

Introduktion til R

26. januar 2023

Kristian G. Kjellmann (kgk@adm.aau.dk)

CALDISS

Det Samfundsvidenskabelige og Humanistiske Fakultet, AAU



Dagens program



1. Hvad er R?
2. At arbejde med R sproget i RStudio
3. Basiskoncepter i R (objekter, funktioner og klasser)
4. Udvid hvad R kan med pakker
5. Import af data
6. Simple visualiseringer

Hvem er jeg?



Kristian Kjellmann

Ph.d. studerende ved sociologi

Forskningsupport ved CALDISS

Programmering, data science og
machine learning (særligt med tekst)

Brugt R siden 2016

Hvad er R?



R er et gratis analyseprogram med sit eget kommandosprog/kodesprog.

Programmet egner sig især til kvantitative analyser og visualiseringer af kvantitative data.

R kan arbejde med mange forskellige dataformater. Da programmet er “open source”, findes ufatteligt mange udvidelser til programmet, der tilføjer funktioner.



Hvad er RStudio?



R i sig selv er meget begrænset. RStudio tilføjer en brugerflade ovenpå R, der gør det rarere at arbejde med.

Man arbejder typisk i RStudio, når man bruger R.

Fordele ved RStudio

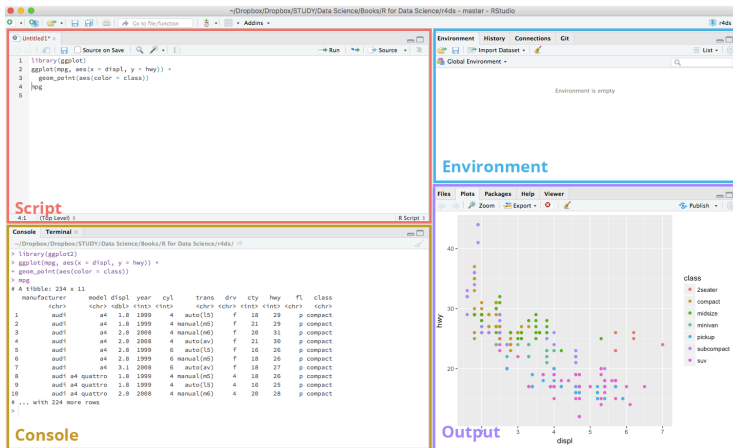
Giver overblik over projektfiler

Hjælper til at tilrette og fuldende kommandoer i script

Overblik over aktivt miljø

Hjælper med at importere data

Hvad er RStudio?

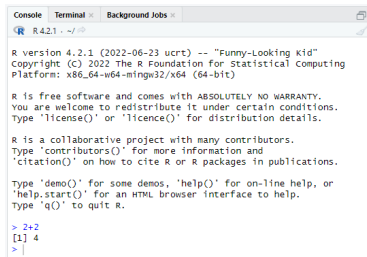


Source: Cecilia Lee

R fungerer ved at man skriver kommandoer i R sproget, som R derefter “fortolker”.

“Fortolkning” i R er blot et spørgsmål om R forstår, hvad du forsøger at gøre. Hvis R forstår det du beder om, gør R den ting. Hvis R ikke forstår det, får man en fejl.

Al fortolkning i R foregår gennem R’s konsol. I konsollen skriver man enten R kode direkte eller også sender man kode dertil fra et script (en tekstfil med R-kode).



```
Console Terminal Background Jobs x
R 4.2.1 - ~/
R version 4.2.1 (2022-06-23 ucrt) -- "Funny-Looking Kid"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 2+2
[1] 4
> |
```

R som en veltrænet hund

Naiv

Forstår kommandoer sagt på en bestemt måde

Gør hvad du beder om

Kan lære nye ting



Operator	Function
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponentiation
%%	Modulo
/%/%	Integer Division

R-sproget: Objekter og funktioner



R fungerer ved at lagre værdier og information i “objekter”. Disse objekter kan derefter bruges i forskellige funktioner. Funktioner kan være alt fra at udregne et gennemsnit, lave en figur, gemme et datasæt osv.

Forsimplet sagt: Et objekt er en eller anden lagret information, mens en funktion er noget, som kan bearbejde eller gøre noget ved informationen i et objekt.

At arbejde med R involverer kontinuerligt at definere objekter. Objekter er blot et navn til at kalde lagret information frem igen.

Objekter kan være mange ting:

- et ord
- et tal
- en talrække
- et datasæt
- en matematisk formel
- et resultat
- en filsti
- en graf
- og så videre...

Når et objekt er defineret, er det tilgængeligt i det aktuelle “miljø”. I R skal et miljø forstås som en samling af objekter og funktioner, som er til rådighed (defineret, “native” eller indlæst).



bolden er tallet 42

`bolden <- 42`



```
print(bold)
```

```
Error in print(bold): object 'bold' not found
```



```
print(bolden)
```

```
[1] 42
```

Vectors (enkeltvariable)



En basal datastruktur i R er en vector.

En vector er en række af værdier af den samme type (fx en række tal, en række ord osv.).

Vectors dukker op i mange forskellige sammenhænge i R, da de bruges hver gang, at man skal angive en samling af flere værdier.

Vectors laves med `c()` - fx `c(1, 2, 3)` (hver værdi adskilt med `,`).

Script-filer er tekst filer med kode, som R kan forstå (“fortolke”).

En script-fil kan forstås som en “analyseopskrift”, der indeholder alle kommandoer nødvendige for at foretage en analyse. Det tillader også, at man nemt kan køre kommandoer igen.

Man bør altid skrive de kommandoer, som er nødvendige for at lave analysen, ind i et script.

Brug konsolvinduet til at finde frem til den rigtige kommando.

kan bruges til kommentarer (ignoreres når man kører koden)

BEMÆRK! Der er ingen fortryd-knap i R! Når koden er kørt, er ændringen sket. Den eneste måde at “fortryde” er ved at genskabe det, som man har lavet, ved at køre tidligere kode igen. Netop derfor er scripts vigtige.

Når man skal importere og eksportere filer i R, har R brug for at vide, hvor filer skal hentes fra og lægges.

Dette holder R styr på med *filstier*. En *filsti* kan siges at være en adresse for, hvor en fil ligger på din computer.

R arbejder med *arbejdsstier*. En *arbejdssti* er det sted, som R tager udgangspunkt i, når R skal finde eller gemme filer.

Man kan tjekke sin nuværende arbejdssti med funktionen `getwd()`.

Arbejdssti kan ændres med `setwd()` (eller i RStudio under Session -> Set Working Directory).

Da man ofte bruger R i flere sammenhænge (fx en introduktion til R i dag, en analyseopgave i morgen), skifter man typisk arbejdssti afhængigt af, hvad man arbejder med.

R Projekter i RStudio letter arbejdet med at skifte mellem arbejdsstier og finde rundt i filer.

Et R Projekt er ikke andet end en mappe på din computer. I denne kan det være en god ide at samle datafiler, R kodefiler og andet relevant materiale, som er relevant for det pågældende projekt.

R ligger en `.Rproj`-fil i den mappe, som man laver til R Projekt. Denne bruges til nemt at skifte arbejdssti til projektet.

ØVELSE: Lav et R Projekt



1. Lav et R Projekt

2. Lav et script, der danner følgende objekter:

`navn1: "araya"`

`navn2: "townsend"`

`aar1: 1961 (uden anførselstegn)`

`aar2: "1972" (med anførselstegn)`

Et objekt, der indeholder alder for `aar1` (2023 - `aar1`)

En vector, der indeholder årene mellem 1960 og 2023
(`c(1960:2023)`)

En vector, der indeholder aldrene for folk født mellem 1960-2023
(`2023 - c(1960:2023)`)

Funktioner er kommandoer brugt til at transformere objekter på en eller anden måde og give et output.

Det, som man sætter i funktionen, kaldes et “argument” eller “input”. Antallet af argumenter varierer mellem funktioner.

Funktioner har alle den samme opbygning:

```
funktionsnavn(arg1, arg2, arg3)
```

funktionsnavn med argumenterne i parentes adskilt med kommaer

Nogle argumenter er krævede, mens andre er valgfrie. Man kan altid konsultere hjælpefil for at se, hvilke argumenter en funktion har (?funktionsnavn).

Funktion:

Mikrobølgeovn

Krævet argument:

Det der skal tilberedes

Valgfrie argumenter:

Mikrobølgeovnens indstillinger



```
microwave(squash, watts = 750, time = 2)
```

Funktioner ændrer aldrig et objekt. Når man bruger en funktion, beder man R om at se et output, men ikke om at ændre noget (hvordan ser min squash ud, når den har været i mikroovnen?).

Hvis man vil ændre et objekt, skal man derfor lagre outputtet i et nyt eller eksisterende objekt (jeg vil gerne erstatte min squash med den, som har været i mikroovnen).

Brug af betingelser i R - logiske værdier



En lang række kommandoer og funktioner i R vil returnere *logiske værdier*.

Logiske værdier kan kun være TRUE eller FALSE (eller missing/NA).

Bruger man fx disse operatorer i R, returneres altid en logisk værdi:

Operator	Betydning
>	Større end
>=	Større end eller lig med
<	Mindre end
<=	Mindre end eller lig med
==	Lig med
!=	Ikke lig med
%in%	Er værdi del af værdisamling?

1. Brug funktionen `mean()` til at udregne gennemsnit af din aldersvector
2. Brug funktionen `gsub()` til at skifte det lille "t" i `navn2` ud med stort "T" (se hjælpefilen).
3. Spørg R om hhv. `aar1` og `aar2` indgår i din vector med årene 1960-2023 (`x %in% y`)

Classes (datatyper)



R adskiller mellem objekter via deres “class”.

En “class” er R’s måde at holde styr på, hvilken type af information, som objektet indeholder.

En class sætter samtidig betingelserne for, hvad der kan lade sig gøre med objektet (fx at vi kan lave udregninger med tal, men at vi ikke kan det med tekst).

Classes (datatype)



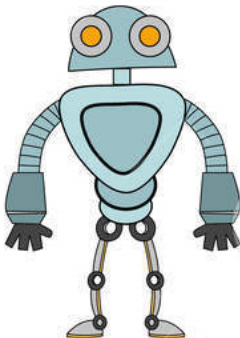
Fordi R er “open source”, bliver der konstant tilføjet nye funktioner til R. Funktioner, som andre har lavet, kan læses ind via “R pakker”, som kan gøres til del af ens “R bibliotek”.

R adskiller mellem installation og indlæsning. Dette for at undgå konflikter mellem pakker.

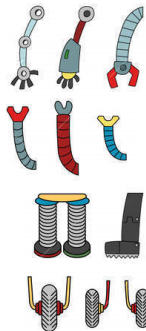
Man behøver kun at installere pakker én gang.

Pakker, som bruges i et script, indlæses i starten af scriptet.

Basis R



R pakker



Tidyverse er en samling af pakker til R, der letter arbejdet med at indlæse, håndtere og arbejde med data.

Pakkerne fra Tidyverse har den fordel, at de alle følger den samme designfilosofi og opbygning i deres funktioner.

Vi vil i disse R introduktioner primært bruge funktioner fra tidyverse til data- og variabelhåndtering.

Alle pakker fra tidyverse kan installeres og indlæses på én gang:

```
install.packages('tidyverse')
```

```
library(tidyverse)
```

Læs mere om tidyverse og se guides på:

<https://www.tidyverse.org/>.

R kan arbejde med data på mange forskellige måder.

For at R kan arbejde med data, skal data indlæses i en af R's *datastrukturer*.

Datasæt kan eksistere i mange forskellige former. Måden vi indlæser data i R, skal derfor hænge sammen med den form og det format, som data har “uden for R”.

En “data frame” er en datastruktur i R, som bruges til at håndtere data i tabelformat (data i rækker og kolonner).

For R er en data frame bare endnu et objekt af en bestemt type (class).

Fordi data frames blot er objekter, kan man arbejde med så mange data frames ad gangen, som man har lyst til.

Af samme grund er man nødt til at fortælle R, når man vil gøre noget med information, som befinder sig i en bestemt data frame.

Indlæsning af data - tabeldata



idno	yrbrn	gndr	eduyrs	wkhtot	health
5816	1974	Male	35	37	Good
7251	1975	Female	13	34	Fair
7887	1958	Male	25	39	Fair
9607	1964	Female	13	34	Good
11688	1952	Female	2	37	Very bad
12355	1963	Male	14	37	Fair

Indlæsning af data - csv-filer



```
idno,yrbrn,gndr,eduyrs,wkhtot,health  
5816,1974,Male,35,37,Good  
7251,1975,Female,13,34,Fair  
7887,1958,Male,25,39,Fair  
9607,1964,Female,13,34,Good  
11688,1952,Female,2,37,Very bad  
12355,1963,Male,14,37,Fair
```

1. Lav en undermappe til data i dit R Projekt
2. Hent datasæt ned i din datamappe

Data findes på: bit.ly/sofia-r-data-jan23

Højreklik på “Raw” og vælg “Gem som” ->
Læg i din datamappe

3. Indlæs datasæt som dataframe i R med
`read_csv()`



For R er værdier (tal, tekst, osv.) *altid* vectors.

Selv objekter, der blot består af ét tal, behandler R som en vector (en vector bestående af én talværdi).

Derfor virker kommandoer og funktioner i R ens, uanset om de bruges på enkelte tal, enkelte vectors eller enkelte kolonner i en data frame.

Data vil ofte indeholde missing-værdier. Missing-værdier angiver ikke-gyldige værdier; fx et manglende svar, ugyldigt svar, information der ikke kunne skaffes eller lignende.

Missing-værdier bruges til at give en værdi uden at give en værdi (cellerne skal indeholde noget). På den måde er man ikke nødt til at fjerne hele rækker fra datasættet, selvom der mangler oplysninger.

I R angives missing-værdier med `NA` og er hverken høje eller lave.

1. Udregn gennemsnitsværdi (`mean()`) for `netustm` (minutter på internettet om dagen) i ESS18 datasættet

BEMÆRK: Variabel indeholder muligvis missing! Tjek hjælpefil for at se, om der er en måde at tage højde for dette.

Pakkerne i `tidyverse` indeholder primært funktioner til datahåndtering (udvælge rækker, kolonner, ændre variable osv.).

Funktioner i `tidyverse` følger den samme opbygning, hvor data altid er første argument:

```
tidyversefunction(dataframe, arg1, arg2)
```

Nogle centrale funktioner er blandt andet:

`filter()` til at selektare rækker

`select()` til at vælge kolonner

`mutate()` til at lave og ændre variable

R (eller mere specifikt `ggplot2`) er rost for de mange måder, som man kan danne og tilpasse visualiseringer af data.

Pakken `ggplot2` er en del af `tidyverse` og bruges til at lave visualiseringer af data. Funktionerne i `ggplot2` gør det muligt at tilpasse nærmest alt, og understøtter mange typer af visualisering.

Overordnet set fungerer `ggplot` ved, at man først angiver, hvad der skal visualiseres (data og “aesthetics”), hvorefter man angiver, hvordan det skal visualiseres (“geom”).

Som hovedregel følger et ggplot denne skabelon:

```
ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

ggplot: Hovedfunktion. Hvad skal visualisering indeholde?

data: Hvilket datasæt skal visualiseres?

mapping: Hvilke informationer fra data skal "mappes" til visualiseringen? (gives ved aes() - fx aes(x = 'alder', y = 'månedsløn'))

GEOM_FUNCTION: Hvordan skal data visualiseres? (hvilken *geometrisk* repræsentation? - prikker, linjer, bjælker osv.)

1. Dan et subset (ny data frame), der kun indeholder mændene i ESS18 sættet (brug `filter()` og variabelen `gndr`).
2. Dan et punktdiagram (`geom_point()`) over fødselsår (`yrbrn`) og tid brugt på internettet (`netustm`). Brug fødselsår til x-akse og tid brugt på internettet som y-akse.
3. Dan det samme punktdiagram for kvinderne.
4. Dan det samme punktdiagram for hele datasættet, men hvor køn tilføjes som `colour` aesthetic.