Prediction and Machine Learning for Economics

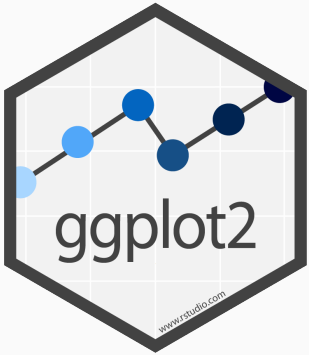
Plotting with `ggplot2`

Sebastian Fossati

University of Alberta | E593 | 2023

Outline

ggplot2



Gapminder data

We'll be working with data from Hans Rosling's **Gapminder project**.

An excerpt of these data can be accessed through an R package called `gapminder`, cleaned and assembled by Jenny Bryan at UBC.

In the console: `install.packages("gapminder")`

Load the package and data:

```
# load library
library(gapminder)
```

Gapminder data

The data frame we will work with is called `gapminder`, available once you have loaded the package. Let's see its structure:

```
# check out data
str(gapminder)
```

```
## tibble [1,704 x 6] (S3: tbl_df/tbl/data.frame)
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ year     : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp  : num [1:1704] 28.8 30.3 32 34 36.1 ...
##  $ pop      : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372 128
##  $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

What's interesting here?

The `gapminder` dataset contains **panel data** on life expectancy, population size, and GDP per capita for 142 countries since the 1950s.

Remarks:

- **factor** variables `country` and `continent`
  - factors are categorical data with an underlying numeric representation
- many observations: n = 1704 rows
- a nested/hierarchical structure: `year` in `country` in `continent`

Some data to visualize

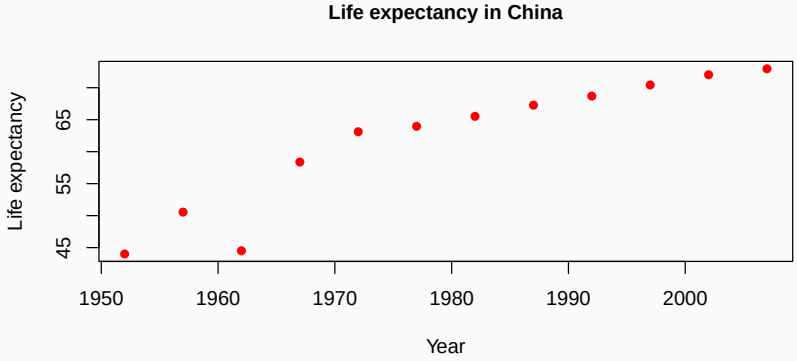
We'll also load `dplyr` to give us tools to manipulate it for visualization.

```
library(dplyr)
China <- gapminder %>% filter(country == "China")
head(China, 4)
```

```
## # A tibble: 4 x 6
##   country continent  year lifeExp       pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>     <int>     <dbl>
## 1 China   Asia       1952    44   556263527      400.
## 2 China   Asia       1957    50.5 637408000      576.
## 3 China   Asia       1962    44.5 665770000      488.
## 4 China   Asia       1967    58.4 754550000      613.
```

Base R plots

```
plot(lifeExp ~ year, data = China,
     xlab = "Year", ylab = "Life expectancy",
     main = "Life expectancy in China",
     col = "red", cex.lab = 1, cex.main= 1, pch = 16)
```



ggplot2

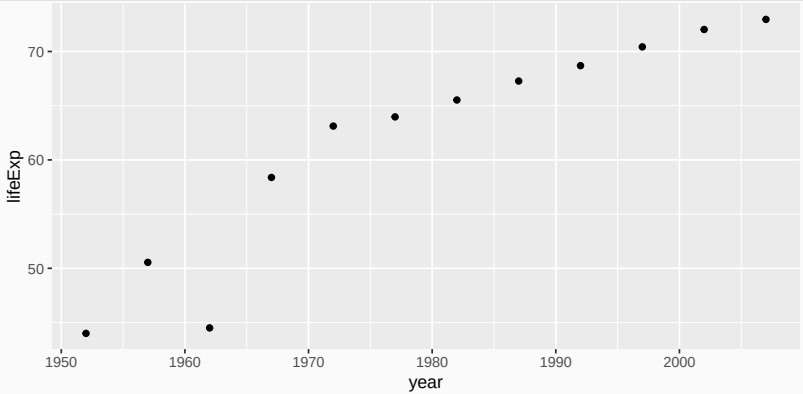
An alternative way of plotting many prefer uses the `ggplot2` package in R, which is part of the `tidyverse`.

```
library(ggplot2)
```

The core idea underlying this package is the **layered grammar of graphics**: we can break up elements of a plot into pieces and combine them.

Chinese Life Expectancy in ggplot

```
#
ggplot(data = China, aes(x = year, y = lifeExp)) +
  geom_point()
```



Structure of a ggplot

`ggplot2` graphics objects consist of two primary components:

1. **layers**, the components of a graph
   - we *add* layers to a `ggplot2` object using `+`
   - this includes lines, shapes, and text
2. **aesthetics**, which determine how the layers appear
   - we *set* aesthetics using *arguments* (e.g. `color="red"`) inside layer functions
   - this includes locations, colors, and sizes
   - aesthetics also determine how data *map* to appearances

Layers

Layers are the components of the graph, such as:

- `ggplot()`: initializes `ggplot2` object, specifies input data
- `geom_point()`: layer of scatterplot points
- `geom_line()`: layer of lines
- `ggtitle()`, `xlab()`, `ylab()`: layers of labels
- `facet_wrap()`: layer creating separate panels stratified by some factor wrapping around
- `facet_grid()`: same idea, but can split by two variables along rows and columns (e.g. `facet_grid(gender ~ age_group)`)
- `theme_bw()`: replace default gray background with black-and-white

Layers are separated by a `+` sign.

Aesthetics

Aesthetics control the appearance of the layers:

- `x`, `y`: *x* and *y* coordinate values to use
- `color`: set color of elements based on some data value
- `group`: describe which points are conceptually grouped together for the plot (often used with lines)
- `size`: set size of points/lines based on some data value
- `alpha`: set transparency based on some data value

Aesthetics: setting vs. mapping

Layers take arguments to control their appearance, such as point/line colors or transparency (`alpha` between 0 and 1).

Arguments like `color`, `size`, `linetype`, `shape`, `fill`, and `alpha` can be used directly on the layers (**setting aesthetics**), e.g. `geom_point(color = "red")`. These *don't depend on the data*.

Arguments inside `aes()` (**mapping aesthetics**) will *depend on the data*, e.g. `geom_point(aes(color = continent))`.

`aes()` in the `ggplot()` layer gives overall aesthetics to use in other layers, but can be changed on individual layers (including switching `x` or `y` to different variables).

Storing plots

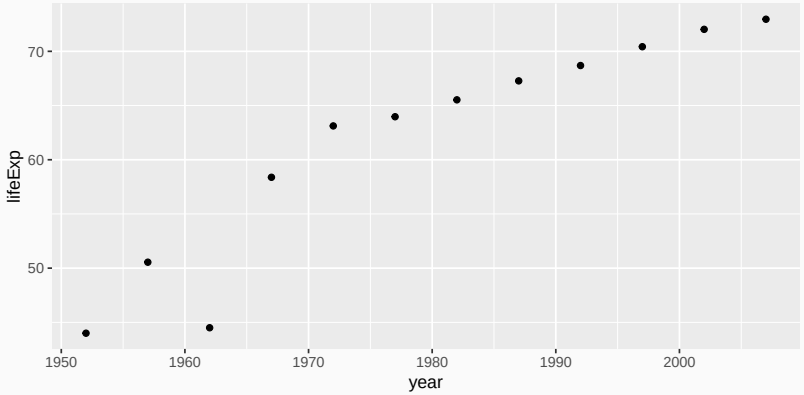
We can assign a `ggplot` object to a name:

```
lifeExp_China <-
  ggplot(data = China, aes(x = year, y = lifeExp)) +
  geom_point()
```

The graph won't be displayed when you do this. You can show the graph using a single line of code with just the object name, *or take the object and add more layers*.

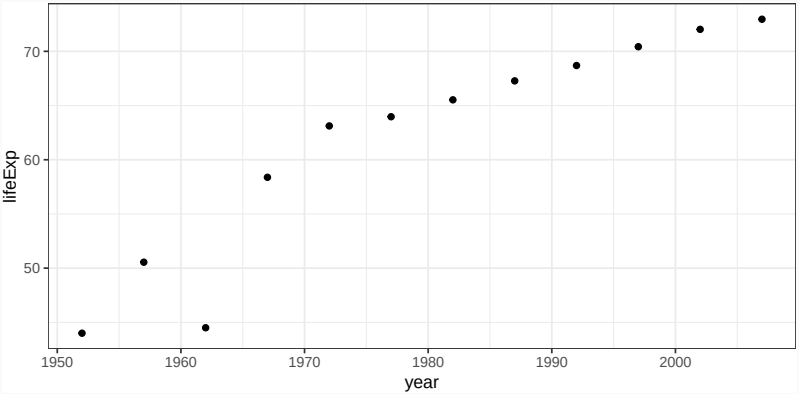
Showing a stored graph

```
#
lifeExp_China
```



Adding a layer

```
#
lifeExp_China + theme_bw()
```



Axis labels, points, no background

1: Base Plot

```
#
lifeExp_China <- ggplot(data = China, aes(x = year, y = lifeExp))
#
#
#
#
#
#
```

Axis labels, points, no background

2: Scatterplot

```
#
lifeExp_China <- ggplot(data = China, aes(x = year, y = lifeExp)) +
  geom_point()
#
#
#
#
#
```

Axis labels, points, no background

3: Point color and size

```
#
lifeExp_China <- ggplot(data = China, aes(x = year, y = lifeExp)) +
  geom_point(color = "red", size = 2)
#
#
#
#
#
```

Axis labels, points, no background

4: x-axis label

```
#
lifeExp_China <- ggplot(data = China, aes(x = year, y = lifeExp)) +
  geom_point(color = "red", size = 2) +
  xlab("Year")
#
#
#
#
```

Axis labels, points, no background

5: y-axis label

```
#
lifeExp_China <- ggplot(data = China, aes(x = year, y = lifeExp)) +
  geom_point(color = "red", size = 2) +
  xlab("Year") +
  ylab("Life expectancy")
#
#
#
```

Axis labels, points, no background

6: Title

```
#
lifeExp_China <- ggplot(data = China, aes(x = year, y = lifeExp)) +
  geom_point(color = "red", size = 2) +
  xlab("Year") +
  ylab("Life expectancy") +
  ggtitle("Life expectancy in China")
#
#
```

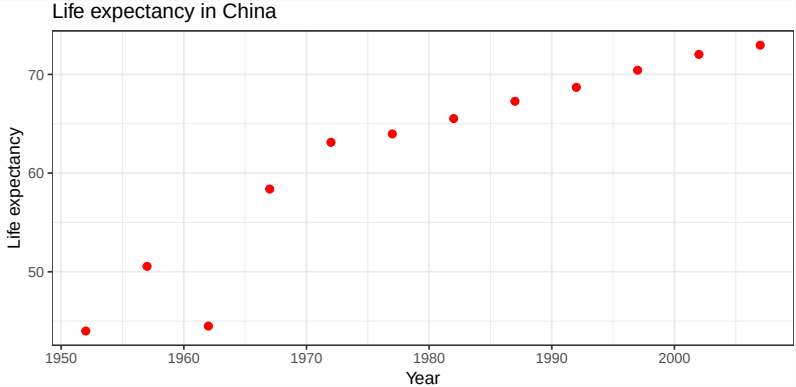
Axis labels, points, no background

7: Theme

```
#
lifeExp_China <- ggplot(data = China, aes(x = year, y = lifeExp)) +
  geom_point(color = "red", size = 2) +
  xlab("Year") +
  ylab("Life expectancy") +
  ggtitle("Life expectancy in China") +
  theme_bw()
#
```

Axis labels, points, no background

```
#
lifeExp_China
```



Plotting all countries

We have a plot we like for China…

… but what if we want *all the countries*?

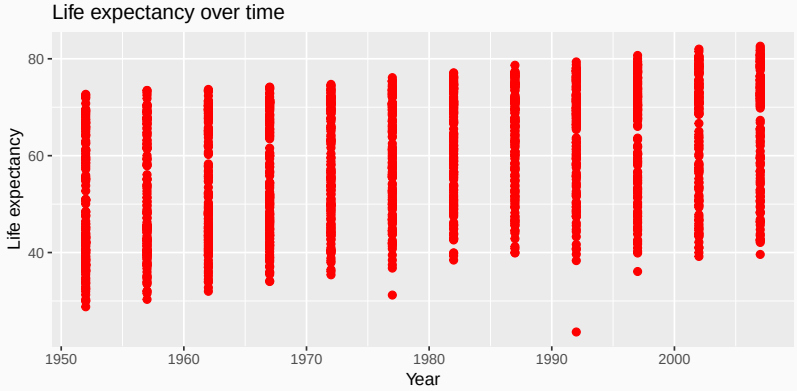
Plotting all countries

1: All countries

```
#
lifeExp_by_year <-
  ggplot(
    data = gapminder,
    aes(x = year, y = lifeExp)
  ) +
  geom_point(color = "red", size = 2) +
  xlab("Year") +
  ylab("Life expectancy") +
  ggtitle("Life expectancy over time")
```

Plotting all countries

```
#
lifeExp_by_year
```



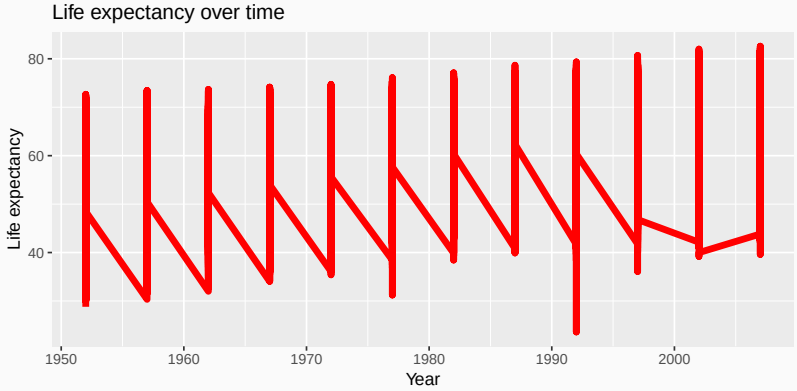
Plotting all countries

2: Lines

```
#
lifeExp_by_year <-
  ggplot(
    data = gapminder,
    aes(x = year, y = lifeExp)
  ) +
  geom_line(color = "red", size = 2) +
  xlab("Year") +
  ylab("Life expectancy") +
  ggtitle("Life expectancy over time")
```

Plotting all countries

```
#
lifeExp_by_year
```



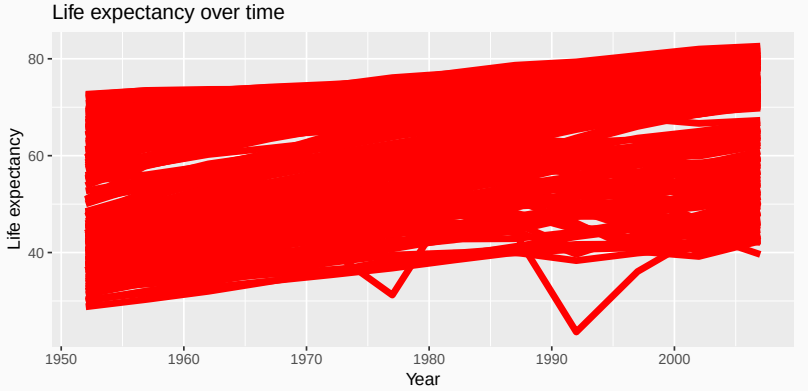
Plotting all countries

3: Grouping

```
#
lifeExp_by_year <-
  ggplot(
    data = gapminder,
    aes(x = year, y = lifeExp, group = country)
  ) +
  geom_line(color = "red", size = 2) +
  xlab("Year") +
  ylab("Life expectancy") +
  ggtitle("Life expectancy over time")
```

Plotting all countries

```
#
lifeExp_by_year
```



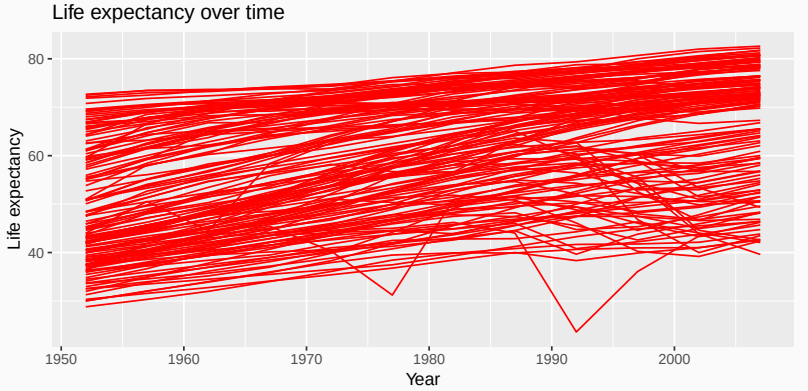
Plotting all countries

4: Size

```
#
lifeExp_by_year <-
  ggplot(
    data = gapminder,
    aes(x = year, y = lifeExp, group = country)
  ) +
  geom_line(color = "red") +
  xlab("Year") +
  ylab("Life expectancy") +
  ggtitle("Life expectancy over time")
```

Plotting all countries

```
#
lifeExp_by_year
```



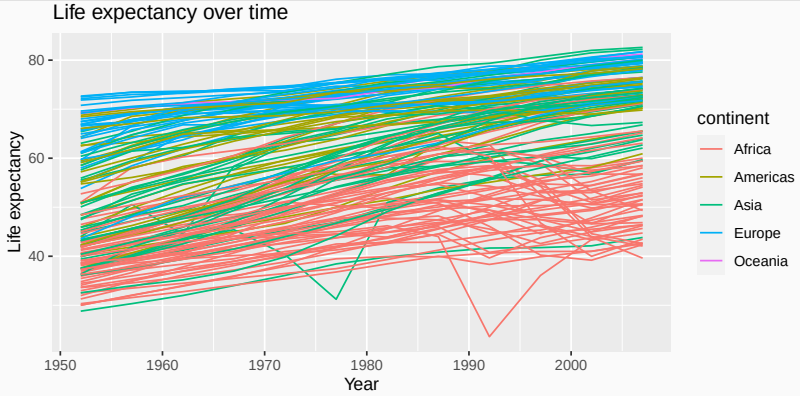
Plotting all countries

5: Color

```
#
lifeExp_by_year <-
  ggplot(
    data = gapminder,
    aes(x = year, y = lifeExp, group = country, color = continent)
  ) +
  geom_line() +
  xlab("Year") +
  ylab("Life expectancy") +
  ggtitle("Life expectancy over time")
```

Plotting all countries

```
#
lifeExp_by_year
```



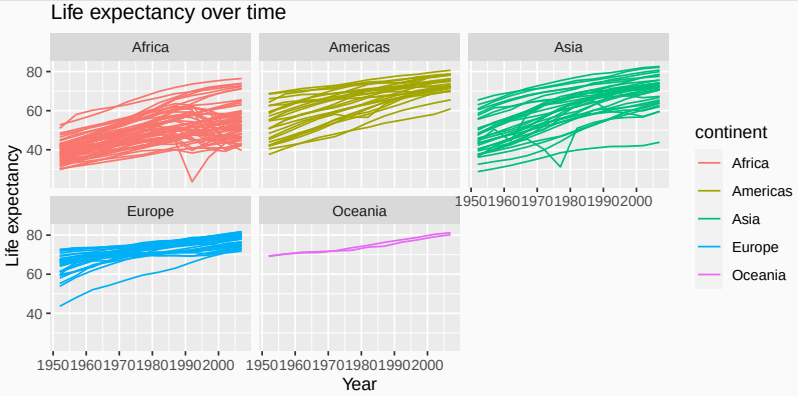
Plotting all countries

6: Facets

```
#
lifeExp_by_year <-
  ggplot(
    data = gapminder,
    aes(x = year, y = lifeExp, group = country, color = continent)
  ) +
  geom_line() +
  xlab("Year") +
  ylab("Life expectancy") +
  ggtitle("Life expectancy over time") +
  facet_wrap(~ continent)
```

Plotting all countries

```
#
lifeExp_by_year
```



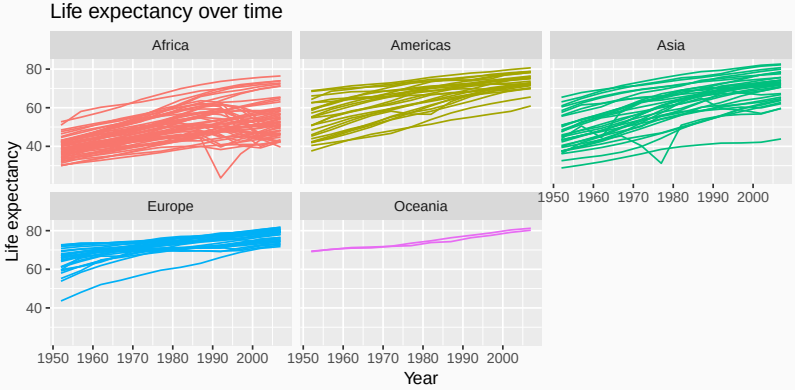
Plotting all countries

7: No legend

```
#
lifeExp_by_year <-
  ggplot(
    data = gapminder,
    aes(x = year, y = lifeExp, group = country, color = continent)
  ) +
  geom_line() +
  xlab("Year") +
  ylab("Life expectancy") +
  ggtitle("Life expectancy over time") +
  facet_wrap(~ continent) +
  theme(legend.position = "none")
```

Plotting all countries

```
#
lifeExp_by_year
```



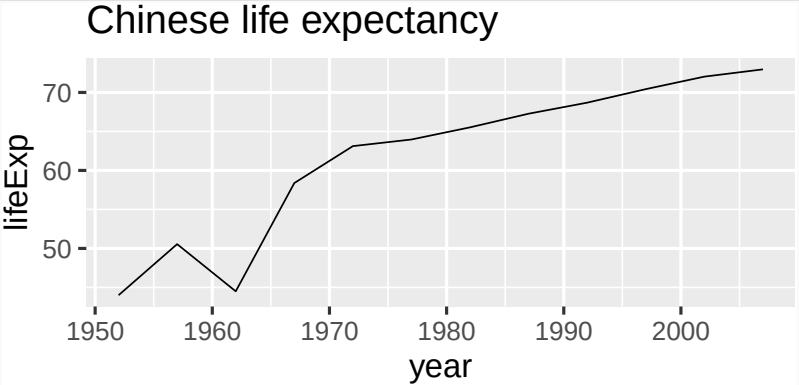
Changing the axes

We can modify the axes in a variety of ways, such as:

- change the *x* or *y* range using `xlim()` or `ylim()` layers
- change to a logarithmic or square-root scale on either axis:
  `scale_x_log10()`, `scale_y_sqrt()`
- change where the major/minor breaks are:
  `scale_x_continuous(breaks =, minor_breaks = )`

Fonts too small?

```
ggplot(data = China, aes(x = year, y = lifeExp)) +
  geom_line() +
  ggtitle("Chinese life expectancy") +
  theme_gray(base_size = 20)
```



Text and tick adjustments

Text size, labels, tick marks, etc. can be messed with more precisely using arguments to the `theme()` layer.

Examples:

- `plot.title = element_text(size = rel(2), hjust = 0)` makes the title twice as big as usual and left-aligns it
- `axis.text.x = element_text(angle = 45)` rotates *x* axis labels
- `axis.text = element_text(colour = "blue")` makes the *x* and *y* axis labels blue
- `axis.ticks.length = unit(.5, "cm")` makes the axis ticks longer

Note: `theme()` is a different layer than `theme_gray()` or `theme_bw()`, which you might also be using in a previous layer.

Scales for color, shape, etc.

Scales are layers that control how the mapped aesthetics appear. You can modify these with a `scale_[aesthetic]_[option]()` layer where `[aesthetic]` is `color`, `shape`, `linetype`, `alpha`, `size`, `fill`, etc. and `[option]` is something like `manual`, `continuous` or `discrete` (depending on nature of the variable).

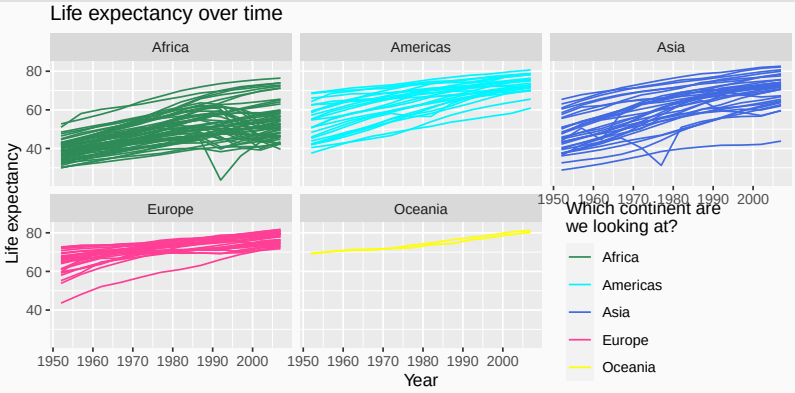
Examples:

- `scale_linetype_manual()`: manually specify the linetype for each different value
- `scale_color_manual()`: manually specify the color for each different value

When confused... Google or StackOverflow it!

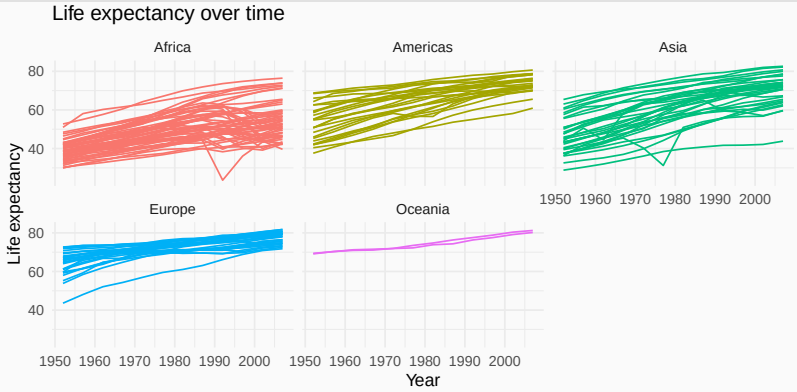
Legend name and manual colors

```
lifeExp_by_year + theme(legend.position = c(0.8, 0.2)) +
  scale_color_manual(
    name = "Which continent are\nwe looking at?", # \n adds a line break
    values = c("Africa" = "seagreen", "Americas" = "turquoise1",
               "Asia" = "royalblue", "Europe" = "violetred1", "Oceania" = "yellow"))
```



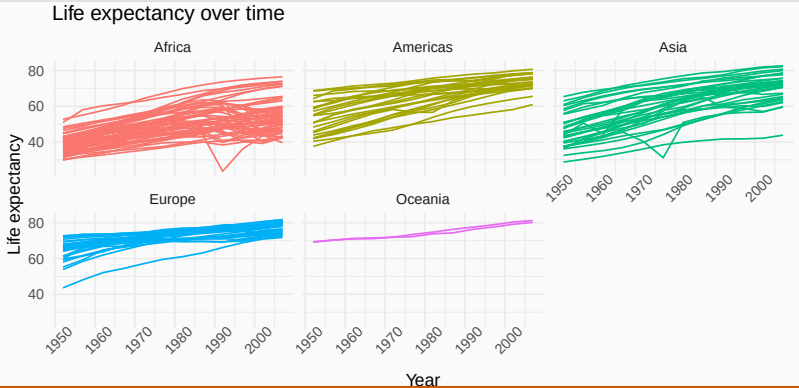
Plotting all countries

```
#
lifeExp_by_year + theme_minimal() + theme(legend.position = "none")
```



Plotting all countries

```
#
lifeExp_by_year + theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45),
    legend.position = "none"
  )
```



Saving `ggplot` plots

When you knit an R Markdown file, any plots you make are automatically saved in the "figure" folder in `.png` format.

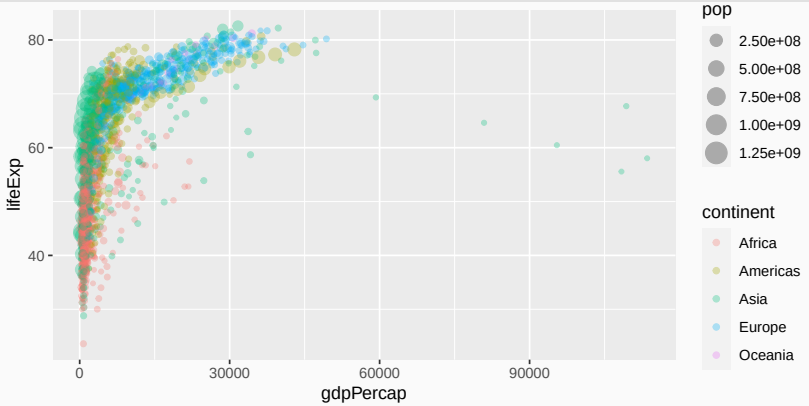
If you want to save another copy (perhaps of a different file type for use in a manuscript), use `ggsave()`:

```
ggsave(
  "I_saved_a_file.pdf",
  plot = lifeExp_by_year,
  height = 3, width = 5, units = "in"
)
```

Outline

One more gapminder example

```
# gapminder scatterplot
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(size = pop, col = continent), alpha = 0.3)
```



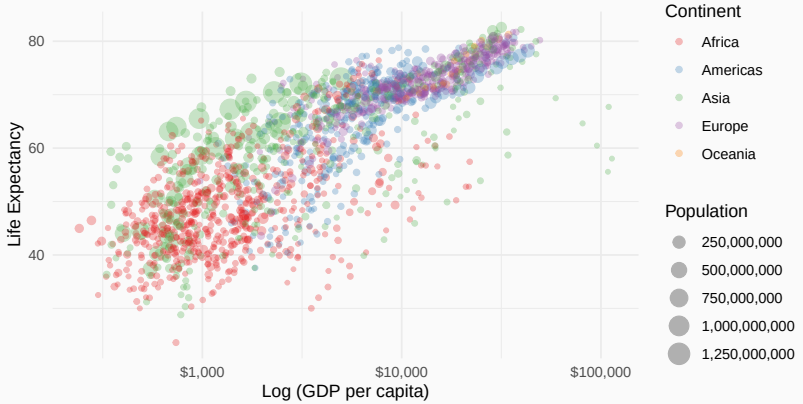
One more `gapminder` example

You don't have to transform your original data.

- `ggplot2` takes care of all of that

```
# logarithmic scale on x-axis, use dollar units
p <- ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(aes(size = pop, col = continent), alpha = 0.3) +
  scale_color_brewer(name = "Continent", palette = "Set1") +
  scale_size(name = "Population", labels = scales::comma) +
  scale_x_log10(labels = scales::dollar) +
  labs(x = "Log (GDP per capita)", y = "Life Expectancy") +
  theme_minimal()
```

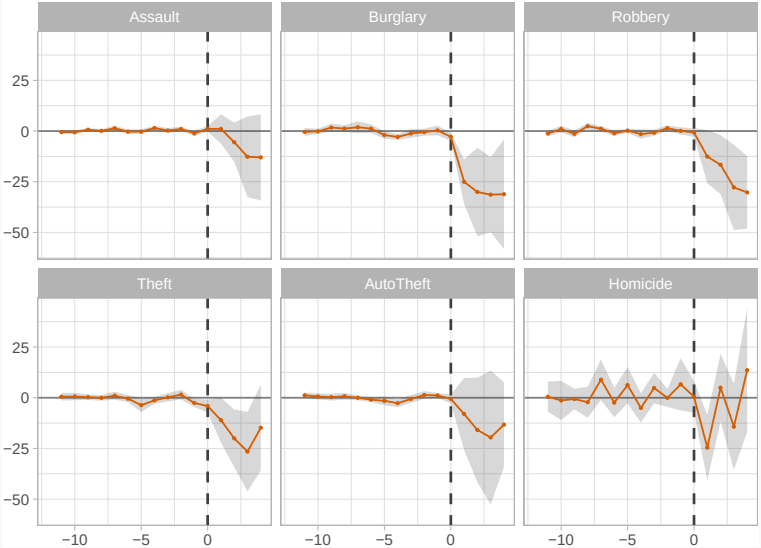
One more `gapminder` example



Some of my plots

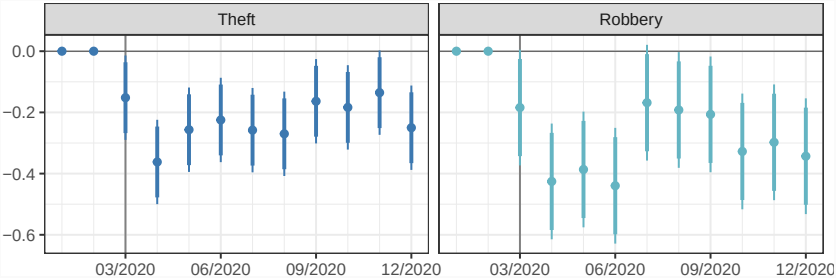
```
ggplot(data, aes(x = time, y = ATT)) +
  geom_hline(yintercept = 0, color = "grey50", size = .5) +
  geom_vline(xintercept = 0, color = "grey25", linetype = "dashed", size = .75) +
  geom_ribbon(aes(x = time, ymin = CI.lower, ymax = CI.upper), fill = "grey25", alpha
  geom_line(size = .5, color = my_colors[3]) +
  geom_point(size = .5, color = my_colors[3]) +
  scale_y_continuous(name = "") +
  scale_x_continuous(name = "", limits = c(-12,4)) +
  theme(legend.position = "none") +
  facet_wrap(
    ~ factor(
      crime,
      levels = c("Assault", "Burglary", "Robbery","Theft", "AutoTheft", "Homicide")
    )
  )
```

Some of my plots



Bonus plots

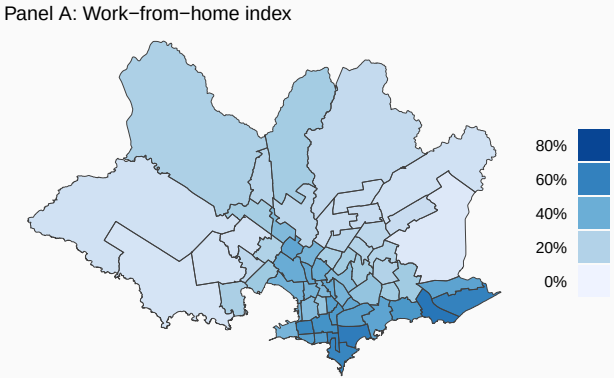
This is a plot from another working paper of mine.[1]



[1] Díaz, Fossati, and Trajtenberg (2022): “Stay at home if you can: COVID-19 stay-at-home guidelines and local crime”, working paper.

Some of my plots

This is a map from the same working paper.[1]



[1] Díaz, Fossati, and Trajtenberg (2022): “Stay at home if you can: COVID-19 stay-at-home guidelines and local crime”, working paper.