# Data Handling: Import, Cleaning and Visualisation

Lecture 11: Visualisation and Dynamic Documents

*Prof. Dr. Ulrich Matter*
*(University of St.Gallen)*

*12/12/2019*

---



---

## 1 Data display

In the last part of a data pipeline we are typically dealing with the visualisation of data and statistical results for presentation/communication. Typical output formats are reports, a thesis (BA, MA, Dissertation chapter), interactive dashboards, and websites. R (and particularly RStudio) provides a very flexible framework to manage the steps involved in visualisation/presentation for all of these output formats. A first (low-level) step in preparing data/results for publication is the formatting data values for publication. Typically, this involves some string operations to make numbers and text look nicer before we show them in a table or graph.

Consider, for example, the following summary statistics.

```
# load packages and data
library(tidyverse)
data("swiss")
# compute summary statistics
swiss_summary <-
  summarise(swiss,
          avg_education = mean(Education, na.rm = TRUE),
          avg_fertility = mean(Fertility, na.rm = TRUE),
          N = n()
          )
swiss_summary
```

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

Quite likely, we do not want to present these numbers with that many decimal places. The function `round()` can take care of this.

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

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

More specific formatting of numeric values is easier when coercing the numbers to character strings (text).[1]

[1] Note that this step only makes sense if we are sure that the numeric values won't be further analyzed or used in a plot (except for labels).

For example, depending on the audience (country/region) to which we want to communicate our results, different standards of how to format numbers are expected. In the English-speaking world it is quite common to use `.` as decimal mark, in the German-speaking world it is rather common to use `,`. The `format()`-function provides an easy way to format numbers in this way (once they are coerced to character).

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

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

# 2 Data visualisation with ggplot2

A key technique to convincingly communicate statistical results and insights from data is visualisation. How can we visualise raw data and insights gained from statistical models with R? It turns out that R is a really useful tool for data visualisation, thanks to its very powerful graphics engine (i.e., the underlying low-level R functions that handle things like colors, shapes, etc.). Building on this graphics engine, there are particularly three R packages with a variety of high-level functions to plot data in R:

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

While all of these packages provide well-documented high-level R functions to plot data, their syntax differs in some important ways. For R beginners it thus makes sense to first learn how to generate plots in R with *one* of these packages. Here, we focus on `ggplot2` because it is part of the `tidyverse`.

## 2.1 'Grammar of Graphics'

A few years back, Leland Wilkinson (statistician and computer scientist) wrote an influential book called 'The Grammar of Graphics'. In this book, Wilkinson develops a formal description ('grammar') of graphics used in statistics, illustrating how different types of plots (bar plot, histogram, etc.) are special cases of an underlying framework. In short, his idea was that we can think of graphics as consisting of different design-layers and thus can build and describe graphics/plots layer by layer (see here for an illustration of this idea).

This framework got implemented in R with the prominent `ggplot2`-package, building on the already powerful R graphics engine. The result is a user-friendly environment to visualise data with enormous potential to plot almost any graphic illustrating data.

## 2.2 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))
```

## 2.3 Tutorial

In the following, we learn the basic functionality of `ggplot` by applying it to the `swiss` dataset.

### 2.3.1 Loading/preparing the data

First, we load and inspect the data. Among other variables it contains information about the share of inhabitants of a given Swiss province who indicate to be of Catholic faith (and not Protestant).

```
# load the R package
library(ggplot2)
# load the data
data(swiss)
# get details about the data set
# ?swiss
# inspect the data
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
```

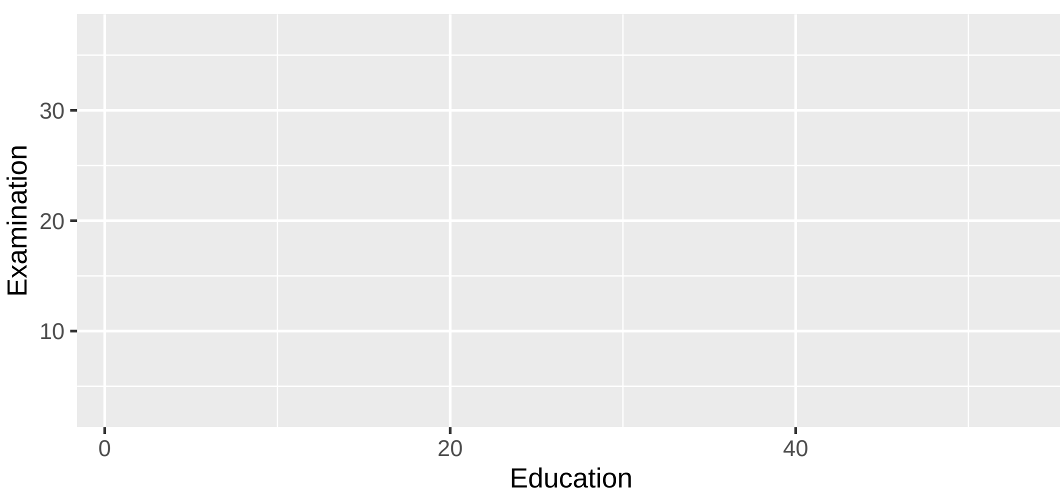
As we do not only want to use this continuous measure in the data visualisation, we generate an additional factor variable called `Religion` which has either the value `'Protestant'` or `'Catholic'` depending on whether more then 50 percent of the inhabitants of the province are Catholics.

```
# code province as 'Catholic' if more than 50% are catholic
swiss$Religion <- 'Protestant'
swiss$Religion[50 < swiss$Catholic] <- 'Catholic'
swiss$Religion <- as.factor(swiss$Religion)
```

### 2.3.2 Data and aesthetics

We initiate the most basic plot with `ggplot()` by defining which data to use and in the plot aesthetics which variable to use on the x and y axes. Here, we are interested in whether the level of education beyond primary school in a given district is related with how well draftees from the same district do in a standardized army examination (% of draftees that get the highest mark in the examination).

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

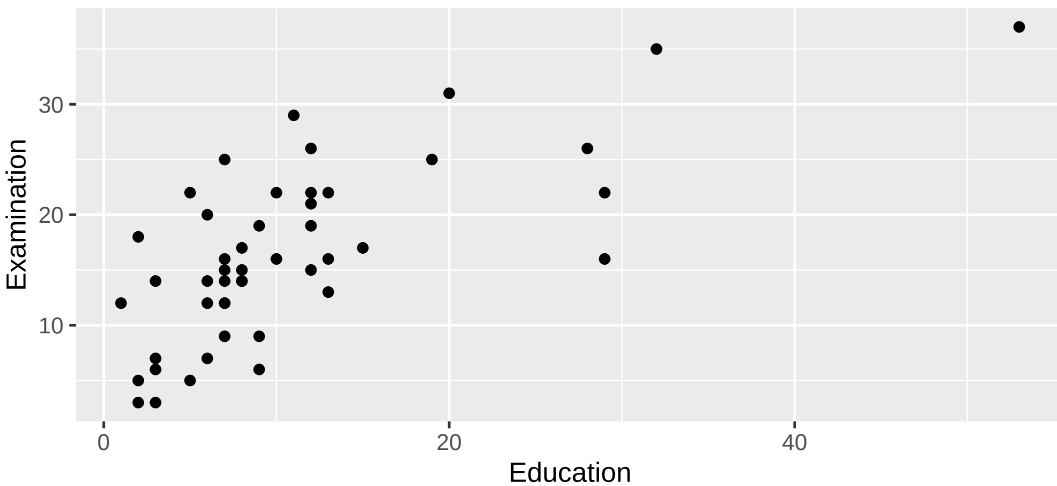


As we have not yet defined according to what rules the data shall be visualised, all we get is an empty 'canvas' and the axes (with the respective label and ticks indicating the range of the values).

## 2.4 Geometries (~ type of plot)

In order to actually plot the data we have to define the 'geometries', defining according to which function the data should be mapped/visualised. In other words, geometries define which 'type of plot' we use to visualise the data (histogram, lines, points, etc.). In the example code below, we use `geom_point()` to get a simple point plot.

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

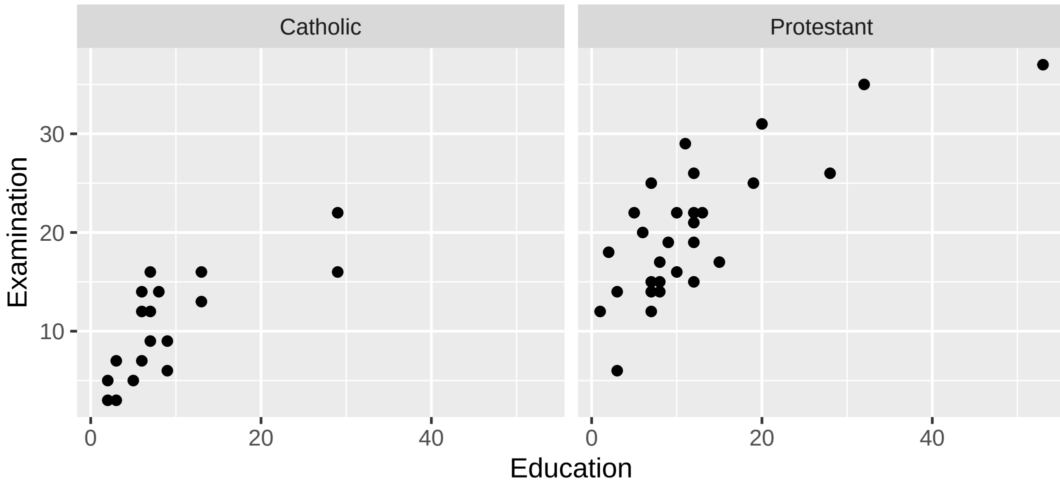



The result indicates that there is a positive correlation between the level of education and how well draftees do in the examination. We want to better understand this correlation. Particularly, what other factors could drive this picture.

### 2.4.1 Facets

According to a popular thesis, the protestant reformation and the spread of the protestant movement in Europe was driving the development of compulsory schooling. It would thus be reasonable to hypothesize that the picture we see is partly driven by differences in schooling between Catholic and Protestant districts. In order to make such differences visible in the data, we use 'facets' to show the same plot again, but this time separating observations from Catholic and Protestant districts:

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

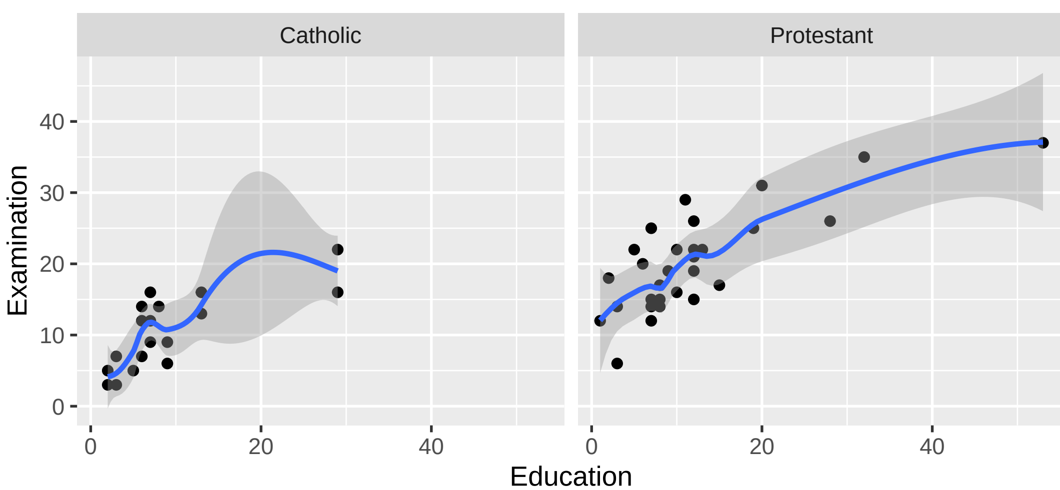



Draftees from protestant districts tend to do generally better (which might be an indication of better primary schools, or a generally stronger focus on scholastic achievements of Protestant children). However, the relationship between education (beyond primary schools) and examination success seems to hold for either type of districts.

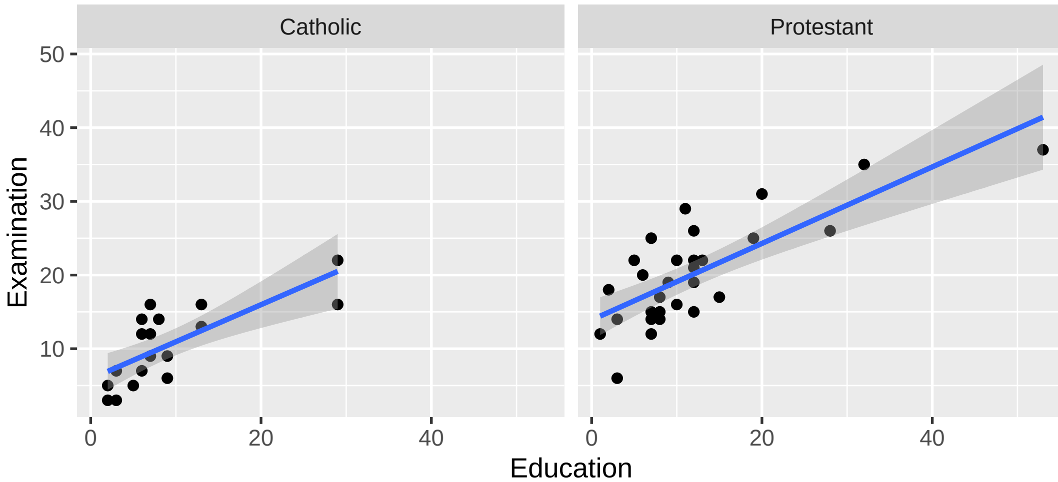
### 2.4.2 Additional layers and statistics

Let's visualise this relationship more clearly by drawing trend-lines through the scatter diagrams. Once with the non-parametric 'loess'-approach and once forcing a linear model on the relationship between the two variables.

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



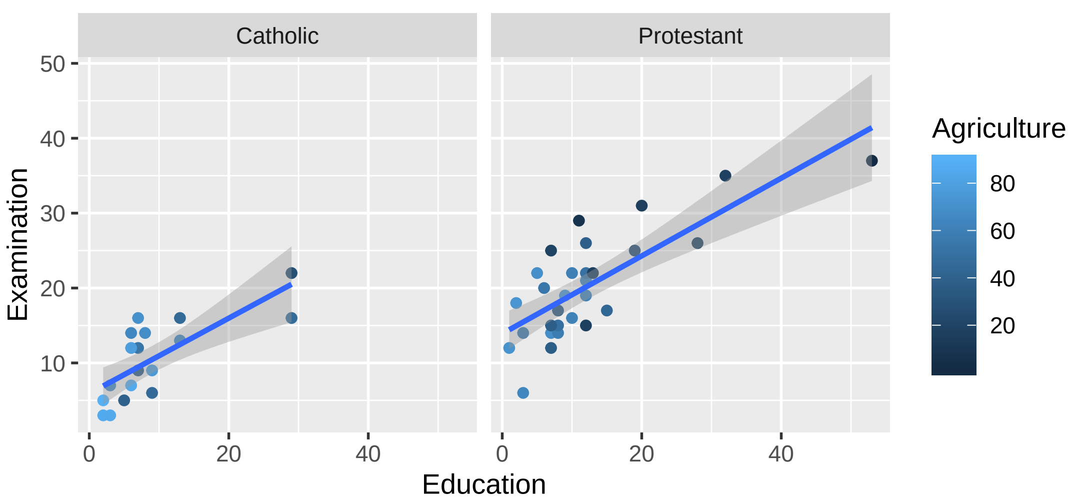

```
ggplot(data = swiss, aes(x = Education, y = Examination)) +
     geom_point() +
     geom_smooth(method = 'lm') +
     facet_wrap(~Religion)
```




### 2.4.3 Additional aesthetics

Knowing a little bit about Swiss history and geography, we realize that particularly rural cantons in mountain regions remained Catholic during the reformation. In addition, cantonal school systems historically took into account that children have to help their parents on the farms during the summers. Thus in some rural cantons schools were closed from spring until autumn. Hence, we might want to indicate in the plot which point refers to a predominantly agricultural district. We use the aesthetics of the point geometry to color the points according to the '`Agriculture`'-variable (the % of males involved in agriculture as occupation).

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



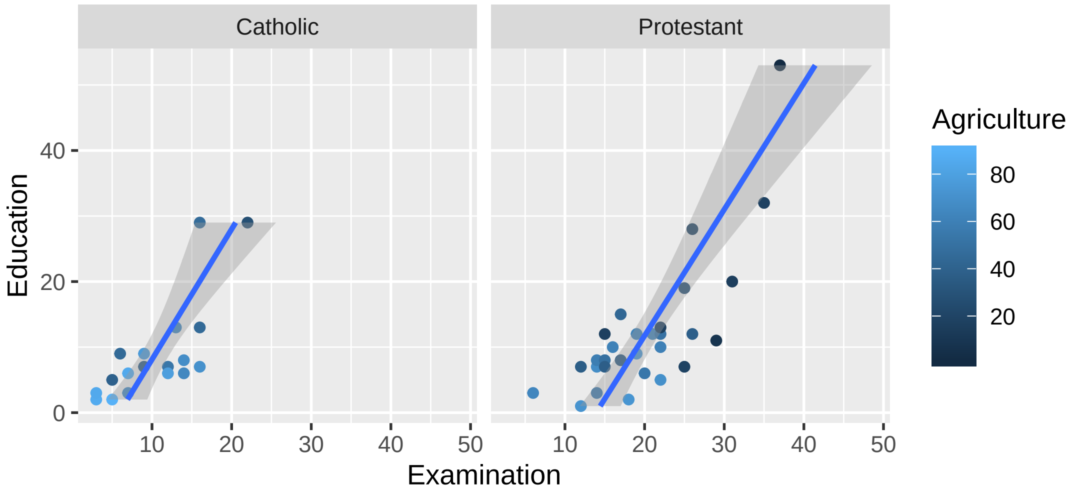

The resulting picture is in line with what we have expected. Overall, the districts with a lower share of occupation in agriculture tend to have rather higher levels of education as well as higher achievements in the examination.

#### 2.4.4 Themes: Fine-tuning the plot

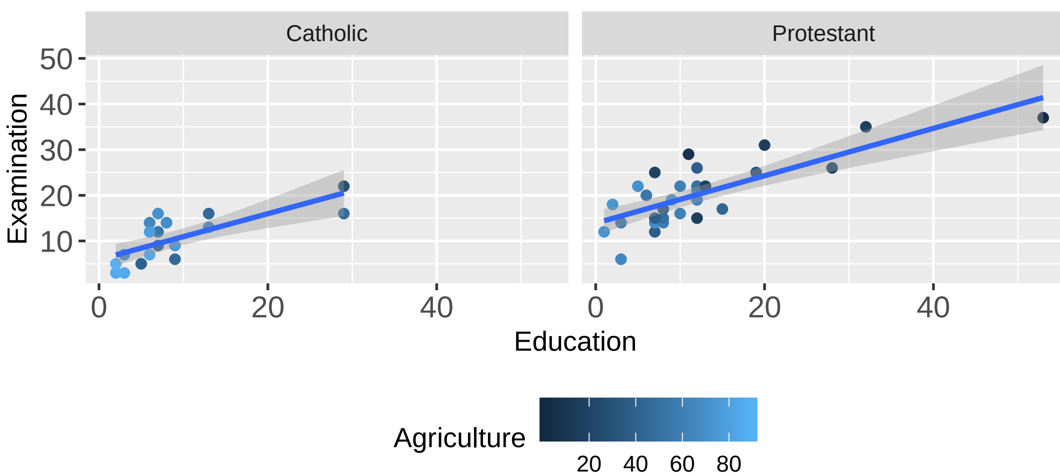
Finally, there are countless options to further refine the plot. For example, we can easily change the orientation/coordinates of the plot:

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




In addition, the theme()-function allows to change almost every aspect of the plot (margins, font face, font size, etc.). For example, we might prefer to have the plot legend at the bottom and have larger axis labels.

```
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) )
```

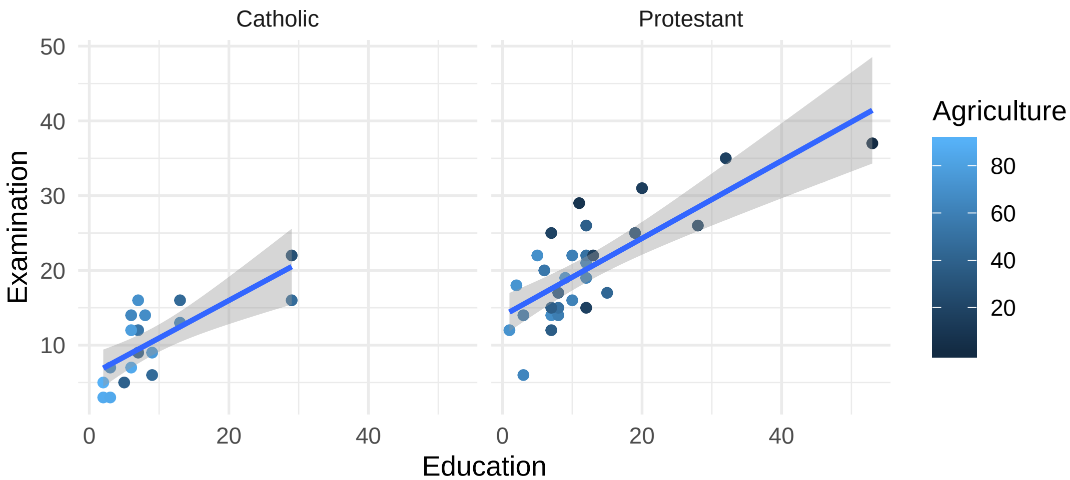



Moreover, several theme-templates offer ready-made designs for plots:

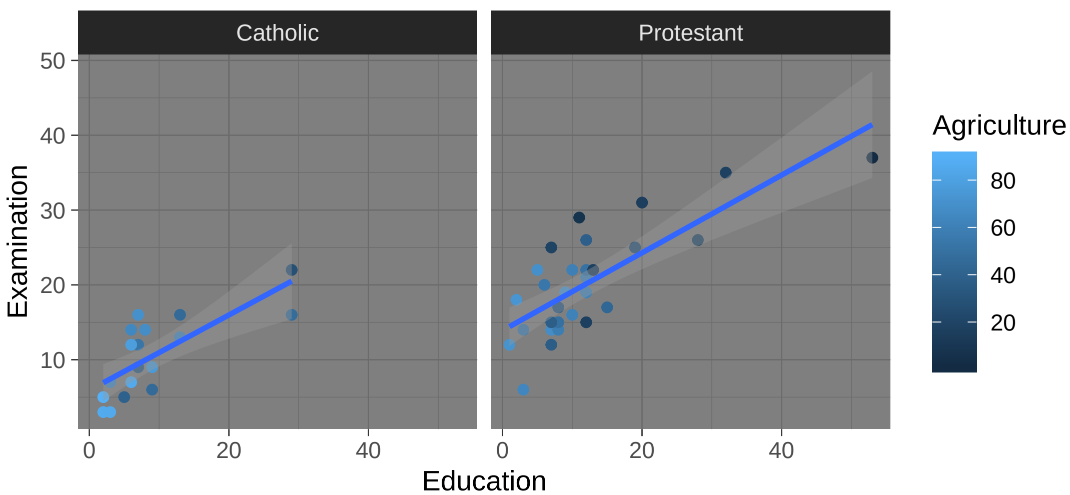
```
ggplot(data = swiss, aes(x = Education, y = Examination)) +
     geom_point(aes(color = Agriculture)) +
     geom_smooth(method = 'lm') +
```

```
    facet_wrap(~Religion) +
    theme_minimal()
```




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




# 3 Dynamic documents

Dynamic documents are a way to directly/dynamically integrate results of an analysis in R (numbers, tables, plots) in written text (a report, thesis, slide set, website, etc.). That is, we can write a report in so-called 'R-Markdown' format and place directly in the same document 'chunks' of R code which we want to be executed each time we 'knit' the report. Knitting the document means that the following steps are executed under the hood:

1. The code in the R chunks is executed and the results cached and formatted for print.
2. The formatted R output is embedded in a so-called 'Markdown'-file (`.md`).
3. The markdown-file is rendered as either a PDF, HTML, or Word-file (with additional formatting options, depending on the output format).

The entire procedure from importing, cleaning, analyzing, and visualising data can thus be combined in one document, based on which we can generate a meaningful output to communicate our results (in fact, the initial dashboard example, as well as all notes and slides of this course are generated in this way).

# 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.