Data Handling: Import, Cleaning and Visualisation

Lecture 11:

Visualisation and Dynamic Documents

Prof. Dr. Ulrich Matter

16/12/2021

Updates

Last exercise session: on Zoom!!

- Thursday, 21 December
  - 16:00-18:15
  - On Zoom

Last lecture: on Zoom!!

- Thursday, 23 December
  - online only!
  - Wrap up
  - Exam info
  - Feedback
  - Q&A (send questions until tomorrow! ulrich.matter@unisg.ch)!

Exchange students exam

- Thursday, 23 December
  - **In person/on campus!**
  - Decentral exam for **exchange students**!
  - See Canvas for details on place/time.

Data Display

Data display

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

Data display

```r
# 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
```

Problems?

Data display: round numeric values

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

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

Data display: 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

Data display: `format()` example

```
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()`/`tolowe()`.
- Remove white spaces: `trimws()`,

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

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

```
tolower(string)
```

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

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

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

Data Visualisation with R (`ggplot2`)

Data visualisation

- Final step of data pipeline/data science procedure!
    - Convincingly communicating insights from data.
- R is a very powerful tool to do this!
    - (Very powerful graphics engine)

Data visualisation in R

Three main approaches:

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

Data visualisation in 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.

Data visualisation in 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

`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
# load the R package
library(tidyverse) # automatically loads 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.
## Delemont          83.1        45.1           6         9    84.84             22.
## Franches-Mnt      92.5        39.7           5         5    93.40             20.
## Moutier           85.8        36.5          12         7    33.77             20.
## Neuveville        76.9        43.5          17        15     5.16             20.
## Porrentruy        76.1        35.3           9         7    90.57             26.
```

Add indicator variable

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

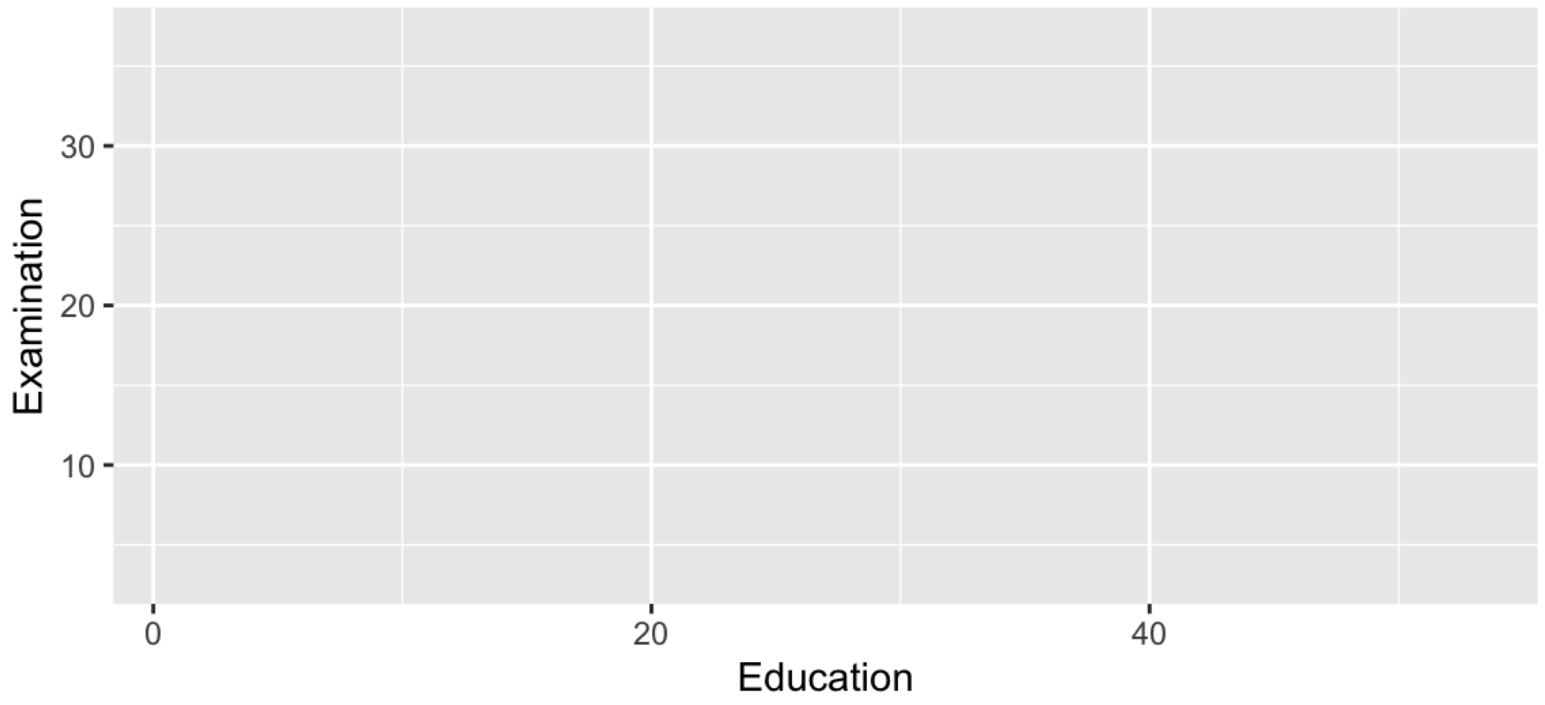
```r
# via tidyverse/mutate
swiss <- mutate(swiss,
                Religion =
                  ifelse(50 < Catholic, 'Catholic', 'Protestant'))

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

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

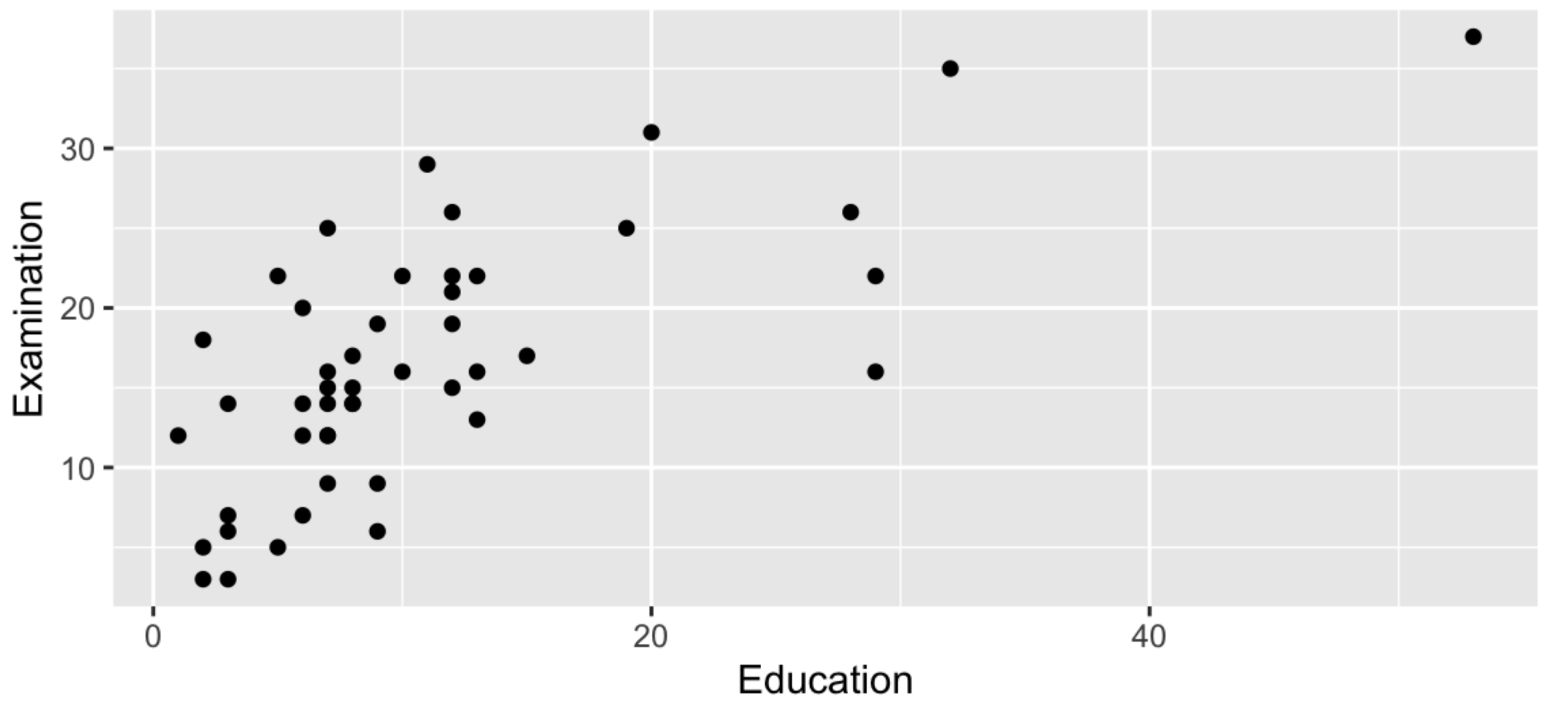
Data and aesthetics

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



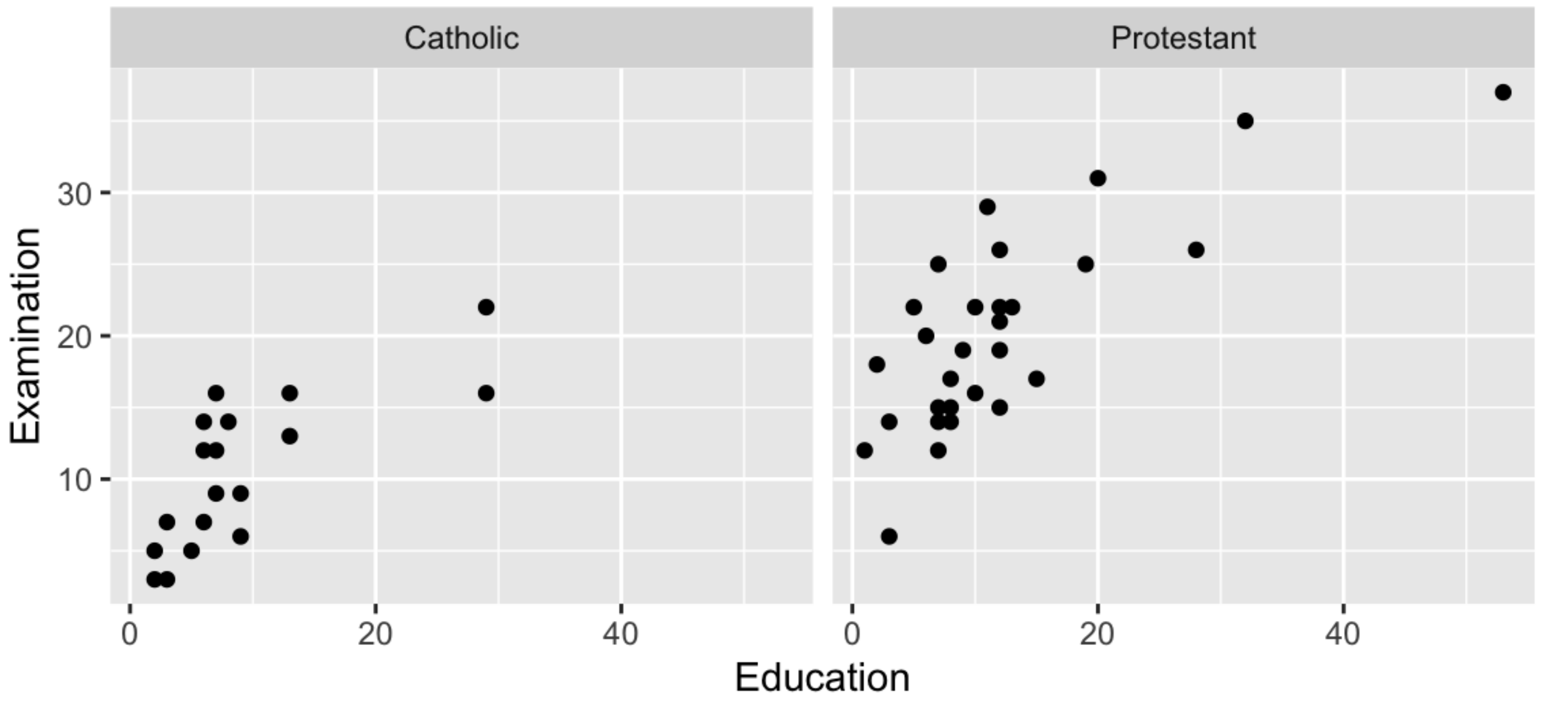
Geometries (~the type of plot)

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



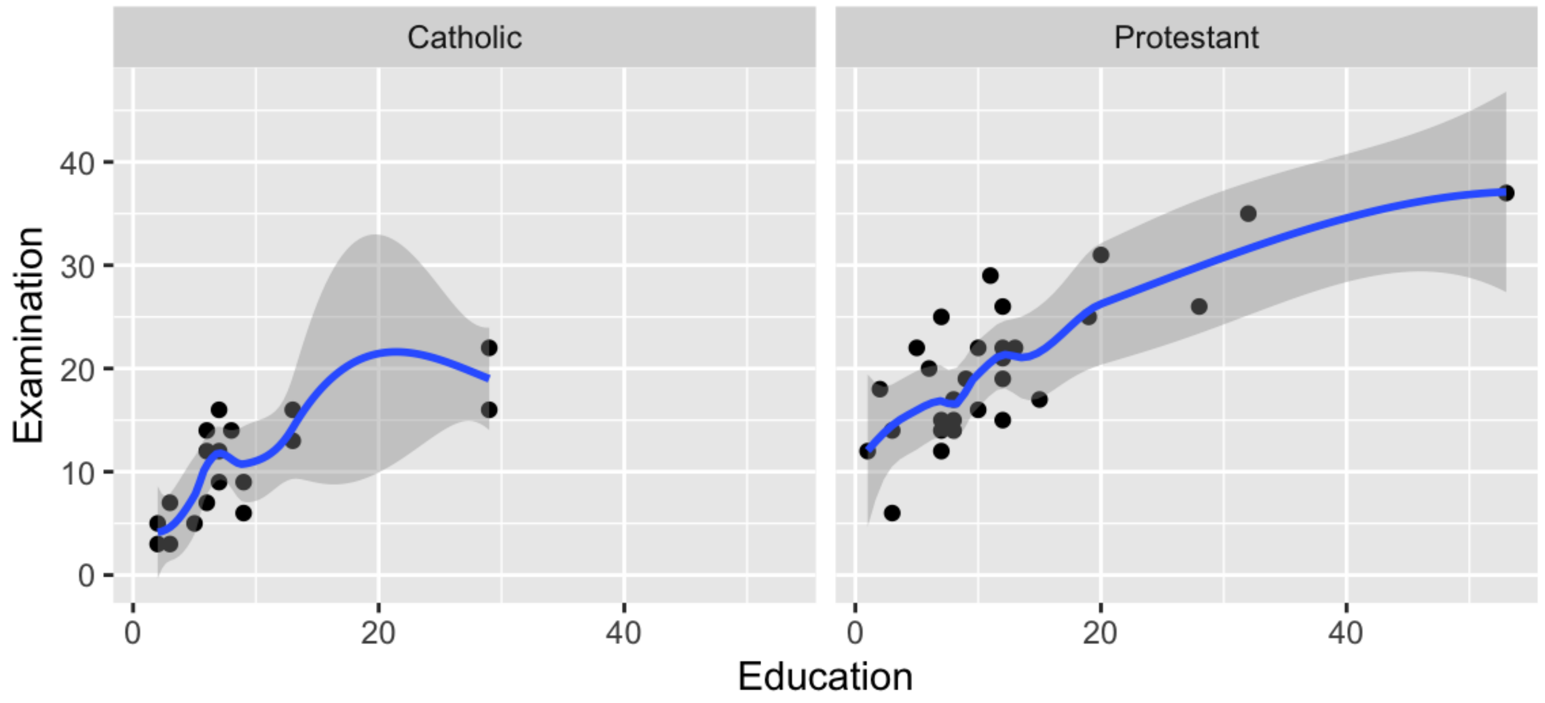
Facets

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



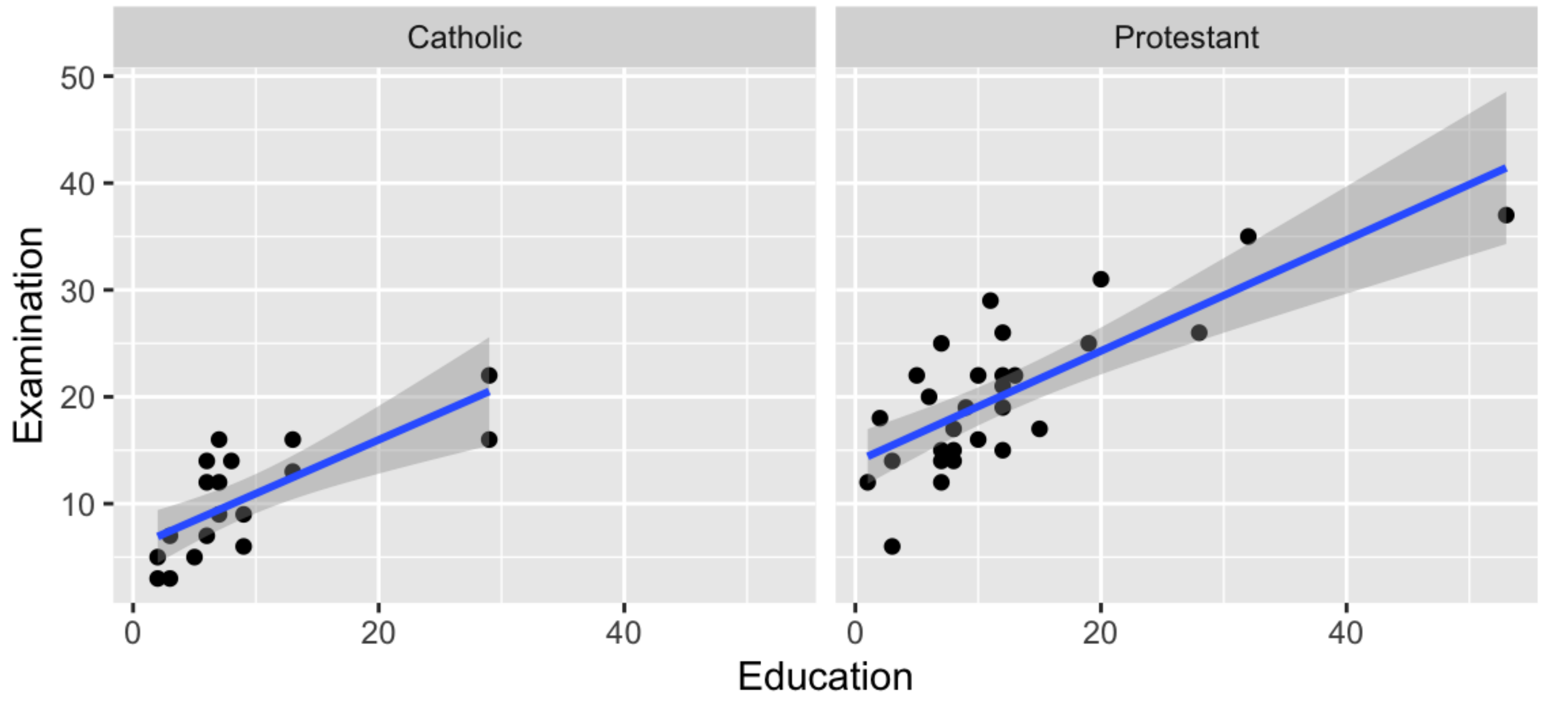
Additional layers and statistics

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



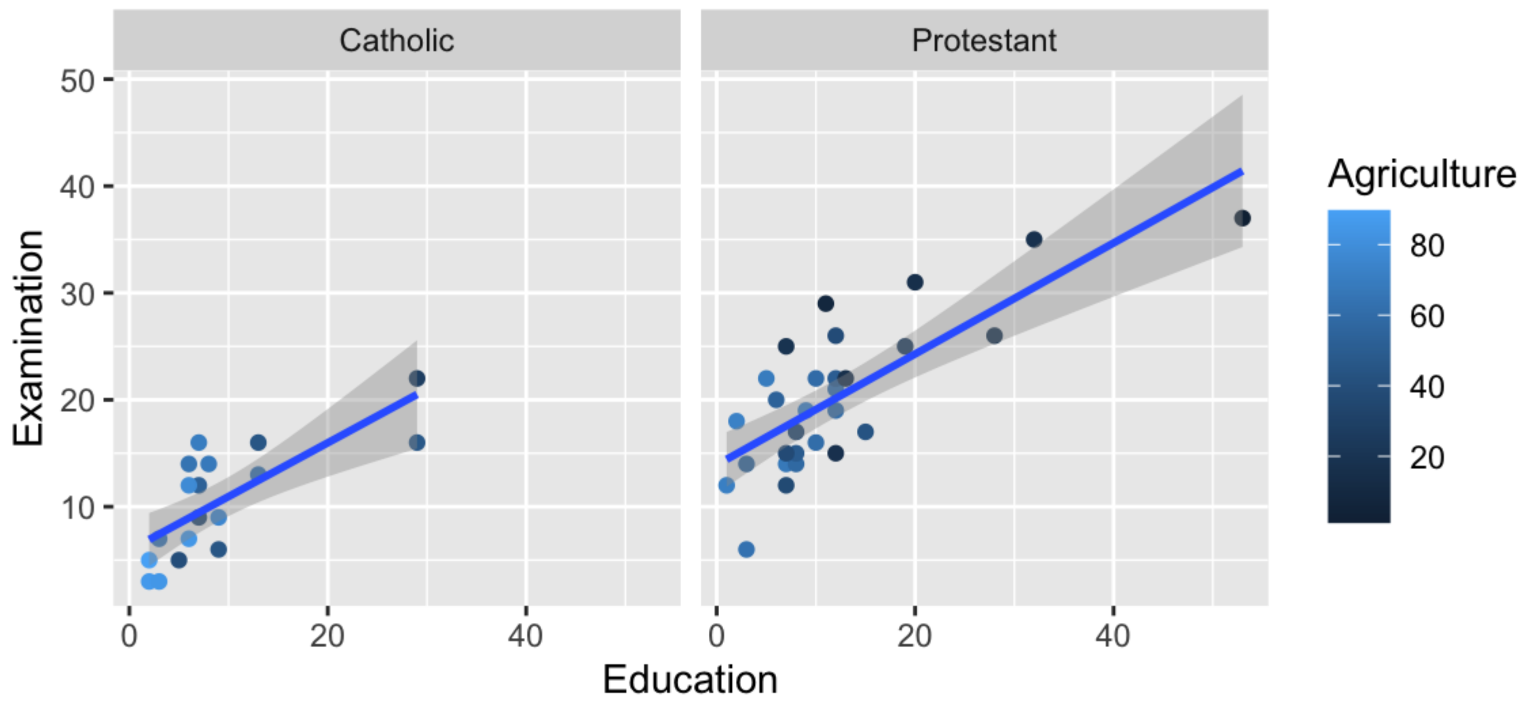
Additional layers and statistics

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



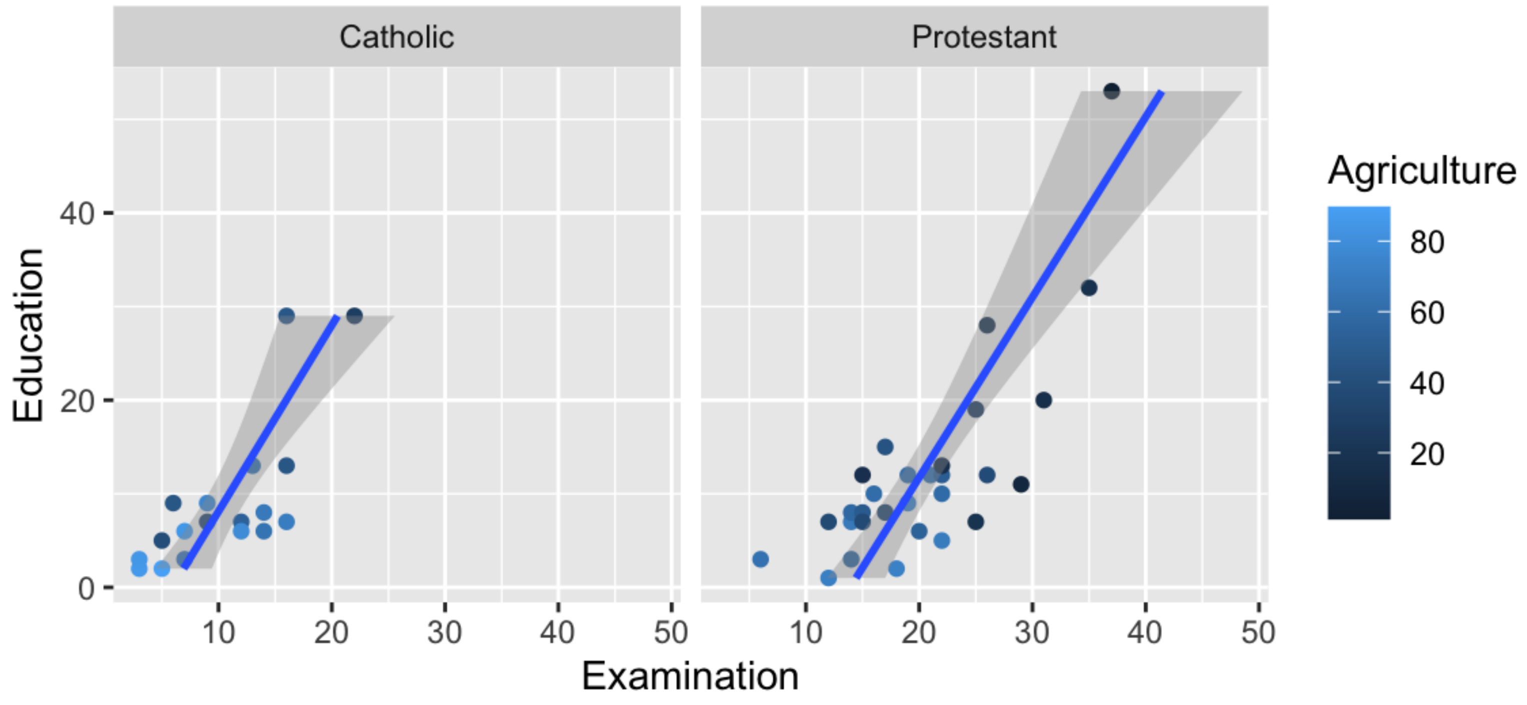
Additional aesthetics

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



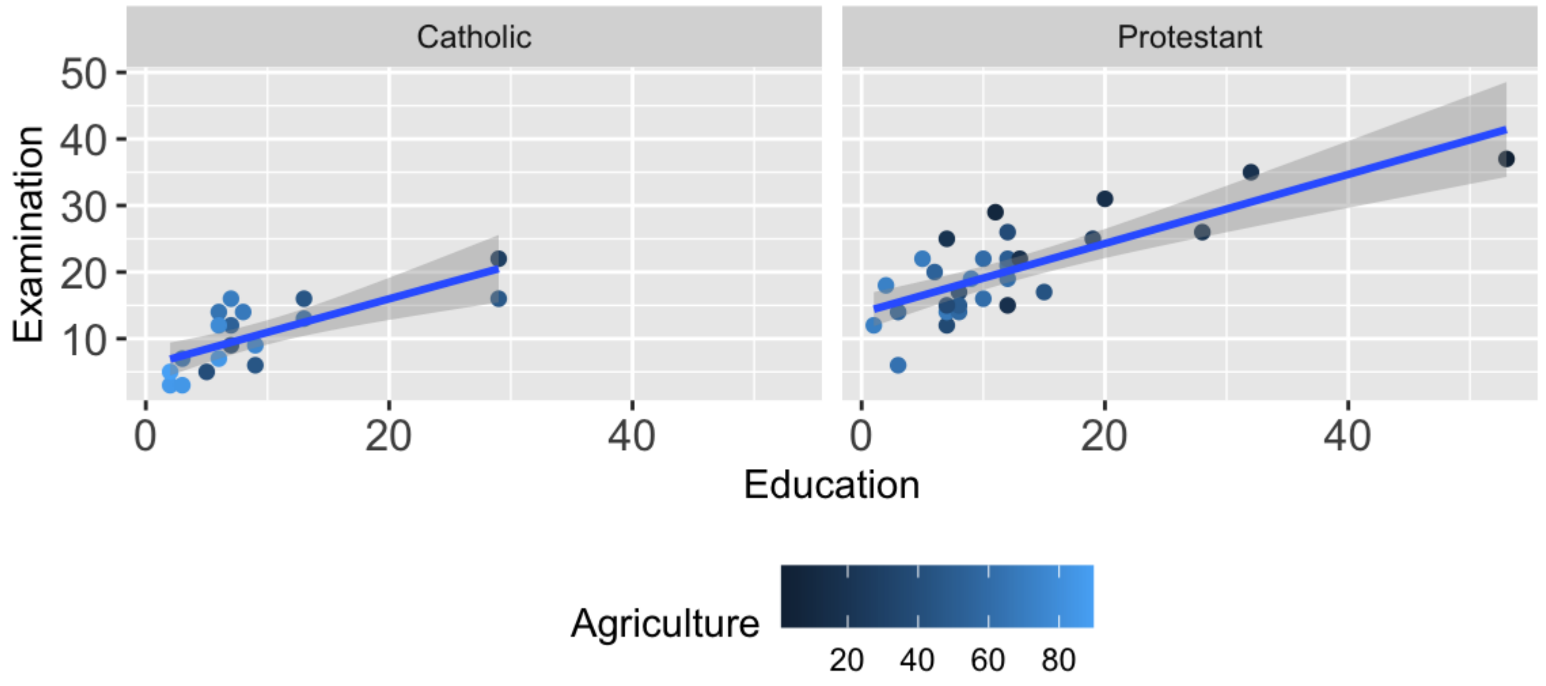
Change coordinates

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



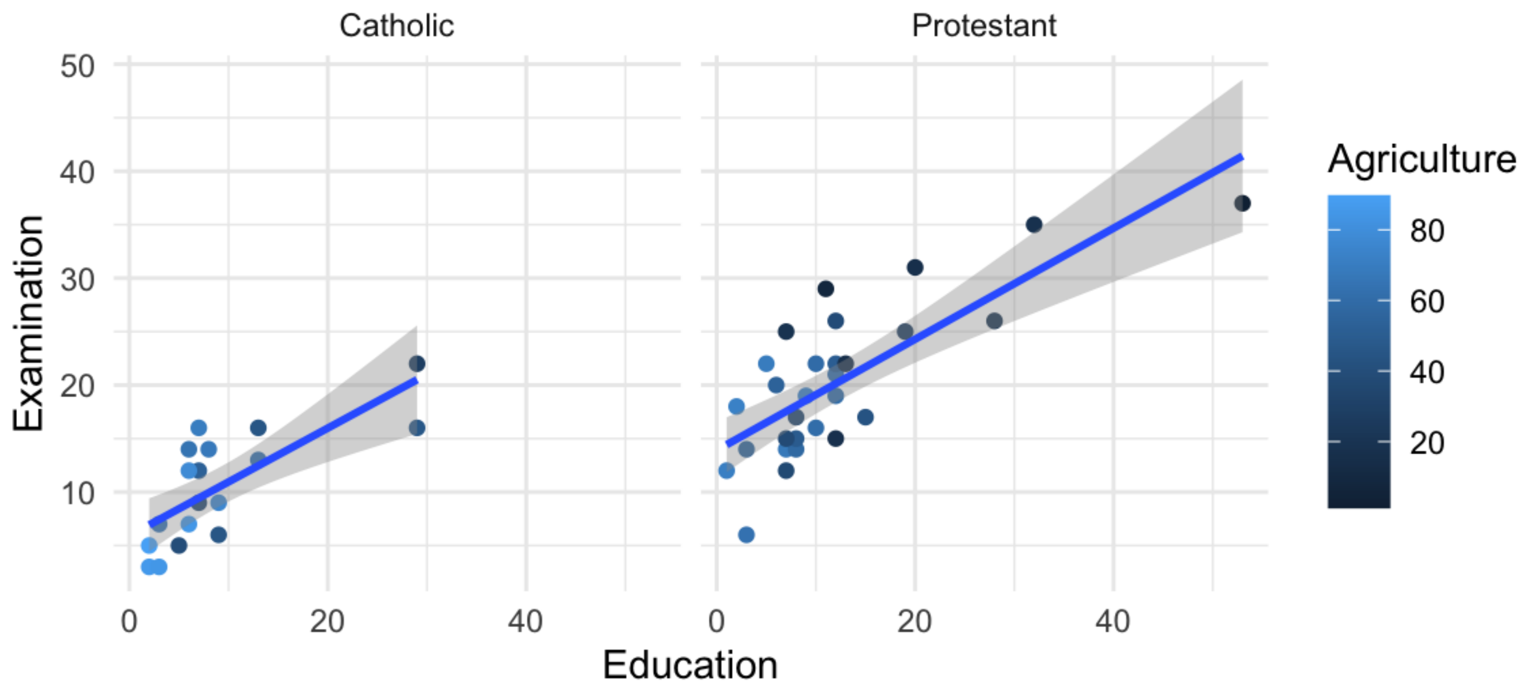
Themes

```
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) )
```



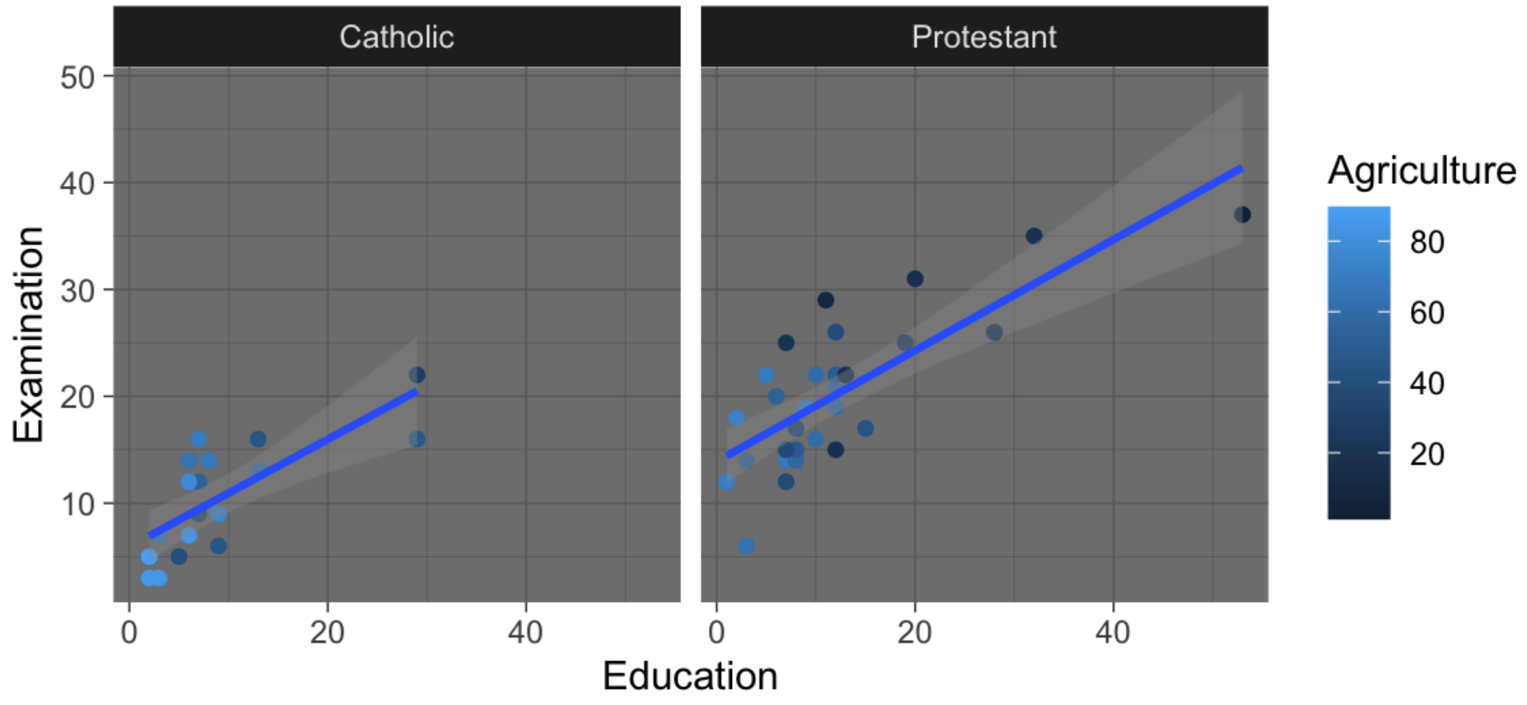
Themes

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



Themes

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



Dynamic Documents

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.