

## Plan for the day

Introduction

A direct dialogue

Workflow

Working with data

End of session

# Introduction to R

Silje Synnøve Lyder Hermansen

05 februar 2025

# Plan for the day

# Plan for the day

- ▶ alternative reading
- ▶ lecture: why use R?
- ▶ we divide the class:
  - ▶ proficient R-users meet up to discuss workshops on dynamic reporting and shinyapps (voluntary)
  - ▶ R-newbies stay in class for an intro
- ▶ first encounter: lecture/exercises

# Introduction

# Suggestions for alternative reading

For non-Danish speakers:

- ▶ week 1: workflow, objects, data types, subsetting
  - ▶ [Lusseau \(2024\)](#): chapter 1-3 and/or
  - ▶ [datacamp](#): chapters 1-5
- ▶ week 2: descriptive statistics and graphics
  - ▶ [Lusseau \(2024\)](#): chapter 4-6 and/or
  - ▶ [datacamp](#) and
  - ▶ [datacamp](#)
- ▶ week 3: linear regression and interpretation
  - ▶ [datacamp](#): linear regression in R

## Why R?

# Why I'm addicted to R

## R is an open-source tool for data analysis

- ▶ **reproducible research**
  - ▶ syntax: you can document what you did
  - ▶ “dynamic reporting”: you can knit analysis with text
- ▶ **fills every analytical need** from beginner to advanced
  - ▶ qualitative methods (maps, text analysis, barplots...)
  - ▶ state-of-the-art statistical methods (Bayesian estimation, machine learning...)
  - ▶ data collection (“scraping”)
- ▶ **looks good**
  - ▶ good visualization tool (graphs, maps, etc)

# Why I'm addicted to R (cont.)

## R is an open-source tool for data analysis

- ▶ **free**
  - ▶ I need the permission from none
- ▶ **versatile:**
  - ▶ I can tweak it to my needs (functions, packages ... )

⇒ *first thing I do in the morning, last program to close in the evening*

# Why you should care

**It will help you through your time at KU and make you attractive afterwards**

- ▶ **transferable skill**
  - ▶ the data analytic skills you develop through using R
  - ▶ a programming language → other languages
  - ▶ in itself → a popular software
- ▶ **useful for your methods training and BA/MA thesis**
  - ▶ R is where you practice what I preach

## What is R?

# R is many things

- ▶ a statistical language
- ▶ a programming language
- ▶ a part of a universe

# Statistical language

- ▶ calculator
- ▶ drawing table

⇒ *boosted: pre-packaged solutions in “R packages”*

programming language

# programming language

- ▶ vocabulary and syntax
- ▶ dialects with different syntax
  - ▶ “base R”: math operations; no packages needed
  - ▶ pipes (e.g. `ggplot`, `tidyverse`)

## Part of a universe

# Part of a universe

- ▶ **Collect data** and fit it to your needs (data “wrangling”)
- ▶ **Knit text, calculations and images together**
  - ▶ Word, PowerPoint
  - ▶ LaTeX → pdf
  - ▶ HTML: websites and dynamic web applications

## How it looks

# Difference between R and RStudio

- ▶ R is the actual program
- ▶ RStudio is an interface between R and us
- ▶ This is why you install and update both at the same time

⇒ *You will always be talking to R through RStudio*

# Code along with me

**The best way to learn is to play**

- ▶ Open RStudio and let us start



# The screen

## Your screen consists in four windows

- ▶ Your notebook (top left; you'll have to open it): Where I'm working
- ▶ Your dialogue with R (the "console"; bottom left)
- ▶ Environment (top right): my objects, history etc.
- ▶ The external environment (bottom right): my plots, files, help, etc.

# A direct dialogue

# Let's talk with R

- ▶ Your notebook (top left; you'll have to open it): Where I'm working
- ▶ **Your dialogue with R** (the “console”; bottom left)
- ▶ Environment (top right): my objects, history etc.
- ▶ The external environment (bottom right): my plots, files, help, etc.

# R as a parrot

I can say “hi”

```
"Hei"
```

```
## [1] "Hei"
```

- ▶ The quotation marks say “repeat after me”

# R has selective auditory capacity

## I can talk to myself

```
#I'm just talking to myself
```

- ▶ The # indicates I don't want R to listen

# I can get answers

## R knows math

2+2

```
## [1] 4
```

- ▶ No quotation mark == give me an answer
- ▶ Hit “Enter” to send message to R

# I can store information

## I can store information in objects

- ▶ the <- or = means I'm assigning a value to an object

```
two <- 2
# the same as
two = 2
```

- ▶ the object is listed in the “environment” (upper right)
- ▶ I get no answer unless I ask

```
two
```

```
## [1] 2
```

# I can store information

⇒ *Information is lost unless I store it in objects*

R can use the stored information

**R can do operations on the objects (stored information)**

```
two + two
```

```
## [1] 4
```

# R can use the stored information

## I can ask yes/no questions

- ▶ Is two larger than 1?

```
two > 1
```

```
## [1] TRUE
```

- ▶ Is two equal to 2? (note the double ==)

```
two == 2
```

```
## [1] TRUE
```

- ▶ Is two not 3?

```
two != 3
```

```
## [1] TRUE
```

R can use the stored information

**I can ask yes/no questions**

⇒ *The basis of an algorithm* ⇒ *Useful when “grabbing” observations*

# R can update information

## I can update the information

```
two <- two +1  
two
```

```
## [1] 3
```

- ▶ The information is overwritten; old information is lost

## R can remove objects

### I can remove objects

```
rm(two)
```

- ▶ The disappears from the “environment”

## A few pieces of advice

- ▶ R is nit-picky : capital letters, commas, parentheses. . .
  - ▶ e.g. Two is something else than two
- ▶ R is English speaking
  - ▶ avoid Scandinavian letters

# Your turn 1

## Play around for a few minutes

- ▶ Create an object two and three
- ▶ Sum over the two and store them in object five
- ▶ Update two to a new value
- ▶ Sum over two and three
- ▶ Ask if the sum of two and three is equal to five

⇒ Share your answers on [https://padlet.com/siljesynnove/r\\_coding](https://padlet.com/siljesynnove/r_coding)

# Main takeaways

- ▶ You work in RStudio, not R
- ▶ R is an object-oriented language
  - ▶ information is stored in objects
  - ▶ information is lost unless you store it
- ▶ R is never wrong; you are
  - ▶ you'll have spelling mistakes
  - ▶ none saw that; try again

# Workflow

# Workflow involves several elements

- ▶ A master notepad with all your work
- ▶ A place to put it all

## The notepad

# Let's create a workflow

- ▶ **Your notebook** (top left; you'll have to open it): Where I'm working
- ▶ Your dialogue with R (the “console”; bottom left)
- ▶ Environment (top right): my objects, history etc.
- ▶ The external environment (bottom right): my plots, files, help, etc.

# Open and use it

## Usually, you prepare your dialogue on a notepad

- ▶ Open a notepad: File -> New file -> R script
- ▶ Here, you can write whatever
- ▶ Send lines down to R for a dialogue
  - ▶ put your cursor on the selected line + hit “Run” or ctr+enter

# Why a notepad?

## This is where you do all the work!

- ▶ you re-run the script next time you open R
  - ▶ store questions, not answers (exception is your data)
  - ▶ you should be able to run the script from A to Z without errors
- ▶ it is reproducible
  - ▶ you know what you did
  - ▶ me too
  - ▶ you can share!

# How it looks

## Some good rules of thumb

- ▶ Take notes for yourself using #my notes
- ▶ Make it chronological; R doesn't know what is to come
- ▶ Have a second notepad: your "draft" where you work out a code

```
##My notes for week 1##
```

```
#Store my info first
```

```
four <- 2+2
```

```
#Ask if true second
```

```
four == 4
```

Save your work

# Save all of your work

**You obviously want to save your work**

- ▶ your notepad
- ▶ your data
- ▶ your project (everything related)

# A step back: Filing system

- ▶ your computer is *not* a bucket
- ▶ it is a filing system with drawers (folders)
- ▶ you store your work in a drawer (folder)

⇒ *R relies on a folder*

# Where do I work now?

- ▶ ask where you're working now ("working directory")

```
getwd()
```

```
## [1] "C:/Users/dhf568/Dropbox/Teaching/Universitetet i Kober
```

- ▶ you'll find your stuff here by using "File explorer"/"Path finder"

# Where do I want to work?

**You can decide yourself where you want to work**

- ▶ Tell R directly

```
setwd("C:/Users/ssherman/Dropbox/Teaching/Universitetet i København")
```

- ▶ ... or use the menu
  - ▶ Session -> Set working directory -> choose/create a folder

⇒ *Good places are “Documents” or “Dropbox” (or any other local version of cloud)*

# Save your notepad

**You can save your notepad in the same way**

- ▶ File -> Save as... ; create a folder
- ▶ File extension ".R"
  - ▶ e.g. "first\_encounter.R"
- ▶ Don't use scandinavian letters and space

⇒ *Notepad is red when it is unsaved, black otherwise*

# Project: Save it all

**You can create a “project” folder where everything is stored**

- ▶ Upper right menu: New project -> Existing folder (your created folder)
- ▶ Your desktop is stored there
- ▶ Your working directory is automatically set

⇒ *you can open your notepad again in new project*

# Data

⇒ *Later, you'll save the data the same way using ".rda"*

# Working with data

# Some vocabulary

- ▶ **data structures:** ways to store information in objects
  - ▶ vector
  - ▶ matrix/data frame
  - ▶ list
- ▶ **indexation** a way of “grabbing” pieces of information from objects
- ▶ **functions:** the operations you want to do on the data

# R is a language

## You communicate to R as you do with sentences

- ▶ functions are verbs (you *do* stuff)
- ▶ objects are object (you do stuff to *something*)
- ▶ syntax (the order in which you do it)

# Vector

# What is a vector?

## Vectors are a “ribbon”/“line” of information

- ▶ I can concatenate (glue) pieces of information together `c()`

```
c(1,2,3,4)
```

```
## [1] 1 2 3 4
```

- ▶ note the
  - ▶ `c`
  - ▶ parenthesis
  - ▶ comma between values

# Different vectors

## Vectors can store different information

- ▶ Letters (quotation marks)

```
party <- c("DF", "SD", "V")
```

- ▶ Numbers
  - ▶ note that . is decimal separator
  - ▶ no quotation marks

```
econ <- c(4.5, 3.9, 7.3)
```

## Indexation of vectors

# What is indexation?

I can grab values in the vector by using square brackets

- ▶ see only the second observation

```
party[2]
```

```
## [1] "SD"
```

- ▶ see first and second observation

```
party[c(1,2)]
```

```
## [1] "DF" "SD"
```

## About the example

**The two vectors come from Chapel Hill Expert Survey on parties**

- ▶ National experts rate parties political preferences
- ▶ econ is the economic left (0) to right (10) value

⇒ *did you notice that the two vectors were equally long?*

## Advanced indexation

We can index one vector based on values of the other

- ▶ Which observation is Socialdemokraterne?

```
party == "SD"
```

```
## [1] FALSE TRUE FALSE
```

- ▶ Stash the question as an index to get the preference of Socialdemokraterne

```
econ[party == "SD"]
```

```
## [1] 3.9
```

## Your turn 2

### Can you do the same?

- ▶ create the vectors (if you haven't)
- ▶ find the preference of Dansk folkeparti using indexation

```
party <- c("DF", "SD", "V")
econ <- c(4.5, 3.9, 7.3)
```

⇒ Share your answers on [https://padlet.com/siljesynnove/r\\_coding](https://padlet.com/siljesynnove/r_coding)

# Functions

# What are functions?

## Functions are ready-made operations for objects

- ▶ some are stored
  - ▶ in base R
  - ▶ in “packages”
- ▶ at the core of R language
  - ▶ none knows all the functions
  - ▶ you google (<https://stackoverflow.com> is great)
  - ▶ you ask ChatGPT

⇒ *You remember the ones you need/use the most*

## An example: mean()

- ▶ I can take the mean of my numbers

```
mean(c(1,2))
```

```
## [1] 1.5
```

- ▶ I take the mean of my econ vector

```
mean(econ)
```

```
## [1] 5.2
```

# Functions

# Functions

**Functions requires the data to be stored at the right measurement level**

- ▶ You can't take the mean of non-numbers

```
class(party)
```

```
## [1] "character"
```

- ▶ You can try

```
mean(party)
```

# Functions have arguments

## All functions require arguments

- ▶ they are documented in the “help” pages (bottom right)

```
?mean()
```

- ▶ `x` = is the vector you want to take the mean of

```
mean(x = econ)
```

```
## [1] 5.2
```

# Specifying the argument

- ▶ **some arguments are compulsory** (e.g. what object are you applying this on?)
  - ▶ sometimes you have to specify which argument you're using

```
mean(x = econ)
```

- ▶ sometimes not

```
mean(econ)
```

- ▶ **other arguments are optional**
  - ▶ here, I trim the mean (remove the 50% outliers)

```
mean(x = econ, trim = 0.5)
```

```
## [1] 4.5
```

## Mix functions, indexes and vectors

# The power of the R language

**You can piece together amazing things with simple vocabulary**

- ▶ use two vectors
- ▶ a function
- ▶ indexation

⇒ *An example*

# Sorting out your data

## It is useful to sort your data

- ▶ you can sort a vector according to value

```
sort(econ)
```

```
## [1] 3.9 4.5 7.3
```

- ▶ if you don't store the sorting in a new object, you lose it

```
econ
```

```
## [1] 4.5 3.9 7.3
```

⇒ *sort a vector based on its own values*

# Order your data

You can sort one vector on the basis of the values of another

- ▶ you can order a vector

```
order(econ)
```

```
## [1] 2 1 3
```

- ▶ it returns the rank of each observation
- ▶ you can use this to order the other vector

*#compare*

```
party[order(econ)]
```

```
## [1] "SD" "DF" "V"
```

*#with*

```
party
```

```
## [1] "DF" "SD" "V"
```

# Matrix

# Data objects

## The basic data structure in R are matrices

- ▶ they're stored as objects
- ▶ they are vectors glued together as columns
- ▶ “data frames” are a special case of a “matrix”

⇒ *That's what we run our analysis on*

# Create a matrix

## We can create a matrix with our data

- ▶ I bind vectors together as columns
- ▶ ... and store it in df (my favorite object name)

```
df <- cbind(party, econ)
```

- ▶ I can ask what this is

```
class(df)
```

```
## [1] "matrix" "array"
```

# What is a matrix?

**A matrix is a spreadsheet (as in Excel)**

party	econ	imm
DF	4.5	9.7
EL	1.0	1.6
FolkB	1.3	1.5
KF	7.6	7.1
LA	9.1	4.1
RV	6.5	2.6
SD	3.9	5.5
SF	2.3	2.8
V	7.3	7.7

- ▶ each row is an observation (party)
- ▶ each column is a variable (vector)
- ▶ each square is the value of the observation on that variable

## Indexation of a matrix

Matrices can also be indexed: `matrix[n,m]`

- ▶ First observation in first column

```
df[1,1]
```

```
## party
## "DF"
```

- ▶ All observations in first column

```
df[,1]
```

- ▶ All observations in first row

```
df[1,]
```

# From matrix to data frame

## Data frames are a special type of matrices

- ▶ Redefine matrix to data frame

```
df <- as.data.frame(df)
```

⇒ *They're easier to work with when you analyze*

# Indexing a data frame

- ▶ Now you can grab variables using the dollar sign

```
df$party
```

```
## [1] "DF" "SD" "V"
```

- ▶ Ask what variables you have

```
names(df)
```

```
## [1] "party" "econ"
```

# Saving a data frame

**Data frames (matrices) are stored in objects and can be saved on the computer**

- ▶ You can have several data frames (objects) in your environment
- ▶ You can save in R native file format

```
save(party, file = "party.rda")
```

- ▶ file extension is “.rda”
- ▶ if you've set your working directory, you need no more

End of session

# Main takeaways

- ▶ **good workflow:**
  - ▶ know your filing system
  - ▶ two notepad scripts: one draft and one proper
  - ▶ save the script + data
- ▶ **only things stored in objects are kept**
- ▶ **vectors are ribbons of information** → variables
- ▶ **matrices are spreadsheets** → data
- ▶ **functions are operations you do on your objects**

⇒ *Google and ChatGPT are your best friends*

## Next time

### To prepare for next time:

- ▶ assigned readings: gives you base-R
- ▶ my R-notes: introduce dialects

⇒ *My R-notes are already out, but I will update them.*