Data Handling: Import, Cleaning and Visualization

Lecture 9:

Visualization

Aurélien Sallin, PhD

Updates

Recap

Cleaning and analytics

1. Stacking
2. Merging (joining) datasets
3. Data manipulation with `tidyverse()`
4. Aggregation of statistics

Merging (joining) datasets: R

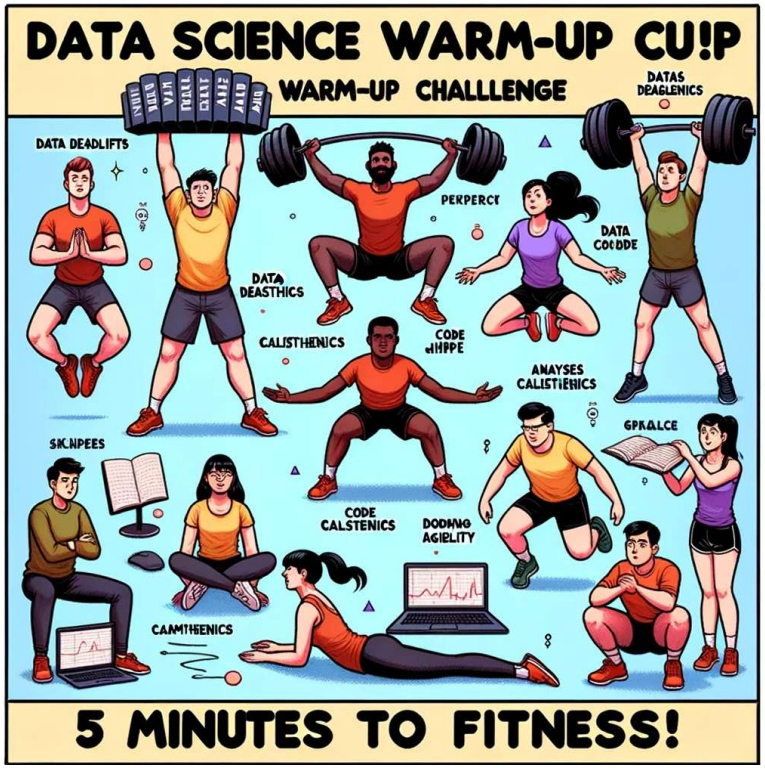
Overview by Wickham and Grolemund (2017):

dplyr (tidyverse)	base::merge
`inner_join(x, y)`	`merge(x, y)`
`left_join(x, y)`	`merge(x, y, all.x = TRUE)`
`right_join(x, y)`	`merge(x, y, all.y = TRUE),`
`full_join(x, y)`	`merge(x, y, all = TRUE)`

Data manipulation

- `arrange()`
- `select()`
- `filter()`
- `mutate()`
- `group_by()`
- `summarize()`

Warm up and test LockDown browser



Warm up and test LockDown browser

On Canvas, open the survey “LockDown browser test”. Answer the questions.

You must install the LockDown browser on your computer

Joining

Consider the following code:

```
temperature_conversions <- data.frame(conversion_factor = c(5/9, 1),
                                      scale = c("Fahrenheit", "Celsius"))

temperature <- data.frame(city = c("StGallen", "Zürich", "Detroit"),
                          temp = c(12, 14, 21),
                          scale = c("Cel", "Cel", "F"))
```

Select statements that are true:

- **`inner_join(temperature, temperature_conversions, by="scale")`** returns a data frame with 3 rows.
- **`left_join(temperature, temperature_conversions, by="scale")`** returns a data frame with 3 rows.
- **`full_join(temperature, temperature_conversions, by="scale")`** returns a data frame with 3 rows.

Data manipulation

Consider the following data frame:

```
> main_dataset
      city temp      scale conversion_factor
1 StGallen   12    Celsius         1.0000000
2   Zürich   14    Celsius         1.0000000
3  Detroit   40 Fahrenheit         0.5555556
```

Select statements that are true:

- `main_dataset |> mutate(temp_celsius = ifelse(scale == "Fahrenheit", (temp-32) * conversion_factor, temp))` replaces the variable `temp` with `temp_celsius`
- `main_dataset |> mutate(temp_celsius = ifelse(scale == "Fahrenheit", (temp-32) * conversion_factor, temp))` has 3 rows and 5 columns
- `main_dataset |> summarize(mean_temp = mean(temp), min_temp = min(temp))` returns a tibble containing 2 columns and 1 row
- `main_dataset |> summarize(mean_scale = mean(scale), sd_scale = sd(scale))` is a good way to get summary statistics about the variable `scale`

Stacking: True or False

Consider the following code:

```
Ostschwiz <- data.frame(
  Region = c("St. Gallen", "Appenzell", "Appenzell"),
  FavoriteBeverage = c("Schützengarten", "Quöllfrisch", "Appenzeller"),
  Year = c(2021, 2021, 2022)
)

Valais <- data.frame(
  Region = c("Valais"),
  FavoriteBeverage = c("Fendant"),
  Year = c(2021)
)

Vaud <- data.frame(
  Region = c("Vaud"),
  FavoriteBeverage = c("Chasselas"),
  Year = c(2021)
)
```

The statement `nrow(rbind(Ostschwiz, Valais, Vaud)) == 5` returns `FALSE`

Stacking and reshaping: True or False

Consider the following data frame

```
Ostschwiz <- data.frame(
  Region = c("St. Gallen", "Appenzell", "Appenzell"),
  FavoriteBeverage = c("Schützengarten", "Quöllfrisch", "Appenzeller"),
  Year = c(2021, 2021, 2022)
)

Valais <- data.frame(
  Region = c("Valais"),
  FavoriteBeverage = c("Fendant"),
  Year = c(2021)
)

Vaud <- data.frame(
  Region = c("Vaud"),
  FavoriteBeverage = c("Chasselas"),
  Year = c(2021)
)

Ostschwiz <- pivot_wider(Ostschwiz,
                         names_from = c(Year),
                         values_from = FavoriteBeverage)
```

The statement **`colnames(Ostschwiz)[2]`** returns **`"FavoriteBeverage`**

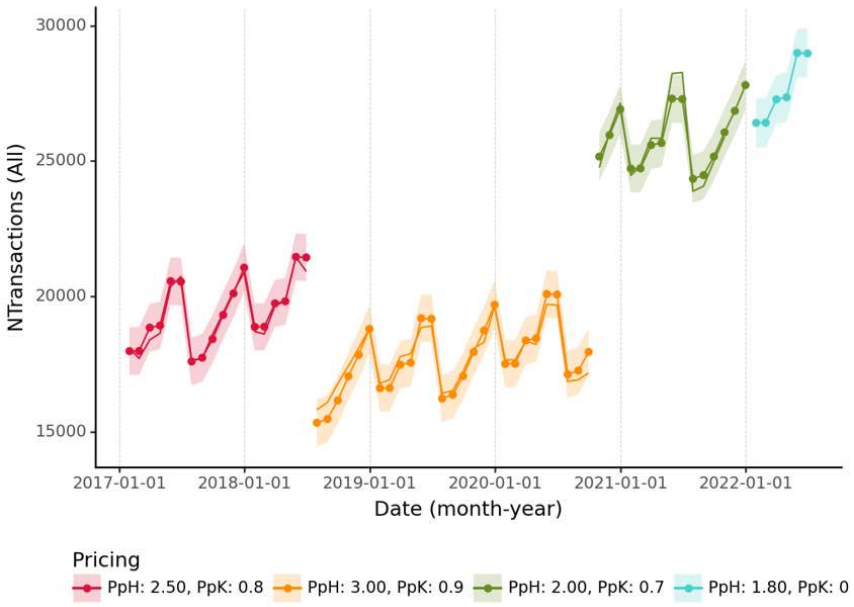
Data visualization

Data visualization

- Final step of data pipeline/data science procedure!
  - Convincingly communicating insights from data.
  - Data visualization is all about **telling a story**.
- **R** is a very powerful tool to do this!

Data visualization

- Data visualization is **THE** most important part of delivering results!
- Best way to convince people about your message (and about your competence 🎓)



Data visualization

Two ways: display data through **tables** or **graphs**.

Depends on the purpose.

Data visualization

- A **chart** typically contains at least one axis, the values are represented in terms of visual objects (dots, lines, bars) and axes typically have scales or labels.
    - If we are interested in exploring, analyzing or communicating **patterns** in the data, **charts are more useful than tables**.

- A **table** typically contains rows and columns, and the values are represented by text.
    - If we are interested in exploring, analyzing or communicating **specific numbers** in the data, **tables are more useful than graphs**.

Tables

Tables

- Formatting data values for publication.
- Typical: String operations to make numbers and text look nicer.
  - Before creating a table or figure…

Tables

```r
# load packages and data
library(tidyverse)
data("swiss")

# compute summary statistics
swiss_summary <- swiss |>
  summarise(avg_education = mean(Education),
            avg_fertility = mean(Fertility),
            N = n()
  )

swiss_summary
```

```
##   avg_education avg_fertility  N
## 1      10.97872      70.14255 47
```

Problems?

Tables: round numeric values

```r
swiss_summary_rounded <- round(swiss_summary, 2)
swiss_summary_rounded
```

```
##   avg_education avg_fertility  N
## 1         10.98         70.14 47
```

Tables: detailed formatting of numbers

- Coerce to text.
- String operations.
- Decimal marks, units (e.g., currencies), other special characters for special formats (e.g. coordinates).
- `format()`-function

Tables: `format()` example

```r
swiss_form <- format(swiss_summary_rounded,
                     decimal.mark = ",")
swiss_form
```

```
##   avg_education avg_fertility  N
## 1         10,98         70,14 47
```

See also the helpful functions for formatting text-strings

- Uppercase/lowercase: **`toupper()`**/**`tolower()`**.
- Remove white spaces: **`trimws()`**,

```
string <- "AbCD "
toupper(string)
```

```
## [1] "ABCD "
```

```
tolower(string)
```

```
## [1] "abcd "
```

```
trimws(tolower(string))
```

```
## [1] "abcd"
```

Get creative with tables: `gtExtras` and sparklines

```r
head(USArrests, 10)
```

```
##             Murder Assault UrbanPop Rape
## Alabama       13.2     236       58 21.2
## Alaska        10.0     263       48 44.5
## Arizona        8.1     294       80 31.0
## Arkansas       8.8     190       50 19.5
## California     9.0     276       91 40.6
## Colorado       7.9     204       78 38.7
## Connecticut    3.3     110       77 11.1
## Delaware       5.9     238       72 15.8
## Florida       15.4     335       80 31.9
## Georgia       17.4     211       60 25.8
```

Problems?

Get creative with tables: gtExtras and sparklines

```r
library(gtExtras)

USArrests_summary <- USArrests |>
  mutate(UrbanPop = case_when(UrbanPop > quantile(UrbanPop, .66) ~ "High",
                              UrbanPop > quantile(UrbanPop, .33) ~ "Middle",
                              UrbanPop > 0 ~ "Low")) |>
  group_by(UrbanPop) |>
  summarize(
    "Mean murder" = mean(Murder),
    "SD murder" = sd(Murder),
    Density = list(Murder),
    .groups = "drop"
  )
```

Get creative with tables: gtExtras and sparklines

```
USArrests_summary |>
  gt() |>
  tab_header(
    title = md("Murder rates"),
    subtitle = md("Per high, middle, and low urban population ")
  ) |>
  gtExtras::gt_plt_dist(Density, type = "density", line_color = "black",
                        fill_color = "red") %>%
  fmt_number(columns = `Mean murder`:`SD murder`, decimals = 2)
```

Murder rates

Per high, middle, and low urban population

UrbanPop	Mean murder	SD murder	Density
High	8.07	3.80	
Low	7.41	5.20	
Middle	7.89	4.18	

Get creative with tables: other sources

- `kable()` for `html` / Markdown reports
- `stargazer` for your LaTeX reports or for your Office Word reports

Get creative with tables: `kable()`

```
knitr::kable(head(USArrests, 5), format = "markdown")
```

	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6

Get creative with tables: `kable()`

```r
knitr::kable(head(USArrests, 5), format = "html")
```

	Murder	Assault	UrbanPop	Rape
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6

Graphs with R (ggplot2)

Graphs with R

Three main approaches:

1. The original `graphics` package ((R Core Team 2018); shipped with the base R installation).

Graphs with R

Three main approaches:

1. The original `graphics` package ((R Core Team 2018); shipped with the base R installation).
2. The `lattice` package (Sarkar 2008), an implementation of the original Bell Labs 'Trellis' system.

Graphs with R

Three main approaches:

1. The original `graphics` package ((R Core Team 2018); shipped with the base R installation).
2. The `lattice` package (Sarkar 2008), an implementation of the original Bell Labs 'Trellis' system.
3. The `ggplot2` package (Wickham 2016), an implementation of Leland Wilkinson's 'Grammar of Graphics'.

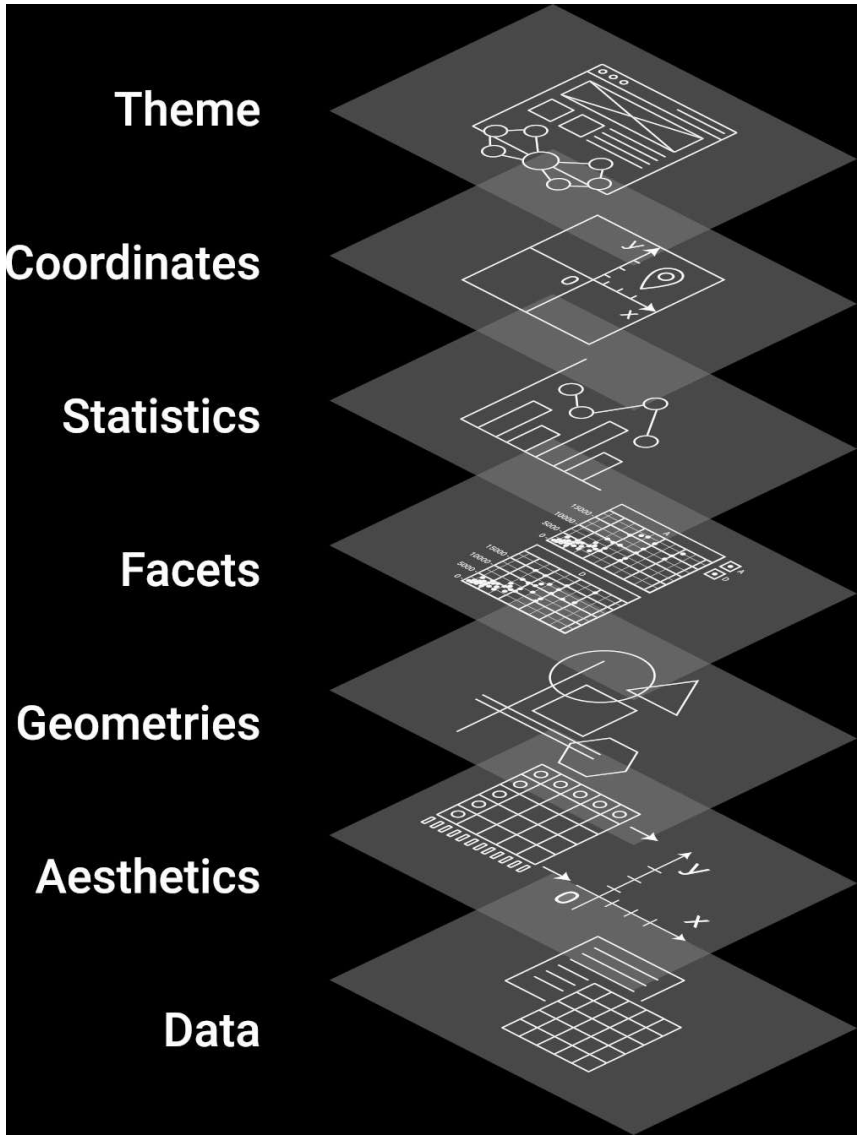
Graphs with R

Three main approaches:

1. The original **`graphics`** package ((R Core Team 2018); shipped with the base R installation).
2. The **`lattice`** package (Sarkar 2008), an implementation of the original Bell Labs 'Trellis' system.
3. The **`ggplot2`** package (Wickham 2016), an implementation of Leland Wilkinson's 'Grammar of Graphics'.

`ggplot2` is so good that it has become **THE** reference [In python, use **`plotnine`** to apply the grammar of graphics.]

Grammar of graphics



ggplot2

`ggplot2` basics

Using `ggplot2` to generate a basic plot in R is quite simple. Basically, it involves three key points:

1. The data must be stored in a `data.frame`/`tibble` (in tidy format!).

`ggplot2` basics

Using `ggplot2` to generate a basic plot in R is quite simple. Basically, it involves three key points:

1. The data must be stored in a `data.frame`/`tibble` (in tidy format!).
2. The starting point of a plot is always the function `ggplot()`.

`ggplot2` basics

Using `ggplot2` to generate a basic plot in R is quite simple. Basically, it involves three key points:

1. The data must be stored in a `data.frame`/`tibble` (in tidy format!).
2. The starting point of a plot is always the function `ggplot()`.
3. The first line of plot code declares the data and the 'aesthetics' (e.g., which variables are mapped to the x-/y-axes):

`ggplot2` basics

Using **`ggplot2`** to generate a basic plot in R is quite simple. Basically, it involves three key points:

1. The data must be stored in a **`data.frame`**/**`tibble`** (in tidy format!).
2. The starting point of a plot is always the function **`ggplot()`**.
3. The first line of plot code declares the data and the 'aesthetics' (e.g., which variables are mapped to the x-/y-axes):

```
ggplot(data = my_dataframe, aes(x= xvar, y= yvar))
```

Example data set: `swiss`

```r
library(tidyverse) # automatically loads ggplot2

# load the data
data(swiss)
head(swiss)
```

```
##               Fertility Agriculture Examination Education Catholic Infant.Mortality
## Courtelary         80.2        17.0          15        12     9.96             22.2
## Delemont           83.1        45.1           6         9    84.84             22.2
## Franches-Mnt       92.5        39.7           5         5    93.40             20.2
## Moutier            85.8        36.5          12         7    33.77             20.3
## Neuveville         76.9        43.5          17        15     5.16             20.6
## Porrentruy         76.1        35.3           9         7    90.57             26.6
```

Add indicator variable

Code a province as ‘Catholic’ if more than 50% of the inhabitants are catholic:

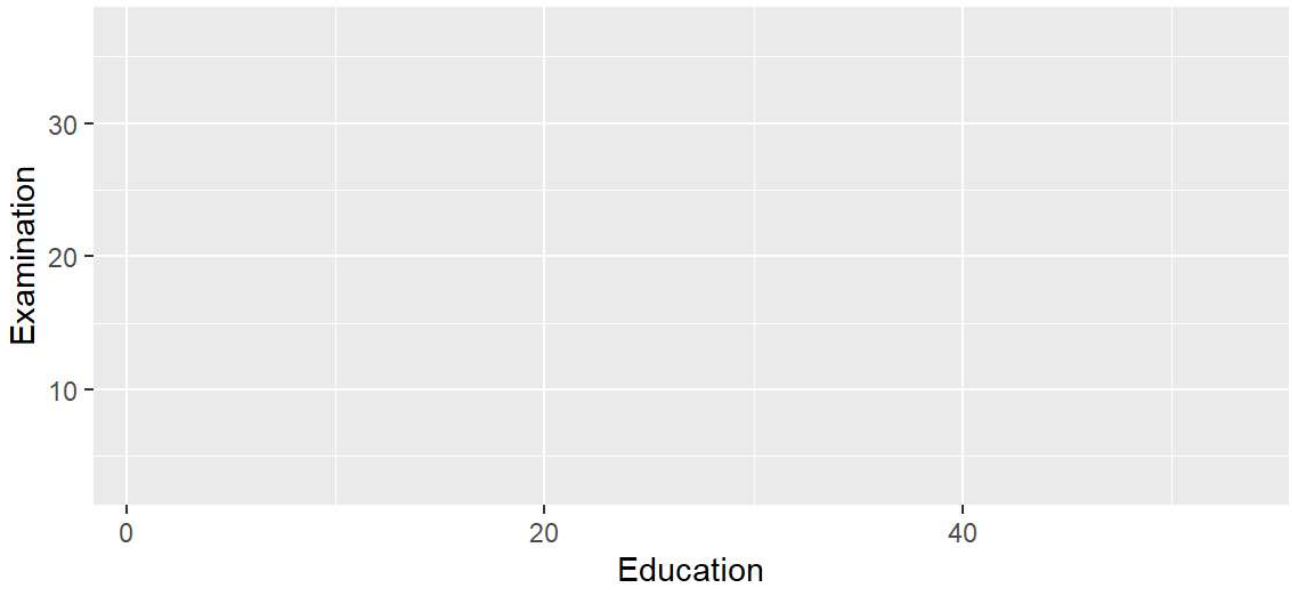
```
# via tidyverse/mutate
swiss <- mutate(swiss,
                Religion =
                  ifelse(50 < Catholic, 'Catholic', 'Protestant'))

# 'old school' alternative
swiss$Religion <- 'Protestant'
swiss$Religion[50 < swiss$Catholic] <- 'Catholic'

# set to factor
swiss$Religion <- as.factor(swiss$Religion)
```

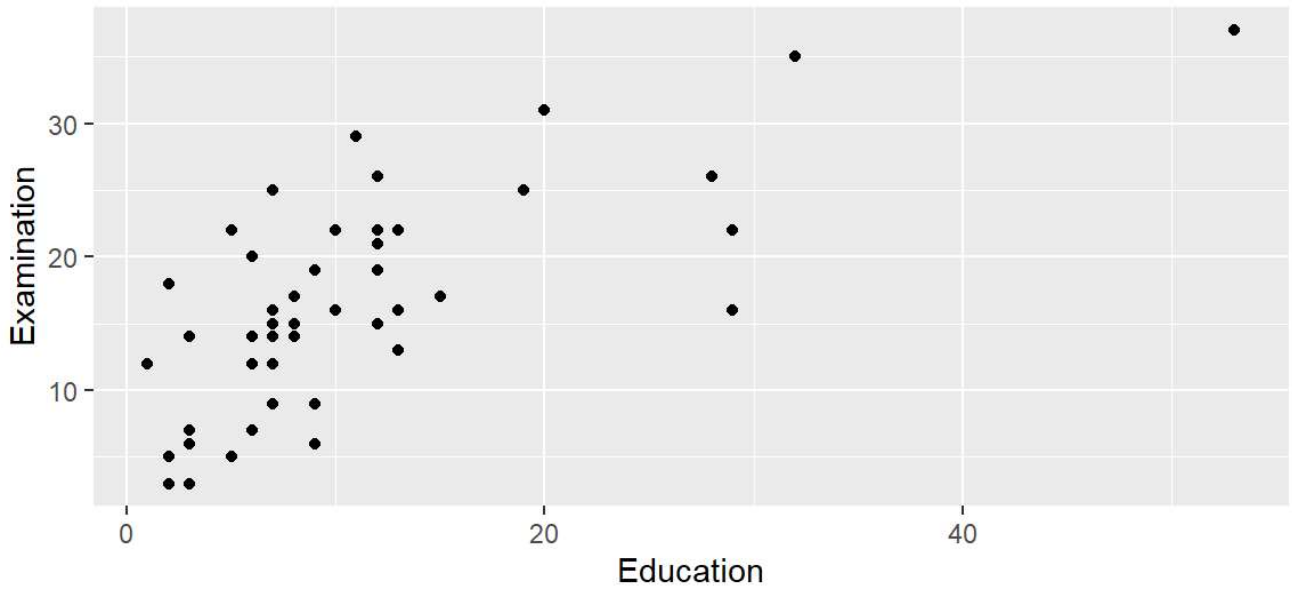
Data and aesthetics

```
ggplot(data = swiss, aes(x = Education, y = Examination))
```



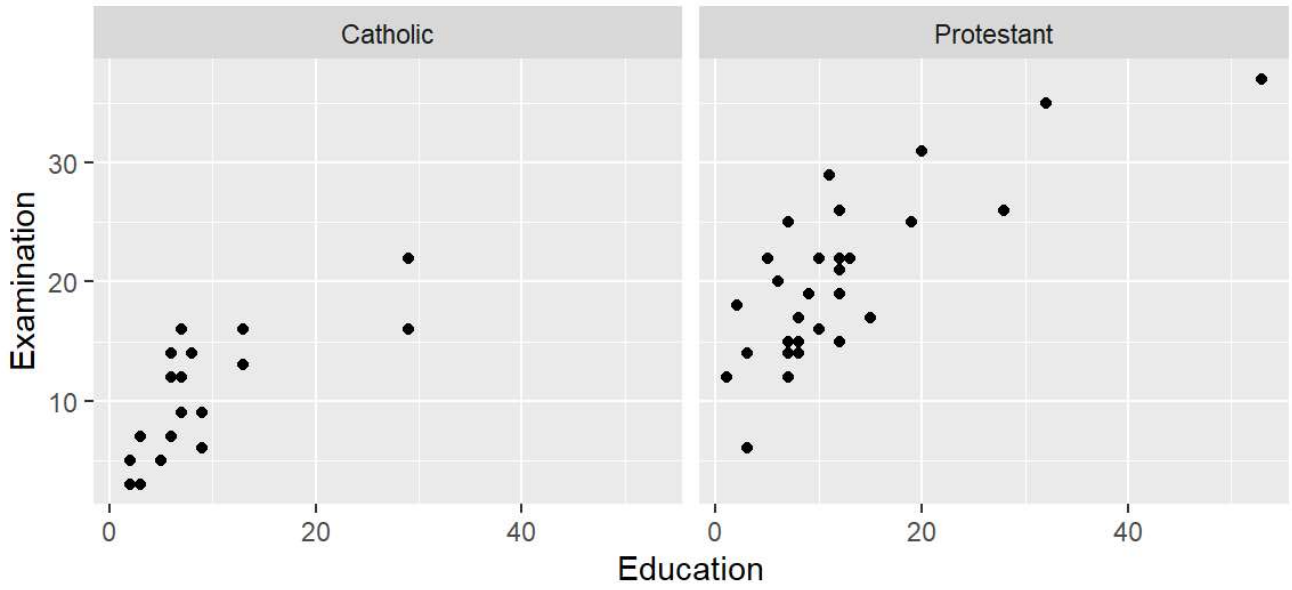
Geometries (~the type of plot)

```
ggplot(data = swiss, aes(x = Education, y = Examination)) +
     geom_point()
```



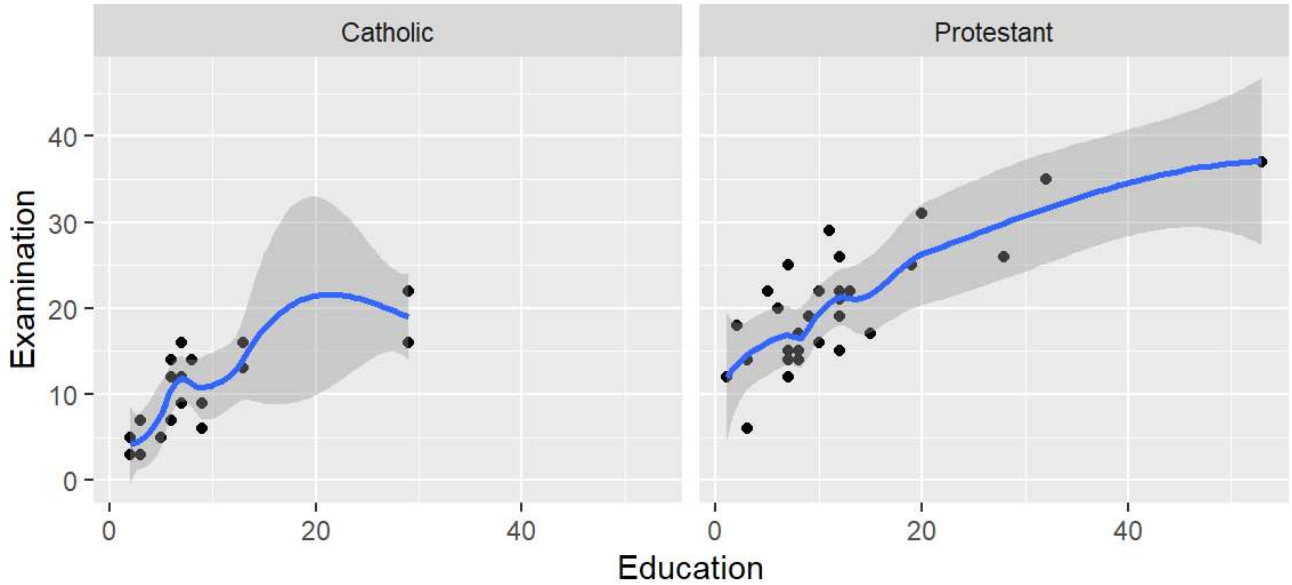
Facets

```
ggplot(data = swiss, aes(x = Education, y = Examination)) +
     geom_point() +
     facet_wrap(~Religion)
```



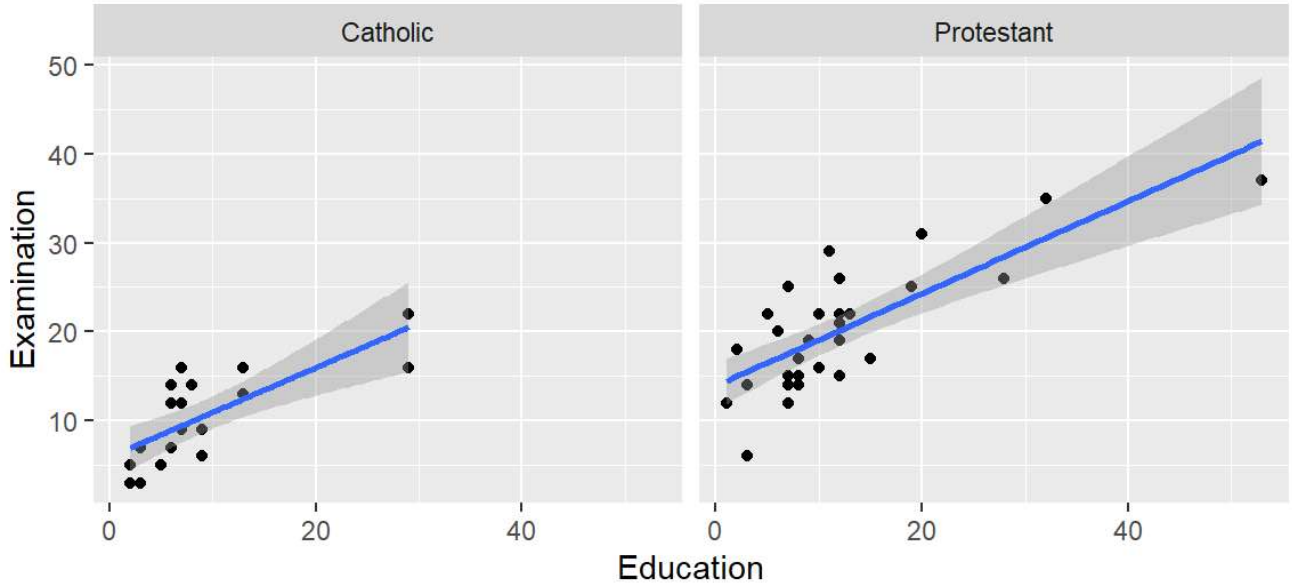
Additional layers and statistics

```
ggplot(data = swiss, aes(x = Education, y = Examination)) +
     geom_point() +
     geom_smooth(method = 'loess') +
     facet_wrap(~Religion)
```



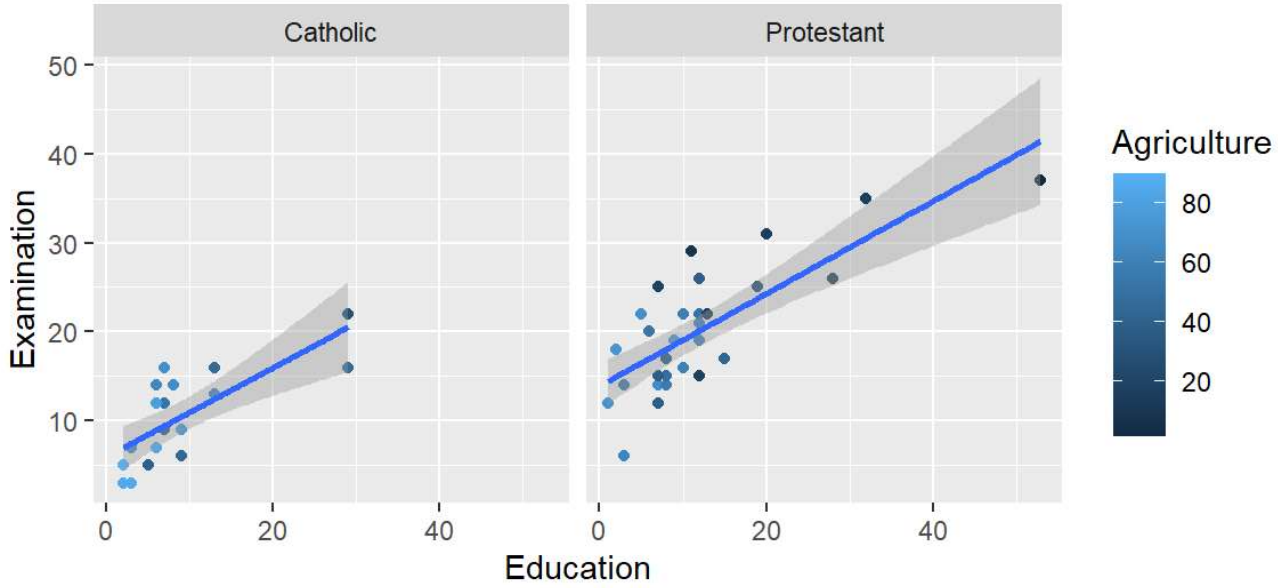
Additional layers and statistics

```
ggplot(data = swiss, aes(x = Education, y = Examination)) +
     geom_point() +
     geom_smooth(method = 'lm') +
     facet_wrap(~Religion)
```



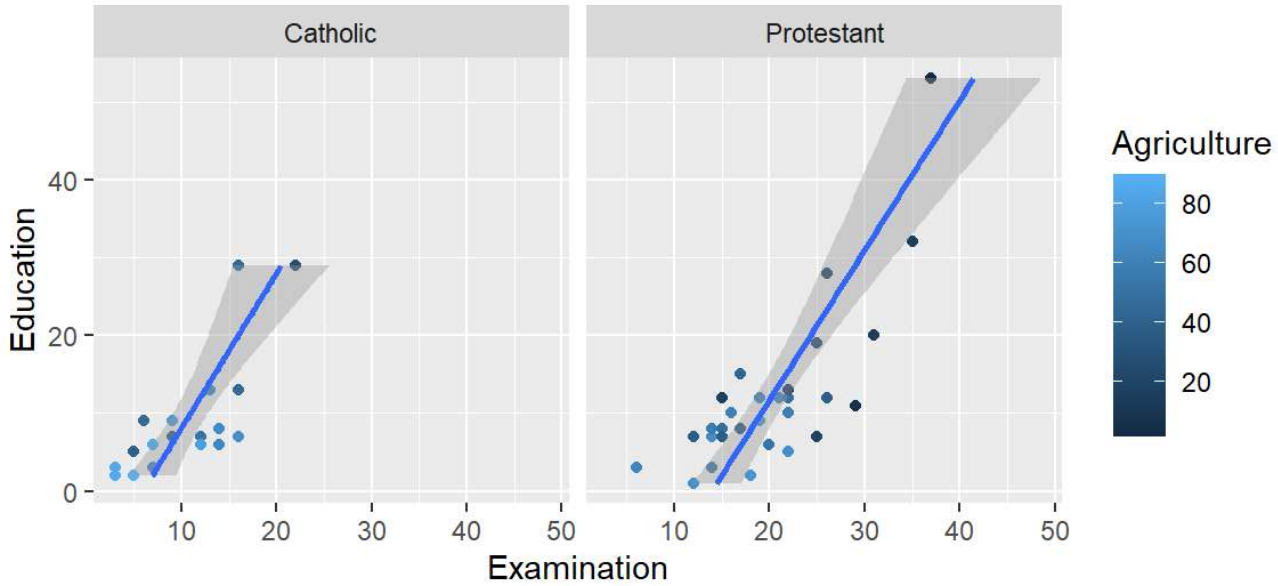
Additional aesthetics

```
ggplot(data = swiss, aes(x = Education, y = Examination)) +
     geom_point(aes(color = Agriculture)) +
     geom_smooth(method = 'lm') +
     facet_wrap(~Religion)
```



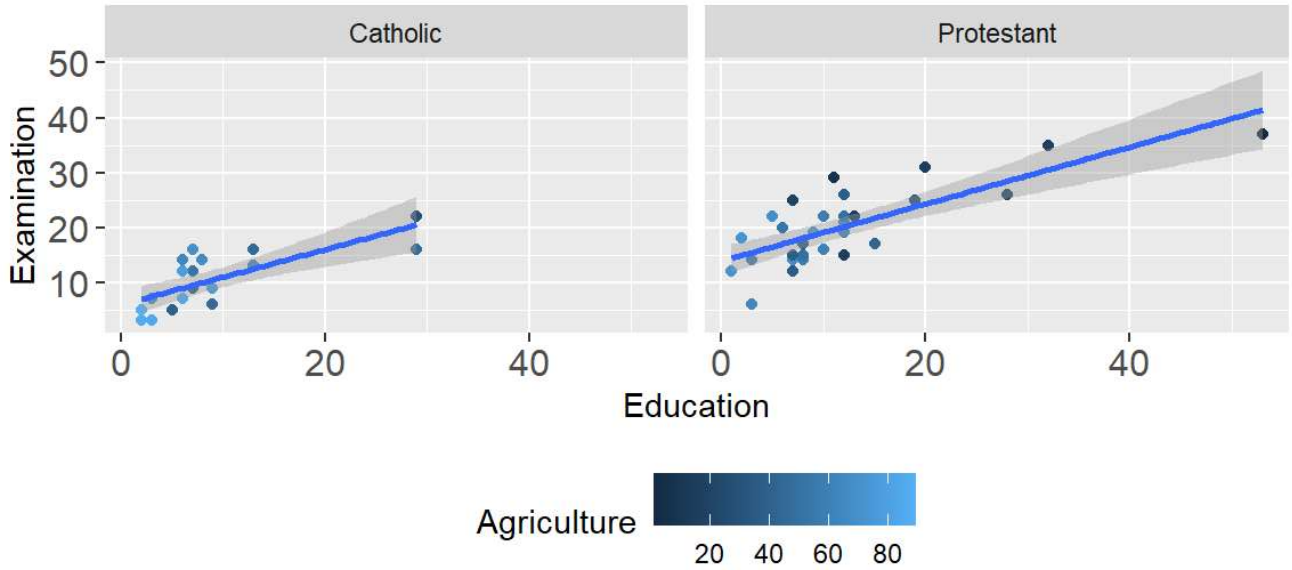
Change coordinates

```
ggplot(data = swiss, aes(x = Education, y = Examination)) +
     geom_point(aes(color = Agriculture)) +
     geom_smooth(method = 'lm') +
     facet_wrap(~Religion) +
     coord_flip()
```



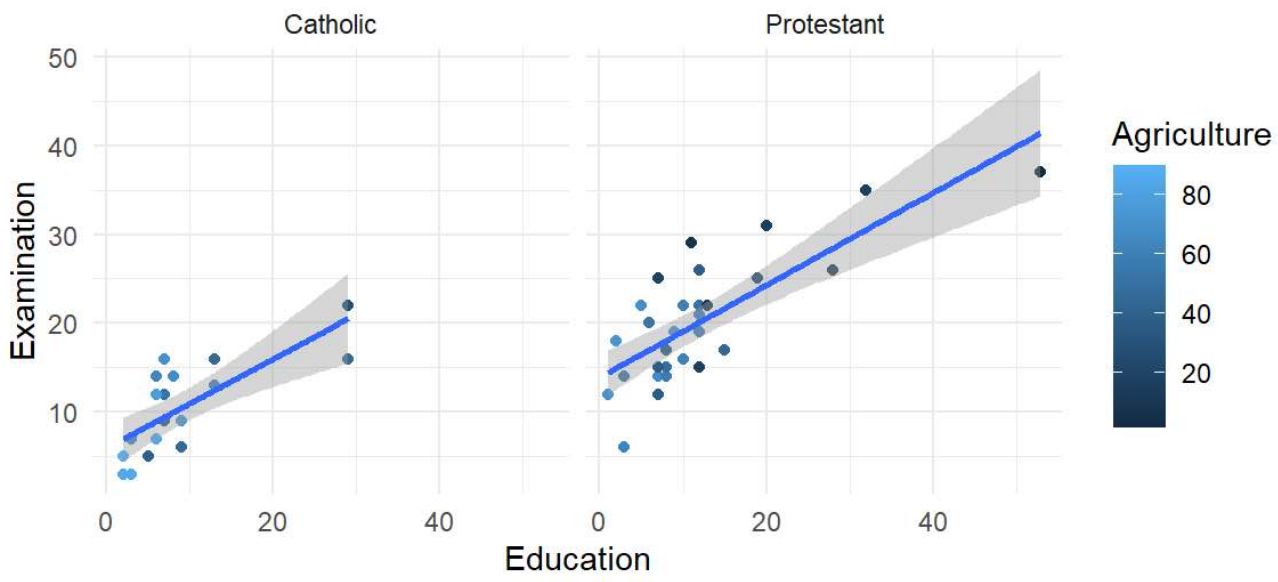
Themes

```
ggplot(data = swiss, aes(x = Education, y = Examination)) +
     geom_point(aes(color = Agriculture)) +
     geom_smooth(method = 'lm') +
     facet_wrap(~Religion) +
     theme(legend.position = "bottom", axis.text=element_text(size=12) )
```



Themes

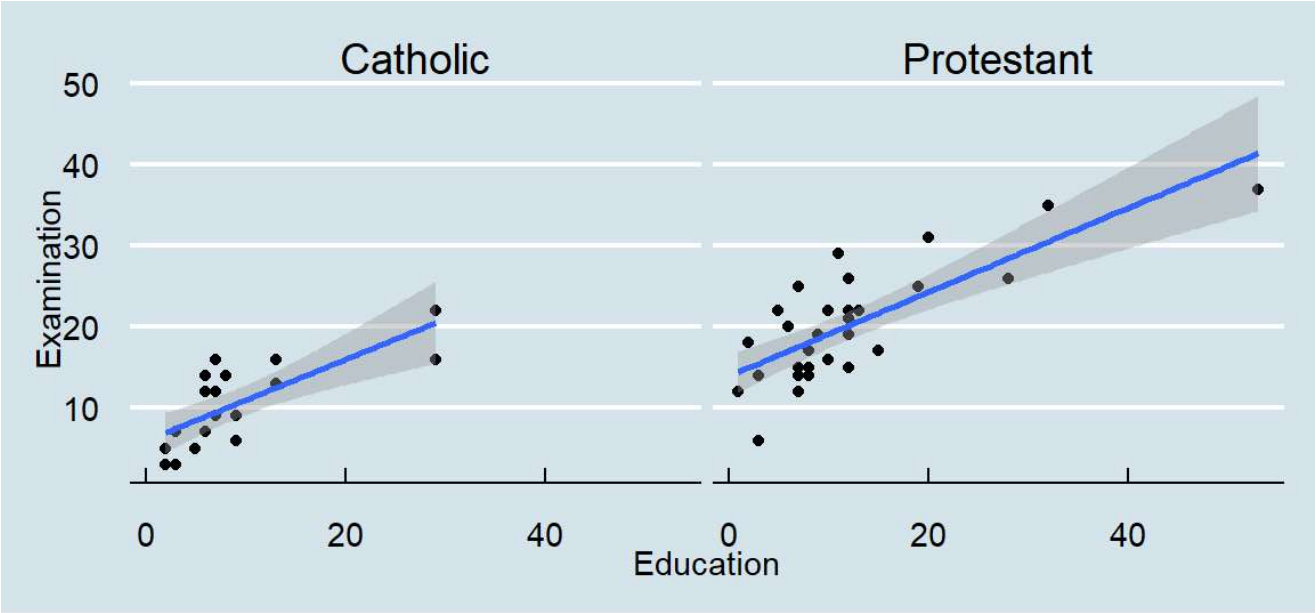
```
ggplot(data = swiss, aes(x = Education, y = Examination)) +
     geom_point(aes(color = Agriculture)) +
     geom_smooth(method = 'lm') +
     facet_wrap(~Religion) +
     theme_minimal()
```



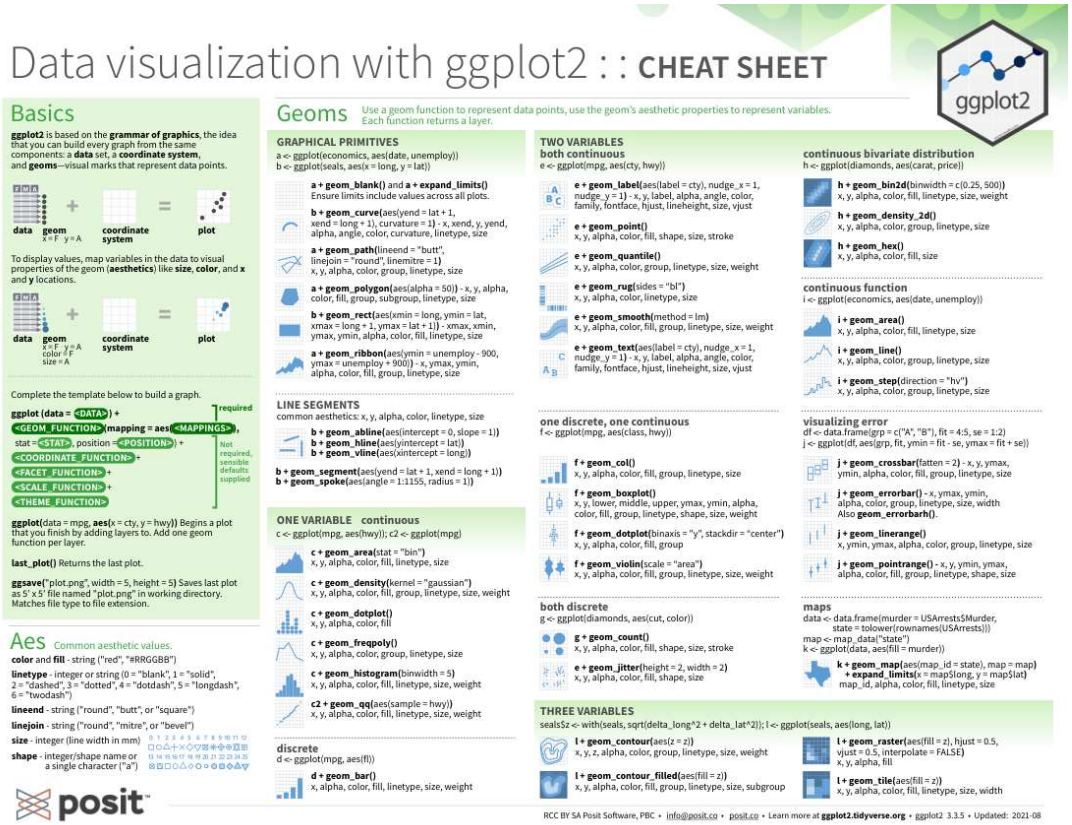
Themes

```
library(ggthemes)

ggplot(data = swiss, aes(x = Education, y = Examination)) +
     geom_point() +
     geom_smooth(method = 'lm') +
     facet_wrap(~Religion) +
     theme_economist()
```



Cheat sheet



Link: https://rstudio.github.io/cheatsheets/html/data-visualization.html

Data visualization

- Values are represented by their **position relative to the axes**: line charts and scatterplots.
- Values are represented by the **size of an area**: bar charts and area charts.
- Values are **continuous**: use chart type that visually connects elements (line chart).
- Values are **categorical**: use chart type that visually separates elements (bar chart).

(Source: https://hhsievertsen.github.io/EconDataBook/data-visualization-basics.html)

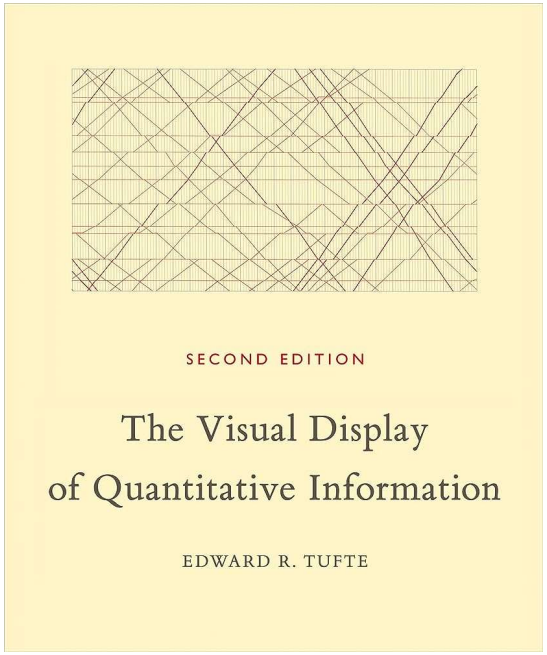
Data visualization: some principles

- Two pieces of advice I personally received:
  - **fit your whole story in one graph**.
  - your audience should understand your graph **without the need of listening to you** or **reading your text.**

- Be simple and avoid fanciness.
- Avoid pie charts and 3D charts.

Data visualization: some principles

Recommendations
from Tufte

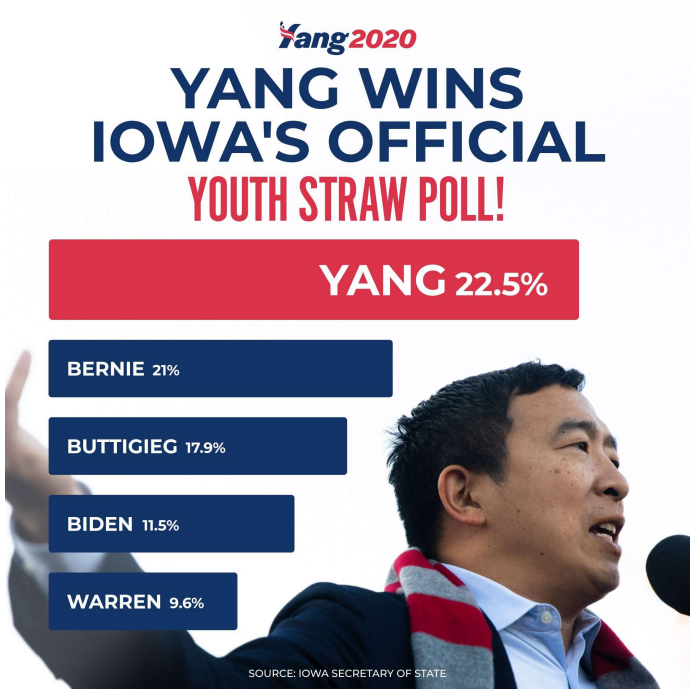


Data visualization: some principles

We can quantify the Lie Factor of a graph

Lie Factor =

$$\frac{\text{size of effect shown in graphic}}{\text{size of effect in data}}$$



Thou shalt not truncate the Y axis.

Tucker Carlson is guilty of committing chart sins

Thou shalt not truncate the Y axis.

JEMIMA KELLY + Add to myFT

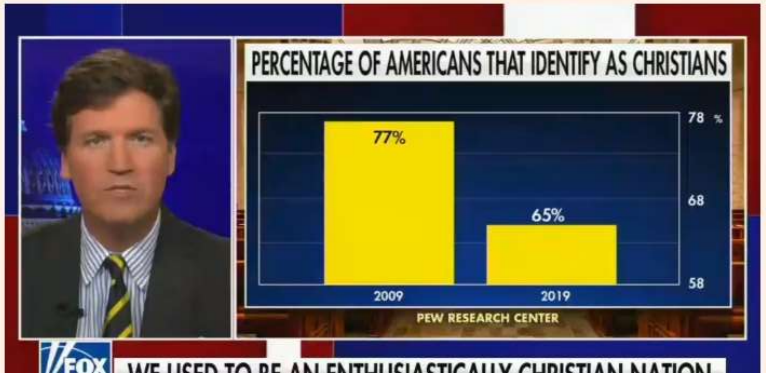
Jemima Kelly SEPTEMBER 28 2021 48

Unlock the Editor's Digest for free

Roula Khalaf, Editor of the FT, selects her favourite stories in this weekly newsletter.

Enter your email address Sign up

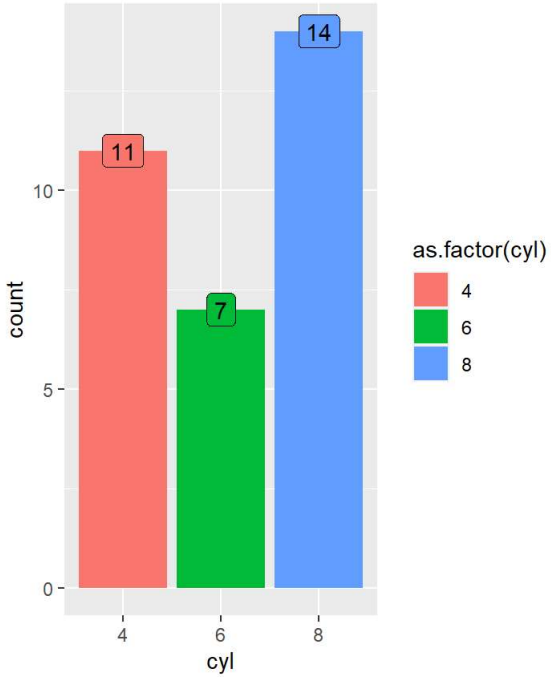
True chart crime fans, rejoice! For unto you a new atrocity is born:

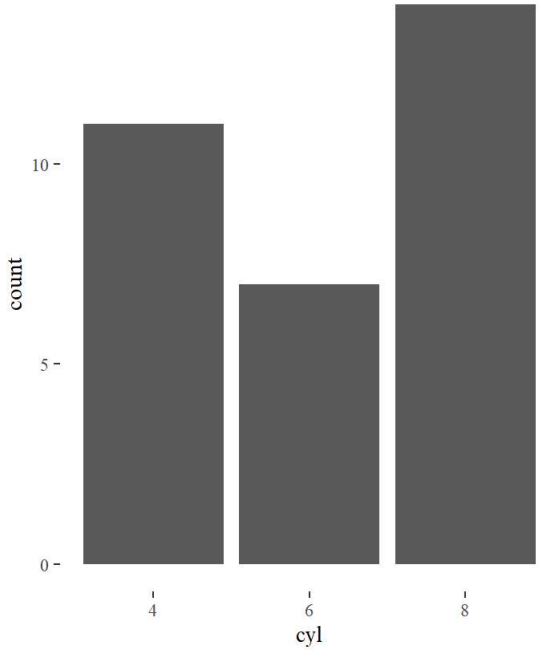


Data visualization: some principles

Only what matters should be reported (**Data-ink Ratio**):

$$\text{Data-ink Ratio} = \frac{\text{ink used for data}}{\text{total ink used to print the graphic}}$$





Conclusion

Data visualization is an art of story-telling, deception, and scientific exactitude 🧐.

Q&A

References

R Core Team. 2018. **R: A Language and Environment for Statistical Computing**. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Sarkar, Deepayan. 2008. **Lattice: Multivariate Data Visualization with r**. New York: Springer. http://lmdvr.r-forge.r-project.org.

Wickham, Hadley. 2016. **Ggplot2: Elegant Graphics for Data Analysis**. Springer-Verlag New York. http://ggplot2.org.

Wickham, Hadley, and Garrett Grolemund. 2017. Sebastopol, CA: O'Reilly. http://r4ds.had.co.nz/.