03 - Creating Visualizations with `ggplot2`

VisAnalytics Exercise

Summer Term

Data Visualization - RECAP

> *"The greatest value of a picture is when it forces us to notice what we never expected to see."* - John W. Tukey

Why data visualization?

- To discover data problems, such as mistakes, biases, systematic errors and unexpected variability preventing flawed analyses and false discoveries
- To discover patterns which would otherwise be missed
- To convincingly communicate a data-driven finding

Example: infographics in newspapers

Infographic examples 1/8

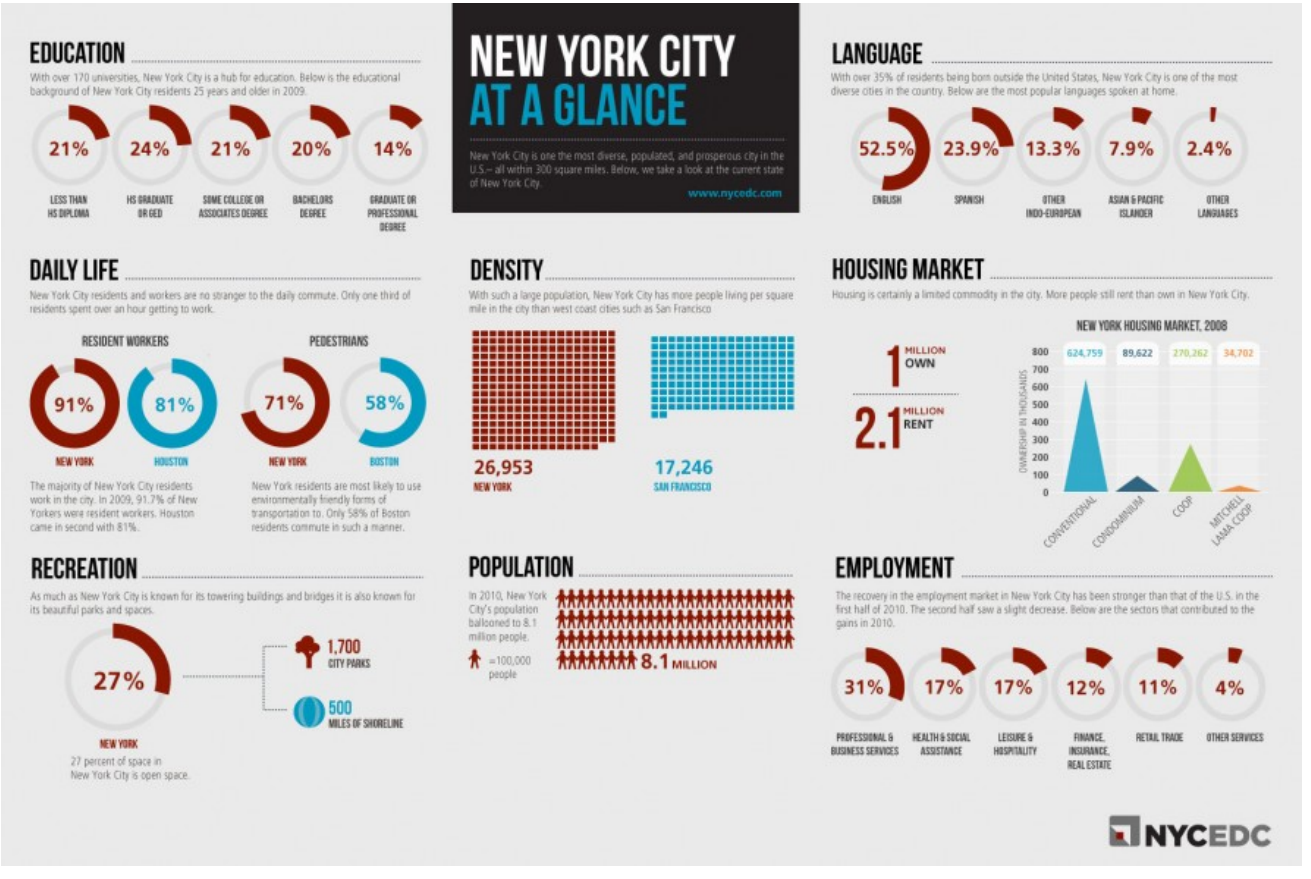


Figure source: http://www.hireahelper.com/_static/img/public/landing-content/Infographic-New-York-City.jpg

Infographic examples 2/8

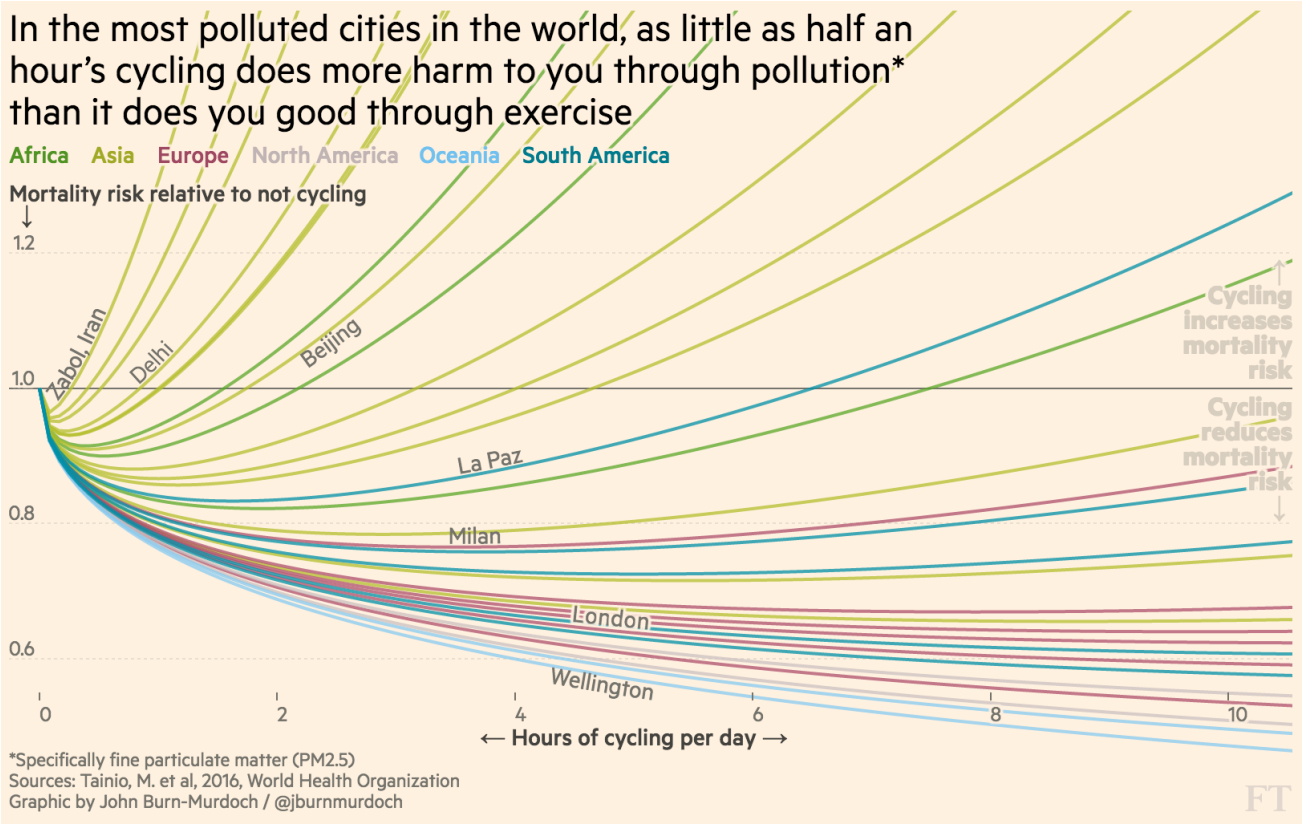


Figure source: http://johnburnmurdoch.github.io/slides/r-ggplot/greatest-hits-1.png

Infographic examples 3/8

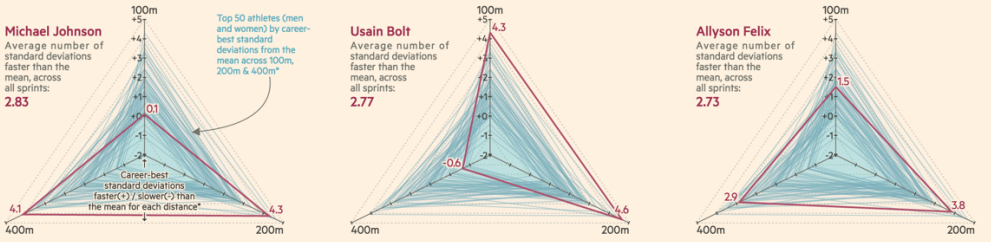


Figure source: http://johnburnmurdoch.github.io/slides/r-ggplot/greatest-hits-2.png

Infographic examples 4/8

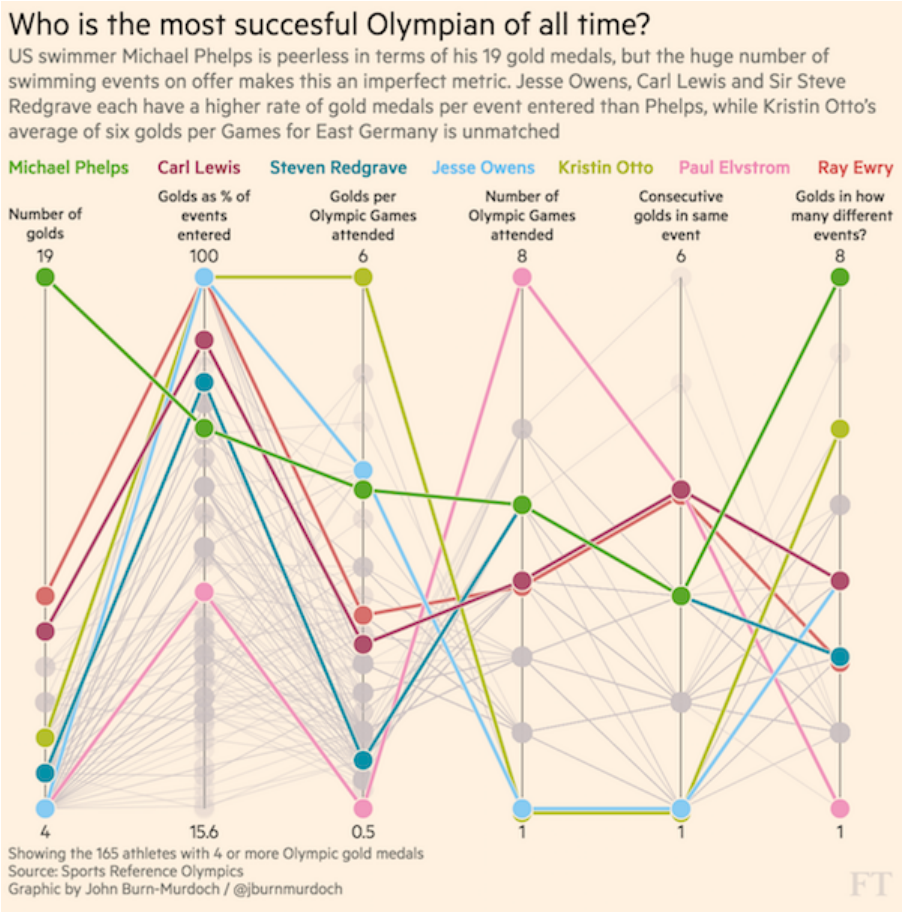


Figure source: http://johnburnmurdoch.github.io/slides/r-ggplot/greatest-hits-3.png

Infographic examples 5/8

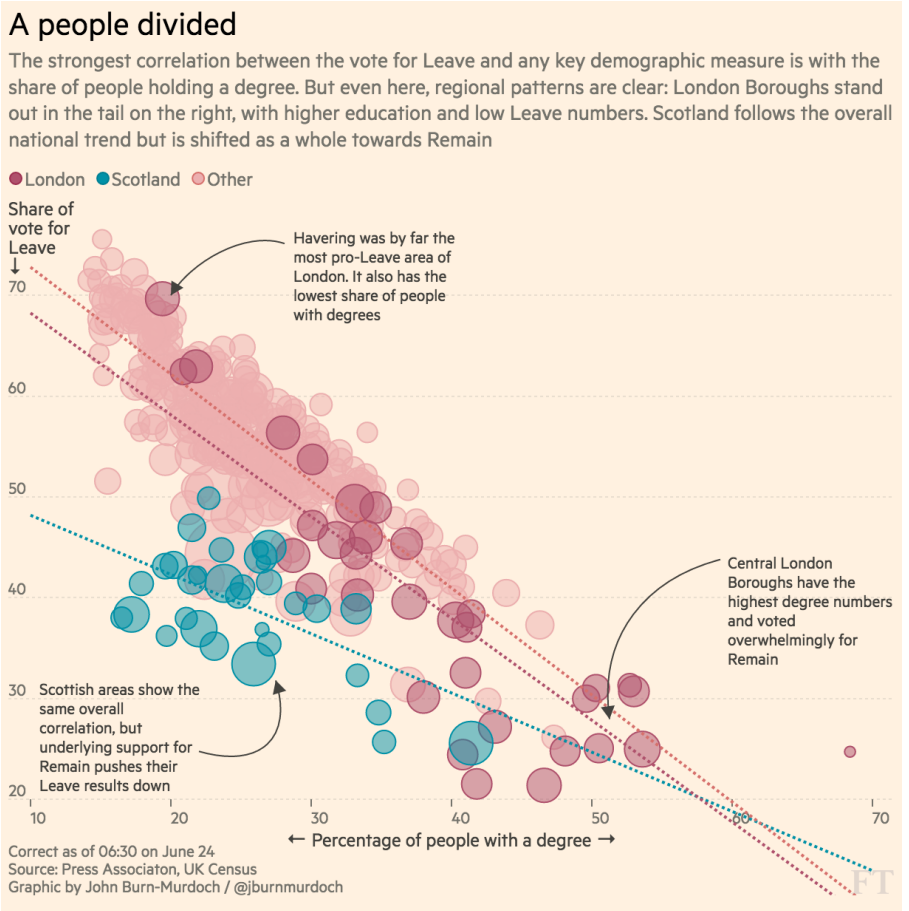


Figure source: http://johnburnmurdoch.github.io/slides/r-ggplot/greatest-hits-6.png

Infographic examples 6/8

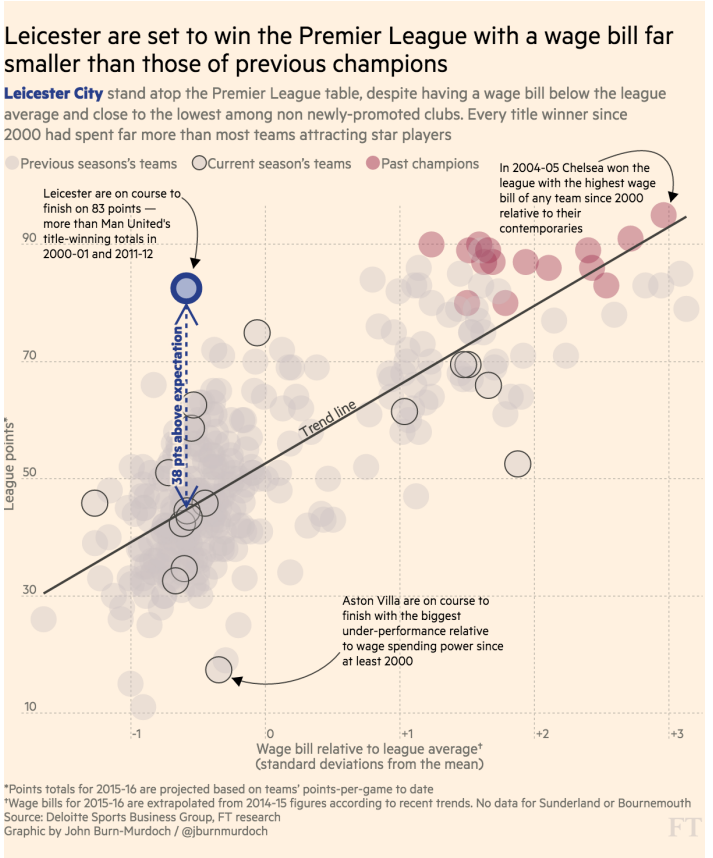


Figure source: http://johnburnmurdoch.github.io/slides/r-ggplot/greatest-hits-8.png

Infographic examples 7/8

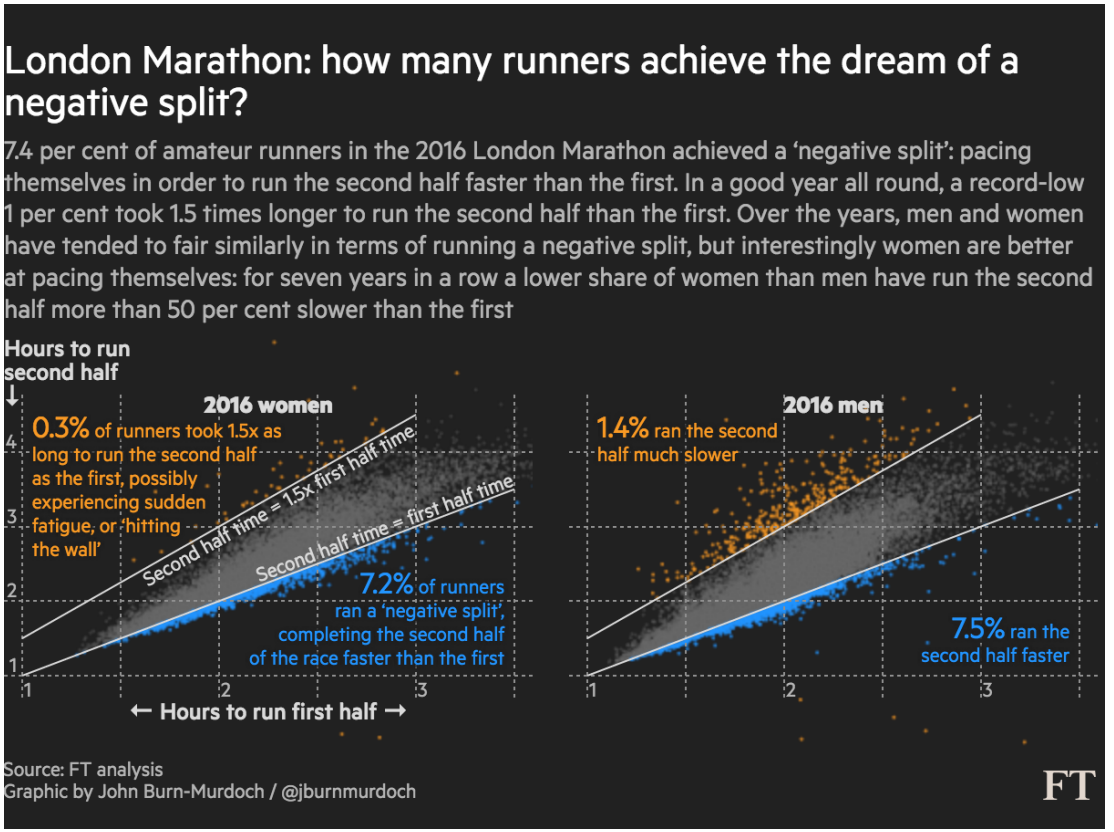


Figure source: http://johnburnmurdoch.github.io/slides/r-ggplot/greatest-hits-9.png

Infographic examples 8/8

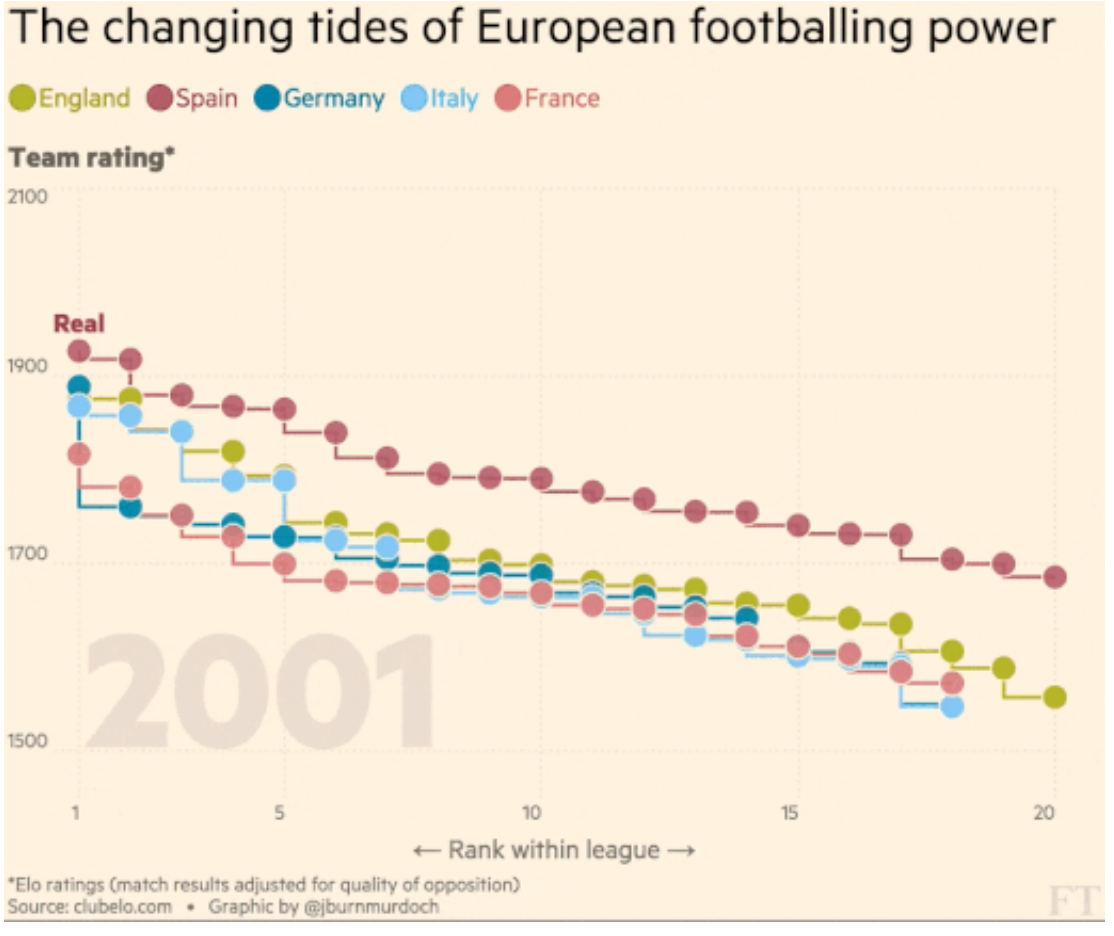


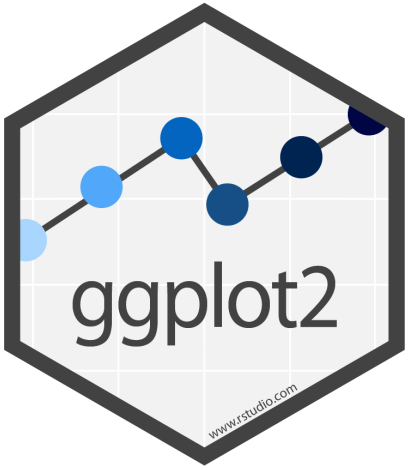
Figure source: http://johnburnmurdoch.github.io/slides/r-ggplot/all.gif

ggplot2

`ggplot2` is a data exploration and visualization package for graphics generation in R.

Base R has inbuild plotting capabilities. What are the advantages of `ggplot2`?

- `ggplot2` is inspired by the *Grammar of Graphics*[1]
- idea: break the graph into components and handle each component individually $\rightarrow$ ensure versatility and control
- `ggplot2` package contains a set of functions to build the features of the graph in a series of layers



[1] Leland Wilkinson. *The grammar of graphics*. Springer Science & Business Media, 2006.

The grammar of graphics

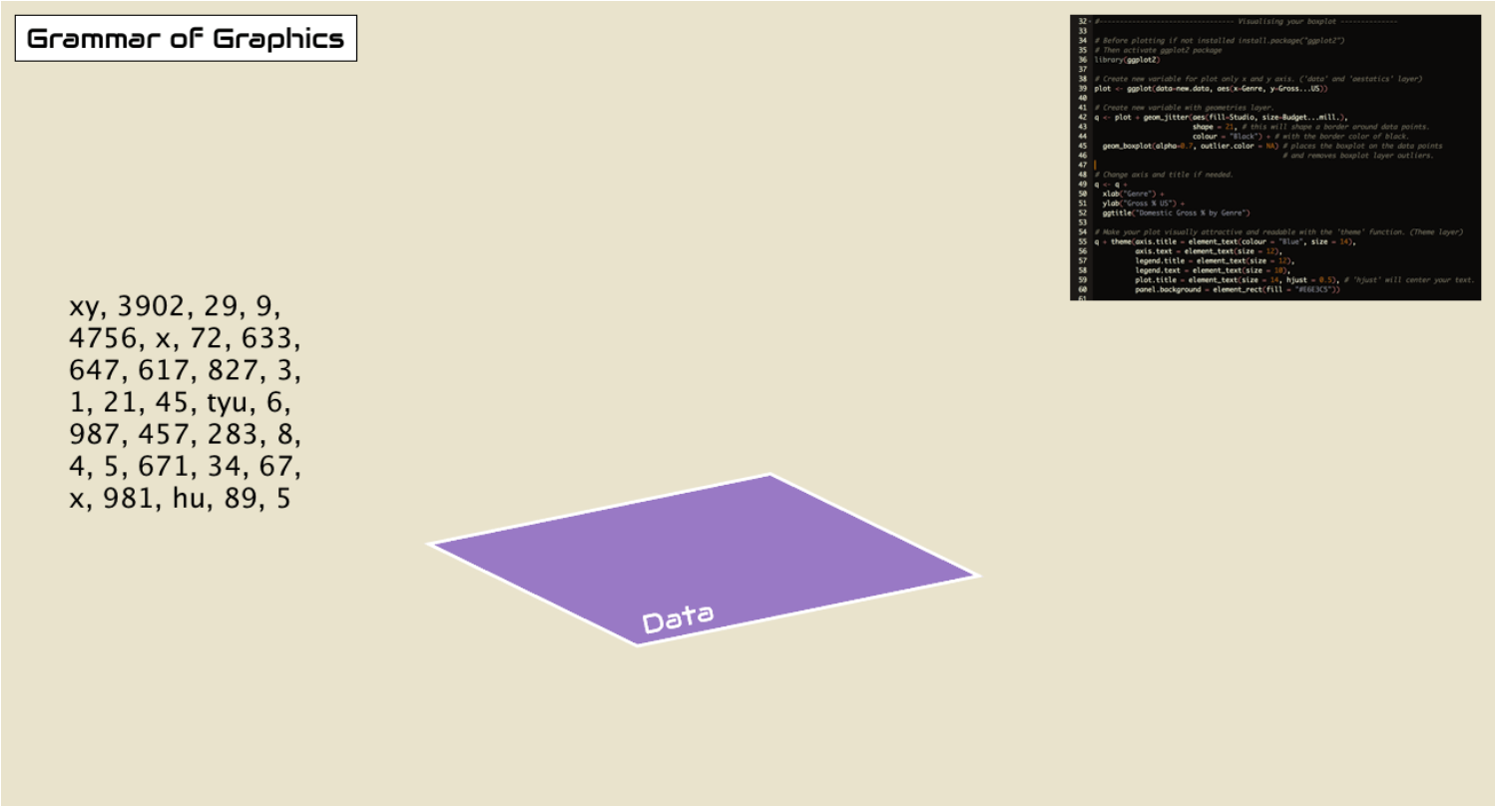


Figure source: Thomas de Beus. *Think About the Grammar of Graphics When Improving Your Graphs.* Medium, 2017.

First `ggplot` graph 1/3

Gapminder dataset: global health and economic data for 142 countries between 1952 and 2007 in increments of 5 years

```
library(dplyr)
library(gapminder)
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

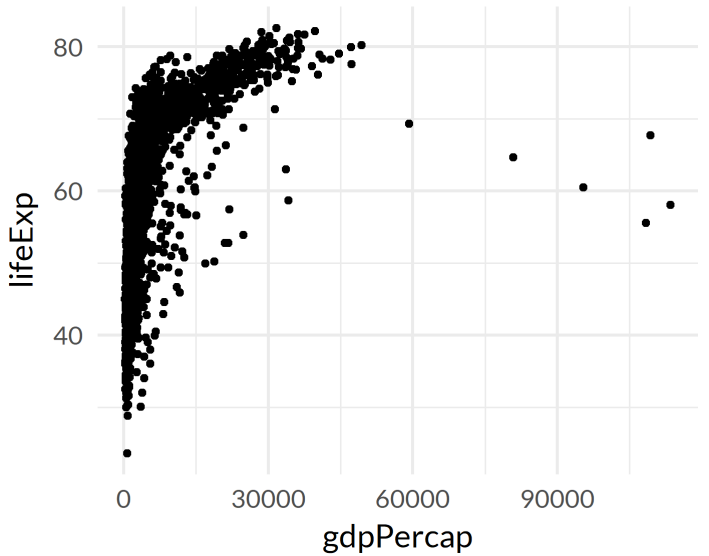
First `ggplot` graph 2/3

Gapminder dataset:

variable	meaning
country	factor with 142 levels
continent	factor with 5 levels
year	ranges from 1952 to 2007 in increments of 5 years
lifeExp	life expectancy at birth, in years
pop	population
gdpPercap	GDP per capita (US$, inflation-adjusted)

First `ggplot` graph 3/3

```
library(ggplot2)
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```



`ggplot2` terminology [1]

- **ggplot** - the main function: specify data frame

[1] `ggplot2` function reference

`ggplot2` terminology [1]

- **ggplot** - the main function: specify data frame
- **aes** - aesthetics: *mapping* of variables onto visual properties
    - examples: coordinates, shape, transparency, color, fill, linetype

[1] `ggplot2` function reference

`ggplot2` terminology [1]

- **ggplot** - the main function: specify data frame
- **aes** - aesthetics: *mapping* of variables onto visual properties
  - examples: coordinates, shape, transparency, color, fill, linetype
- **geoms** - geometric objects
  - examples: `geom_point()`, `geom_bar()`, `geom_line()`, `geom_histogram()`

[1] `ggplot2` function reference

`ggplot2` terminology [1]

- **ggplot** - the main function: specify data frame
- **aes** - aesthetics: *mapping* of variables onto visual properties
  - examples: coordinates, shape, transparency, color, fill, linetype
- **geoms** - geometric objects
  - examples: `geom_point()`, `geom_bar()`, `geom_line()`, `geom_histogram()`
- **stats** - aggregated data summaries
  - examples: regression lines, confidence intervals, summary statistics

[1] `ggplot2` function reference

`ggplot2` terminology [1]

- **ggplot** - the main function: specify data frame
- **aes** - aesthetics: *mapping* of variables onto visual properties
  - examples: coordinates, shape, transparency, color, fill, linetype
- **geoms** - geometric objects
  - examples: `geom_point()`, `geom_bar()`, `geom_line()`, `geom_histogram()`
- **stats** - aggregated data summaries
  - examples: regression lines, confidence intervals, summary statistics
- **scales** - scale modifications
  - examples: continuous, discrete, log-transformed, etc.

[1] `ggplot2` function reference

`ggplot2` terminology [1]

- **ggplot** - the main function: specify data frame
- **aes** - aesthetics: *mapping* of variables onto visual properties
  - examples: coordinates, shape, transparency, color, fill, linetype
- **geoms** - geometric objects
  - examples: `geom_point()`, `geom_bar()`, `geom_line()`, `geom_histogram()`
- **stats** - aggregated data summaries
  - examples: regression lines, confidence intervals, summary statistics
- **scales** - scale modifications
  - examples: continuous, discrete, log-transformed, etc.
- **facets** - spread data onto multiple subplots

[1] `ggplot2` function reference

`ggplot2` terminology [1]

- **ggplot** - the main function: specify data frame
- **aes** - aesthetics: *mapping* of variables onto visual properties
  - examples: coordinates, shape, transparency, color, fill, linetype
- **geoms** - geometric objects
  - examples: `geom_point()`, `geom_bar()`, `geom_line()`, `geom_histogram()`
- **stats** - aggregated data summaries
  - examples: regression lines, confidence intervals, summary statistics
- **scales** - scale modifications
  - examples: continuous, discrete, log-transformed, etc.
- **facets** - spread data onto multiple subplots
- **coordinates** - adjust the coordinate system

[1] `ggplot2` function reference

`ggplot2` terminology [1]

- **ggplot** - the main function: specify data frame
- **aes** - aesthetics: *mapping* of variables onto visual properties
  - examples: coordinates, shape, transparency, color, fill, linetype
- **geoms** - geometric objects
  - examples: `geom_point()`, `geom_bar()`, `geom_line()`, `geom_histogram()`
- **stats** - aggregated data summaries
  - examples: regression lines, confidence intervals, summary statistics
- **scales** - scale modifications
  - examples: continuous, discrete, log-transformed, etc.
- **facets** - spread data onto multiple subplots
- **coordinates** - adjust the coordinate system
- **themes** - additional non-data settings, e.g. font size or background color

[1] `ggplot2` function reference

aes()

- function `aes()` defines the **aesthetic mapping**, i.e., which variable is mapped onto the x-axis, y-axis, color attribute, etc.
- either specified in `ggplot()` function (**global** setting) or within a layer (**local** setting)

aes()

- function `aes()` defines the **aesthetic mapping**, i.e., which variable is mapped onto the x-axis, y-axis, color attribute, etc.
- either specified in `ggplot()` function (**global** setting) or within a layer (**local** setting)

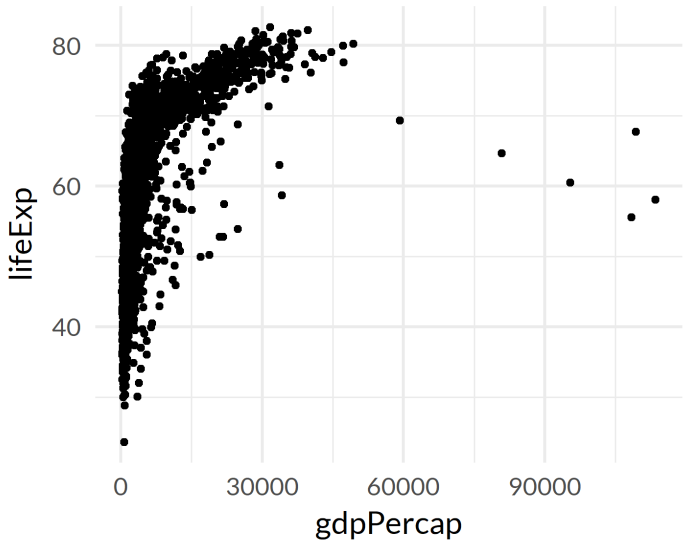
```
p ← ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))
```

aes()

- function `aes()` defines the **aesthetic mapping**, i.e., which variable is mapped onto the x-axis, y-axis, color attribute, etc.
- either specified in `ggplot()` function (**global** setting) or within a layer (**local** setting)

```
p ← ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))
```

```
# Add a scatterplot layer
p + geom_point()
```

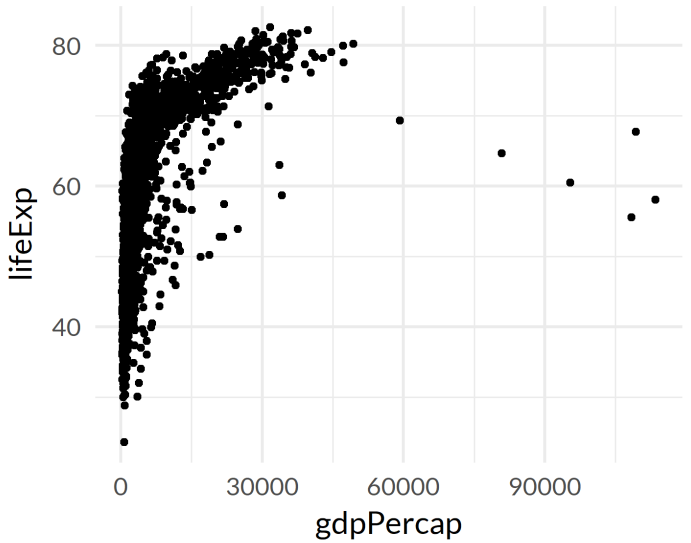


aes()

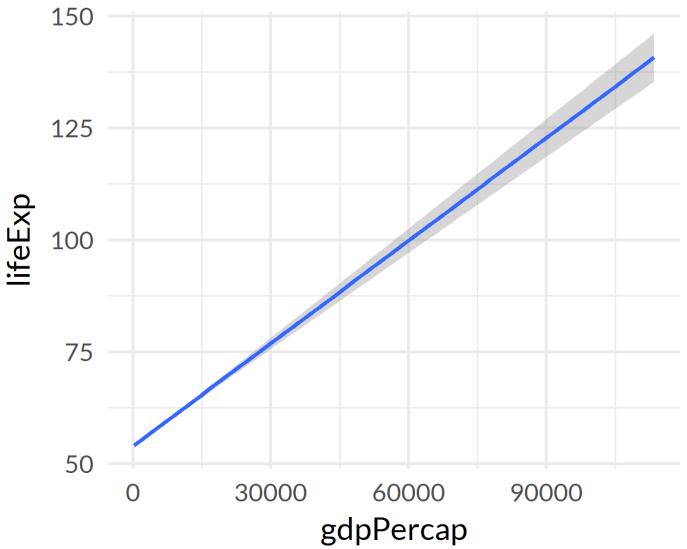
- function `aes()` defines the **aesthetic mapping**, i.e., which variable is mapped onto the x-axis, y-axis, color attribute, etc.
- either specified in `ggplot()` function (**global** setting) or within a layer (**local** setting)

```
p ← ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp))
```

```
# Add a scatterplot layer
p + geom_point()
```

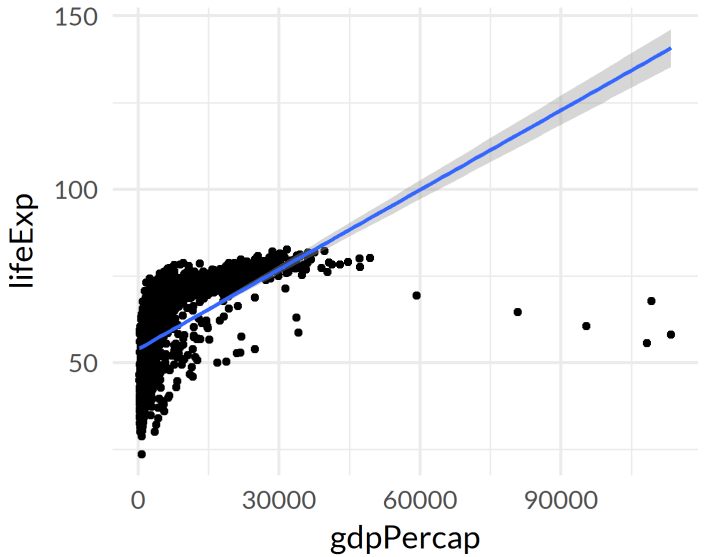



```
# Add a trend line layer
p + geom_smooth(method = "lm")
```



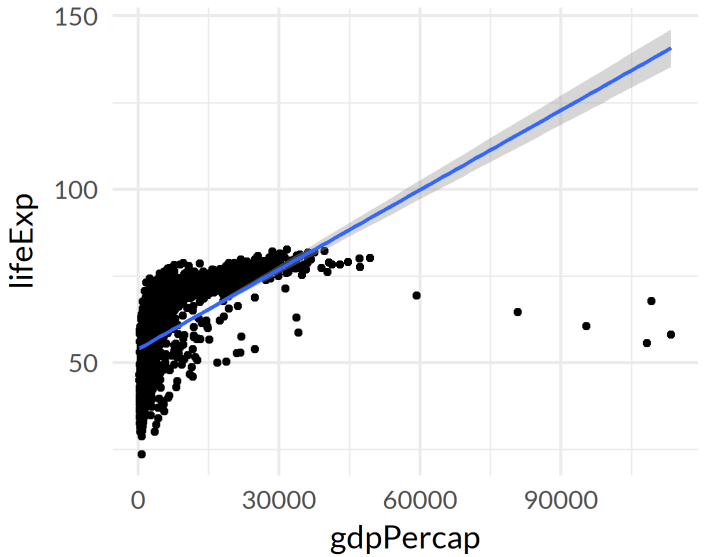
Adding multiple geoms layer by layer

```
# Add both scatterplot layer and trend line layer
p +
  geom_point() +
  geom_smooth(method = "lm")
```



Adding multiple geoms layer by layer

```
# Add both scatterplot layer and trend line layer
p +
  geom_point() +
  geom_smooth(method = "lm")
```



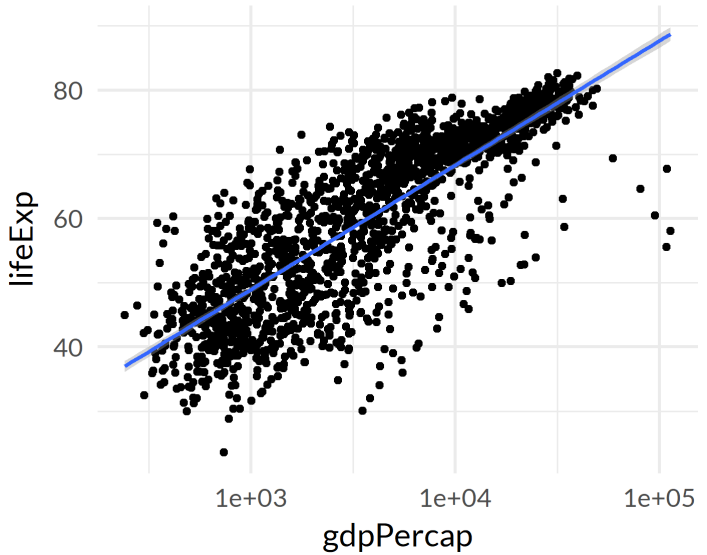
→ plot reveals log relationship between GDP per capita and life expectency.

→ improve visualization by log-transforming x-axis.

Adding multiple layers

Log-transform x-axis

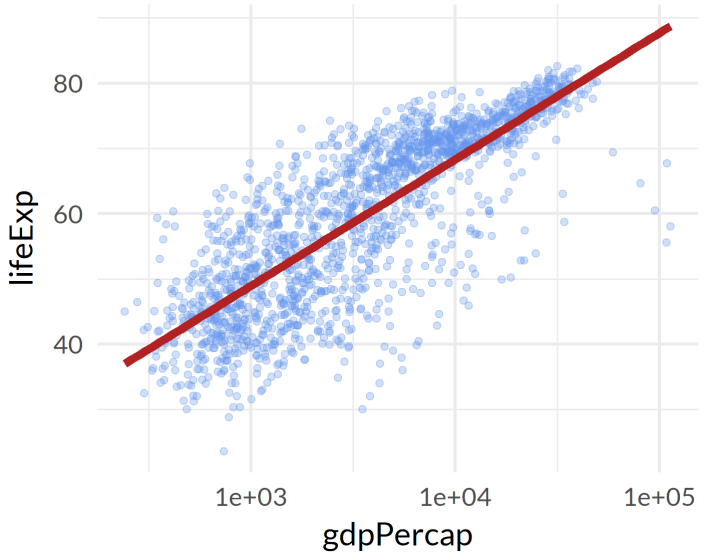
```
p +
  geom_point() +
  geom_smooth(method = "lm") +
  scale_x_log10()
```



Non-aesthetic layer arguments

Make some cosmetic adjustments

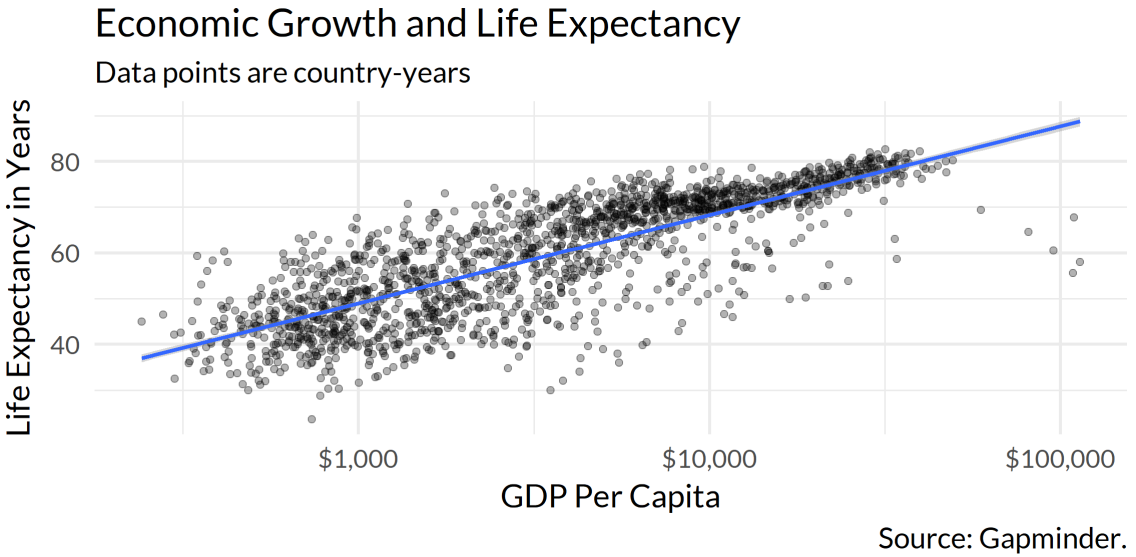
```
p +
  geom_point(alpha = 0.3, color = "cornflowerblue") +
  geom_smooth(method = "lm",
              color = "firebrick",
              se = FALSE,
              size = 2) +
  scale_x_log10()
```



Publication-ready version of the plot

- polish x-axis labels
- add axis and plot titles

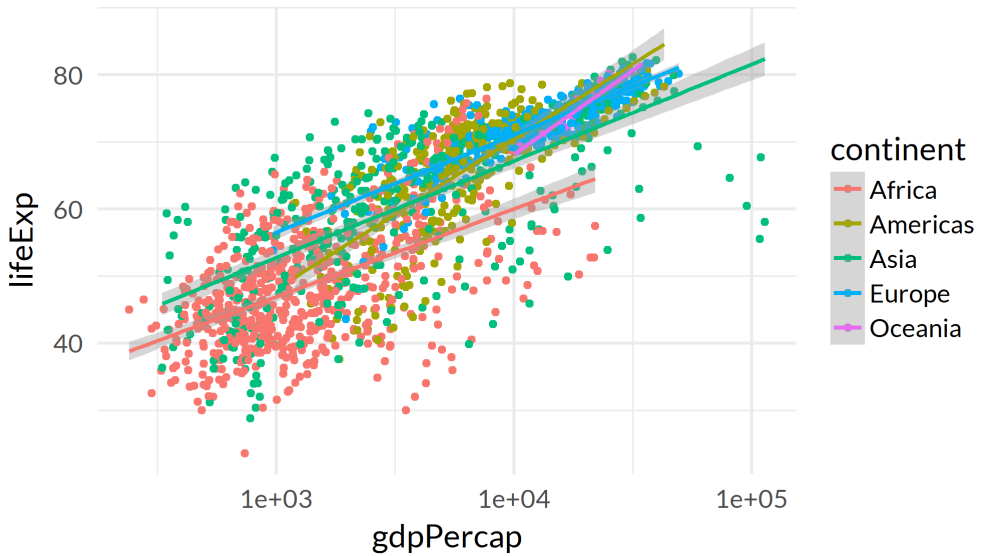
```
p + geom_point(alpha = 0.3) +
    geom_smooth(method = "lm") +
    scale_x_log10(labels = scales::dollar) +
    labs(x = "GDP Per Capita", y = "Life Expectancy in Years",
         title = "Economic Growth and Life Expectancy",
         subtitle = "Data points are country-years",
         caption = "Source: Gapminder.")
```



Mapping aesthetics per geom

Set aesthetics **globally** in `ggplot()` function or **individually** per layer in `geom_*()` function.

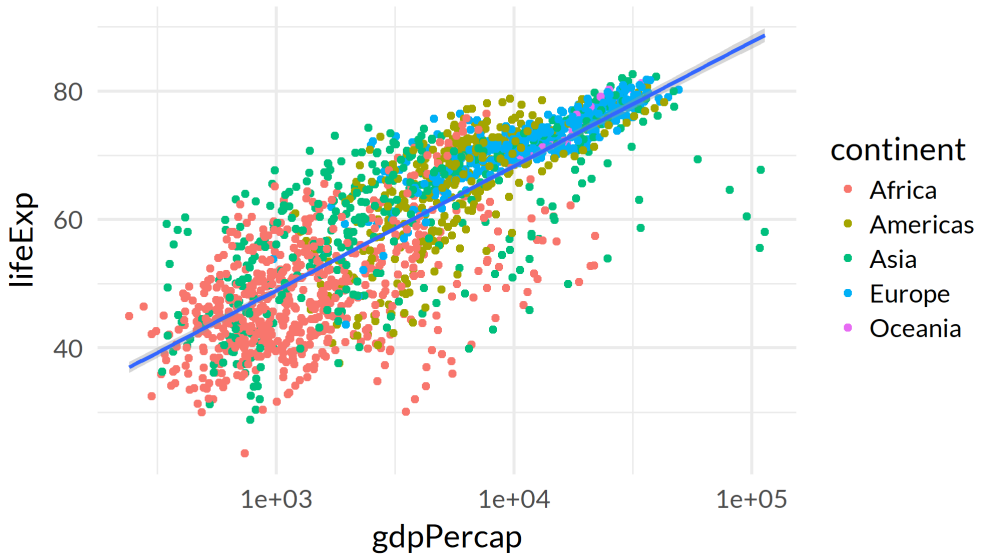
```
# continent is mapped to color for all underlying layers
p ← ggplot(data = gapminder,
           mapping = aes(x = gdpPercap, y = lifeExp, color = continent))
p + geom_point() +
    geom_smooth(method = "lm") +
    scale_x_log10()
```



Mapping aesthetics per geom

Set aesthetics **globally** in `ggplot()` function or **individually** per layer in `geom_*()` function.

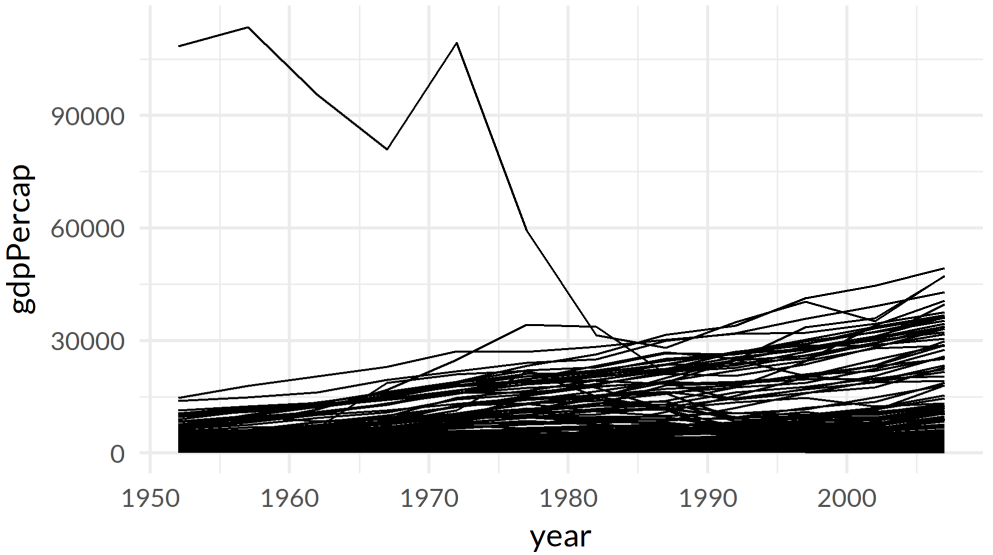
```
# continent is mapped to color only for scatterplot layer
p ← ggplot(data = gapminder,
           mapping = aes(x = gdpPercap, y = lifeExp))
p + geom_point(mapping = aes(color = continent)) +
    geom_smooth(method = "lm") +
    scale_x_log10()
```



Line plot

How did each country's financial capability change over time?

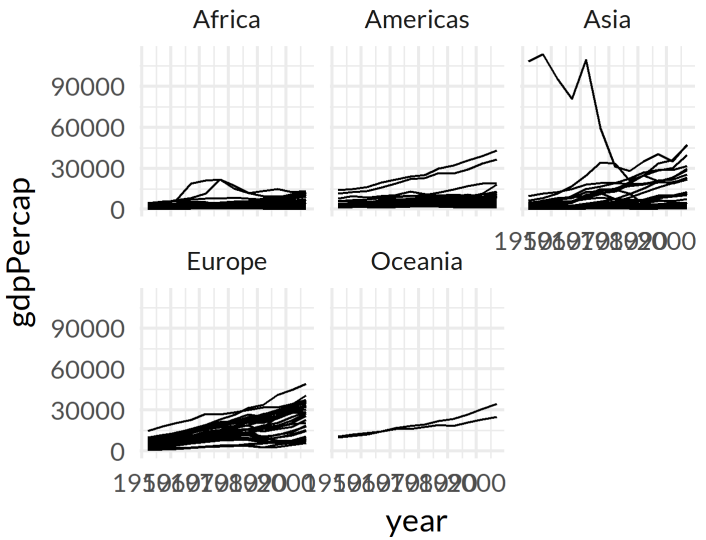
```
p ← ggplot(data = gapminder,
           mapping = aes(x = year, y = gdpPercap))
p + geom_line(aes(group = country))
```



Facetting 1/2

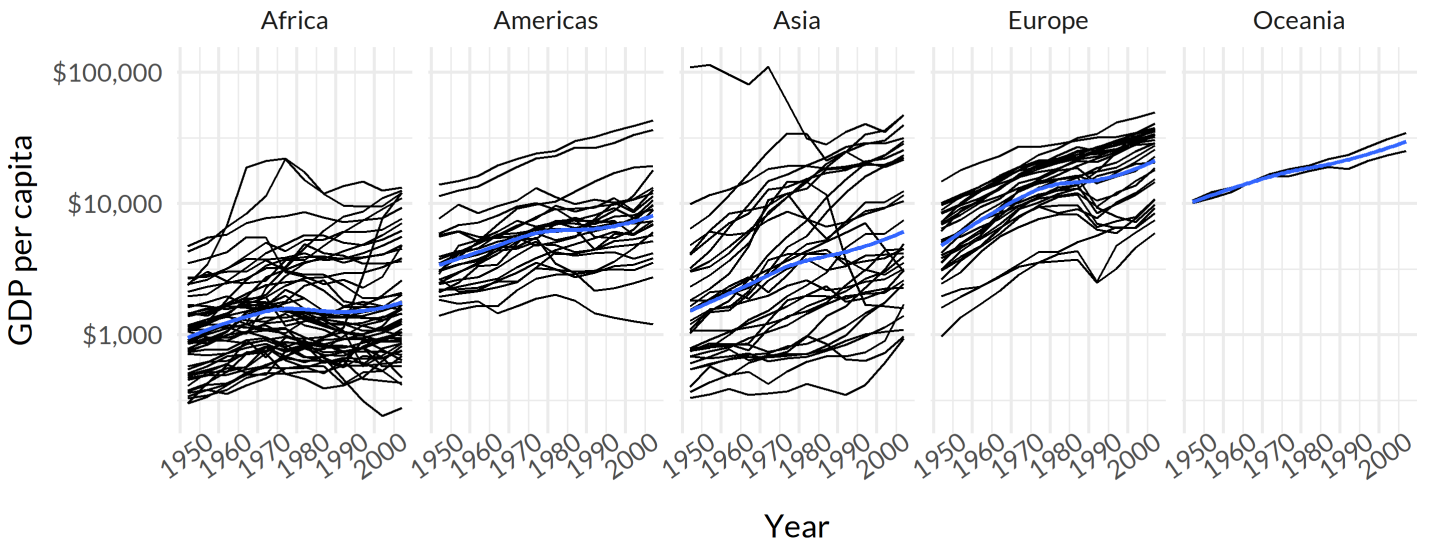
Compactly present a lot of information by **facetting** a plot, i.e., via stratifying by a third variable.

```
p ← ggplot(data = gapminder,
           mapping = aes(x = year, y = gdpPercap))
p + geom_line(aes(group = country)) +
  facet_wrap(~ continent)
```



Facetting 2/2

```
p ← ggplot(data = gapminder,
           mapping = aes(x = year,y = gdpPercap))
p + geom_line(aes(group = country)) +
  geom_smooth(method = "loess", se = FALSE) +
  scale_y_log10(labels = scales::dollar) +
  facet_wrap(~ continent, nrow = 1) +
  labs(x = "Year", y = "GDP per capita") +
  theme(axis.text.x = element_text(angle = 35))
```



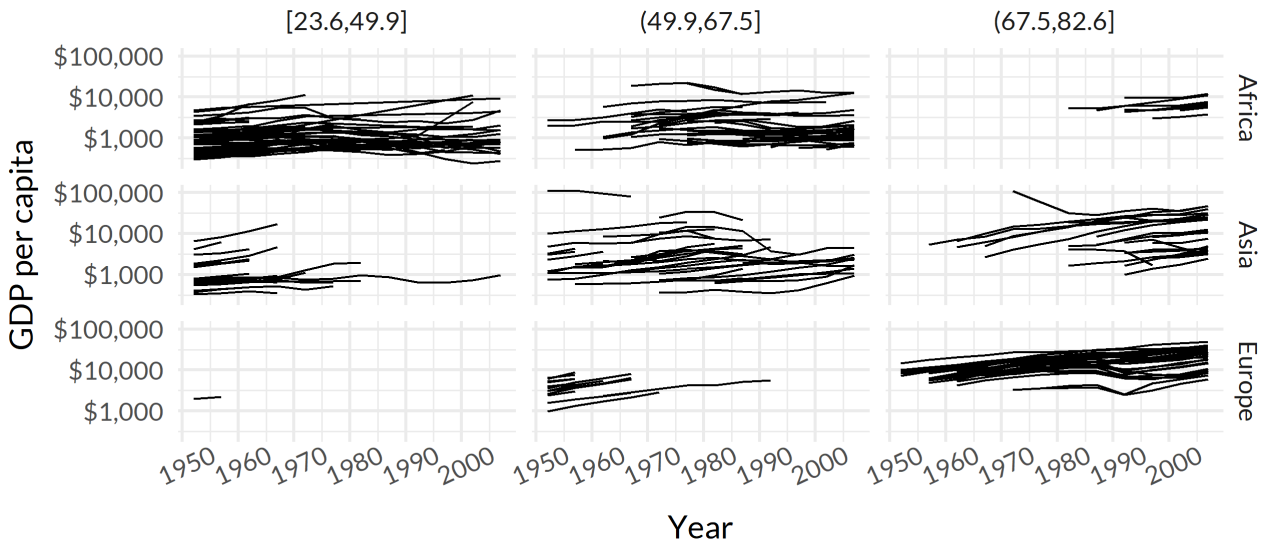
`facet_wrap()` vs. `facet_grid()` 1/2

- `facet_wrap()`: one categorical stratification variable
- `facet_grid()`: two or more categorical stratification variables

`facet_wrap()` vs. `facet_grid` 2/2

Show GDP development, stratified by life expectancy groups:

```
my_gapminder ← gapminder %>%
  filter(continent %in% c("Africa", "Asia", "Europe")) %>%
  mutate(lifeExp = cut_number(lifeExp, 3))
#Make three equally-sized groups based on life expectancy
p ← ggplot(data = my_gapminder, mapping = aes(x = year, y = gdpPercap))
p + geom_line(aes(group = country)) +
  scale_y_log10(labels = scales::dollar) +
  facet_grid(continent ~ lifeExp) +
  labs(x = "Year", y = "GDP per capita") +
  theme(axis.text.x = element_text(angle = 25))
```

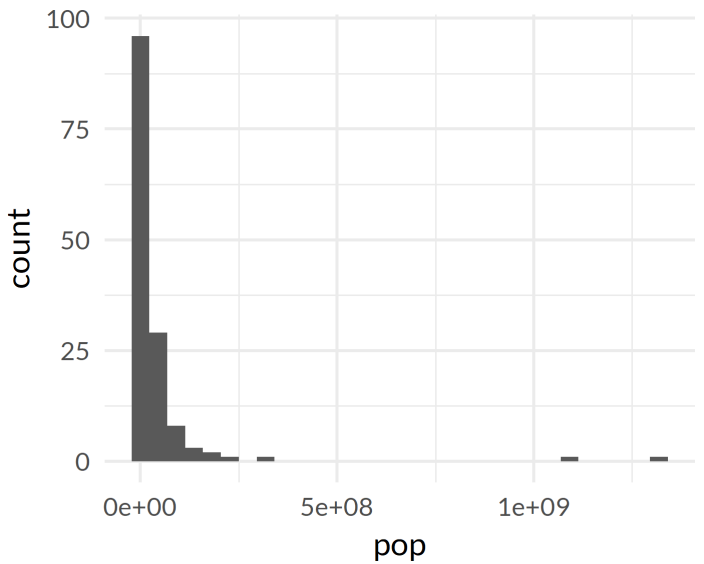


Simple histograms 1/3

Histogram that shows the population number distribution in 2007.

```
p ← ggplot(data = gapminder %>% filter(year == 2007),
           mapping = aes(x = pop))
p + geom_histogram()
```

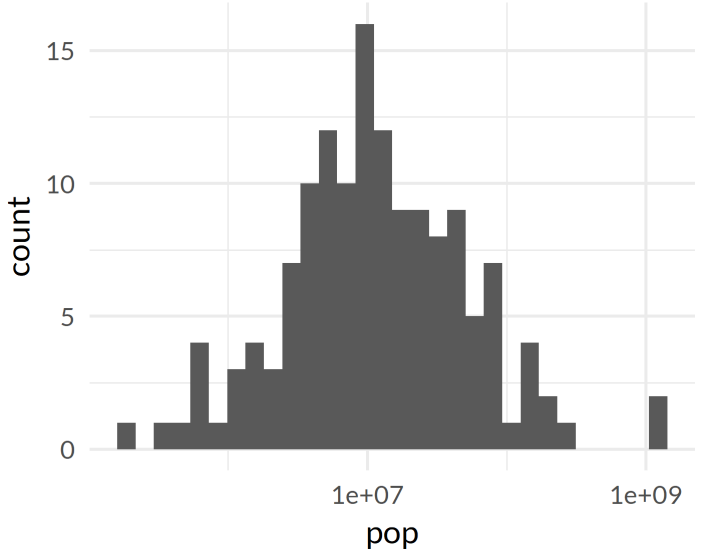
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Simple histograms 2/3

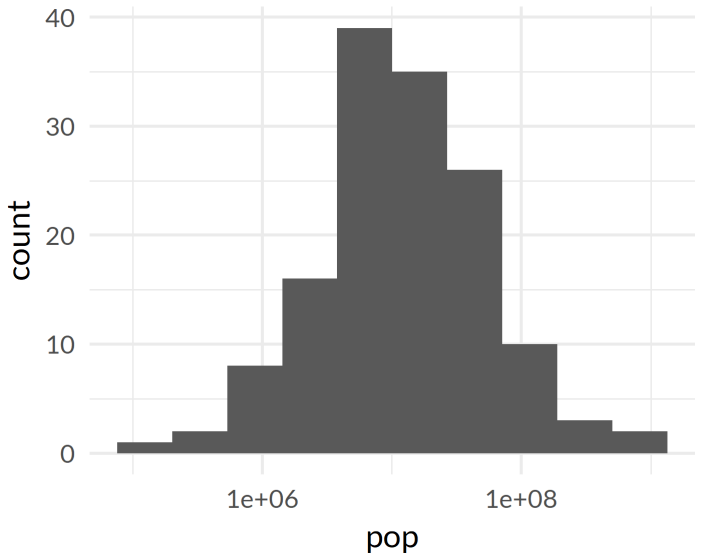
```
p ← ggplot(data = gapminder %>% filter(year == 2007),
           mapping = aes(x = pop))
p + geom_histogram() +
  scale_x_log10()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Simple histograms 3/3

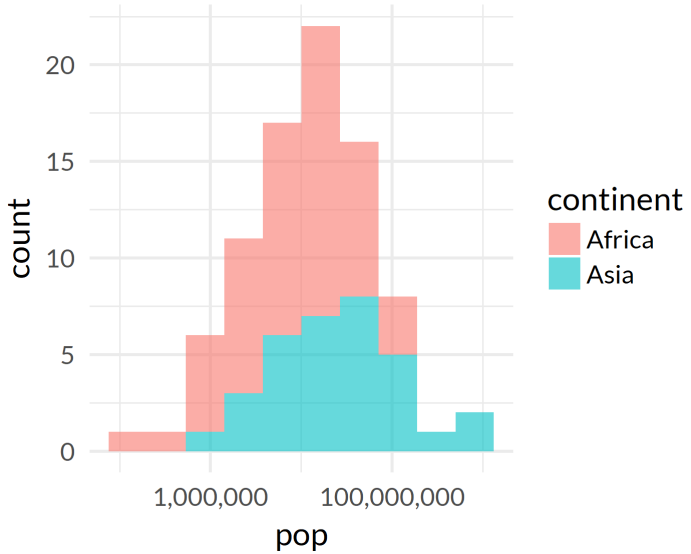
```
p ← ggplot(data = gapminder %>% filter(year == 2007),
           mapping = aes(x = pop))
p + geom_histogram(bins = 10) +
  scale_x_log10()
```



Multiple distributions with histograms 1/3

Histograms for the population numbers of African and Asian countries in 2007.

```
my_gapminder ← gapminder %>%
  filter(year == 2007,
         continent %in% c("Africa", "Asia"))
p ← ggplot(data = my_gapminder,
           mapping = aes(x = pop,
                         fill = continent))
p + geom_histogram(bins = 10, alpha = 0.6) +
  scale_x_log10(labels = scales::comma)
```



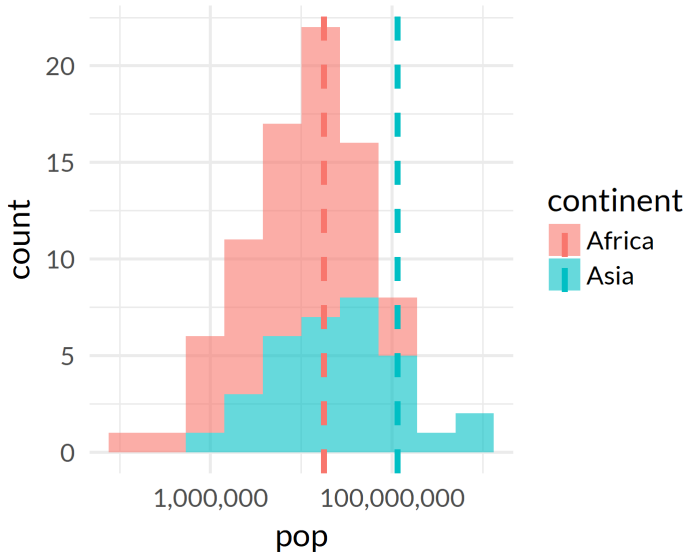
Multiple distributions with histograms 2/3

```
# Calculate mean per continent
my_gapminder_summary ← my_gapminder %>%
  group_by(continent) %>%
  summarize(pop_mean = mean(pop))
my_gapminder_summary
```

```
## # A tibble: 2 x 2
##   continent   pop_mean
##   <fct>          <dbl>
## 1 Africa     17875763.
## 2 Asia      115513752.
```

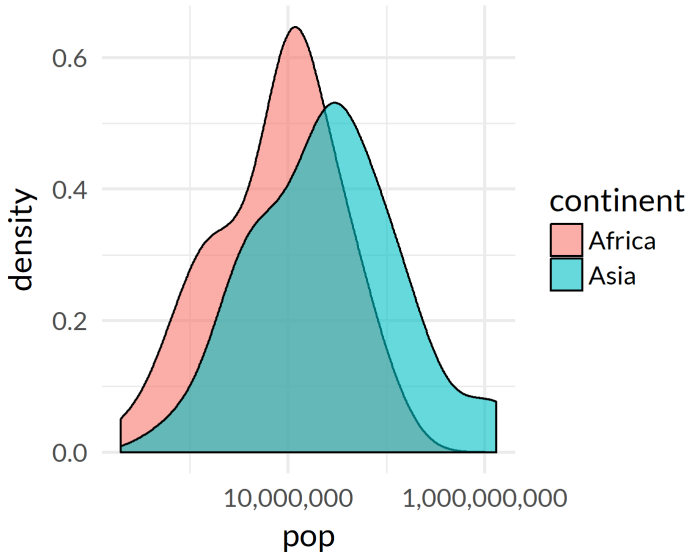
Multiple distributions with histograms 3/3

```
p ← ggplot(data = my_gapminder,
           mapping = aes(x = pop, fill = continent))
p + geom_histogram(bins = 10, alpha = 0.6) +
  scale_x_log10(labels = scales::comma) +
  geom_vline(data = my_gapminder_summary,
             aes(xintercept = pop_mean, color = continent),
             linetype = 2, size = 1.5)
```



Multiple distributions with density plots

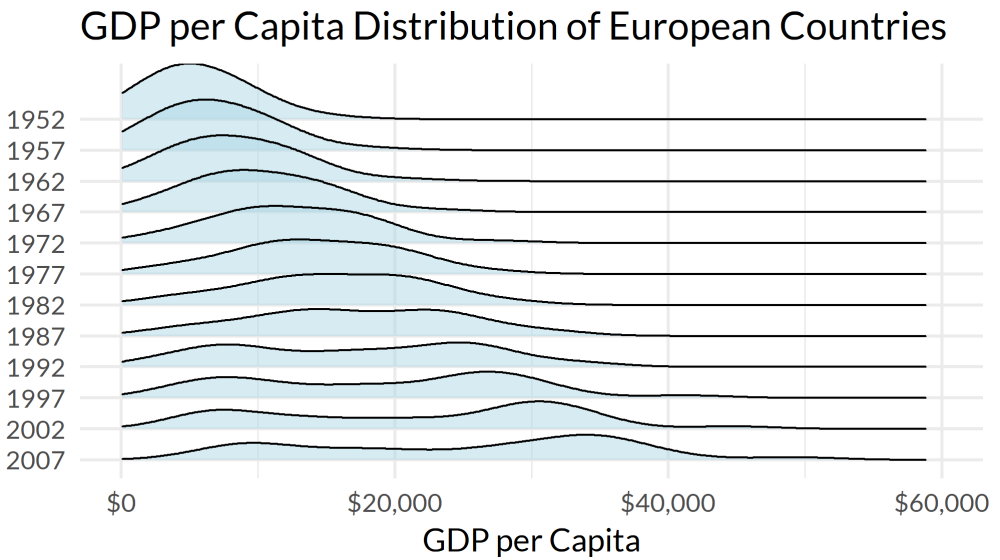
```
my_gapminder ← gapminder %>%
  filter(year == 2007,
         continent %in% c("Africa", "Asia"))
p ← ggplot(data = my_gapminder,
           mapping = aes(x = pop, fill = continent))
p + geom_density(alpha = .6) +
  scale_x_log10(labels = scales::comma)
```



Multiple distributions with ridgeplots

Visualize Europe's GDP per capita distribution over time!

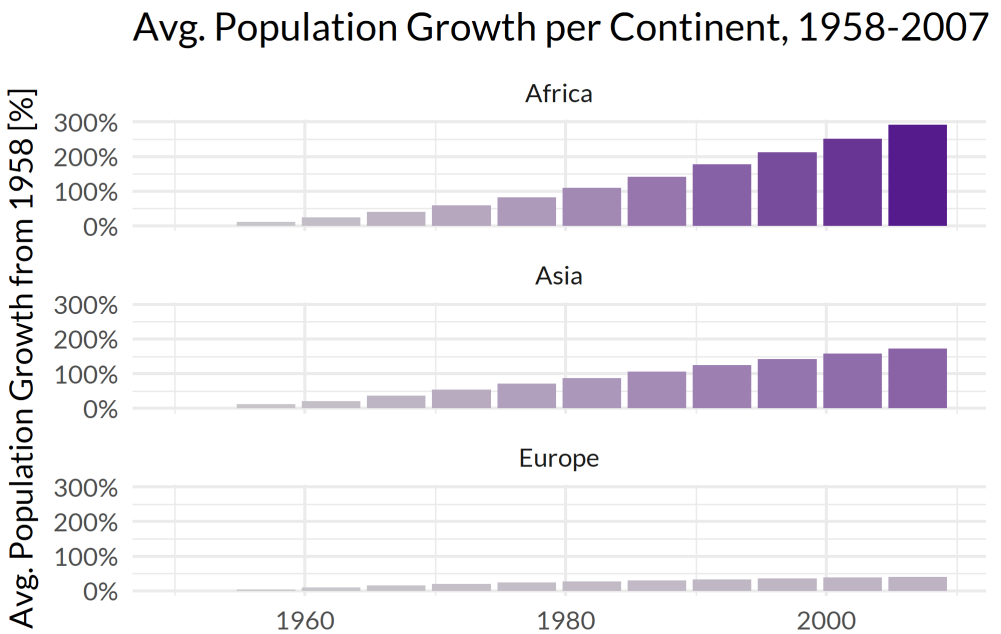
```
library(ggridges)

ggplot(data = gapminder %>% filter(continent == "Europe"),
       mapping = aes(x = gdpPercap,
                     y = factor(year, levels = rev(unique(year))))) +
  geom_density_ridges(alpha = .5, fill = "lightblue") +
  scale_x_continuous(limits = c(0, 60000), labels = scales::dollar) +
  labs(x = "GDP per Capita", y = NULL,
       title = "GDP per Capita Distribution of European Countries")
```



Multiple distributions with barcharts 1/5

For each of the continents Asia, Africa and Europe, show the percental average population growth from 1952 to 2007 in a facetted plot!

Anticipated result:



Multiple distributions with barcharts 2/5

For each of the continents Asia, Africa and Europe, show the percental average population growth from 1952 to 2007 in a facetted plot!

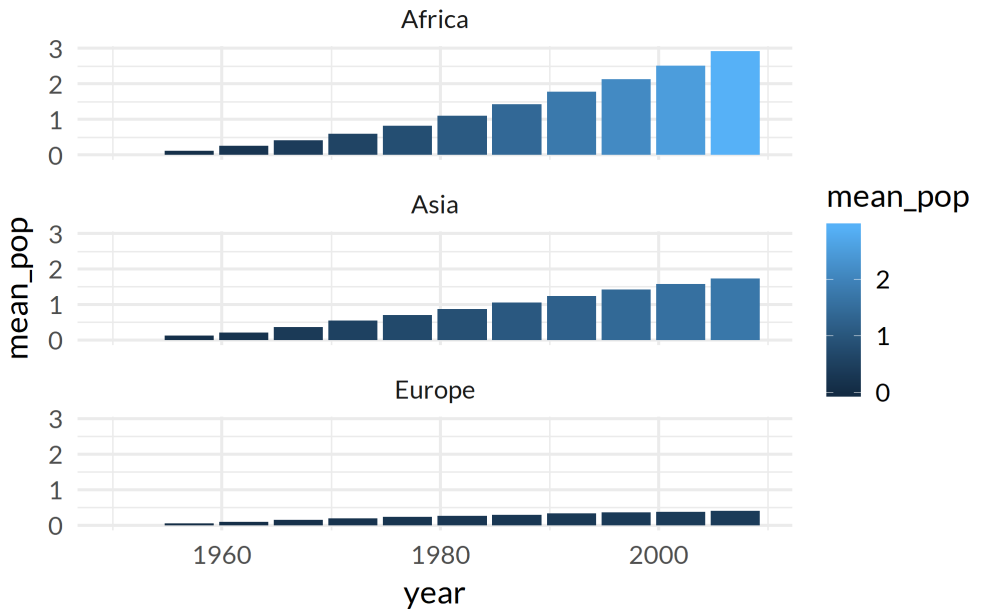
```
# Prepare Data
my_gapminder ← gapminder %>%
  filter(continent %in% c("Africa", "Asia", "Europe")) %>%
  group_by(continent, year) %>%
  summarize(mean_pop = mean(pop)) %>%
  ungroup() %>%
  group_by(continent) %>%
  mutate(mean_pop = (mean_pop - mean_pop[1]) / mean_pop[1]) %>%
  ungroup()

head(my_gapminder, 6)
```

```
## # A tibble: 6 x 3
##   continent  year mean_pop
##   <fct>     <int>    <dbl>
## 1 Africa     1952    0.
## 2 Africa     1957    0.114
## 3 Africa     1962    0.248
## 4 Africa     1967    0.411
## 5 Africa     1972    0.599
## 6 Africa     1977    0.822
```

Multiple distributions with barcharts 3/5

For each of the continents Asia, Africa and Europe, show the percental average population growth from 1952 to 2007 in a facetted plot!

```
# Plot data
ggplot(data = my_gapminder,
       mapping = aes(x = year, y = mean_pop, fill = mean_pop)) +
  geom_col() +
  facet_wrap(~ continent, ncol = 1)
```



Multiple distributions with barcharts 4/5

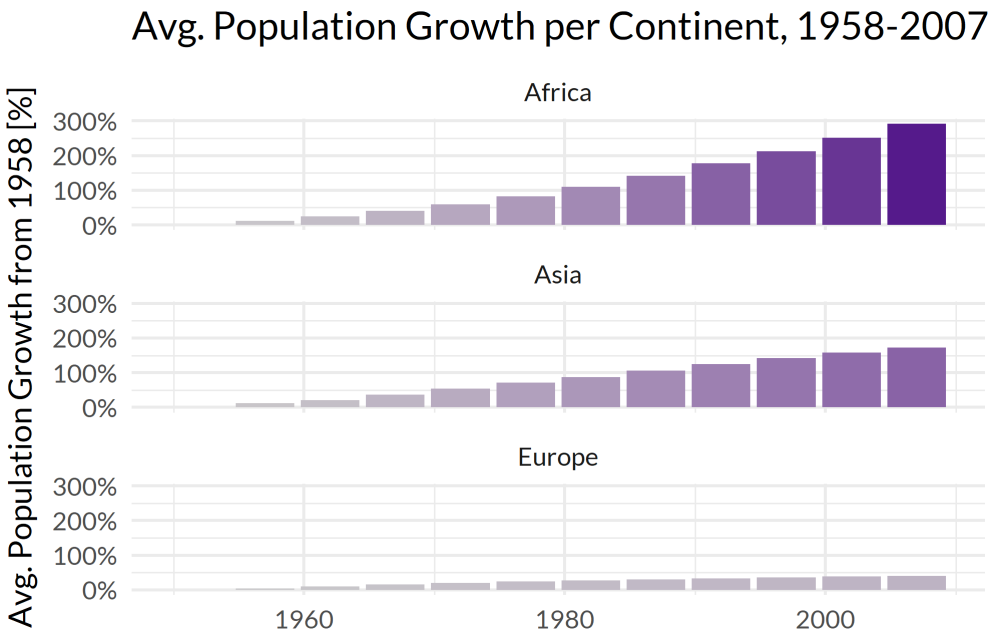
For each of the continents Asia, Africa and Europe, show the percental average population growth from 1952 to 2007 in a facetted plot!

```
# Apply cosmetic modfications
ggplot(data = my_gapminder,
       mapping = aes(x = year, y = mean_pop, fill = mean_pop)) +
  geom_col() +
  facet_wrap(~ continent, ncol = 1) +
  scale_y_continuous(labels = scales::percent) +
  scale_fill_continuous(low = "grey80", high = "purple4") +
  guides(fill = FALSE) +
  labs(x = NULL, y = "Avg. Population Growth from 1958 [%]",
       title = "Avg. Population Growth per Continent, 1958-2007") -> p
```

Multiple distributions with barcharts 5/5

For each of the continents Asia, Africa and Europe, show the percental average population growth from 1952 to 2007 in a facetted plot!

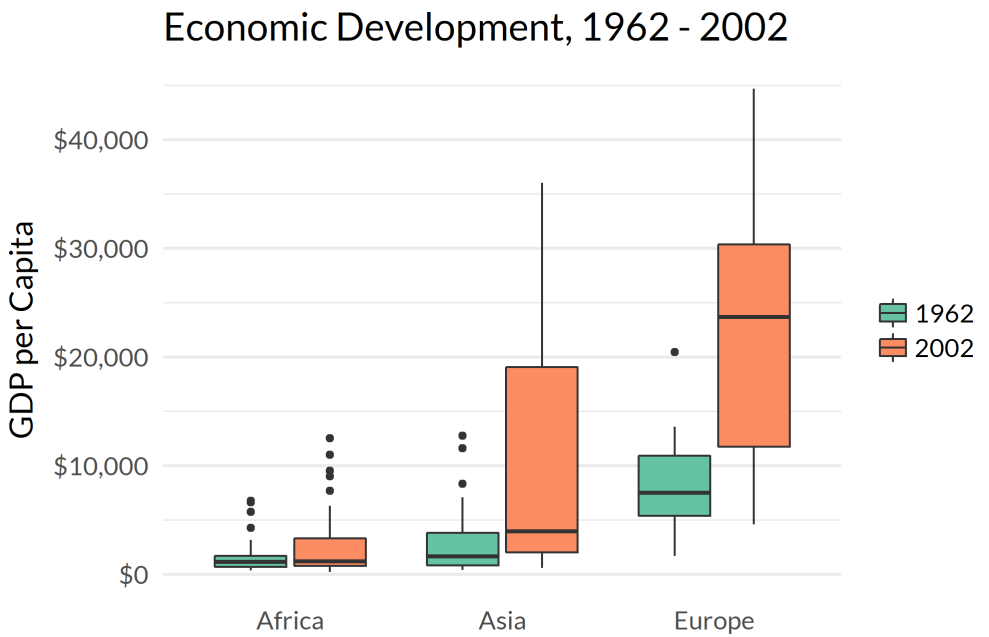
```
p
```



Multiple distributions with boxplots 1/7

For each of the continents Asia, Africa and Europe, juxtapose the GDP per capita distribution of 1962 with 2002!

Anticipated output:



Multiple distributions with boxplots 2/7

For each of the continents Asia, Africa and Europe, juxtapose the GDP per capita distribution of 1962 with 2002!

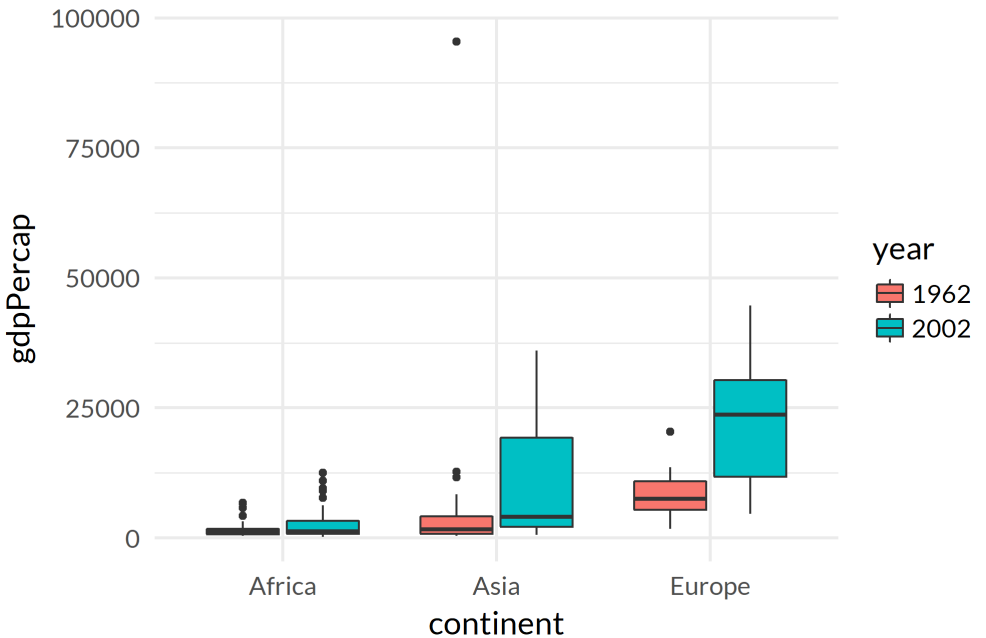
```
# Prepare data
my_gapminder ← gapminder %>%
  filter(continent %in% c("Asia", "Africa", "Europe"),
         year %in% c(1962, 2002)) %>%
  mutate(year = factor(year))

head(my_gapminder, 6)
```

```
## # A tibble: 6 x 6
##   country     continent year  lifeExp      pop gdpPercap
##   <fct>       <fct>     <fct>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia      1962     32.0 10267083      853.
## 2 Afghanistan Asia      2002     42.1 25268405      727.
## 3 Albania     Europe    1962     64.8  1728137     2313.
## 4 Albania     Europe    2002     75.7  3508512     4604.
## 5 Algeria     Africa    1962     48.3 11000948     2551.
## 6 Algeria     Africa    2002     71.0 31287142     5288.
```

Multiple distributions with boxplots 3/7

For each of the continents Asia, Africa and Europe, juxtapose the GDP per capita distribution of 1962 with 2002!

```
# Plot data
ggplot(data = my_gapminder,
       mapping = aes(x = continent, y = gdpPercap, fill = year)) +
  geom_boxplot()
```



Multiple distributions with boxplots 4/7

For each of the continents Asia, Africa and Europe, juxtapose the GDP per capita distribution of 1962 with 2002!

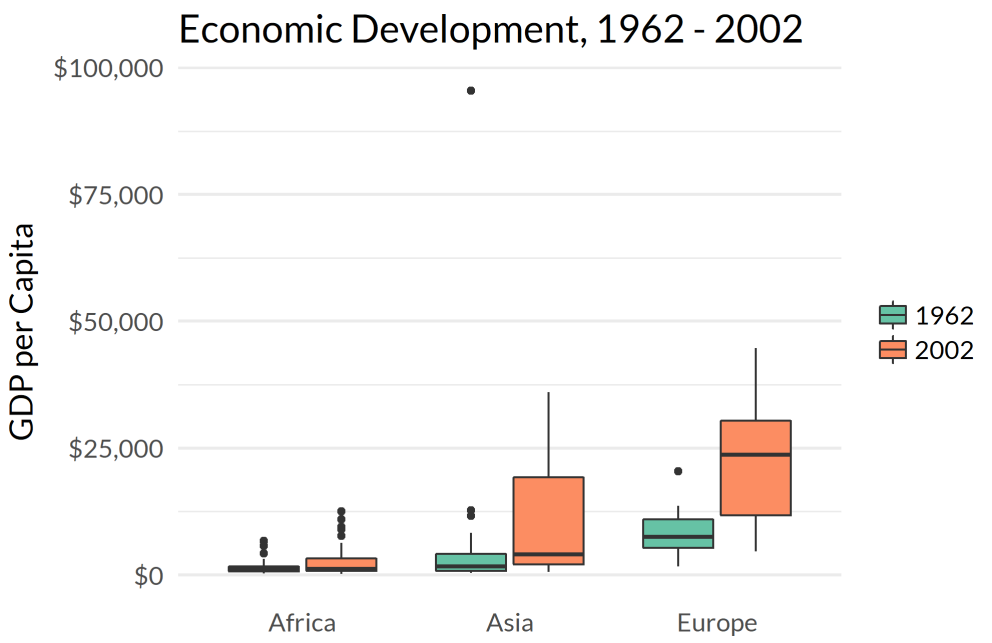
```
# Apply cosmetic modifications
ggplot(data = my_gapminder,
       mapping = aes(x = continent, y = gdpPercap, fill = year)) +
  geom_boxplot() +
  scale_y_continuous(labels = scales::dollar) +
  scale_fill_brewer(palette = "Set2") +
  labs(x = NULL, y = "GDP per Capita", fill = NULL,
       title = "Economic Development, 1962 - 2002") +
  theme(panel.grid.minor.x = element_blank()) +
  theme(panel.grid.major.x = element_blank()) -> p
```

Package for ColorBrewer palettes RColorBrewer

Multiple distributions with boxplots 5/7

For each of the continents Asia, Africa and Europe, juxtapose the GDP per capita distribution of 1962 with 2002!

```
p
```



Multiple distributions with boxplots 6/7

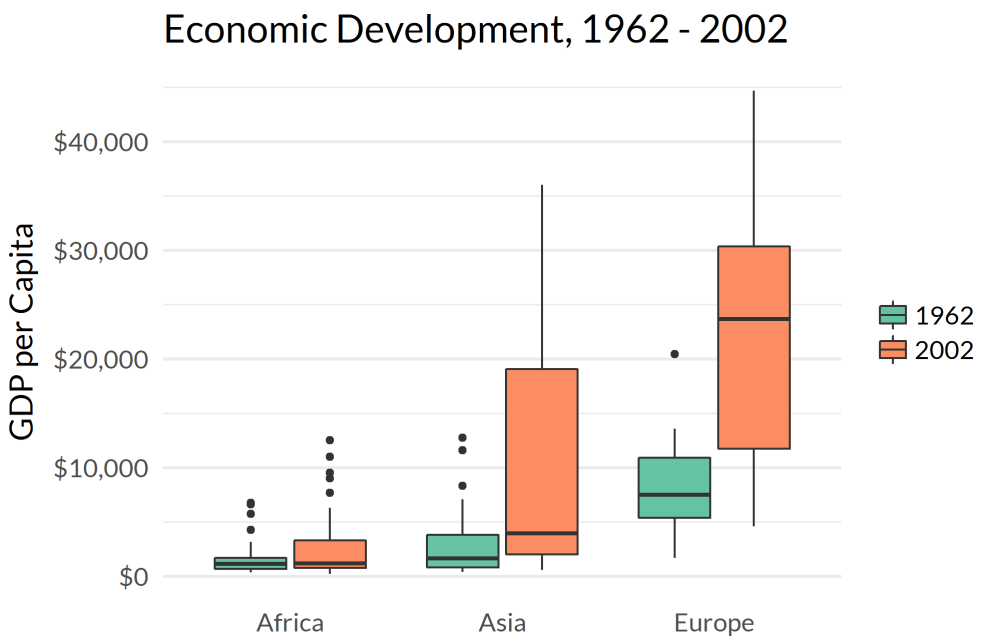
For each of the continents Asia, Africa and Europe, juxtapose the GDP per capita distribution of 1962 with 2002!

```
# Remove outlier Kuwait
my_gapminder ← my_gapminder %>%
  filter(country ≠ "Kuwait")
ggplot(data = my_gapminder,
       mapping = aes(x = continent, y = gdpPercap, fill = year)) +
  geom_boxplot() +
  scale_y_continuous(labels = scales::dollar) +
  scale_fill_brewer(palette = "Set2") +
  labs(x = NULL, y = "GDP per Capita", fill = NULL,
       title = "Economic Development, 1962 - 2002") +
  theme(panel.grid.minor.x = element_blank()) +
  theme(panel.grid.major.x = element_blank()) → p
```

Multiple distributions with boxplots 7/7

For each of the continents Asia, Africa and Europe, juxtapose the GDP per capita distribution of 1962 with 2002!

```
p
```



Extracting plot details 1/2

Whenever a ggplot object is printed to the screen, the function `ggplot_build()` is invoked to render the plot.

```
class(p)
```

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

Extracting plot details 1/2

Whenever a ggplot object is printed to the screen, the function `ggplot_build()` is invoked to render the plot.

```
class(p)
```

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

```
gr ← ggplot_build(p) # execute all neccessary steps to render the plot
class(gr)
```

```
## [1] "ggplot_built"
```

```
names(gr)
```

```
## [1] "data"   "layout" "plot"
```

- `data`: details for each plot layer, e.g. the 5-point summary of a boxplot.
- `layout`: axis information, e.g. breaks, ranges and labels
- `plot`: the rendered plot itself

Extracting plot details 2/2

```
gr$data[[1]] # or: layer_data(p, i = 1)
```

```
##      fill      ymin      lower    middle     upper      ymax
## 1 #66C2A5  355.2032   673.3290  1133.784  1702.219  3173.723
## 2 #FC8D62  241.1659   780.5778  1215.683  3314.887  6316.165
## 3 #66C2A5  388.0000   820.0531  1635.623  3802.884  7105.631
## 4 #FC8D62  611.0000  2010.6485  3967.921 19069.403 36023.105
## 5 #66C2A5 1709.6837  5373.5366  7515.734 10931.085 13583.314
## 6 #FC8D62 4604.2117 11721.8515 23674.863 30373.363 44683.975
##                                                    outliers notchupper
## 1                          4269.277, 6631.459, 6757.031, 5768.730   1359.220
## 2 11003.605, 7703.496, 12521.714, 9534.677, 9021.816, 7710.946   1770.967
## 3                                12753.275, 8341.738, 11626.420   2468.749
## 4                                                                 8732.554
## 5                                                    20431.09   9118.904
## 6                                                               29055.213
##   notchlower      x PANEL group ymin_final ymax_final    xmin    xmax
## 1   908.3471 0.8125     1     1   355.2032   6757.031 0.64375 0.98125
## 2   660.3994 1.1875     1     2   241.1659  12521.714 1.01875 1.35625
## 3   802.4968 1.8125     1     3   388.0000  12753.275 1.64375 1.98125
## 4  -796.7113 2.1875     1     4   611.0000  36023.105 2.01875 2.35625
## 5  5912.5631 2.8125     1     5  1709.6837  20431.093 2.64375 2.98125
## 6 18294.5136 3.1875     1     6  4604.2117  44683.975 3.01875 3.35625
##   weight colour size alpha shape linetype
## 1      1 grey20  0.5    NA    19    solid
## 2      1 grey20  0.5    NA    19    solid
## 3      1 grey20  0.5    NA    19    solid
## 4      1 grey20  0.5    NA    19    solid
## 5      1 grey20  0.5    NA    19    solid
## 6      1 grey20  0.5    NA    19    solid
```

Maps 1/2

Example **choropleth maps** (*Flächenkartogramme*) showing the poll results of the 2016 United States Presidental Elections:

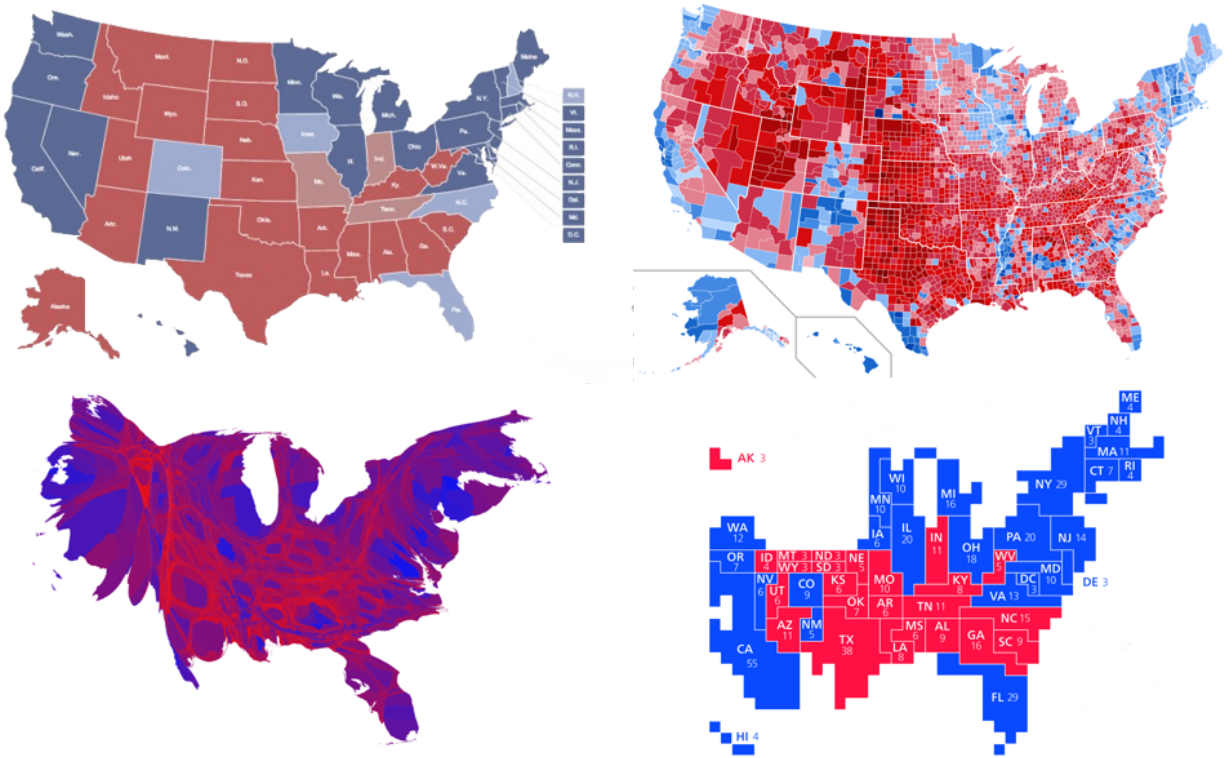


Figure source: Kieran Healy. *Data Visualization. A practical introduction.* Princeton University Press, 2018. URL: http://socviz.co/

Maps 2/2

Draw a map of the USA:

```
library(maps)
library(ggmap)
usa ← map_data("usa")
str(usa)
```

```
## 'data.frame':    7243 obs. of  6 variables:
## $ long : num -101 -101 -101 -101 -101 ...
## $ lat : num 29.7 29.7 29.7 29.6 29.6 ...
## $ group : num 1 1 1 1 1 1 1 1 1 1 ...
## $ order : int 1 2 3 4 5 6 7 8 9 10 ...
## $ region : chr "main" "main" "main" "main" ...
## $ subregion: chr NA NA NA NA ...
```

Maps 2/2

Draw a map of the USA:

```
library(maps)
library(ggmap)
usa ← map_data("usa")
str(usa)
```

```
## 'data.frame':    7243 obs. of  6 variables:
## $ long : num -101 -101 -101 -101 -101 ...
## $ lat : num 29.7 29.7 29.7 29.6 29.6 ...
## $ group : num 1 1 1 1 1 1 1 1 1 1 ...
## $ order : int 1 2 3 4 5 6 7 8 9 10 ...
## $ region : chr "main" "main" "main" "main" ...
## $ subregion: chr NA NA NA NA ...
```

```
p ← ggplot(usa,
           aes(x = long,
               y = lat,
               group = group)) +
  geom_polygon() +
  coord_map() +
  theme_void()
```

Draw maps from shape files 1/4

The `maps` package contains map data only for a handful of countries, including USA, France, Italy and New Zealand, as well as 2 world maps.

Draw maps from shape files 1/4

The `maps` package contains map data only for a handful of countries, including USA, France, Italy and New Zealand, as well as 2 world maps.

Generally, using **shapefiles** is more flexible in accessing geographic and political boundaries than built-in maps. A shapefile is geospatial vector data format for geographic information system (GIS) software.

Draw maps from shape files 1/4

The `maps` package contains map data only for a handful of countries, including USA, France, Italy and New Zealand, as well as 2 world maps.

Generally, using **shapefiles** is more flexible in accessing geographic and political boundaries than built-in maps. A shapefile is geospatial vector data format for geographic information system (GIS) software.

Shapefiles actually consist of several sub-files, see e.g. Wikipedia.

Draw maps from shape files 2/4

Draw a map showing the unemployment rates of each federal state of Germany[1]!

Available levels:

- `DEU_adm0`: administrative (political) boundaries of the entire country
- `DEU_adm1`: administrative boundaries for each of the 16 states
- `DEU_adm2`: administrative boundaries for each of the 42 *kreisfreie Städte*
- `DEU_adm3`: administrative boundaries for each of the 435 districts (*Landkreise*)

Draw maps from shape files 2/4

Draw a map showing the unemployment rates of each federal state of Germany[1]!

Available levels:

- `DEU_adm0`: administrative (political) boundaries of the entire country
- `DEU_adm1`: administrative boundaries for each of the 16 states
- `DEU_adm2`: administrative boundaries for each of the 42 *kreisfreie Städte*
- `DEU_adm3`: administrative boundaries for each of the 435 districts (*Landkreise*)

```
library(rgdal)
germany ← readOGR(dsn = "Data/DEU_adm", # folder of shapefiles
                  layer = "DEU_adm1", # level of interest
                  use_iconv = TRUE, # convert input strings to ...
                  # ... native system encoding
                  encoding = "UTF-8", # set encoding explicitely
                  verbose = FALSE)
```

[1] Download shapefiles for Germany from DIVA-GIS

Draw maps from shape files 3/4

```
class(germany)
```

```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
# convert to dataframe
df_germany ← fortify(germany)
```

```
## Regions defined for each Polygons
```

```
head(df_germany)
```

```
##       long      lat order  hole piece id group
## 1 9.650460 49.77634     1 FALSE     1  0   0.1
## 2 9.650968 49.76515     2 FALSE     1  0   0.1
## 3 9.656839 49.76145     3 FALSE     1  0   0.1
## 4 9.640400 49.75014     4 FALSE     1  0   0.1
## 5 9.652028 49.74276     5 FALSE     1  0   0.1
## 6 9.652208 49.73903     6 FALSE     1  0   0.1
```

Draw maps from shape files 4/4

```
ggplot(df_germany,
       aes(x = long, y = lat,
           group = group)) +
    geom_polygon(fill = "steelblue2",
                 col = "white") +
    coord_map() +
    theme_void()
```

Unemployment data taken from Wikipedia (November 2017). Download of unemployment data `unemp`

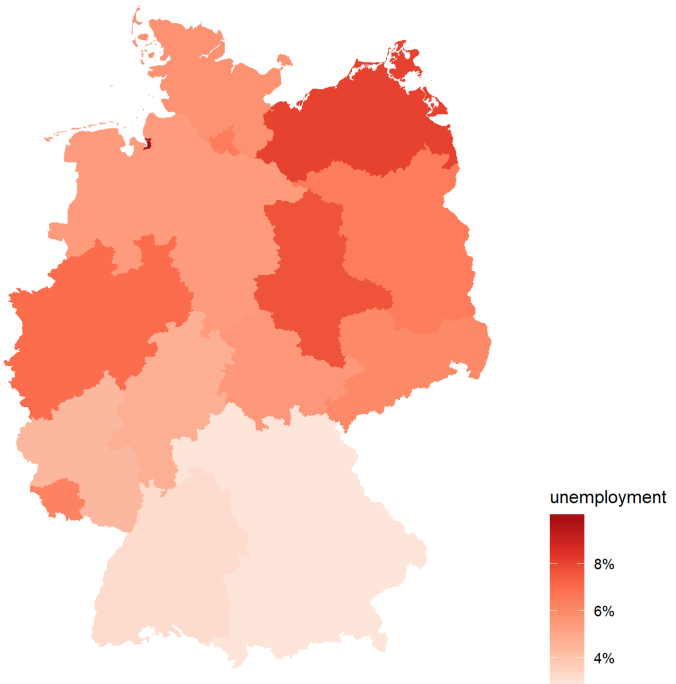
Draw maps from shape files 4/4

```
ggplot(df_germany,
       aes(x = long, y = lat,
           group = group)) +
    geom_polygon(fill = "steelblue2",
                 col = "white") +
    coord_map() +
    theme_void()
```

```
# merge df_germany with unemp via state
df_germany ← df_germany %>%
  mutate(id = as.numeric(id)) %>%
  inner_join(unemp, by = "id")
```

Unemployment data taken from Wikipedia (November 2017). Download of unemployment data `unemp`

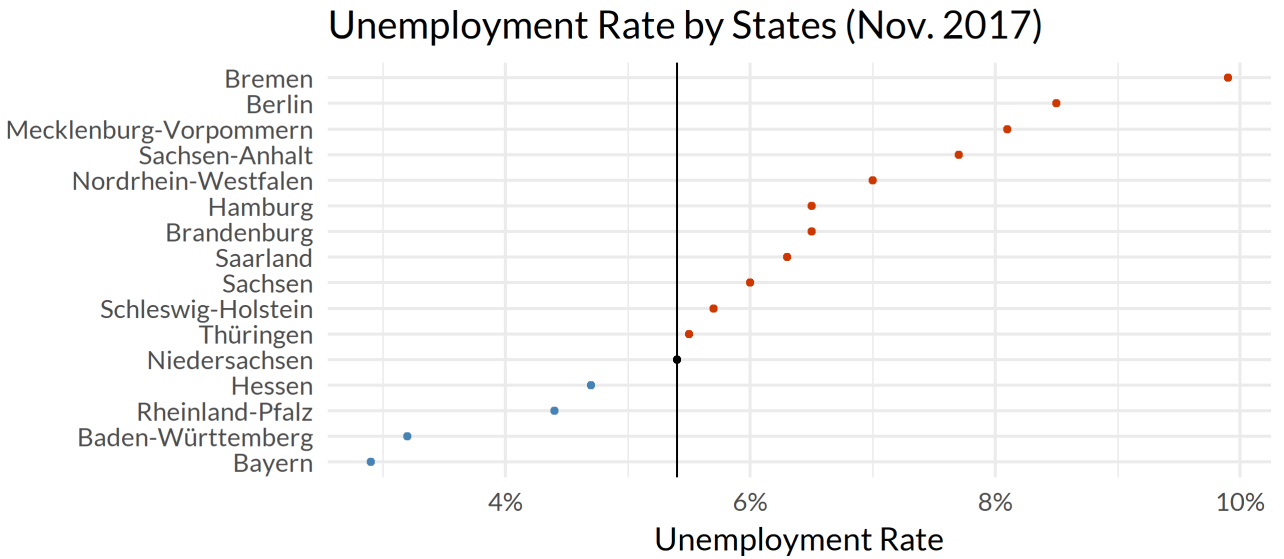
```r
ggplot(df_germany, aes(x = long, y = lat, group = group,
                       fill = unemployment)) +
  geom_polygon() +
  coord_map() +
  scale_fill_distiller(palette = "Reds", direction = 1,
                       labels = scales::percent) +
  ggthemes::theme_map() +
  theme(legend.position = "right")
```



Maps vs. dotplots 1/2

Spatial data doesn't always have to be represented spatially.

For example, it can be more informative to show a **dotplot**.



Maps vs. dotplots 2/2

```
library(forcats)
avg_germany ← 0.054
unemp %>%
  mutate(below_above_avg = case_when(unemployment < avg_germany ~ "01_below",
                                     unemployment > avg_germany ~ "02_above",
                                     TRUE ~ "03_equal")) %>%
  mutate(state = fct_reorder(state, unemployment)) %>%
  ggplot(mapping = aes(x = state, y = unemployment,
                       color = below_above_avg)) +
  geom_point(size = 1.5) +
  geom_hline(yintercept = avg_germany) +
  scale_y_continuous(labels = scales::percent) +
  coord_flip() +
  guides(color = FALSE) +
  scale_color_manual(values = c("steelblue", "orangered3", "black")) +
  labs(x = NULL, y = "Unemployment Rate",
       title = "Unemployment Rate by States (Nov. 2017)") → p
```

Cartographic maps using Google Maps 1/4

Draw a map of Magdeburg and annotate 11 sights as labeled points!

```
magdeburg_sites ← c("Fakultaet fuer Informatik, Magdeburg",
                    "Dom, Magdeburg",
                    "Jahrtausendturm, Magdeburg",
                    "Gruene Zitadelle, Magdeburg",
                    "Rathaus, Magdeburg",
                    "Elbauenpark, Magdeburg",
                    "Gruson Gewaechshaeuser, Magdeburg",
                    "Trogbruecke, Magdeburg",
                    "Schiffshebewerk, Magdeburg",
                    "Magdeburger Zoo, Magdeburg",
                    "Johanniskirche, Magdeburg")
# Get longitude and latitude from Google Maps
xx ← geocode(magdeburg_sites)
# Remove ', Magdeburg' suffix from location names
xx$location ← str_replace(magdeburg_sites, ", Magdeburg", "")
```

Cartographic maps using Google Maps 2/4

Draw a map of Magdeburg and annotate 11 sights as labeled points!

```
head(xx)
```

```
## # A tibble: 6 x 3
##     lon   lat location
##   <dbl> <dbl> <chr>
## 1  11.6  52.1 Fakultaet fuer Informatik
## 2  11.6  52.1 Dom
## 3  11.7  52.1 Jahrtausendturm
## 4  11.6  52.1 Gruene Zitadelle
## 5  11.6  52.1 Rathaus
## 6  11.7  52.1 Elbauenpark
```

Cartographic maps using Google Maps 3/4

Draw a map of Magdeburg and annotate 11 sights as labeled points!

```
# Create bounding box: bbox
bbox ← make_bbox(lon = xx$lon, lat = xx$lat, f = 0.5)
# Get map
magdeburg_loc ← get_map(location = bbox, zoom = 12,
                        scale = 2,
                        source = "google", maptype = "hybrid")
```

```
magdeburg_loc
```

```
## 1280×1280 hybrid map image from Google Maps.  see ?ggmap to plot it.
```

Cartographic maps using Google Maps 4/4

Draw a map of Magdeburg and annotate 11 sights as labeled points

```
# Draw map with labeled points
ggmap(magdeburg_loc) +
  geom_point(data = xx, color = "white", size =3) +
  ggrepel::geom_label_repel(data = xx, aes(label = location), size = 2,
             fontface = "bold", fill = "grey90",
             col = "red") +
  theme_void()
```

`ggplot2` extensions 1/5

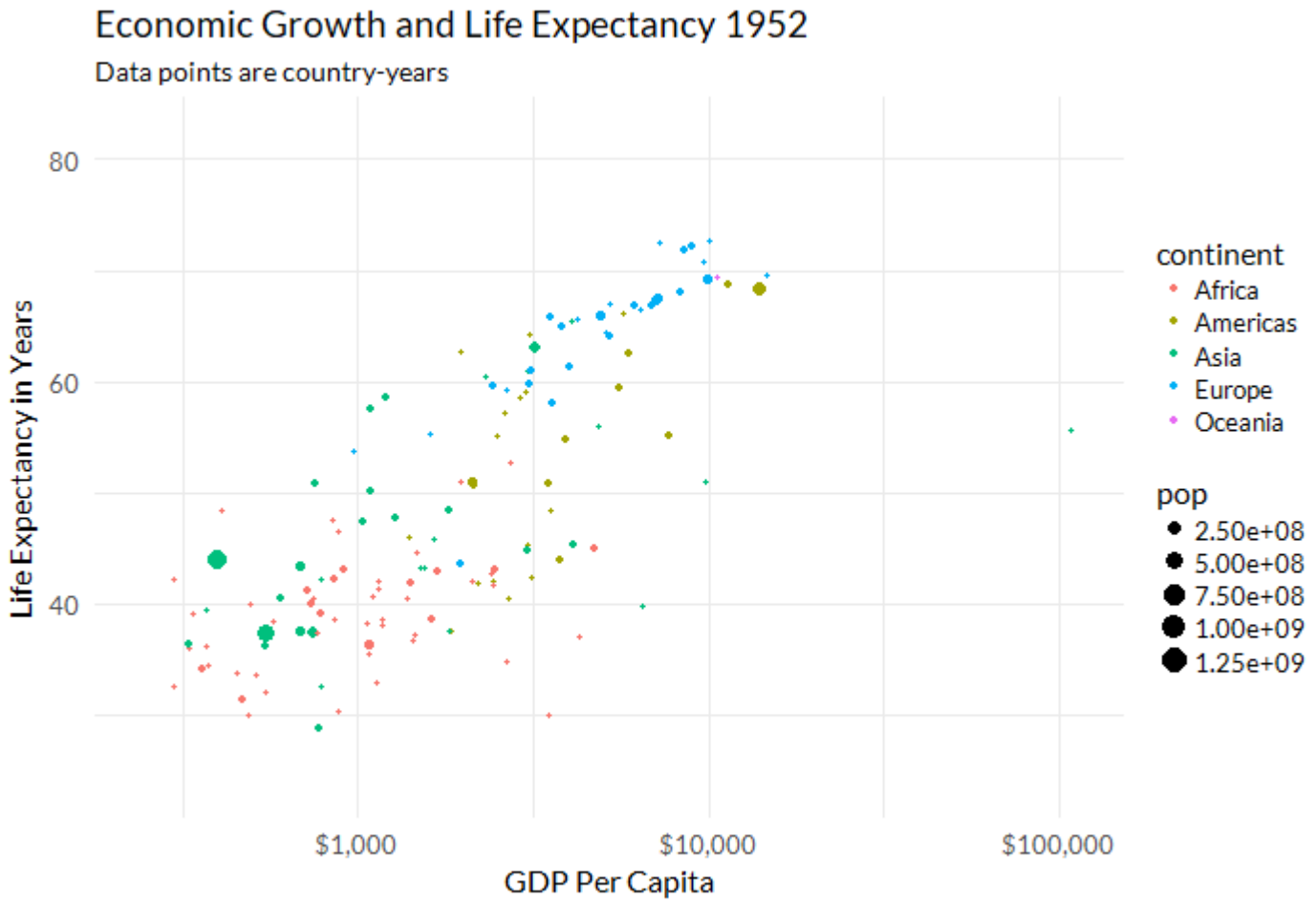
`plotly`[1]: interface to Javascript library for interactive visualizations

- includes convinience function `ggplotly()` for translating `ggplot2` graphs

[1] URL: https://plot.ly/r

`ggplot2` extensions 2/5

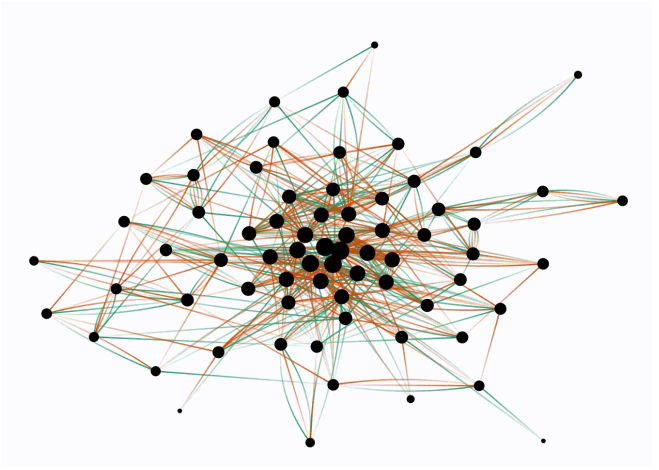
`gganimate`[1]: `ggplot2` with animations

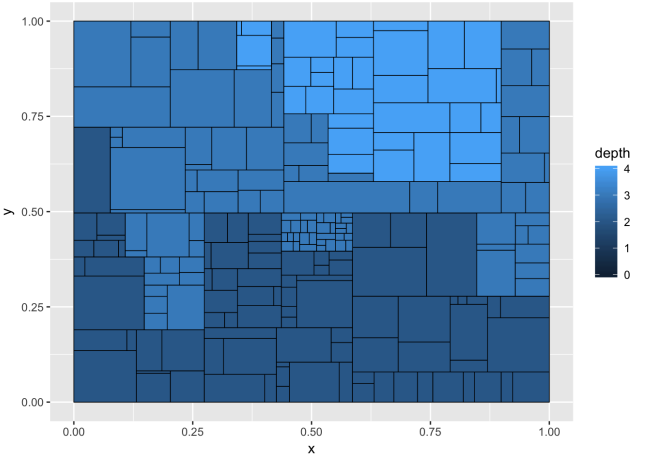


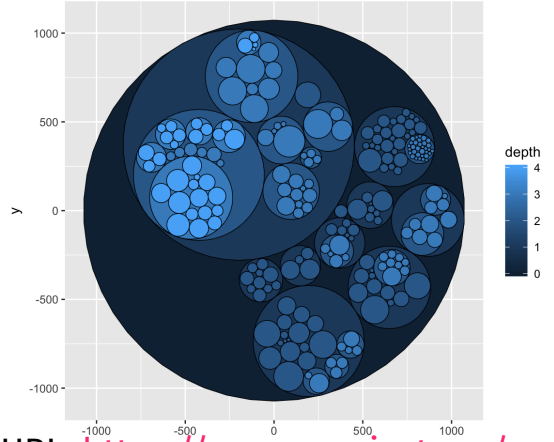
[1] URL: https://github.com/dgrtwo/gganimate

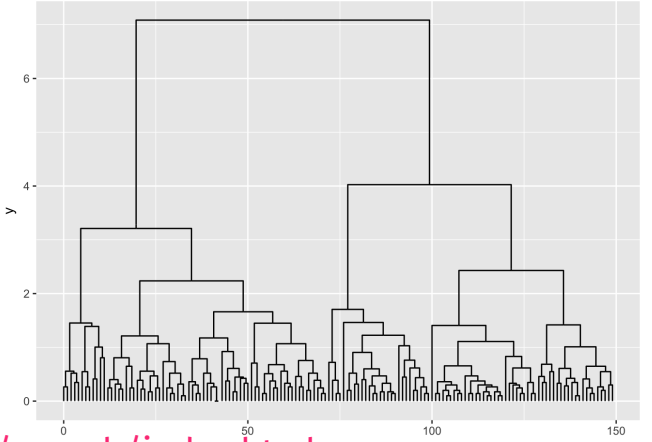
`ggplot2` extensions 3/5

`ggraph`: graph and network visualizations[1]





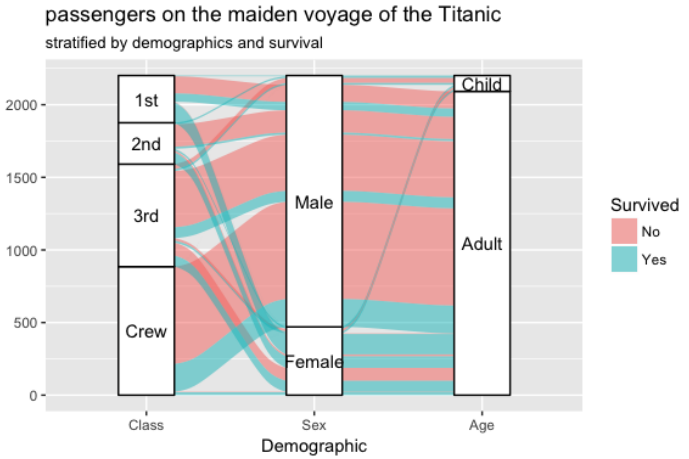


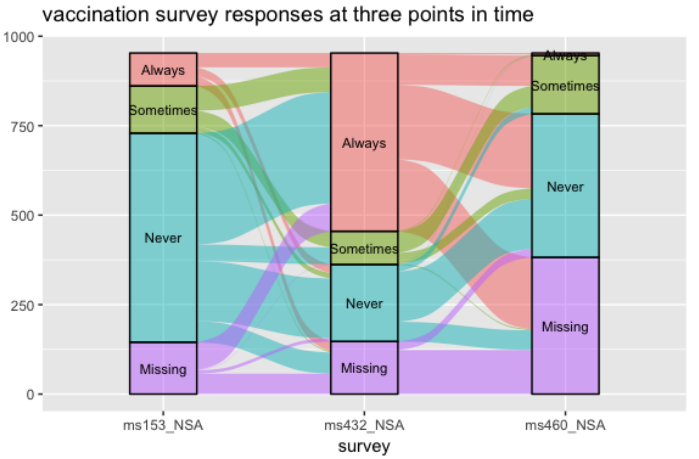


[1] URL: https://cran.r-project.org/web/packages/ggraph/index.html

`ggplot2` extensions 4/5

`ggalluvial`: multi-dimensional frequency data or categorical repeated measures data visualizations[1]

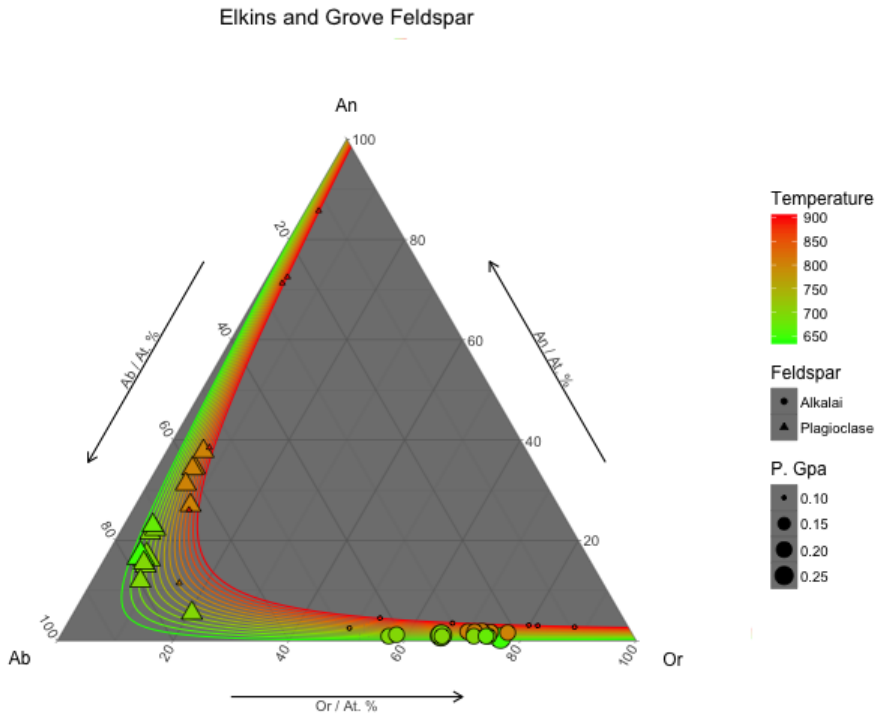




[1] URL: https://cran.r-project.org/web/packages/ggalluvial/index.html

`ggplot2` extensions 5/5

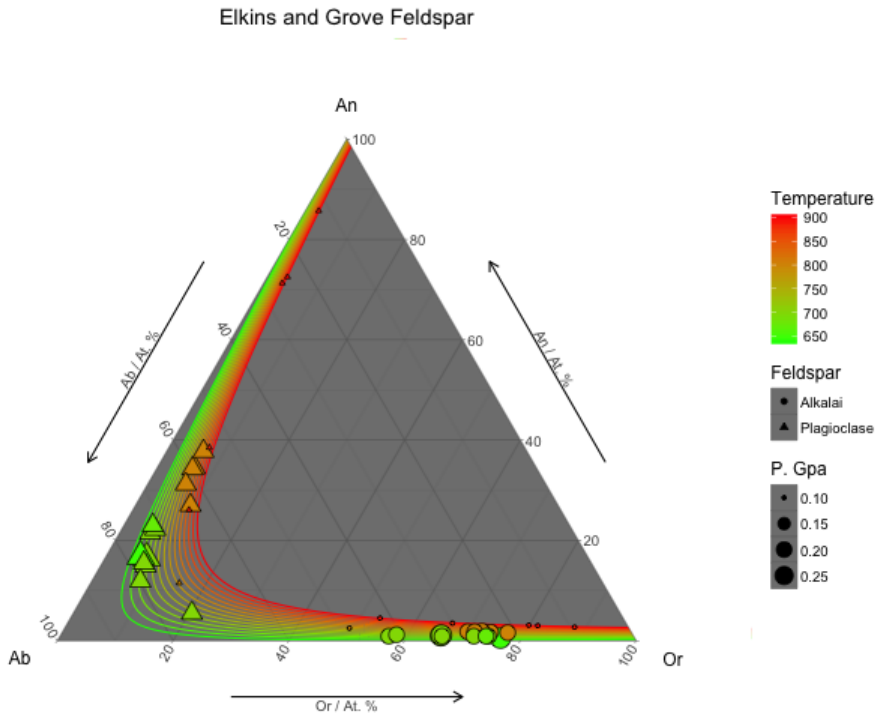
`ggtern`[1]: triangle-plot for trivariate data (single components sum up to 100%)



[1] URL: https://cran.r-project.org/web/packages/ggtern/index.html

ggplot2 extensions 5/5

ggtern[1]: triangle-plot for trivariate data (single components sum up to 100%)



Further ggplot2 extensions: http://www.ggplot2-exts.org/gallery/

[1] URL: https://cran.r-project.org/web/packages/ggtern/index.html

Session info 1/2

```
##  setting  value
##  version  R version 3.4.1 (2017-06-30)
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  German_Germany.1252
##  tz       Europe/Berlin
##  date     2018-05-02
```

	package	version	date	source
1	base	3.4.1	2017-06-30	local
2	bindrcpp	0.2	2017-06-17	CRAN (R 3.4.3)
3	datasets	3.4.1	2017-06-30	local
4	dplyr	0.7.4	2017-09-28	CRAN (R 3.4.2)
5	extrafont	0.17	2014-12-08	CRAN (R 3.4.1)
6	forcats	0.3.0	2018-02-19	CRAN (R 3.4.3)
7	Formula	1.2-2	2017-07-10	CRAN (R 3.4.1)
8	gapminder	0.3.0	2017-10-31	CRAN (R 3.4.3)
9	ggmap	2.6.1	2016-01-23	CRAN (R 3.4.3)
10	ggplot2	2.2.1.9000	2018-04-03	Github (thomasp85/ggplot2@f1ba983)
11	ggridges	0.5.0.9000	2018-04-24	Github (clauswilke/ggridges@9a29ec4)
12	graphics	3.4.1	2017-06-30	local
13	grDevices	3.4.1	2017-06-30	local
14	HistData	0.8-2	2017-08-16	CRAN (R 3.4.3)

	package	version	date	source
15	Hmisc	4.1-1	2018-01-03	CRAN (R 3.4.4)
16	lattice	0.20-35	2017-03-25	CRAN (R 3.4.3)
17	maps	3.2.0	2017-06-08	CRAN (R 3.4.3)
18	MASS	7.3-49	2018-02-23	CRAN (R 3.4.4)
19	methods	3.4.1	2017-06-30	local
20	readr	1.1.1	2017-05-16	CRAN (R 3.4.1)
21	rgdal	1.2-18	2018-03-17	CRAN (R 3.4.4)
22	sp	1.2-7	2018-01-19	CRAN (R 3.4.4)
23	stats	3.4.1	2017-06-30	local
24	stringr	1.3.0	2018-02-19	CRAN (R 3.4.3)
25	survival	2.41-3	2017-04-04	CRAN (R 3.4.1)
26	UsingR	2.0-5	2015-08-06	CRAN (R 3.4.3)
27	utils	3.4.1	2017-06-30	local

Session info 2/2

```
# global ggplot2 theme setting
theme_set(theme_minimal(base_size = 16, base_family = "Lato"))
```