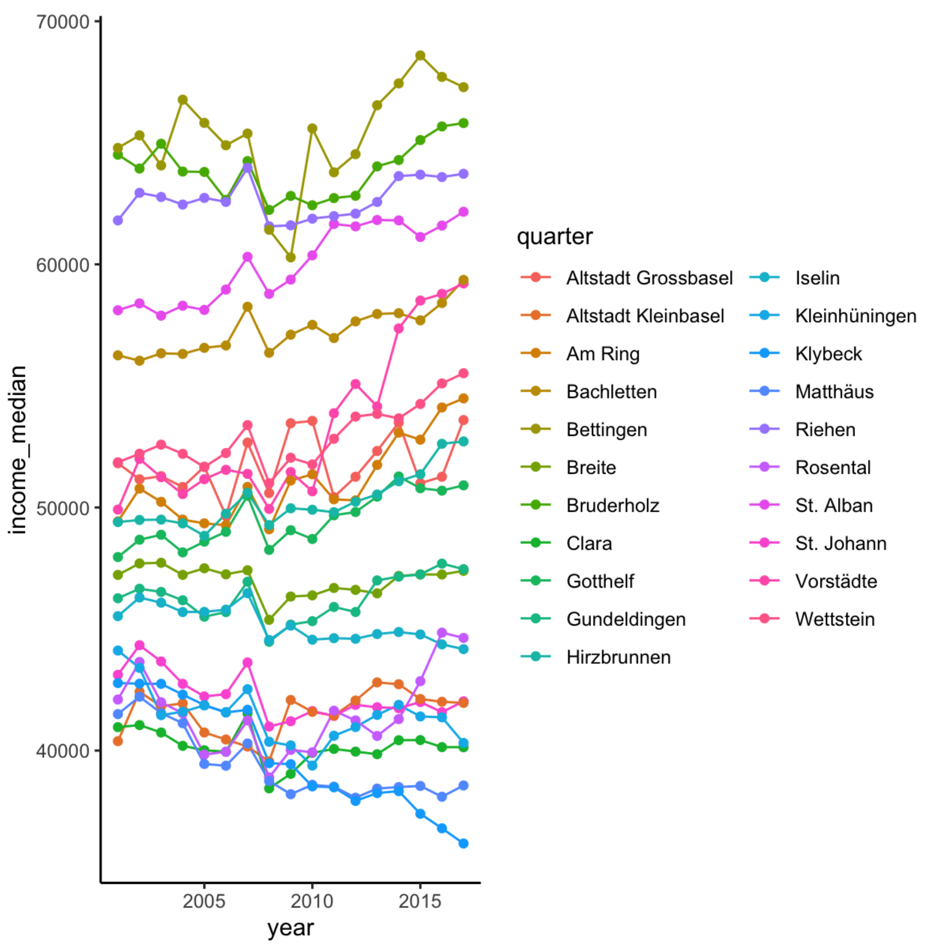
theme_*()

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

2. A few `theme`s:

   - `theme_gray()`
   - `theme_classic()`
   - `theme_void()`
   - `theme_minimal()`
   - `theme_excel() (ggthemes)`
   - `theme_economist() (ggthemes)`

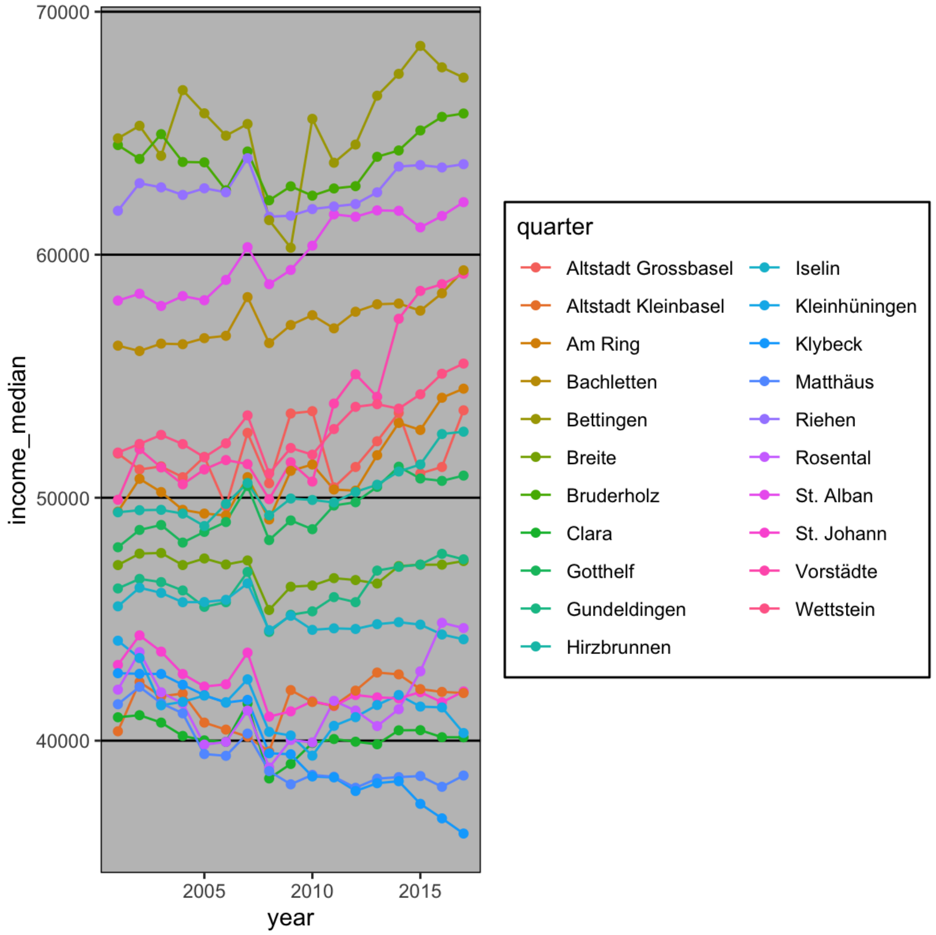
```
my_plot + theme_classic()
```



theme_*()

1. Using theme_*() the plot can be styled according to various presets.
2. A few themes:
   - theme_gray()
   - theme_classic()
   - theme_void()
   - theme_minimal()
   - theme_excel() (ggthemes)
   - theme_economist() (ggthemes)

```
my_plot + theme_excel()
```



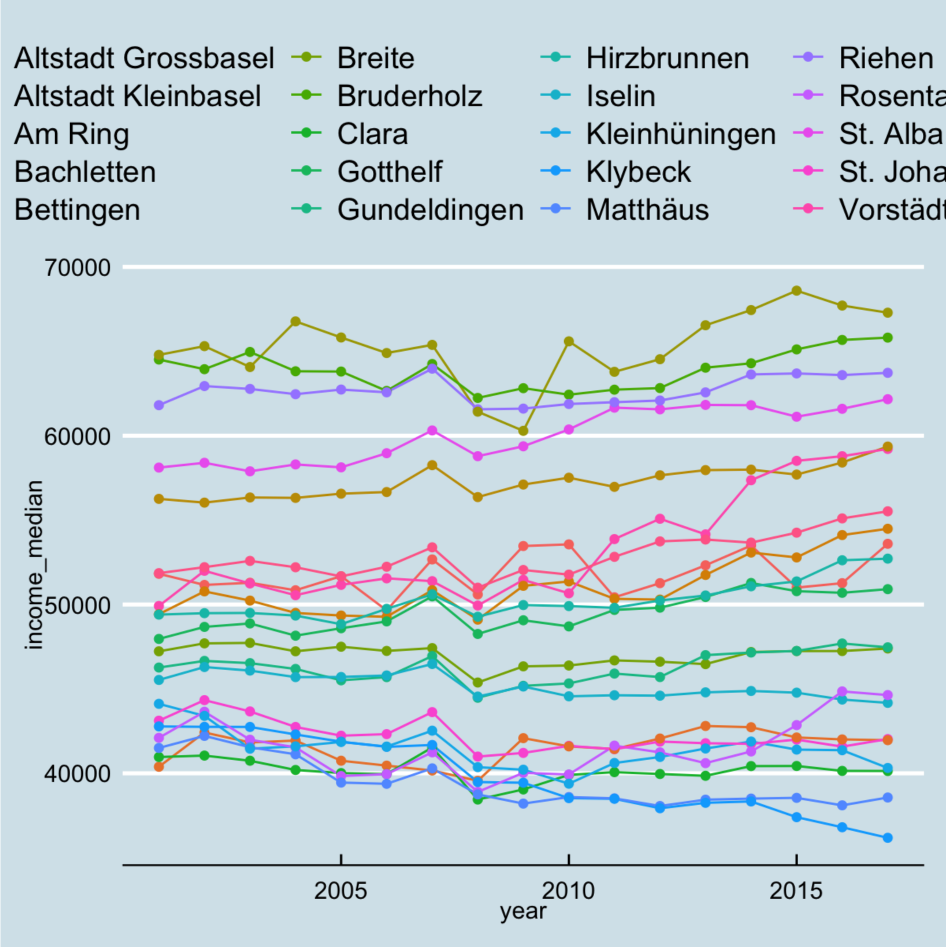
theme_*()

1. Using theme_*() the plot can be styled according to various presets.

2. A few themes:

   - theme_gray()
   - theme_classic()
   - theme_void()
   - theme_minimal()
   - theme_excel() (ggthemes)
   - theme_economist() (ggthemes)

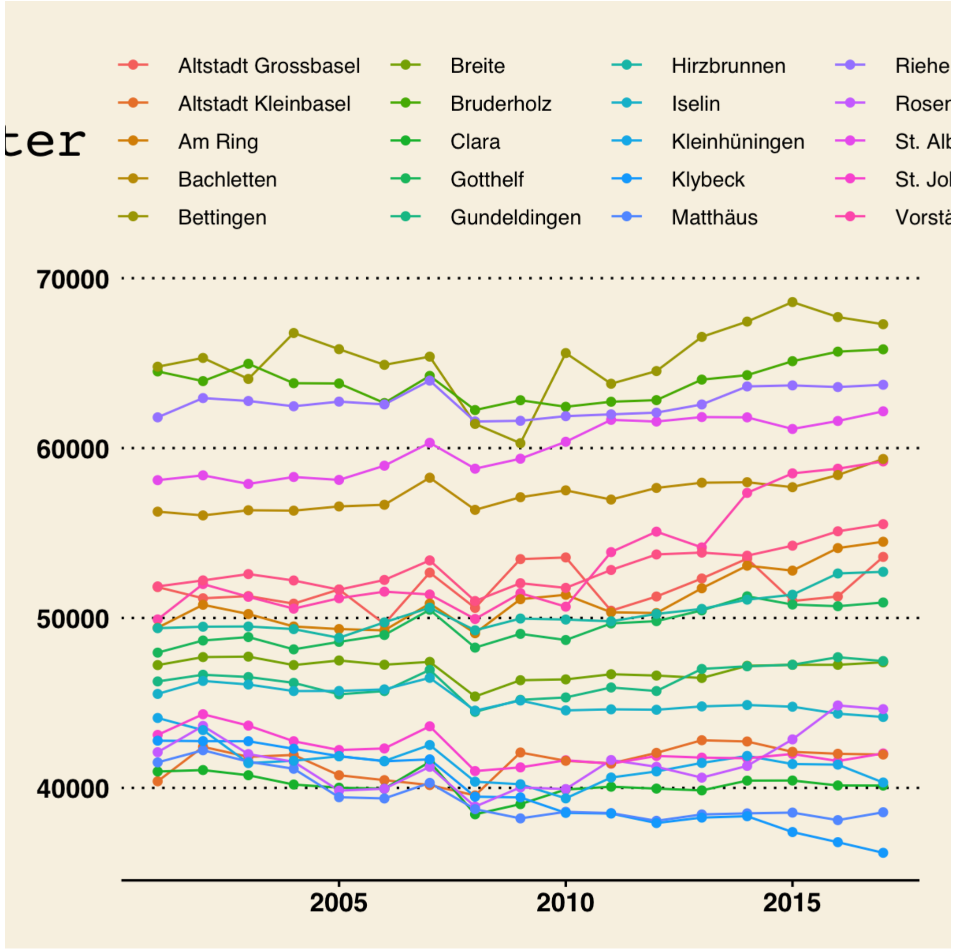
```
my_plot + theme_economist()
```



theme_*()

1. Using `theme_*()` the plot can be styled according to various presets.
2. A few `themes`:
    - `theme_gray()`
    - `theme_classic()`
    - `theme_void()`
    - `theme_minimal()`
    - `theme_excel() (ggthemes)`
    - `theme_economist() (ggthemes)`

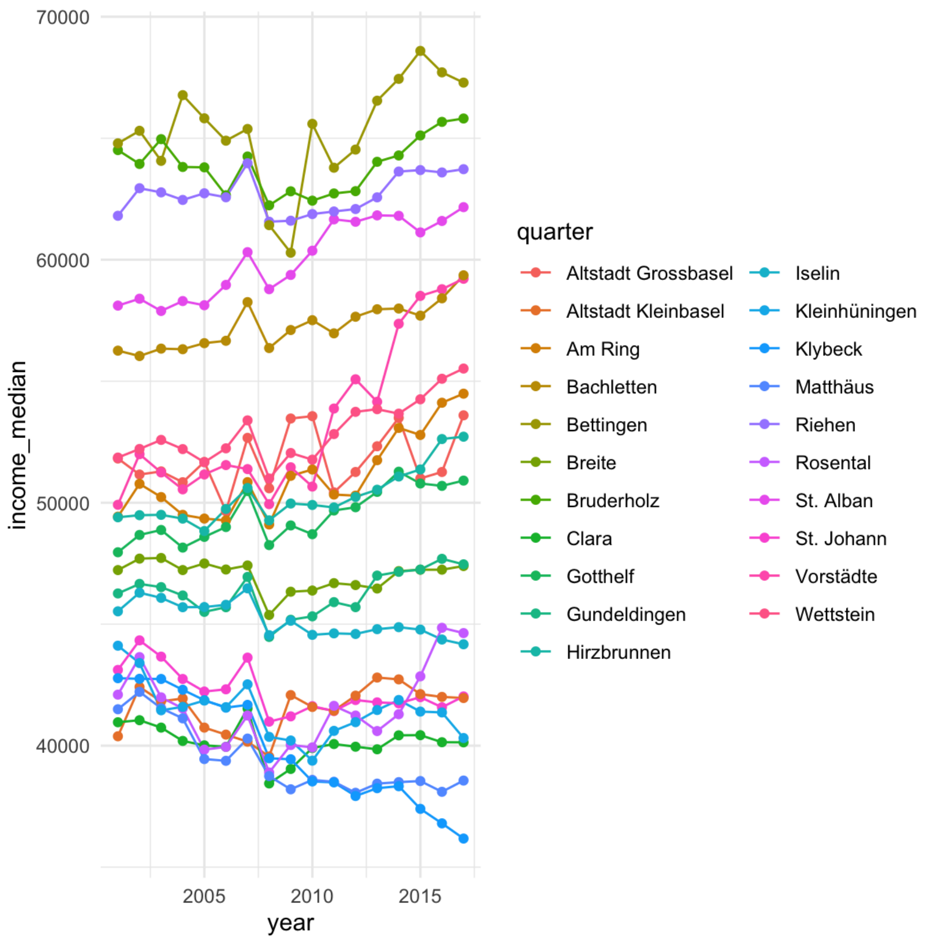
```
my_plot + theme_wsj()
```



theme_*()

1. Using `theme_*()` the plot can be styled according to various presets.
2. A few `themes`:
   - `theme_gray()`
   - `theme_classic()`
   - `theme_void()`
   - `theme_minimal()`
   - `theme_excel() (ggthemes)`
   - `theme_economist() (ggthemes)`

```
my_plot + theme_minimal()
```

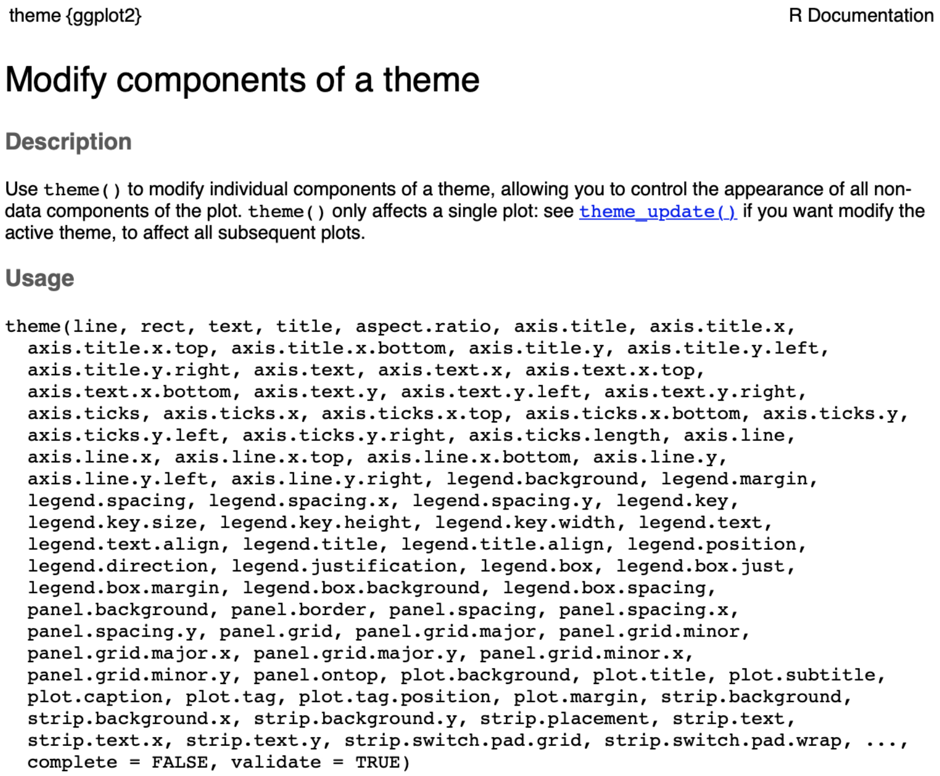


theme()

1. With **87 arguments** `theme()` permits specification of all aesthetic details.

2. Makes uses 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.

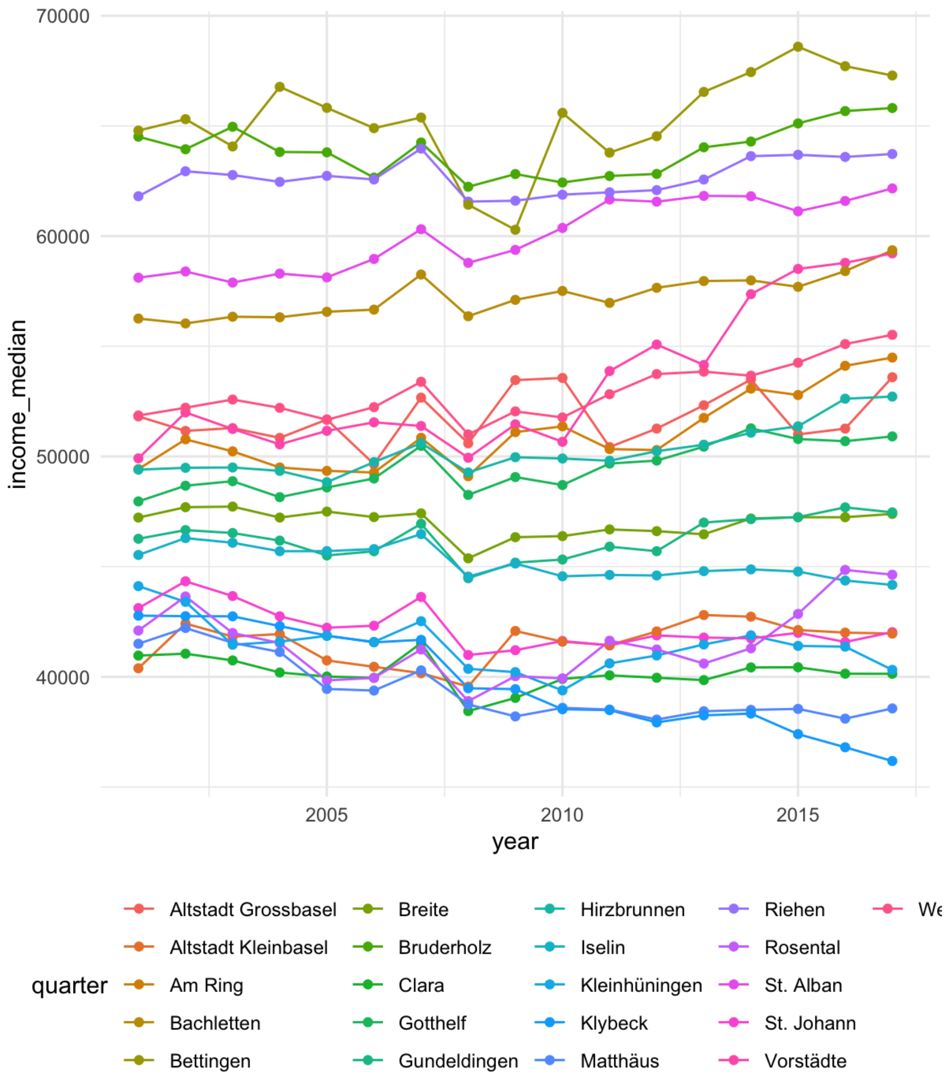
Usage

```
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.ontop, 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()

1. With **87 arguments** `theme()` permits specification of all aesthetic details.
2. Makes uses of helper functions:
   - `element_rect()` | for rectangles
   - `element_line()` | for lines
   - `element_text()` | for text
   - `element_blank()` | for removals

```r
# Fixing the legend
my_plot +
  theme_minimal() +

  # move legend to bottom
  theme(legend.position = "bottom")
```

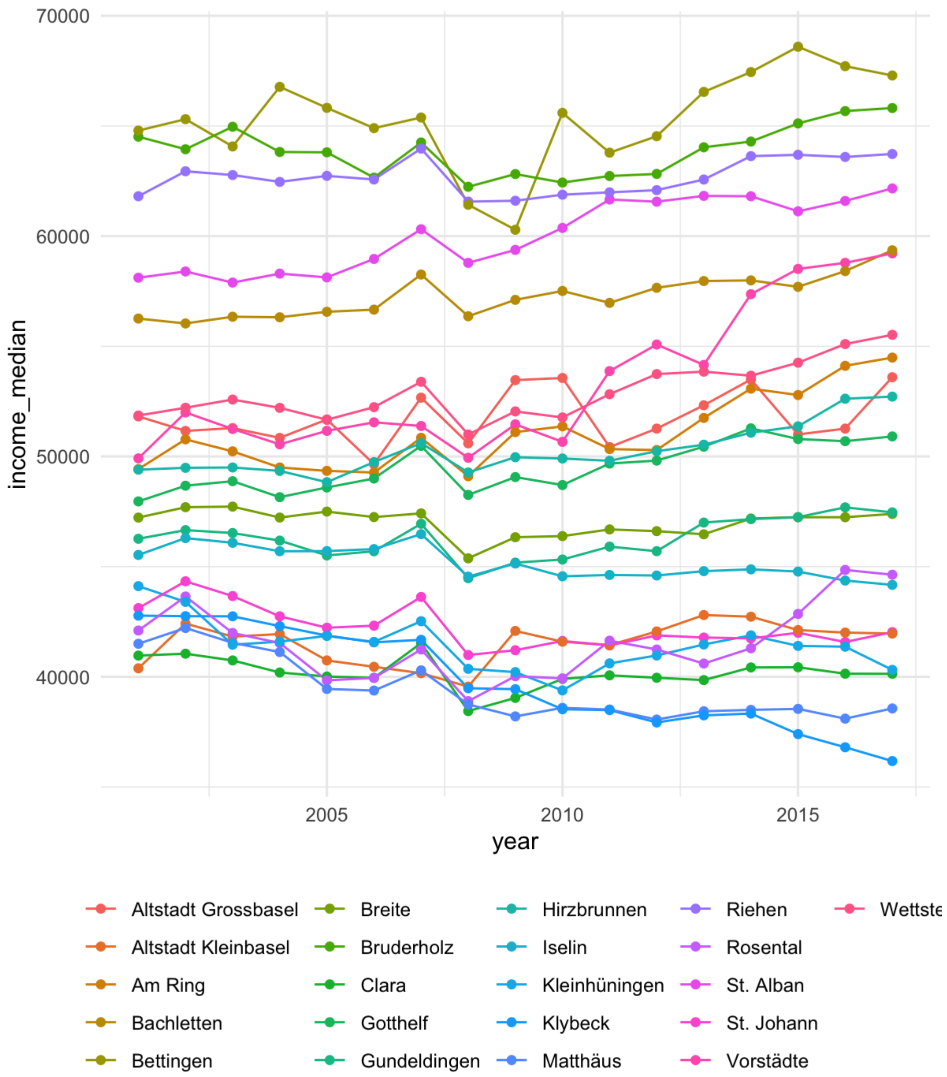


`theme()`

1. With **87 arguments** `theme()` permits specification of all aesthetic details.

2. Makes uses of helper functions:
   - `element_rect()` | for rectangles
   - `element_line()` | for lines
   - `element_text()` | for text
   - `element_blank()` | for removals

```r
# Fixing the legend
my_plot +
  theme_minimal() +
  theme(legend.position = "bottom",

        # remove legend title
        legend.title = element_blank())
```

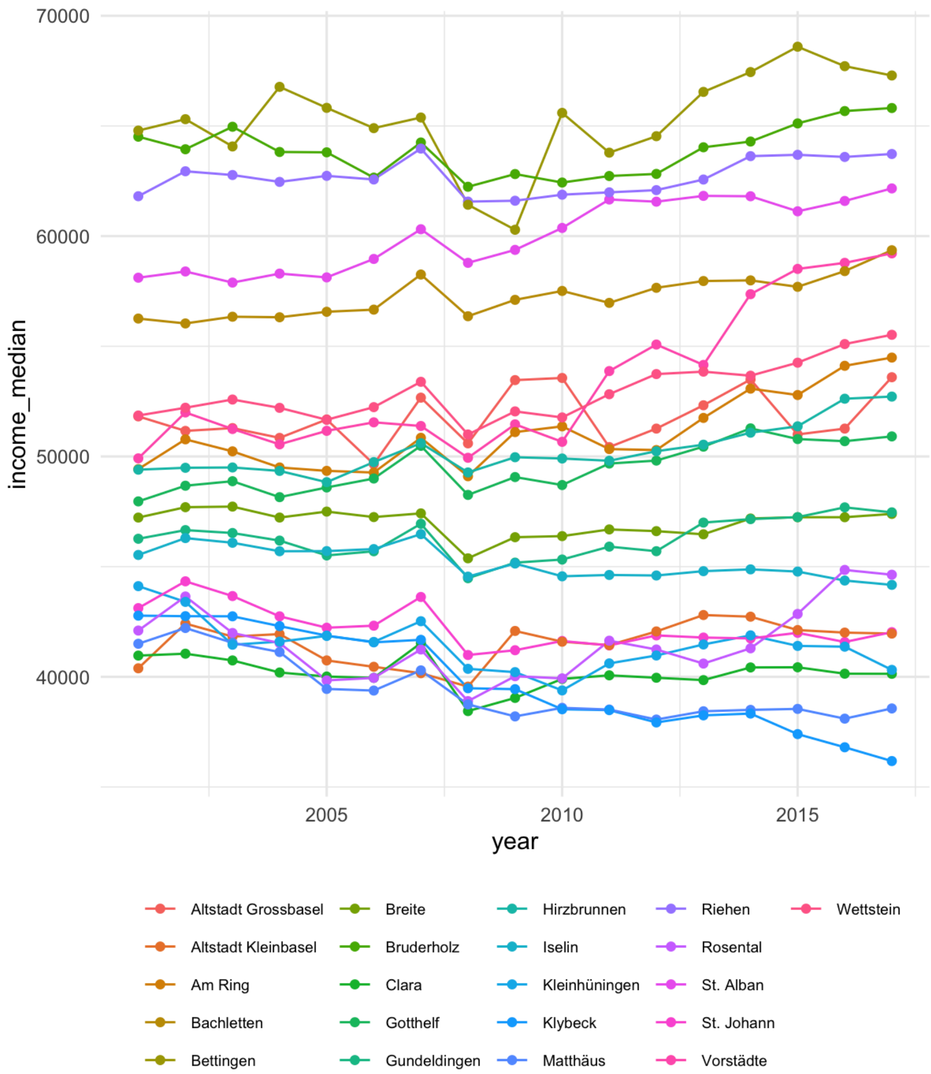


theme()

1. With **87 arguments** `theme()` permits specification of all aesthetic details.

2. Makes uses of helper functions:
   - `element_rect()` | for rectangles
   - `element_line()` | for lines
   - `element_text()` | for text
   - `element_blank()` | for removals

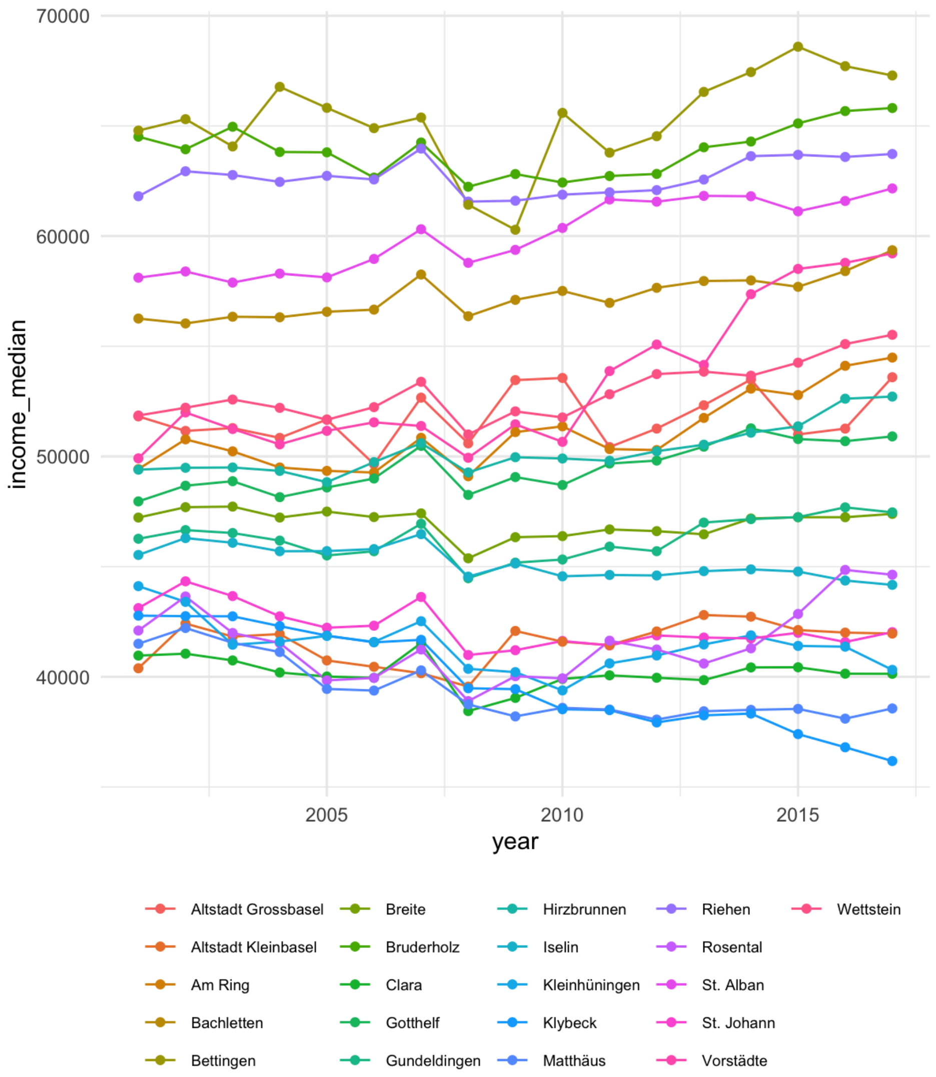
```
# Fixing the legend
my_plot +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    legend.title = element_blank(),

    # reduce legend text size
    legend.text = element_text(size = 7))
```



`theme`

1. `theme`s can be stored in an object.
2. `theme` objects are not functions.

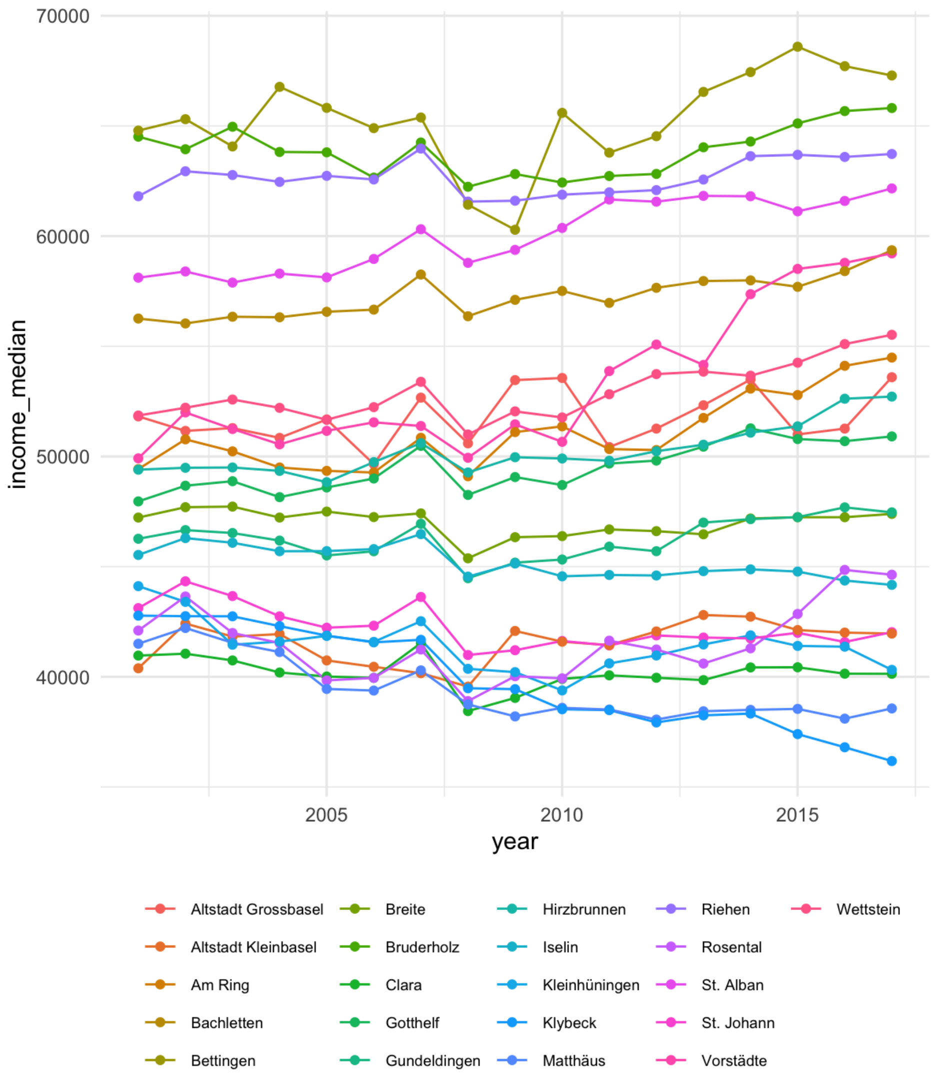
```r
# save my theme
my_theme <- theme(
  legend.position = "bottom",
  legend.title = element_blank(),
  legend.text = element_text(size = 7))


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



scale_*()

1. Various scale_*() permit specification of all **dimensions**, inclding axes, colors, sizes, etc.

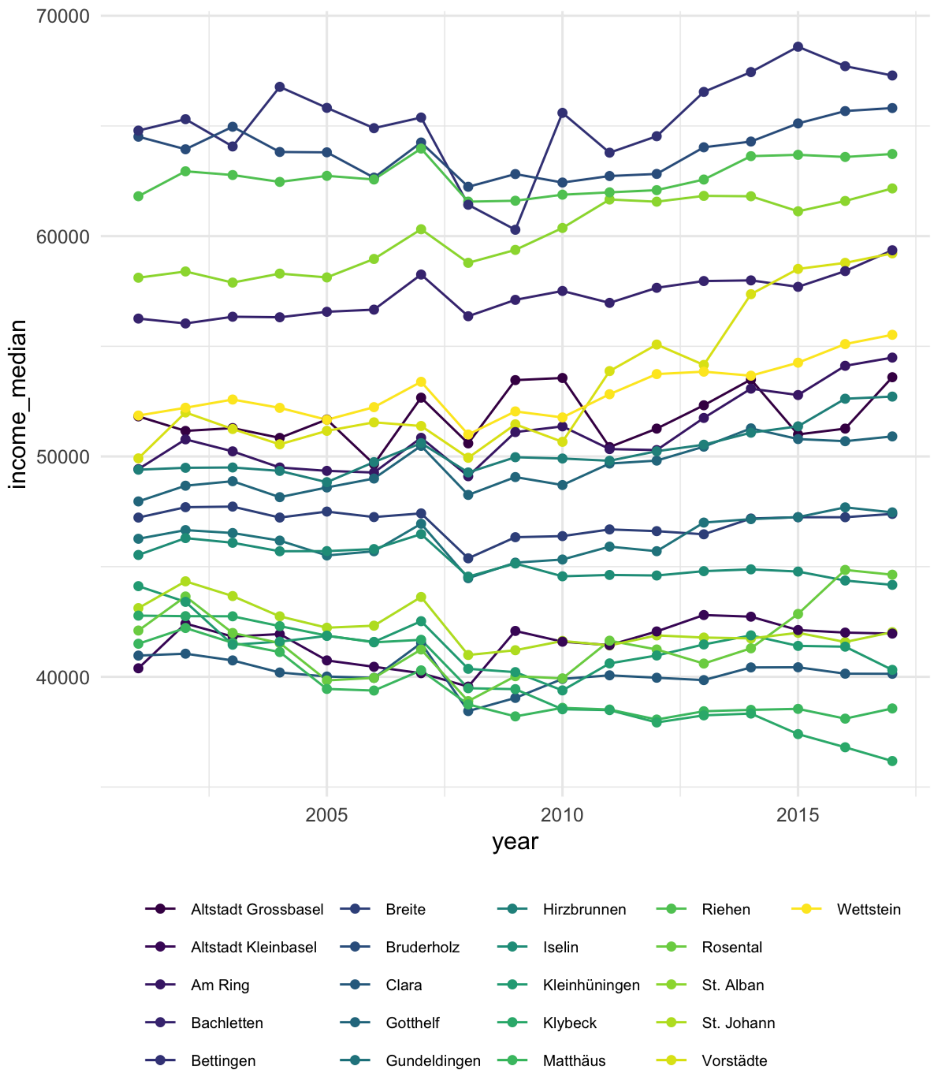
2. Groups of scale_*() functions:
   - scale_xy_* | Scales axes
   - scale_color_* | Scales colors
   - scale_size_* | Sclaes sizes
   - scale_alpha_* | Scales opacity



scale_*()

1 Various `scale_*()` permit specification of all **dimensions**, inclding axes, colors, sizes, etc.

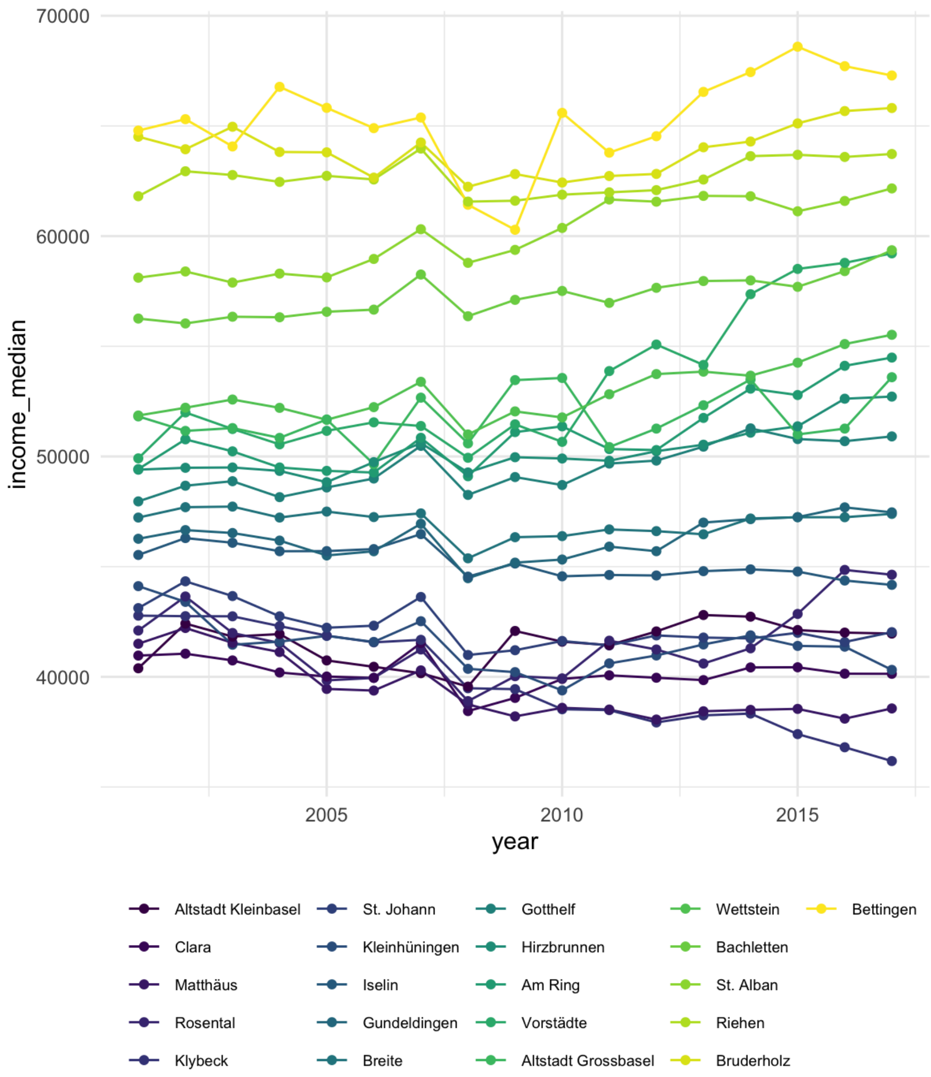
```
# Fixing the legend
my_plot +
  theme_minimal() +
  theme(
    legend.position = "bottom",
    legend.title = element_blank(),
    legend.text = element_text(size=7)) +

  # color using viridis
  scale_color_viridis_d()
```



Wrangling

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

```
basel %>%

  # sort by income and factor quarter
  arrange(year, income_median) %>%
  mutate(quarter = as_factor(quarter)) %>%

  # 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)) +
  scale_color_viridis_d()
```

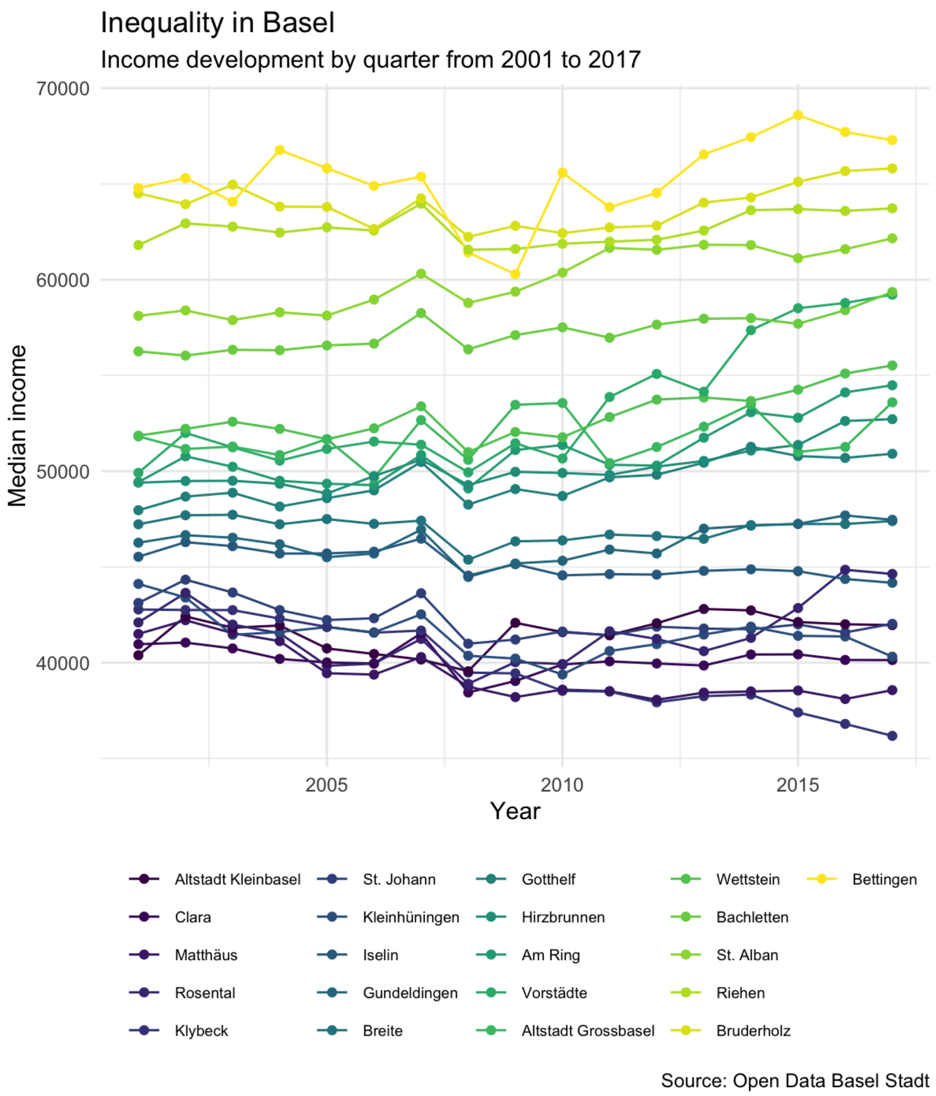


`labs()`

1. Vaious annotions can be added using `labs()`.

2. 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...")
```



Inequality in Basel

Income development by quarter from 2001 to 2017

Source: Open Data Basel Stadt

ggsave()

1. Saves plots to the harddrive.
2. Key arguments:
   - filename | Filename/path
   - device | e.g., ".pdf" or ".png"
   - path | Path to folder
   - height, width | Height, Width
   - unit | Unit for Height, Width
   - dpi | Resolution

```r
# Save as pdf
ggsave(filename = "my_plot.pdf",
       plot = my_plot,
       device = "pdf",
       path = "3_Figures",
       width = 7,
       height = 7)

# Save as png
ggsave(filename = "my_plot.png",
       plot = my_plot,
       device = "png",
       path = "3_Figures",
       width = 7,
       height = 7)
```

Schedule