# A Brief Introduction to Programming with R

MEcon & MiQE/F Introductory Week

Erik Senn & Jeremia Stalder

2025-09-05

# Part I: Background / Tools

# Schedule

## Morning Sessions

09:15 - 10:00
Introduction, Background, Tools

10:00 - 10:15
Break, support with installations

10:15 - 11:00
Exercises, First steps with R

11:00 - 11:15
Break, Q&A

11:15 - 11:45
First steps with R, Concepts

## Afternoon Sessions

11:45 - 13:15
Lunch (individually)

13:15 - 14:00
Exercises

14:00 - 14:15
Break, Q&A

14:15 - 15:00
Working with Data

# Welcome

1. **Fire up** your notebooks!
2. **Download** (or clone) the course materials
    - GitHub Repository
    - Course slides available online

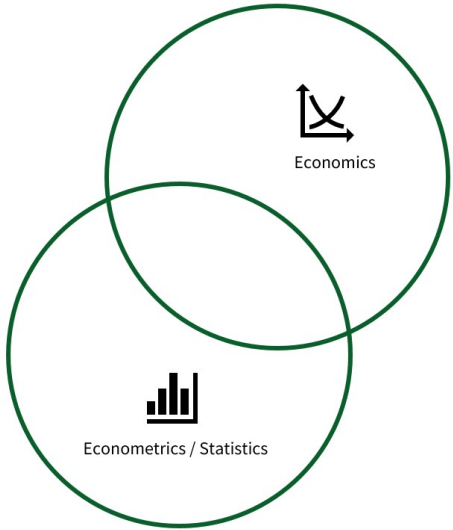
# Why learn to program (now)?

# Background: Technological change

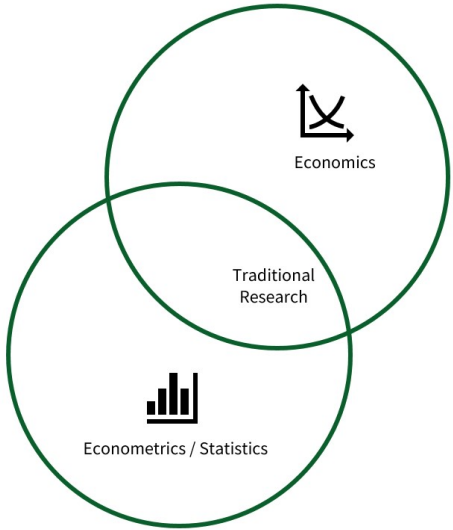
- Computers have become **omnipresent**
- Data is one of the world's most **valuable resources**
- **AI and machine learning** are reshaping every business and industry

# Perspective: Data Science and Economics

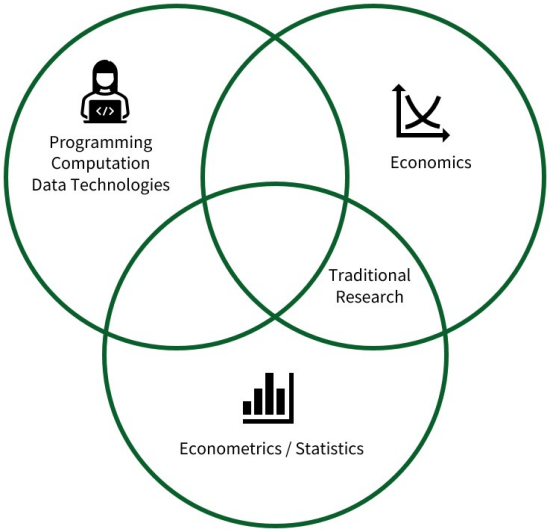
# Perspective: Data Science and Economics



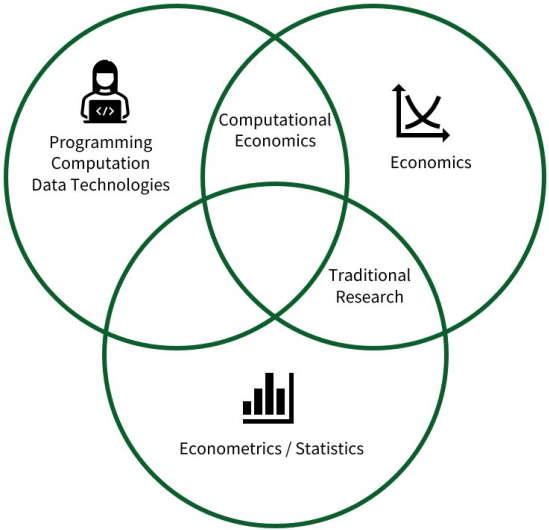
# Perspective: Data Science and Economics



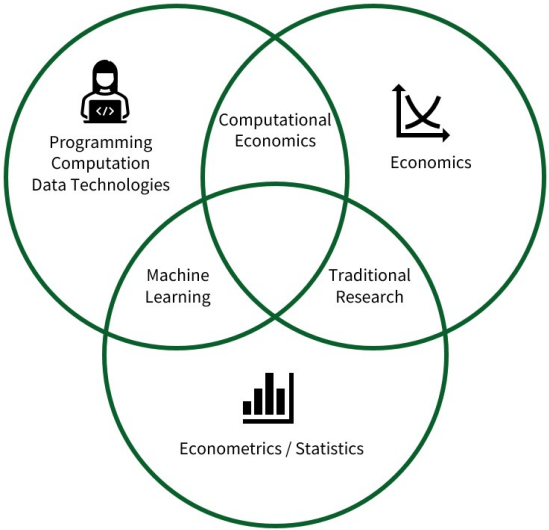
# Perspective: Data Science and Economics



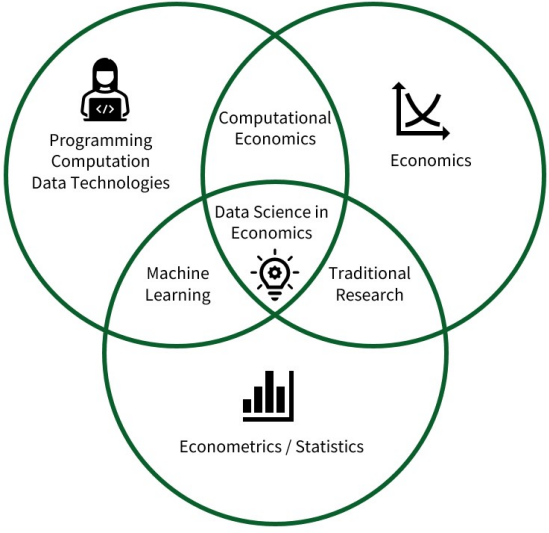
# Perspective: Data Science and Economics



# Perspective: Data Science and Economics



# Perspective: Data Science and Economics



# Why R?

# A data language

- Widely used in **data science** jobs
- Particularly adapted to program with **data**
- Originally designed for **statistical analysis**
- Competing with Python as the **top data science language**

# A high-level language

- **Relatively easy** to learn

# A high-level language

- **Relatively easy** to learn
- Extensive **free resources**:

# A high-level language

- **Relatively easy** to learn
- Extensive **free resources**:
  - DataCamp: Introduction to R

# A high-level language

- **Relatively easy** to learn
- Extensive **free resources**:
  - DataCamp: Introduction to R
  - RStudio Cheatsheets

# A high-level language

- **Relatively easy** to learn
- Extensive **free resources**:
  - DataCamp: Introduction to R
  - RStudio Cheatsheets
  - Stack Overflow

# The Tools

# R

**R** is the **programming language.**

You can download R from: https://cran.r-project.org/

# RStudio

**RStudio** is the **integrated development environment (IDE).**

You can download RStudio from: https://www.rstudio.com/

# RStudio Cloud

**RStudio Cloud** lets you use RStudio **without a local installation**

You can use RStudio Cloud by registering here: https://rstudio.cloud

# Exercises

# Exercise A: Setting up a Working Environment

1. Open **RStudio** and navigate to your desired working folder
2. Create a new folder called **r_course**
3. Set it as your **working directory**
4. Create subfolders: **data** and **code**
5. Create a new **R Project** in the `r_course` folder

TODO Add screenshots of Rstudio and explain each part TODO Add quick intro to here package for clean project management TODO Add how to use copilot?

# Exercise B: R Scripts

1. In the R console, type:

```
print("Hello world")
```

2. Create a new **R script**: `File > New File > R Script`
3. Type the same code in the script and **run it**

# Part II: First Steps and Basic Concepts

# First Steps in R

# Variables and Vectors

```
a <- 2
a
```

```
[1] 2
```

# Working with Vectors

R easily allows to work with **vectors** of data!

```
# Instantiate an integer vector 'a'
a <- c(10, 22, 33, 22, 40)
# Give names to the vector's elements
names(a) <- c("Andy", "Betty", "Claire",
              "Daniel", "Eva")
a  # Display the vector
```

```
  Andy  Betty Claire Daniel    Eva
    10     22     33     22     40
```

# Indexing

We can access **single** (or **multiple**) elements of a vector:

```
a[3]     # Access the 3rd element
```

```
Claire
    33
```

```
a[3:5]   # Access the 3rd to 5th elements
```

```
Claire Daniel    Eva
    33     22     40
```

```
a["Claire"]  # Access by name
```

```
Claire
    33
```

# Inspecting variables

```
class(a)  # Display the class
```

```
[1] "numeric"
```

```
str(a)    # Display the structure
```

```
 Named num [1:5] 10 22 33 22 40
 - attr(*, "names")= chr [1:5] "Andy" "Betty" "Claire" "Daniel" ...
```

# Math Operators

## Basic operators:

- $+$ : `+`
- $-$ : `-`
- $\times$ : `*`
- $\div$ : `/`

## More operators:

- $a^n$ : `a^n`
- $\sqrt{a}$ : `sqrt(a)`
- $\ln a$ : `log(a)`
- $e^n$ : `exp(n)`

# Basic Programming Concepts

# Loops

- **Repeatedly execute** a sequence of commands
- For a **known** or **unknown** number of iterations
  - **for-loop**: number of iterations typically known
  - **while-loop**: iterate until a condition is met

# for-loops in R

```r
n_iter <- 5  # Define number of iterations
# Specify the loop
for (i in 1:n_iter) {
    print(i)  # Print the number 'i'
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

# for-loops: Summing numbers

```
numbers <- c(72, 42, 150, 13, 36, 19)
total_sum <- 0  # Initialize sum

# Specify the loop
for (n in numbers) {
    total_sum <- total_sum + n
}
total_sum
```

```
[1] 332
```

# Nested for-loops

```
n_iter_inner <- 100
n_iter_outer <- 500

# Start outer loop
for (i in 1:n_iter_outer) {
    # Code for outer loop

    # Start inner loop
    for (j in 1:n_iter_inner) {
        # Code for inner loop
    }
}
```

How many iterations does this combination amount to?

# Booleans and Logical Statements

```
2 + 2 == 4  # Is 2+2 equal to 4?
```

```
[1] TRUE
```

```
3 + 3 == 7  # Is 3+3 equal to 7?
```

```
[1] FALSE
```

```
4 != 7      # Is 4 not equal to 7?
```

```
[1] TRUE
```

# Control Flow with Booleans

```
condition <- TRUE

if (condition) {
    print("The condition is true!")
} else {
    print("The condition is false!")
}
```

```
[1] "The condition is true!"
```

# R Functions

- Functions take **parameter values** as input, process these values, and **return** results
- Many functions are **provided with R**
- Additional functions via **packages**

```
numbers <- c(13, 25, 39, 881)
mean(numbers)    # Compute the mean
```

```
[1] 240
```

```
sd(numbers)      # Standard deviation
```

```
[1] 428
```

```
median(numbers)  # Median
```

# Creating Custom Functions

```
# Define custom mean function
my_mean <- function(x) {
    x_bar <- sum(x) / length(x)
    return(x_bar)
}

# Test the function
my_mean(numbers)
```

```
[1] 240
```

```
mean(numbers)  # Compare with built-in
```

```
[1] 240
```

# Data Structures and Indices

# Vectors and Lists

```
# Integer vector
integer_vector <- 9:20
integer_vector[2]    # Second element
```

```
[1] 10
```

```
integer_vector[2:5]  # Second to fifth
```

```
[1] 10 11 12 13
```

```
# String vector
string_vector <- c("a", "b", "c")
string_vector[-3]    # All except third
```

```
[1] "a" "b"
```

# Lists

Lists can contain **different types** of elements:

```
# Create a list
my_list <- list(
    numbers = integer_vector,
    letters = string_vector,
    condition = TRUE
)
str(my_list)
```

```
List of 3
 $ numbers  : int [1:12] 9 10 11 12 13 14 15 16 17 18 ...
 $ letters  : chr [1:3] "a" "b" "c"
 $ condition: logi TRUE
```

# Accessing List Elements

```
# Access by name
my_list$numbers[1:3]
```

```
[1]  9 10 11
```

```
my_list[["letters"]]
```

```
[1] "a" "b" "c"
```

```
# Access by index
my_list[[1]][1:3]
```

```
[1]  9 10 11
```

# Matrices

```
# Create a matrix
my_matrix <- matrix(integer_vector, nrow = 4)
my_matrix
```

```
     [,1] [,2] [,3]
[1,]    9   13   17
[2,]   10   14   18
[3,]   11   15   19
[4,]   12   16   20
```

```
my_matrix[2,]      # Second row
```

```
[1] 10 14 18
```

```
my_matrix[, 1:2]   # First two columns
```

# Data Frames

```r
# Create a dataframe
my_df <- data.frame(
    Name = c("Alice", "Betty", "Claire"),
    Age = c(20, 30, 45)
)

my_df
```

| Name | Age |
|---|---|
| Alice | 20 |
| Betty | 30 |
| Claire | 45 |

```r
my_df$Age          # Access column
```

# Exercises

# Exercise A: Write a Sum Function

Write a function that takes a **numeric vector** as input and returns the **sum** of the vector's elements.

```
my_sum <- function(x) {
    # Your code here
}
```

Test by comparing with built-in `sum()` function.

# Exercise B: Robustness and Warnings

Test your `my_sum()` function with:

```
numbers2 <- c("1", "2", "3")
```

Add **error checking** to make the function robust.

# Exercise C: Standard Deviation Function

Implement a function to compute the **standard deviation**:

$$\text{SD} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

# Exercise D: Standard Error Function

Building on Exercise C, implement:

$$\text{SE}_{\bar{x}} = \frac{\text{SD}}{\sqrt{N}}$$

# Exercise E: T-test

Implement a **one-sample t-test** function:

$$t = \frac{\bar{x} - \mu_0}{\text{SE}_{\bar{x}}}$$

# Exercise F: Fibonacci Sequence

Generate the first 30 **Fibonacci numbers** where:

- $F_0 = 0, F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$ for $n > 1$

# Exercise G: Multiples of 3 and 5 (*)

Write a function that computes the sum of all **multiples of 3 or 5** up to a number $N$.

**Hint:** Multiples of both 3 and 5 should only be added once!

# Exercise H: Prime numbers (*)

Write a function that computes the sum of all **prime numbers** up to a number $N$.

# Part III: Working with Data

# Loading/Importing Data

# Loading built-in R data sets

```
# Load built-in dataset
data(swiss)

# Check if loaded
class(swiss)
```

```
[1] "data.frame"
```

The `data()` function loads **built-in R datasets** into your environment.

# Inspect the data structure

```
# Structure of the swiss dataset
str(swiss)
```

```
'data.frame':   47 obs. of  6 variables:
 $ Fertility       : num  80.2 83.1 92.5 85.8 76.9 76.1 83.8 92.4 82.
 $ Agriculture     : num  17 45.1 39.7 36.5 43.5 35.3 70.2 67.8 53.3
 $ Examination     : int  15 6 5 12 17 9 16 14 12 16 ...
 $ Education       : int  12 9 5 7 15 7 7 8 7 13 ...
 $ Catholic        : num  9.96 84.84 93.4 33.77 5.16 ...
 $ Infant.Mortality: num  22.2 22.2 20.2 20.3 20.6 26.6 23.6 24.9 21
```

# First few rows of the data

```
head(swiss, 4)
```

| | 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 |

# Comma Separated Values (CSV)

Example of **CSV format**:

```
"","Fertility","Agriculture","Examination",...
"Courtelary",80.2,17,15,...
"Delemont",83.1,45.1,6,...
```

# Importing data from different sources

```
# CSV files
swiss_csv <- read.csv("./data/swiss.csv")

# Excel files (requires readxl)
library(readxl)
swiss_excel <- read_excel("./data/swiss.xlsx")

# SPSS files (requires haven)
library(haven)
swiss_spss <- read_spss("./data/swiss.sav")
```

# Introduction to the Tidyverse

# What is the Tidyverse?

*"The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures."*

To install and load:

```
install.packages("tidyverse") # Install once

library(tidyverse) # Load in each session
```

# The Pipe Operator |>

Transform nested functions into **readable pipelines**:

```
# Traditional approach
head(select(swiss, Fertility, Education), 3)

# With pipe operator
swiss |>
  select(Fertility, Education) |>
  head(3)
```

Note: |> is built into R (since version 4.1.0). %>% is from the `magrittr` package, which has additional features.

# Select columns with `select()`

```
# Select specific columns
swiss |>
  select(Fertility, Education, Catholic) |>
  head(3)
```

| | Fertility | Education | Catholic |
|---|---|---|---|
| Courtelary | 80.2 | 12 | 9.96 |
| Delemont | 83.1 | 9 | 84.84 |
| Franches-Mnt | 92.5 | 5 | 93.40 |

# Select columns - Advanced patterns

```
# Select by pattern
swiss |>
  select(starts_with("E") | contains("Mort")) |>
  head(3)
```

| | Examination | Education | Infant.Mortality |
|---|---|---|---|
| Courtelary | 15 | 12 | 22.2 |
| Delemont | 6 | 9 | 22.2 |
| Franches-Mnt | 5 | 5 | 20.2 |

# Filter rows with `filter()`

```
# Keep rows meeting conditions
swiss |>
  filter(Education > 20) |>
  select(Fertility, Education, Catholic) |>
  head()
```

| | Fertility | Education | Catholic |
|---|---|---|---|
| Lausanne | 55.7 | 28 | 12.1 |
| Neuchatel | 64.4 | 32 | 16.9 |
| V. De Geneve | 35.0 | 53 | 42.3 |
| Rive Droite | 44.7 | 29 | 50.4 |
| Rive Gauche | 42.8 | 29 | 58.3 |

# Create new variables with `mutate()`

```
swiss |>
  mutate(
    High_Education = Education > 15,
    Fert_per_100 = Fertility / 100
  ) |>
  select(Education, High_Education,
         Fertility, Fert_per_100) |>
  head(3)
```

| | Education | High_Education | Fertility | Fert_per_100 |
|---|---|---|---|---|
| Courtelary | 12 | FALSE | 80.2 | 0.802 |
| Delemont | 9 | FALSE | 83.1 | 0.831 |
| Franches-Mnt | 5 | FALSE | 92.5 | 0.925 |

# Arrange rows with `arrange()`

```
swiss |>
  arrange(desc(Education)) |>
  select(Education, Examination) |>
  head(4)
```

| | Education | Examination |
|---|---|---|
| V. De Geneve | 53 | 37 |
| Neuchatel | 32 | 35 |
| Rive Droite | 29 | 16 |
| Rive Gauche | 29 | 22 |

# Summarize data with `summarize()`

```
swiss |>
  summarize(
    mean_edu = mean(Education),
    sd_edu = sd(Education),
    median_fert = median(Fertility),
    n = n()
  )
```

| mean_edu | sd_edu | median_fert | n |
|---|---|---|---|
| 11 | 9.62 | 70.4 | 47 |

# Group operations with `group_by()`

```
# Group and summarize
swiss |>
  group_by(Catholic > 50) |>
  summarize(
    mean_education = mean(Education),
    mean_fertility = mean(Fertility),
    count = n()
  )
```

| Catholic > 50 | mean_education | mean_fertility | count |
|---|---|---|---|
| FALSE | 12.14 | 66.2 | 29 |
| TRUE | 9.11 | 76.5 | 18 |

# Combining multiple operations

```
swiss |>
  filter(Agriculture < 50) |>
  group_by(Catholic > 50) |>
  summarize(
    avg_exam = mean(Examination),
    n = n(),
    .groups = "drop"
  ) |>
  arrange(desc(avg_exam))
```

| Catholic > 50 | avg_exam | n |
|---|---|---|
| FALSE | 23.1 | 15 |
| TRUE | 12.3 | 6 |

# Reshape data: Wide to Long

```
# Original wide format
swiss_subset <- swiss |>
  slice(1:2) |>
  select(Fertility, Agriculture)

swiss_subset
```

| | Fertility | Agriculture |
|---|---|---|
| Courtelary | 80.2 | 17.0 |
| Delemont | 83.1 | 45.1 |

# Reshape data: `pivot_longer()`

```
# Convert to long format
swiss_subset |>
  pivot_longer(
    cols = everything(),
    names_to = "Metric",
    values_to = "Value"
  )
```

| Metric | Value |
|---|---|
| Fertility | 80.2 |
| Agriculture | 17.0 |
| Fertility | 83.1 |
| Agriculture | 45.1 |

# Join data frames (preparation)

```r
# Sample data
df1 <- data.frame(
  Student_ID = c(2501, 2502, 2503, 2504),
  Master = c("Mecon", "Mecon", "MiQE/F", "MiQE/F")
)

df2 <- data.frame(
  Student_ID = c(2501, 2502, 2503, 2504, 2505, 2506),
  Grades = c(6, 5, 5.5, 5.5, 4, 5)
)
```

# Join data frames

```
# Left join keeps all rows from df1
left_join(df1, df2, by = "Student_ID")
```

| Student_ID | Master | Grades |
|---|---|---|
| 2501 | Mecon | 6.0 |
| 2502 | Mecon | 5.0 |
| 2503 | MiQE/F | 5.5 |
| 2504 | MiQE/F | 5.5 |

# Working with dates using lubridate (preparation)

```
# Create and parse dates
dates <- c("2025-09-05", "2025-09-06")
parsed_dates <- ymd(dates)
```

# Working with dates using lubridate

```
# Extract components
data.frame(
  date = parsed_dates,
  year = year(parsed_dates),
  month = month(parsed_dates),
  weekday = wday(parsed_dates, label = TRUE)
)
```

| date | year | month | weekday |
|---|---|---|---|
| 2025-09-05 | 2025 | 9 | Fri |
| 2025-09-06 | 2025 | 9 | Sat |

# purrr: map functions

```
# Apply a function to multiple columns
swiss |>
  select(Fertility, Agriculture, Education) |>
  map_dbl(mean) |>
  round(2)
```

```
  Fertility Agriculture   Education
       70.1        50.7        11.0
```

# String manipulation with stringr

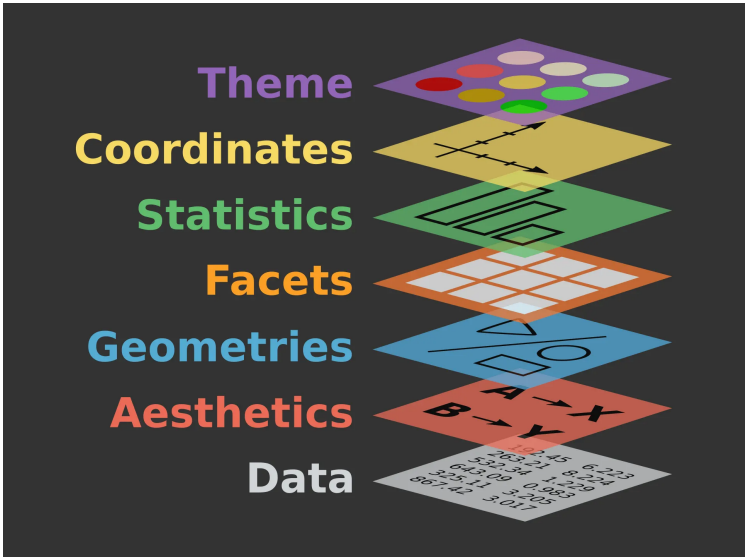
```r
masters <- c("Mecon", "MiQE/F", "MACFin") # Example text

# String operations
tibble(
  original = masters,
  lower = str_to_lower(masters),
  length = str_length(masters),
  contains_e = str_detect(masters, "e")
)
```

| original | lower | length | contains_e |
|---|---|---|---|
| Mecon | mecon | 5 | TRUE |
| MiQE/F | miqe/f | 6 | FALSE |
| MACFin | macfin | 6 | FALSE |

# Visualization with R (ggplot2)

# Grammar of Graphics - Layers



# ggplot2 basics

```
# Basic structure
ggplot(data = <DATA>,
       mapping = aes(x = <X>, y = <Y>)) +
  <GEOM_FUNCTION>() +
  <OPTIONAL_LAYERS>
```

# Prepare the data

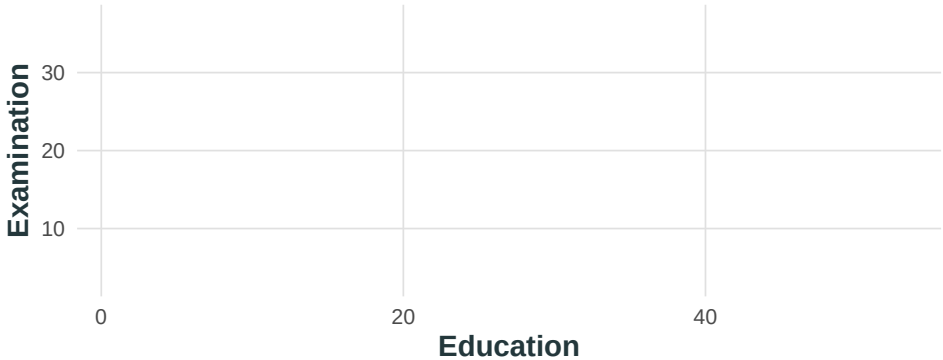
```
# Add Religion indicator
swiss <- swiss |>
  mutate(Religion = ifelse(Catholic > 50,
                           'Catholic', 'Protestant') |> factor())

# Check the data
table(swiss$Religion)
```

```
  Catholic Protestant
        18         29
```

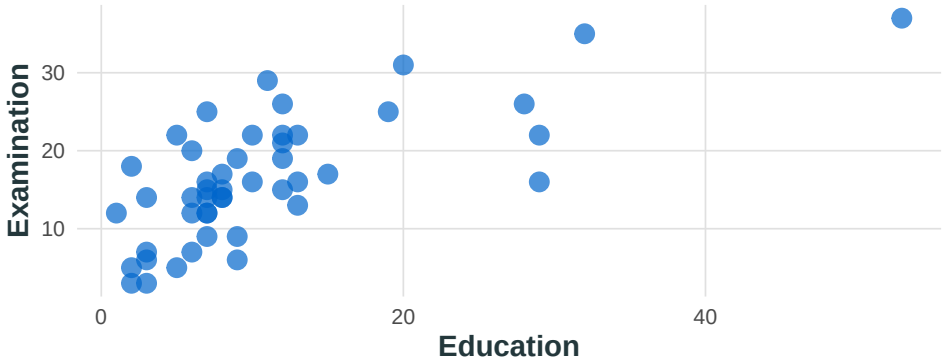
# Building a plot: Data + Aesthetics

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



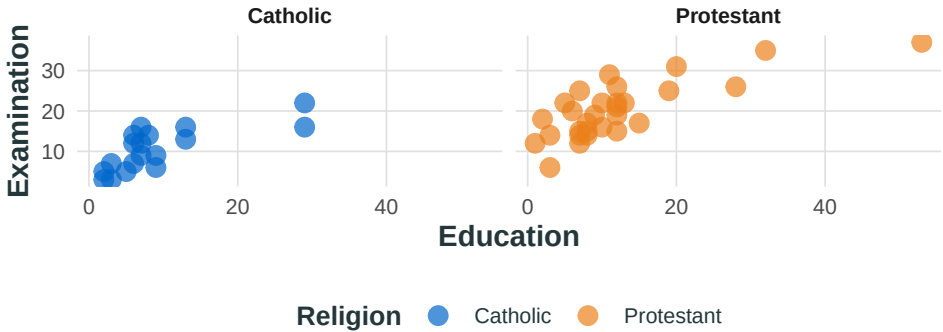
# Adding Geometries

```
ggplot(data = swiss,
       aes(x = Education, y = Examination)) +
  geom_point(size = 3, alpha = 0.7, color = "#0066CC")
```



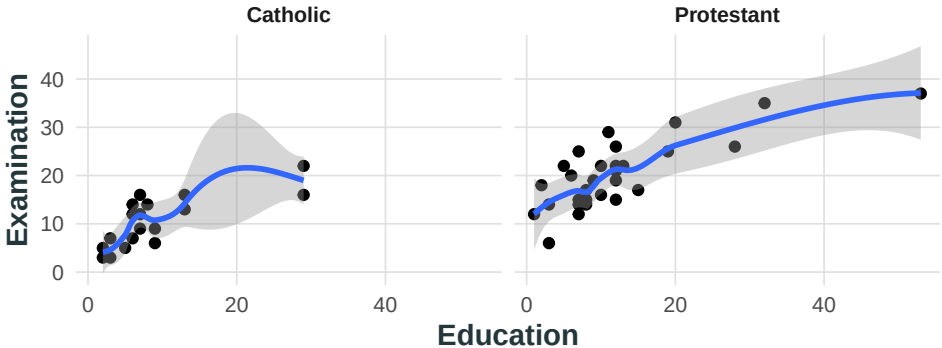
# Using Facets

```
ggplot(data = swiss,
       aes(x = Education, y = Examination)) +
  geom_point(size = 3, alpha = 0.7, aes(color = Religion)) +
  scale_color_modern() +
  facet_wrap(~Religion)
```



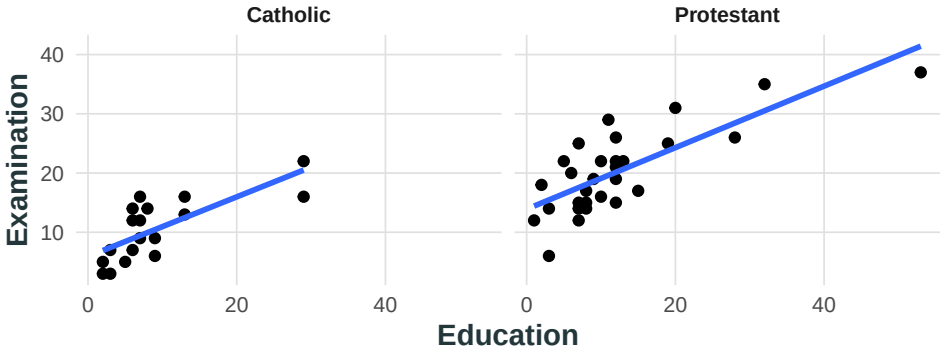
# Adding Statistics with loess

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



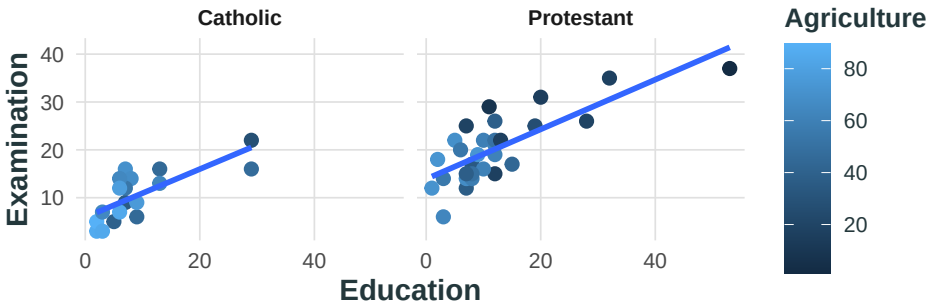
# Adding Statistics with lm

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



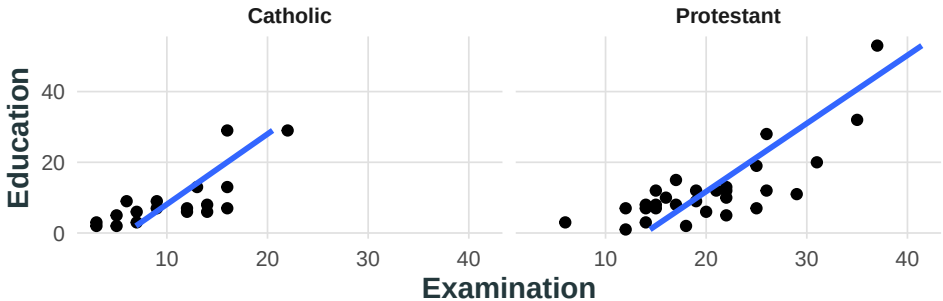
# Multiple Aesthetics

```
ggplot(data = swiss,
       aes(x = Education, y = Examination)) +
  geom_point(aes(color = Agriculture), size = 2) +
  geom_smooth(method = 'lm', se = FALSE) +
  facet_wrap(~Religion) +
  theme(legend.position = "right")
```



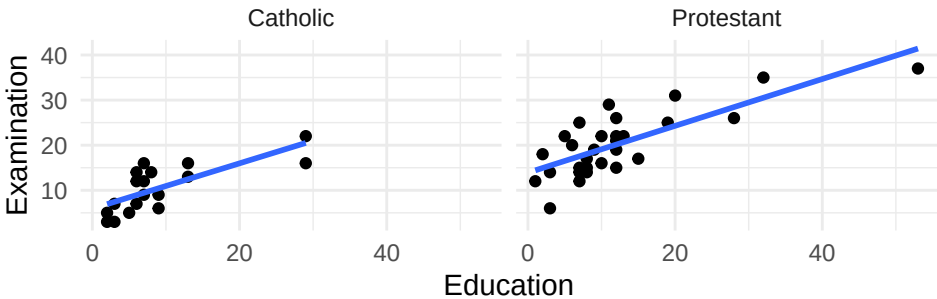
# Change coordinates

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



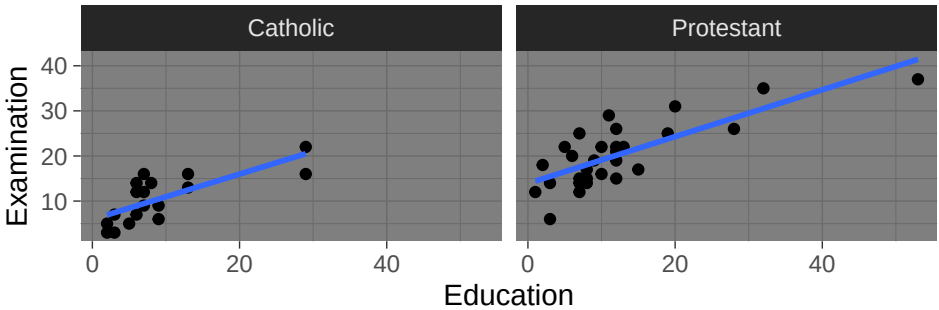
# Customizing Themes

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



# Pre-built Themes

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



# Save plots

```
# Save the last plot
ggsave("my_plot.png",
       width = 8, height = 5,
       dpi = 300)

# Save a specific plot
p <- ggplot(swiss, aes(Education, Examination)) +
  geom_point()

ggsave("scatter.pdf", plot = p,
       width = 6, height = 4)
```

# Basic Statistics with R

# Descriptive Statistics

```
# Create sample data
x <- c(10, 22, 33, 22, 40)

# Basic statistics
tibble(
  mean = mean(x),
  median = median(x),
  sd = sd(x),
  min = min(x),
  max = max(x)
)
```

| mean | median | sd | min | max |
|---|---|---|---|---|
| 25.4 | 22 | 11.5 | 10 | 40 |

# T-test Example

```r
# Generate sample data
set.seed(123)
sample <- rnorm(30, mean = 10, sd = 2)

# One-sample t-test
t_result <- t.test(sample, mu = 10)
t_result$p.value
```

```
[1] 0.794
```

```r
t_result$conf.int
```

```
[1]  9.17 10.64
attr(,"conf.level")
[1] 0.95
```

# Linear Regression - Setup

```r
# Define the model formula
model1 <- Examination ~ Education

# Fit the model
fit1 <- lm(model1, data = swiss)

# View coefficients
coef(fit1) # Use 'summary(fit1)' for more details
```

```
(Intercept)   Education
     10.127       0.579
```

# Multiple Linear Regression

```
# Fit model with multiple predictors
fit2 <- lm(Examination ~ Education + Catholic + Agriculture,
           data = swiss)

# Model fit statistics
tibble(
  R2 = summary(fit2)$r.squared,
  Adj_R2 = summary(fit2)$adj.r.squared,
  RMSE = sigma(fit2))
```

| R2 | Adj_R2 | RMSE |
|---|---|---|
| 0.728 | 0.709 | 4.3 |

# Regression Coefficients Table

```
# Extract coefficient information
coef_summary <- summary(fit2)$coefficients
round(coef_summary, 3)
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    18.537      2.637    7.03    0.000
Education       0.424      0.087    4.89    0.000
Catholic       -0.080      0.017   -4.75    0.000
Agriculture    -0.068      0.040   -1.71    0.095
```

# Other Econometric Models

```r
# Binary outcomes - Logit
glm(y ~ x1 + x2,
    family = binomial(link = "logit"))

# Binary outcomes - Probit
glm(y ~ x1 + x2,
    family = binomial(link = "probit"))

# Count data - Poisson
glm(y ~ x1 + x2,
    family = poisson())

# Panel data - Fixed effects
library(fixest) # install.packages("fixest")
feols(y ~ x1 + x2 | x3,
    data = panel_data)
```

# Print Regression Results - Console

```
# Basic regression table
library(modelsummary)

models <- list(
  "Model 1" = lm(Examination ~ Education,
                 data = swiss),
  "Model 2" = lm(Examination ~ Education + Catholic + Agriculture,
                  data = swiss)
)

modelsummary(models,
             stars = TRUE,
             gof_omit = "AIC|BIC|Log.Lik.")
```

# Export Results to LaTeX

```
# For LaTeX documents
stargazer(fit1, fit2,
          type = "latex",
          title = "Regression Results",
          label = "tab:reg",
          digits = 2,
          header = FALSE,
          font.size = "small",
          column.sep.width = "1pt",
          single.row = TRUE)
```

# Save Regression Output

```
# Save table directly to file
modelsummary(models, output = "table.docx")
modelsummary(models, output = "table.html")
modelsummary(models, output = "table.tex")
modelsummary(models, output = "table.md")
modelsummary(models, output = "table.txt")
modelsummary(models, output = "table.png")

# Raw table
modelsummary(models, output = "html")
modelsummary(models, output = "latex")
modelsummary(models, output = "markdown")
```

# Final Remarks

# Best Practices for R Programming

1. **Organization**
   - Start scripts with `library()` calls
   - Use meaningful variable names
   - Comment your code with #

2. **Efficiency**
   - Use functions to avoid repetition
   - Vectorize operations when possible
   - Use the tidyverse for data manipulation

3. **Reproducibility**
   - Set seeds for random operations
   - Use relative paths (`./data/`)
   - Document package versions

# Resources for Learning R

**Online Resources:**

- **Stack Overflow**
- **R for Data Science** book
- **RStudio cheatsheets**
- **CRAN documentation**

**AI Assistance:**

- **ChatGPT**
- **GitHub Copilot**
- **Google Bard**

# Upcoming Courses

## Next Week

### R Programming Course

Instructor: Erik Senn
Dates: September 8-11, 2025
Time: 9:00-12:00 and 13:00-17:00
Location: Rosenbergstrasse 30, Room 61-152

# Thank You!

Thanks for joining our R workshop!

Happy Coding!

# Survey

Please take a moment to provide feedback on today's workshop.

**Your input helps us improve future courses!**