Basics of R

Great Leap - Summer Training School

Emre SARI

emre@norceresearch.no
Senior Researcher, *Heatlh Economist*
NORCE Norwegian Research Centre, Oslo

August 24, 2025

Funded by the European Union

1. Part 1: Getting Started with R & RStudio
2. Part 2: Importing and Inspecting HMD Data
3. Part 3: Core Data Structures & Basic Manipulation
4. Part 4: Focused Exercise: Who Dies More and When?
5. Part 5: AI for R: Prompting Your First Steps
6. Part 6: Closing

Special Issue Announcement

- We are preparing a **Special Issue** on topics such as:
  - Health inequalities
  - Historical causes of death
  - Public health perspectives
  - ...
- A platform for you to publish:
  - Editorials, original research, and reviews
- Details will be announced!

Lets Get to Know Each Other

- Please share briefly:
  - Your name and current role (student, PhD, postdoc, etc.)
  - Your main research interest
  - Your experience with R (first time, beginner, intermediate)

R and RStudio

What is **R**?

- Open-source language for statistical computing
- Performs data analysis, modeling, and graphics
- Install packages to extend functionality

What is **RStudio**?

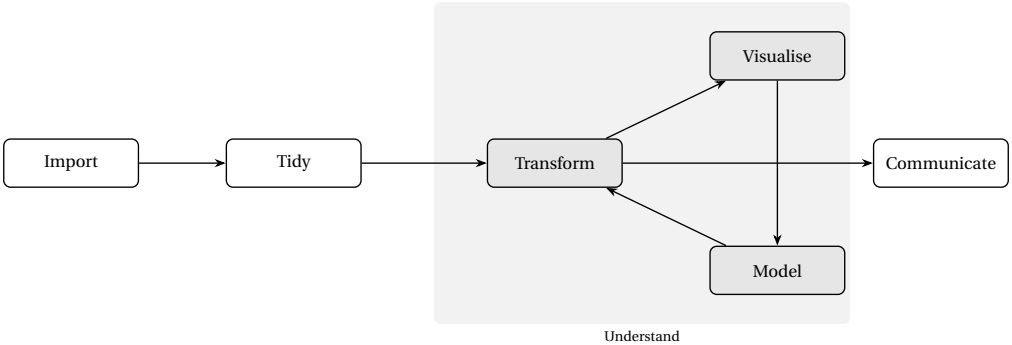
- Integrated Development Environment (IDE) for R
- Provides a user-friendly interface and tools
- Requires R to be installed

R is the cars engine; RStudio is the steering wheel that make it easier to drive.

Why Use **R**?

- One environment for the entire workflow:
  - Import and clean historical health data
  - Tidy and transform large datasets
  - Model inequalities with advanced statistical tools
  - Visualise demographic trends and mortality patterns
  - Communicate results in reproducible reports
- Open-source and completely free
- Widely used in health, demography, and social sciences
- One of the two most common programming languages in data science (**R and Python**)

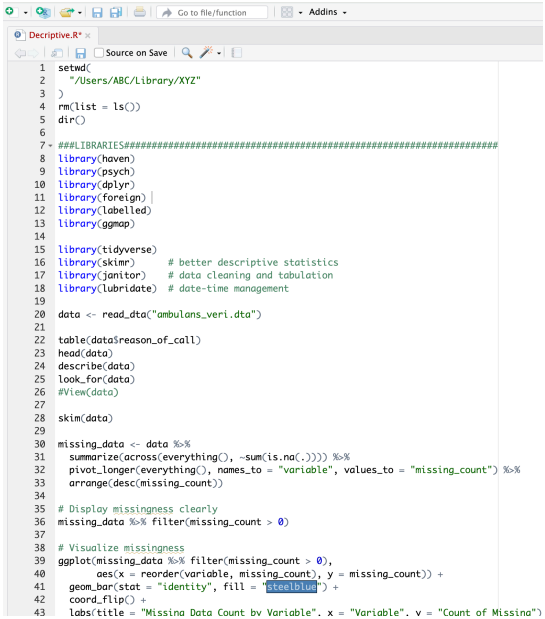
The R Data Science Workflow



Source: Adapted from Rick Mourits, Data management for historians, Utrecht University (2025)

R Compared to Excel and Python

- **Versus Excel:**
  - Better suited for large and complex datasets
  - Enables reproducible research and automation
  - Advanced statistical and visualization capabilities
- **Versus Python:**
  - Stronger tradition in statistics and demography
  - Rich ecosystem for health and mortality research
  - Easier to adopt for non-programmers via tidyverse
- **Unique Strength:** Large community and CRAN packages tailored to historical and demographic data




```r
setwd(
  "/Users/ABC/Library/XYZ"
)
rm(list = ls())
dir()

###LIBRARIES#############################################################################
library(haven)
library(psych)
library(dplyr)
library(foreign)
library(labelled)
library(ggmap)

library(tidyverse)
library(skimr)      # better descriptive statistics
library(janitor)    # data cleaning and tabulation
library(lubridate)  # date-time management

data <- read_dta("ambulans_veri.dta")

table(data$reason_of_call)
head(data)
describe(data)
look_for(data)
#View(data)

skim(data)

missing_data <- data %>%
  summarize(across(everything(), ~sum(is.na(.)))) %>%
  pivot_longer(everything(), names_to = "variable", values_to = "missing_count") %>%
  arrange(desc(missing_count))

# Display missingness clearly
missing_data %>% filter(missing_count > 0)

# Visualize missingness
ggplot(missing_data %>% filter(missing_count > 0),
       aes(x = reorder(variable, missing_count), y = missing_count)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(title = "Missing Data Count by Variable", x = "Variable", y = "Count of Missing")
```

Why Scripts Matter

- Scripts = Your Research Recipe
  - A script is a **series of commands** that documents every step
  - Original data + script = **reproducible study**
  - Allows others (and your future self) to replicate results exactly
- Good Practices for Writing Scripts
  - **Never** modify your original raw data
  - Always keep scripts under **version control** (Git or RStudio projects)
  - Use **clear and consistent naming conventions**
  - Add **comments** explaining each major step
  - Save and date your scripts and outputs systematically
  - Test scripts on a fresh session to ensure reproducibility

Tip: Think of a script as your research diary it documents every decision, like a historians source log.

Recommended Materials

Tutorials and Books

- R Programming Tutorial (YouTube, 2 hours)
- *R for Data Science (2e)* by Hadley Wickham (Free Online Book)

Data for Practice

- Human Mortality Database
  (Free registration required)

Tip: Bookmark these resources theyll be useful throughout the summer school and in your own projects

Essential R Commands and Console Practice

- Key RStudio Areas
  - **Console:** Where commands run immediately
  - **Source Editor:** Write and save scripts
  - **Environment:** View loaded data and variables
  - **Files/Plots/Help:** Access files, view plots, and find help
  - **Packages:** Install and load new R packages

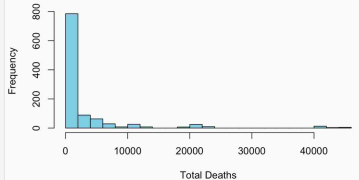
RStudio

Source Editor

```
82
83  # 3A. Base tidyverse approaches
84  glimpse(mortality)      # column names, types, and first values
85  head(mortality, 5)      # first 5 rows
86  summary(mortality)      # quick summary for numeric columns
87  names(mortality)        # list all column names
88  nrow(mortality)         # number of rows
89  ncol(mortality)         # number of columns
90
91  # 3B. Using psych package for more detailed descriptive stats
92  # install.packages("psych") # run this once if not installed
93  library(psych)
94  describe(mortality)
95
96  # 3C. Calculate basic statistics (average, min, max) for numeric variables
97  # Goal: What is the average number of deaths per sex?
98
99  # Step 1: Filter data by sex = 1 (usually male)
100 male_data <- mortality[mortality$sex == 1, ]
101
102 # Step 2: Calculate statistics for total deaths
103 mean(male_data$total, na.rm = TRUE)
104 sd(male_data$total, na.rm = TRUE)
105 min(male_data$total, na.rm = TRUE)
106 max(male_data$total, na.rm = TRUE)
```

```
Mean   : 638.58   Mean   :  608.87   Mean   : 436.74   Mean   : 172.13
3rd Qu.: 445.23   3rd Qu.:  476.68   3rd Qu.: 321.90   3rd Qu.: 124.50
Max.   :8904.00   Max.   :10037.00   Max.   :6944.00   Max.   :3323.00
> names(mortality)        # list all column names
 [1] "country" "year"    "sex"     "list"    "agef"    "cause"   "total"   "d0"      "d1"      "d5"      "d10"
[12] "d15"     "d20"     "d25"     "d30"     "d35"     "d40"     "d45"     "d50"     "d55"     "d60"     "d65"
[23] "d70"     "d75"     "d80"     "d85p"    "d85"     "d90p"    "d90"     "d95p"    "d95"     "d100p"
> nrow(mortality)         # number of rows
[1] 1071
> ncol(mortality)         # number of columns
[1] 32
>
> # 3B. Using psych package for more detailed descriptive stats
> # install.packages("psych") # run this once if not installed
> library(psych)
> describe(mortality)
         vars    n    mean      sd  median trimmed    mad     min   max    range skew kurtosis    se
country*    1 1071    1.00    0.00    1.00    1.00   0.00    1.00     1     0.00  NaN      NaN  0.00
year        2 1071 2006.00    6.06 2006.00 2006.00   7.41 1996.00  2016    20.00 0.00    -1.21  0.19
sex         3 1071    2.00    0.82    2.00    2.00   1.48    1.00     3     2.00 0.00    -1.50  0.02
list*       4 1071    1.00    0.00    1.00    1.00   0.00    1.00     1     0.00  NaN      NaN  0.00
```

Histogram of Total Deaths



RStudio

```
# 3A. Base tidyverse approaches
glimpse(mortality)      # column names, types, and first values
head(mortality, 5)      # first 5 rows
summary(mortality)      # quick summary for numeric columns
names(mortality)        # list all column names
nrow(mortality)         # number of rows
ncol(mortality)         # number of columns

# 3B. Using psych package for more detailed descriptive stats
# install.packages("psych") # run this once if not installed
library(psych)
describe(mortality)

# 3C. Calculate basic statistics (average, min, max) for numeric variables
# Goal: What is the average number of deaths per sex?

# Step 1: Filter data by sex = 1 (usually male)
male_data <- mortality[mortality$sex == 1, ]

# Step 2: Calculate statistics for total deaths
mean(male_data$total, na.rm = TRUE)
sd(male_data$total, na.rm = TRUE)
min(male_data$total, na.rm = TRUE)
max(male_data$total, na.rm = TRUE)
```

```
Median : 189.68   Median :  283.99   Median : 151.47   Median :  47.29   Mode  :character   Mode  :character
Mean   : 638.58   Mean   :  608.87   Mean   : 436.74   Mean   : 172.13
3rd Qu.: 445.23   3rd Qu.:  476.68   3rd Qu.: 321.90   3rd Qu.: 124.50
Max.   :8904.00   Max.   :10037.00   Max.   :6944.00   Max.   :3323.00
> names(mortality)        # list all column names
  [1] "country"  "year"     "sex"      "list"     "agef"     "cause"    "total"    "d0"       "d1"       "d5"       "d10"
 [12] "d15"      "d20"      "d25"      "d30"      "d35"      "d40"      "d45"      "d50"      "d55"      "d60"      "d65"
 [23] "d70"      "d75"      "d80"      "d85"      "d85p"     "d90p"     "d90"      "d95p"     "d95"      "d100p"
> nrow(mortality)         # number of rows
[1] 1071
> ncol(mortality)         # number of columns
[1] 32
>
> # 3B. Using psych package for more detailed descriptive stats
> # install.packages("psych") # run this once if not installed
> library(psych)
> describe(mortality)
          vars    n    mean      sd median trimmed    mad     min    max    range skew kurtosis    se
country*     1 1071    1.00    0.00   1.00    1.00   0.00    1.00      1     0.00  NaN      NaN  0.00
year         2 1071 2006.00    6.06 2006.00 2006.00   7.41 1996.00   2016    20.00 0.00    -1.21  0.19
sex          3 1071    2.00    0.82    2.00    2.00   1.48    1.00      3     2.00 0.00    -1.50  0.02
list*        4 1071    1.00    0.00    1.00    1.00   0.00    1.00      1     0.00  NaN      NaN  0.00
```

Console

Environment | History | Connections | Tutorial

Import Dataset | 262 MiB | List

Global Environment

Data

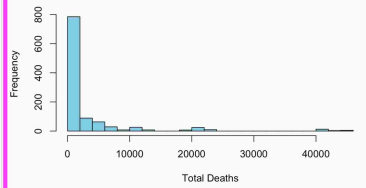
female_1996	17 obs. of 32 variables
female_data	357 obs. of 32 variables
male_1996	17 obs. of 32 variables
male_data	357 obs. of 32 variables
mort_1996	51 obs. of 32 variables
mortality	1071 obs. of 32 variables
yearly_deaths	21 obs. of 2 variables

Values

cause_counts	'table' int [1:17(1d)] 63 63 63 63 63 63 63 63 63 63 ...
deaths_by_year	num [1:21(1d)] 175440 178380 176448 180680 176008 ...
my_table	'table' int [1:3, 1:17] 21 21 21 21 21 21 21 21 21 21 ...
year_counts	'table' int [1:21(1d)] 51 51 51 51 51 51 51 51 51 51 ...

Files | Plots | Packages | Help | Viewer | Presentation

Zoom | Export | Publish



RStudio

R_Script_Lecture_1.R

```
82
83  # 3A. Base tidyverse approaches
84  glimpse(mortality)      # column names, types, and first values
85  head(mortality, 5)      # first 5 rows
86  summary(mortality)      # quick summary for numeric columns
87  names(mortality)        # list all column names
88  nrow(mortality)         # number of rows
89  ncol(mortality)         # number of columns
90
91  # 3B. Using psych package for more detailed descriptive stats
92  # install.packages("psych") # run this once if not installed
93  library(psych)
94  describe(mortality)
95
96  # 3C. Calculate basic statistics (average, min, max) for numeric variables
97  # Goal: What is the average number of deaths per sex?
98
99  # Step 1: Filter data by sex = 1 (usually male)
100 male_data <- mortality[mortality$sex == 1, ]
101
102 # Step 2: Calculate statistics for total deaths
103 mean(male_data$total, na.rm = TRUE)
104 sd(male_data$total, na.rm = TRUE)
105 min(male_data$total, na.rm = TRUE)
106 max(male_data$total, na.rm = TRUE)
```

99:42 3. INSPECTING THE DATA R Script

Console Terminal

R 4.4.3 · ~/Library/CloudStorage/OneDrive-NORCE/Projects/05. COST - GREAT LEAP/Summer School/Course materials/

```
 Median :  189.68   Median :  283.99   Median : 151.47   Median :  47.29   Mode  :character   Mode  :character
 Mean   :  638.58   Mean   :  608.87   Mean   : 436.74   Mean   : 172.13
 3rd Qu.:  445.23   3rd Qu.:  476.68   3rd Qu.: 321.90   3rd Qu.: 124.50
 Max.   :8904.00    Max.   :10037.00   Max.   :6944.00   Max.   :3323.00
> names(mortality)          # list all column names
 [1] "country"  "year"     "sex"      "list"     "agef"     "cause"    "total"    "d0"       "d1"       "d5"       "d10"
[12] "d15"      "d20"      "d25"      "d30"      "d35"      "d40"      "d45"      "d50"      "d55"      "d60"      "d65"
[23] "d70"      "d75"      "d80"      "d85p"     "d85"      "d90p"     "d90"      "d95p"     "d95"      "d100p"
> nrow(mortality)           # number of rows
[1] 1071
> ncol(mortality)           # number of columns
[1] 32
>
> # 3B. Using psych package for more detailed descriptive stats
> # install.packages("psych") # run this once if not installed
> library(psych)
> describe(mortality)
          vars    n    mean      sd median trimmed    mad    min    max    range skew kurtosis    se
country*     1 1071    1.00    0.00   1.00    1.00   0.00   1.00      1     0.00  NaN      NaN  0.00
year         2 1071 2006.00    6.06 2006.00 2006.00   7.41 1996.00 2016    20.00 0.00    -1.21  0.19
sex          3 1071    2.00    0.82   2.00    2.00   1.48   1.00      3     2.00 0.00    -1.50  0.02
list*        4 1071    1.00    0.00   1.00    1.00   0.00   1.00      1     0.00  NaN      NaN  0.00
```

Environment | History | Connections | Tutorial

Import Dataset · 262 MiB · List

Global Environment

Data

female_1996	17 obs. of 32 variables
female_data	357 obs. of 32 variables
male_1996	17 obs. of 32 variables
male_data	357 obs. of 32 variables
mort_1996	51 obs. of 32 variables
mortality	1071 obs. of 32 variables
yearly_deaths	21 obs. of 2 variables

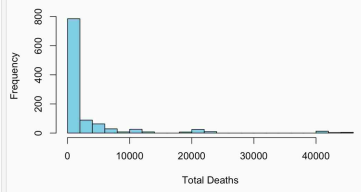
Values

cause_counts	'table' int [1:17(1d)] 63 63 63 63 63 63 63 63 63 63 ...
deaths_by_year	num [1:21(1d)] 175440 178380 176448 180680 176008 ...
my_table	'table' int [1:3, 1:17] 21 21 21 21 21 21 21 21 21 21 ...
year_counts	'table' int [1:21(1d)] 51 51 51 51 51 51 51 51 51 51 ...

Environment

Files | Plots | Packages | Help | Viewer | Presentation

Zoom · Export · Publish



RStudio

```
82
83  # 3A. Base tidyverse approaches
84  glimpse(mortality)      # column names, types, and first values
85  head(mortality, 5)      # first 5 rows
86  summary(mortality)      # quick summary for numeric columns
87  names(mortality)        # list all column names
88  nrow(mortality)         # number of rows
89  ncol(mortality)         # number of columns
90
91  # 3B. Using psych package for more detailed descriptive stats
92  # install.packages("psych") # run this once if not installed
93  library(psych)
94  describe(mortality)
95
96  # 3C. Calculate basic statistics (average, min, max) for numeric variables
97  # Goal: What is the average number of deaths per sex?
98
99  # Step 1: Filter data by sex = 1 (usually male)
100 male_data <- mortality[mortality$sex == 1, ]
101
102 # Step 2: Calculate statistics for total deaths
103 mean(male_data$total, na.rm = TRUE)
104 sd(male_data$total, na.rm = TRUE)
105 min(male_data$total, na.rm = TRUE)
106 max(male_data$total, na.rm = TRUE)
99:42  3. INSPECTING THE DATA                                   R Script
```

```
Console  Terminal
R 4.4.3 · ~/Library/CloudStorage/OneDrive-NORCE/Projects/05. COST - GREAT LEAP/Summer School/Course materials/
Mean   :  638.58   Mean   :  608.87   Mean   :  436.74   Mean   :  172.13
3rd Qu.:  445.23   3rd Qu.:  476.68   3rd Qu.:  321.90   3rd Qu.:  124.50
Max.   : 8904.00   Max.   : 10037.00  Max.   : 6944.00   Max.   : 3323.00
> names(mortality)        # list all column names
 [1] "country" "year"    "sex"     "list"    "agef"    "cause"   "total"   "d0"      "d1"      "d5"      "d10"
[12] "d15"     "d20"     "d25"     "d30"     "d35"     "d40"     "d45"     "d50"     "d55"     "d60"     "d65"
[23] "d70"     "d75"     "d80"     "d85p"    "d85"     "d90p"    "d90"     "d95p"    "d95"     "d100p"
> nrow(mortality)         # number of rows
[1] 1071
> ncol(mortality)         # number of columns
[1] 32
>
> # 3B. Using psych package for more detailed descriptive stats
> # install.packages("psych") # run this once if not installed
> library(psych)
> describe(mortality)
        vars    n    mean      sd  median trimmed    mad     min    max   range skew kurtosis    se
country*    1 1071    1.00    0.00    1.00    1.00   0.00    1.00      1    0.00  NaN      NaN  0.00
year        2 1071 2006.00    6.06 2006.00 2006.00   7.41 1996.00 2016   20.00 0.00    -1.21  0.19
sex         3 1071    2.00    0.82    2.00    2.00   1.48    1.00      3    2.00 0.00    -1.50  0.02
list*       4 1071    1.00    0.00    1.00    1.00   0.00    1.00      1    0.00  NaN      NaN  0.00
```

Environment | History | Connections | Tutorial

Import Dataset · 262 MiB · List

R · Global Environment

Data

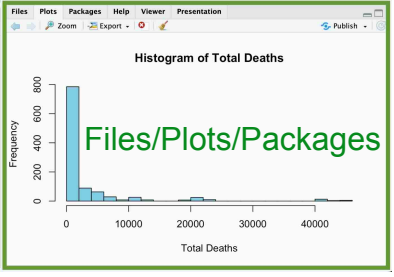
female_1996	17 obs. of 32 variables
female_data	357 obs. of 32 variables
male_1996	17 obs. of 32 variables
male_data	357 obs. of 32 variables
mort_1996	51 obs. of 32 variables
mortality	1071 obs. of 32 variables
yearly_deaths	21 obs. of 2 variables

Values

cause_counts	'table' int [1:17(1d)] 63 63 63 63 63 63 63 63 63 63 ...
deaths_by_year	num [1:21(1d)] 175440 178380 176448 180680 176008 ...
my_table	'table' int [1:3, 1:17] 21 21 21 21 21 21 21 21 21 21 ...
year_counts	'table' int [1:21(1d)] 51 51 51 51 51 51 51 51 51 51 ...

Files | Plots | Packages | Help | Viewer | Presentation

Zoom · Export · Publish



Comments: R ignores text after #

```
1 # This is a comment
2 2 + 2    # R ignores this note
```

Hints

Use # to write notes to yourself.

Basics of R First Steps

Running code in RStudio

```
1 print("Hello, world!")
2 sqrt(144)       # square root
```

Hints

Run code: `Ctrl+Enter` (Win) or `Cmd+Enter` (Mac).

Basics of R First Steps

Objects show up in Environment tab

```
1 x <- 42          # save value into x
2 x                # print value of x
```

Hints

Objects you create appear in RStudios Environment panel.

Lets Get Started in RStudio!

- Open **RStudio** on your computer
- Create a new script: `File → New File → R Script`
- Save it as `my_first_R_script.R`
- Try writing and running your first lines of R code in the **Console**:
  - `print("Hello, world!")` *(your first message)*
  - `2 + 2` *(basic math)*
  - `sqrt(144)` *(square root)*
  - `paste("Today is", Sys.Date())` *(text + todays date)*
  - `c(1, 5, 9)` *(make a small number list)*
  - `mean(c(10, 20, 30))` *(average of numbers)*
- Dont worry if its not perfect — well do it together!

Take 5 minutes now to get comfortable with the interface. Lets break the ice with R!

Basics of R Comparisons & Assignment

Logical comparisons (TRUE/FALSE)

```
2 == 2      # equal to
3 != 2      # not equal to
5 < 10      # less than
5 > 10      # greater than
5 <= 5      # less than or equal to
5 >= 6      # greater than or equal to
```

Basics of R Comparisons & Assignment

Assignment: store a value in an object

```
1 x <- 5        # assign 5 to object x
2 x             # prints 5
```

Assignment: store a value in an object

```
1 x + 10        # reuse x: add 10
2 y <- x * 2    # create a new object using x
3 y
```

- Objects let us **store values** and use them later.
- This is how we build bigger things: datasets = many objects (columns).
- Instead of typing 5 again and again, we reuse x anywhere.

Basics of R Vectors

Create a vector

```
1 ages <- c(0, 1, 5, 10)    # combine values into a vector
2 ages                      # see all values
3
4 mean(ages)                # average age
5 ages * 2                  # multiply all values by 2
```

Why vectors matter?

- A vector is the **basic unit of data** in R
- Every column in a dataset (e.g. year, deaths, sex) is a vector
- Operations apply to all values at once → very powerful

Squences

```
1:5                    # 1, 2, 3, 4, 5
seq(0, 20, by = 5)     # 0, 5, 10, 15, 20
seq(2000, 2020, by = 5)  # years 2000, 2005, ...
```

Why Important?

- Sequences create ranges (ages, years, time steps).
- Useful for loops, plots, and simulated data.
- Example: make a vector of years to plot mortality trends.

Basics of R Repeat Values

Repeat values

```
1 rep("hello", times = 3)      # "hello" "hello" "hello"
2 rep(1:3, times = 2)          # 1 2 3 1 2 3
3 rep(1:3, each = 2)           # 1 1 2 2 3 3
```

Why Important?

- Quickly create repeated patterns of numbers or text.
- Useful for making test data, simulation, or grouping labels.
- Example: repeating years or IDs when building small datasets.

Basics of R Random Sampling

Random sampling

```
sample(1:100, 5)          # pick 5 random numbers
sample(letters, 3)        # pick 3 random letters
sample(1:100, 5, replace = TRUE)  # allow repeats
```

Why Important?

- Useful for practice datasets, bootstrapping, or demos.
- Mimics random draws (e.g., picking survey participants).

Essential R Packages for Beginners

- **tidyverse**
  A collection of tools for data science includes `dplyr`, `readr`, `ggplot2`, and more.
  Use for: data cleaning, importing files, filtering, summarizing, and basic plots.
- **readr**
  Reads CSV and other text files quickly and cleanly. Part of tidyverse.
  Use for: importing data (e.g., `read_csv()`)
- **dplyr**
  A grammar for data manipulation. Part of tidyverse.
  Use for: selecting columns, filtering rows, summarizing, grouping.
- **ggplot2**
  A powerful and flexible system for creating plots and charts. Part of tidyverse.
  Use for: visualising trends and comparisons (youll use this on Day 3).
- **psych**
  Provides useful functions for quick descriptive statistics.
  Use for: understanding numeric variables (e.g., `describe()`).

Introduction to the Human Mortality Database (HMD)

- HMD is a global reference for detailed mortality and population data.
- Developed by:
  - Max Planck Institute for Demographic Research (Germany)
  - University of California, Berkeley (USA)
  - INED (France)
- Goal: Provide harmonised, comparable, and transparent mortality data over time.
- Especially useful in historical health inequality research.

Visit: `https://www.mortality.org`

What to Download from HMD

- Visit: `https://www.mortality.org/Data/ZippedDataFiles`
- Choose a country: We will use **Norway**.
- Download the following three files:
  - `NOR_d_short_idr.csv` *(Reconstructed causes, short list)*
  - `HCD_Method_Explanatory_Notes.pdf` *(How the data were built)*
  - `HCD_Formats.pdf` *(What each column means)*
- **Why these files?**
  - `.csv` = the actual mortality data
  - `Method Notes` = how the data were harmonised
  - `Formats Doc` = what each column and code represents
- **Tip:** Always read metadata before using any dataset!

Whats Inside the CSV File?

- `NOR_d_short_idr.csv` = short-list of causes of death, harmonised across time
- Key variables include:
  - `year` calendar year of death
  - `sex` 1 = male, 2 = female
  - `cause` short-list code for cause of death (1 to 17)
  - `total` number of deaths for that group
- Reconstructed series excludes ill-defined deaths
- Based on ICD codes but harmonised to be consistent over time

Source: HCD Method and Formats documents

Why Use the Reconstructed Short List?

- **Short list** = 17 major cause categories (simpler to start with)
- Reconstructed = adjusted for ICD changes over time
- Easier to analyse trends across decades
- Removes inconsistencies like ill-defined or unclassified causes
- Perfect for first-time analysis and cross-national comparisons

For advanced users, intermediate and long lists are also available.

Appendix 4. Short list

№	Title	Category codes according to ICD10
S001	Certain infectious diseases	A00-B99
S002	Neoplasms	C00-D48
S003	Diseases of the blood and blood-forming organs	D50-D89
S004	Endocrine, nutritional and metabolic diseases	E00-E88
S005	Mental and behavioural disorders	F01-F99
S006	Diseases of the nervous system and the sense organs	G00-G44, G47-H93
S007	Heart diseases	I00-I51
S008	Cerebrovascular diseases	G45, I60-I69
S009	Other and unspecified disorders of the circulatory system	I70-I99, K64
S010	Acute respiratory diseases	J00-J22, U04, U07
S011	Other respiratory diseases	J30-J98
S012	Diseases of the digestive system	K00-K63, K65-K92
S013	Diseases of the skin and subcutaneous tissue, musculoskeletal system and connective tissue	L00-M99
S014	Diseases of the genitourinary system and complcations of pregnancy, childbirth and puerperium	N00-O99
S015	Certains conditions originating in the perinatal period and congenital malformations/anomalies	P00-Q99, R95
S016	External causes	V01-Y89

Step 1: Set Your Working Directory

- R needs to know where to find your data file.
- This location is called your **working directory**.
- Find the folder where you saved `NOR_d_short_idr.csv`
- Then run one of these commands:
  - **Windows:**
    - Right-click on the folder, click `Properties`
    - Copy the folder path (e.g., `"C:/Users/YourName/Documents/R"`)
    - Paste into R:
      `setwd("C:/Users/YourName/Documents/R")`
  - **Mac:**
    - Open Finder, right-click on the folder → `Get Info`
    - Copy the full folder path (e.g., `"/Users/yourname/Documents/R"`)
    - Paste into R:
      `setwd("/Users/yourname/Documents/R")`
- To double-check: `getwd()` shows your current folder.

Step 2: Load the Mortality Data

- Once your working directory is set:
- Make sure the file `NOR_d_short_idr.csv` is in that folder.
- Use this command to read the data:
  - `mortality <- readr::read_csv("NOR_d_short_idr.csv")`
- If successful, youll see a preview of the data.
- To check:
  - `head(mortality)` *(first few rows)*
  - `glimpse(mortality)` *(column types)*
- If something goes wrong:
  - Check spelling of the file name (watch for typos!)
  - Make sure file is in the correct folder
  - Try running: `dir()` to list files in current folder

Inspecting Structure and Metadata

- Now that weve uploaded the dataset, lets explore whats inside.
- These commands help us understand:
  - What variables we have
  - What types of data they are (e.g. numbers or text)
  - If anything is missing or strange
- Try these in the Console:
  - `glimpse(mortality)` *(quick overview of columns)*
  - `head(mortality, 5)` *(first 5 rows)*
  - `summary(mortality)` *(min, max, mean, NAs)*
  - `names(mortality)` *(column names)*
  - `nrow(mortality), ncol(mortality)` *(size of dataset)*
- *Tip: Dont try to interpret yet just look around!*

Exploring Variables Relevant to Health Inequalities

- The goal is to understand *who*, *when*, *how many* and *why* deaths occurred.
- Start with key variables:
  - `year`, `age`, `sex`, `cause`, `total`
- Examples to try:
  - `table(mortality$cause)` *(what causes are recorded?)*
  - `table(mortality$sex)` *(check coding: 1 = male, 2 = female)*
  - `prop.table(table(mortality$sex, mortality$cause), margin = 1)` *(distribution by group)*
- Optional:
  - `library(psych)` + `describe(mortality)` *(detailed summary)*
- Understanding the structure is the first step toward a meaningful analysis.

Understanding Data Structures in R

- **Vector:** A single column of values (e.g. ages, years, names)
- **Data Frame:** Table of multiple vectors (like Excel sheet)
- **Tibble:** Modern, tidyverse-friendly version of a data frame
- Youll work mostly with `tibbles` in this course
- Use `class()` to check structure
  `class(mortality)` → `"tbl_df"` = tibble

How to Select Specific Data

- **Base R:** Use `[ , ]` to extract rows or columns:
  - `mortality[1:5, ]` → first 5 rows
  - `mortality[, "sex"]` → column named `sex`
  - `mortality[mortality$year == 1996, ]` → only year 1996
- **Tidyverse:** Use `filter()` and `select()`:
  - `filter(mortality, sex == 1)` → only males
  - `select(mortality, year, total)` → just year and total
  - `filter(mortality, year == 1996, sex == 2)` → female deaths in 1996
- Combine operations using the pipe operator (`%>%`):
  - `mortality %>% filter(year == 1996) %>% select(total)`
- Tip: Use `names(mortality)` to view all column names

Creating New Variables from HMD Data

- We can create new variables using the `mutate()` function.
- Examples based on this dataset:
  - `mutate(high_mortality = total > 1000)` *(Logical: TRUE/FALSE)*
  - `mutate(age_group_0_14 = d0 + d1 + d5 + d10)` *(Sum of child deaths)*
  - `mutate(age_group_65plus = d65 + d70 + d75 + d80 + d85p)`
- You can also assign category labels using `case_when()`:
  - `mutate(main_age_group = case_when(...))`
    Based on which age columns have values
- Use `glimpse()` to confirm new variables were created

Summarising with Base R and Tidyverse

- You can calculate summary statistics by filtering for a group first:
  - `male_data <- mortality[mortality$sex == 1, ]`
  - `mean(male_data$total, na.rm = TRUE)`
- Same for females:
  - `female_data <- mortality[mortality$sex == 2, ]`
  - `mean(female_data$total, na.rm = TRUE)`
- With tidyverse:
  - `mortality %>% filter(sex == 1) %>% summarise(mean_deaths = mean(total, na.rm = TRUE))`
- Tip: %>% (pipe) reads left to right like a sentence
  - Take the data, then filter, then summarise

Exercise: Who Dies More and When? (20 mins)

- Use your `mortality` dataset to answer the questions below:

Exercise: Who Dies More and When? (20 mins)

- Use your `mortality` dataset to answer the questions below:

Part 1: Subsetting and Summarising (10 mins)

- Filter for:
  - Years after 1990
  - Sex = 1 (male) and 2 (female)
- For each sex, calculate:
  - Average total deaths
  - Minimum and maximum year

Exercise: Who Dies More and When? (20 mins)

- Use your `mortality` dataset to answer the questions below:

Part 1: Subsetting and Summarising (10 mins)

- Filter for:
  - Years after 1990
  - Sex = 1 (male) and 2 (female)
- For each sex, calculate:
  - Average total deaths
  - Minimum and maximum year

Part 2: Visualise (10 mins)

- Create two basic plots:
  - Line plot of total deaths over time for males
  - Line plot of total deaths over time for females
- Add axis labels and a title

Walkthrough: One Possible Solution (10 mins)

- **Filter data after 1990:**
  - `mortality_90 <- mortality[mortality$year > 1990, ]`
- **Subset by sex:**
  - `male <- mortality_90[mortality_90$sex == 1, ]`
  - `female <- mortality_90[mortality_90$sex == 2, ]`
- **Calculate summaries:**
  - `mean(male$total, na.rm = TRUE)`
  - `mean(female$total, na.rm = TRUE)`
  - `min(male$year),max(male$year)`
- **Plot line graphs:**
  - `plot(male$year, male$total, type = "l")`
  - `plot(female$year, female$total, type = "l")`

Using AI Tools to Support R Coding

Why use AI tools for R?

- Learn faster: AI helps explain code, syntax, errors
- Code smarter: Quickly generate working code snippets
- Work independently: Great when Stack Overflow is too much

Top AI tools to try:

- **ChatGPT (OpenAI):** Ask questions, get explanations, write custom R code
- **GitHub Copilot:** Code suggestions as you type in RStudio or VSCode
- **Codeium:** Free AI autocomplete for R and many other languages
- **Phind.com:** AI-powered technical search for coding/debugging
- **Claude.ai (Anthropic):** Longer context and step-by-step explanations

Tip: Think of AI as a *coding assistant*, not a replacement for your logic!

Crafting Effective Prompts for Data Exploration

Good prompts = better results. Try these:

Prompt Templates for Beginners

- `Explain what this R code does line by line.`
- `Write R code to calculate average deaths by year.`
- `Why does this error appear in my R code?`
- `Plot total deaths over time using base R.`
- `Turn this code into tidyverse style.`

Golden Rule: Be specific. Share your data structure, and say what you want.
Pro Tip: Ask AI to act like a `teaching assistant` or a `data tutor`.

Interactive Example: AI-Assisted R Workflow

Scenario: I want to calculate and plot deaths by age group in R.
What a great AI prompt might look like:

Prompt

```
I have an R dataset with death counts in columns like d0, d1, d5,
etc.
Can you help me group them into age bands (014, 1564, 65+) and plot
trends over time in base R?
```

AI might respond with:

- Group columns using `rowSums()`
- Add `plot()` lines for each age group
- Include `legend()` and `colors`

Your job: Ask, run, read, and iterate!
Tip: Dont just copy-paste understand what each line does.

Final Thought: AI is Your Assistant Not Your Brain

AI wont replace you. But someone using AI better than you might.

What to Keep in Mind:

- **AI doesnt know your data.** It may suggest wrong column names or make false assumptions.
- **AI can hallucinate.** Some answers sound confident but are factually wrong.
- **Over-reliance hurts learning.** Use AI to explore, not to avoid understanding.
- **Ethics matter.** Dont use AI to fake knowledge use it to build real skills.

Common Mistakes:

- Blind copy-pasting without reading the code
- Not checking if output matches your dataset
- Asking vague or overly complex questions

Your next R challenge: Ask AI for help *once*, then explain what it does to someone else.

Ask AI to Build a Tiny R Game

Copy this prompt into your AI (ChatGPT, Claude, etc.)

```
You are an R tutor for absolute beginners.
Write a tiny, self-contained R "number guess" game
using only base R (no packages). Requirements:
- Put all logic in a single function called play_guess().
- The game chooses a random integer from 1 to 100 and stores it.
- The player types guesses via readline(); handle non-numeric input.
- After each guess, print "Higher!" or "Lower!" until correct.
- Count the number of guesses and print it at the end.
- Include clear comments explaining each step in simple language.
- Show how to start the game (a single line to run it).
Return only runnable R code, nothing else.
```

Tip: After you get the code from AI, read the comments and explain the flow to your neighbor.

Mini Game in R Number Guess (Base R)

Try this reference solution (paste into Console):

```
play_guess <- function() {
  secret <- sample(1:100, 1)   # random number
  tries  <- 0
  cat("I'm thinking of a number between 1 and 100.\n")
  repeat {
    input <- readline("Your guess: ")
    guess <- suppressWarnings(as.integer(input))
    if (is.na(guess)) { cat("Please type a whole number.\n"); next }
    tries <- tries + 1
    if (guess < secret) { cat("Higher!\n") }
    else if (guess > secret) { cat("Lower!\n") }
    else { cat("Correct! You needed", tries, "guesses.\n"); break }
  }
}

# Start the game:
play_guess()
```

Uses: `sample()`, `readline()`, `if/else`, and a `repeat` loop.

What You Learned Today

- How to navigate R and RStudio
- How to write your first R commands and scripts
- How to import and inspect real-world health data
- How to filter, summarise, and visualise mortality data
- How to use AI tools as your coding companion

You wrote real code using real data thats a huge first step!

Project Work Time

Now it is time to focus on your data and start using R!

Thank you for listening !

Emre SARI

emre@norceresearch.no