

Introduktion til R

Statistik E24 (15 ECTS)

ved Mikkeline Munk Nielsen



Indhold

- Hvad er R?
- Installation
- Rundvisning
- Objekter, pakker og funktioner
- Åbne data
- Variable i R

Hvad er R?

- R er et kodesprog designet til statistik og datavisualisering
- Det er sammenlignligt med anden software som python, stata, SAS, eller SPSS
- I skal bruge R til at arbejde med data og lave analyser

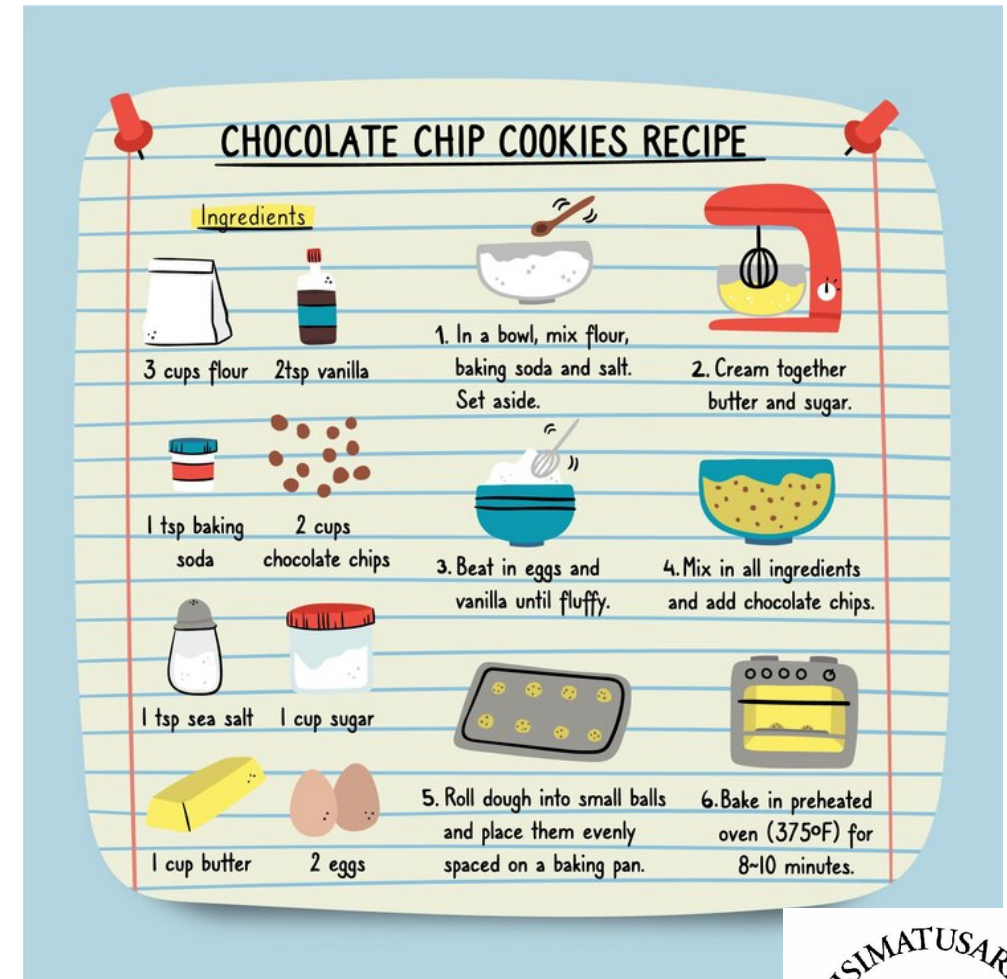


What ´s so special about R?

- **Kodebaseret (vs. klikbaseret)**: Du skriver kode i stedet for at klikke rundt i menuer.
- **Automatisering**: koden automatiserer gentagne opgaver
- **Reproducerbarhed**: koden sikrer, at analyser kan genskabes præcist.
- **Bedre til store datamængder** end f.eks. Excel.
- **Open source** - mulighederne er uendelige!

Hvad er kode?

- Kode er tekstfiler, hvor vi skriver med tekst, hvad programmet skal gøre for os i stedet for at klikke. Tænk på det, som en form for opskrift I skriver til computeren!
- Opskrifter er smarte, fordi vi kan gemme dem og bruge dem igen



Installér R

For at bruge R skal I installere to programmer

- R (software til at køre R programmeringssprog)
- R Studio (Integrated Development Environment “IDE”)

Vi kommer til at åbne og arbejde i R Studio!

... men begge skal installeres



Download

- Gå til: <https://posit.co/download/rstudio-desktop/>
- Klik 'Download and install R'.
- Klik 'Download R for [macOS/Windows]' afhængig af dit styresystem.
- Åbn igen <https://posit.co/download/rstudio-desktop/>.
- Klik 'Download RStudio Desktop for ...' og følg vejledningen.

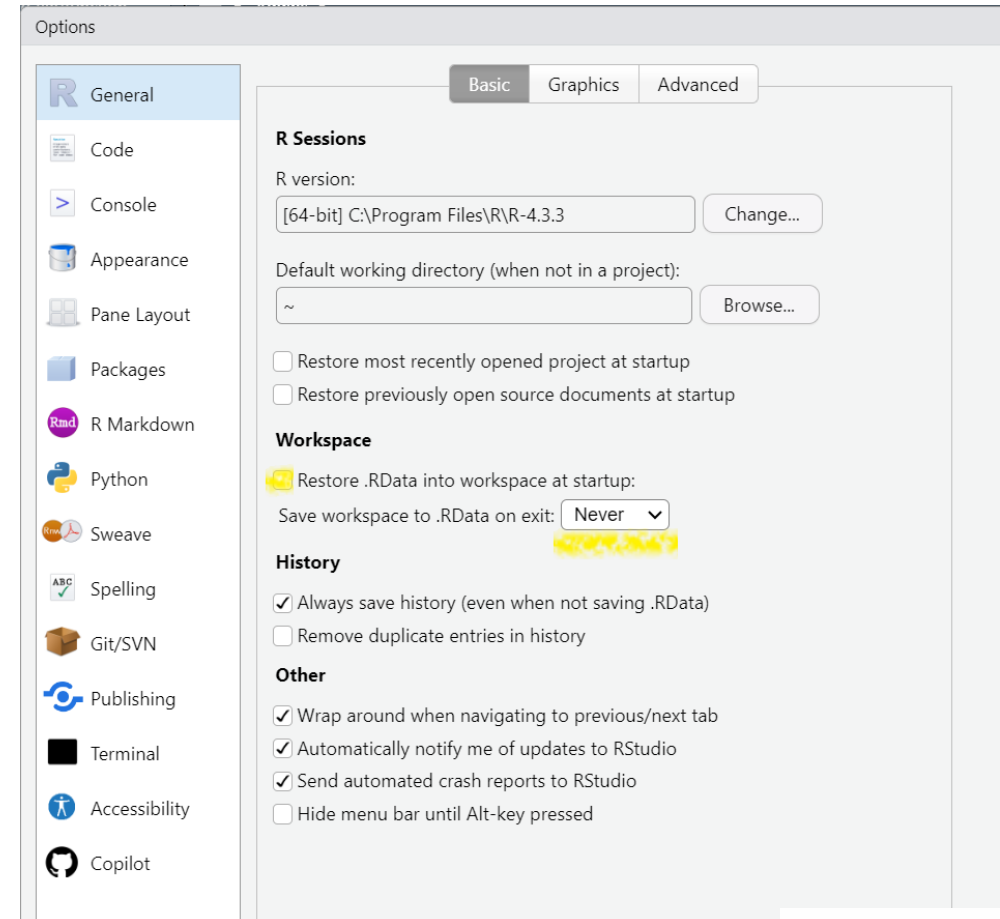
Når du er færdig, skal du have et program på din computer der hedder 'RStudio'.

Indstillinger

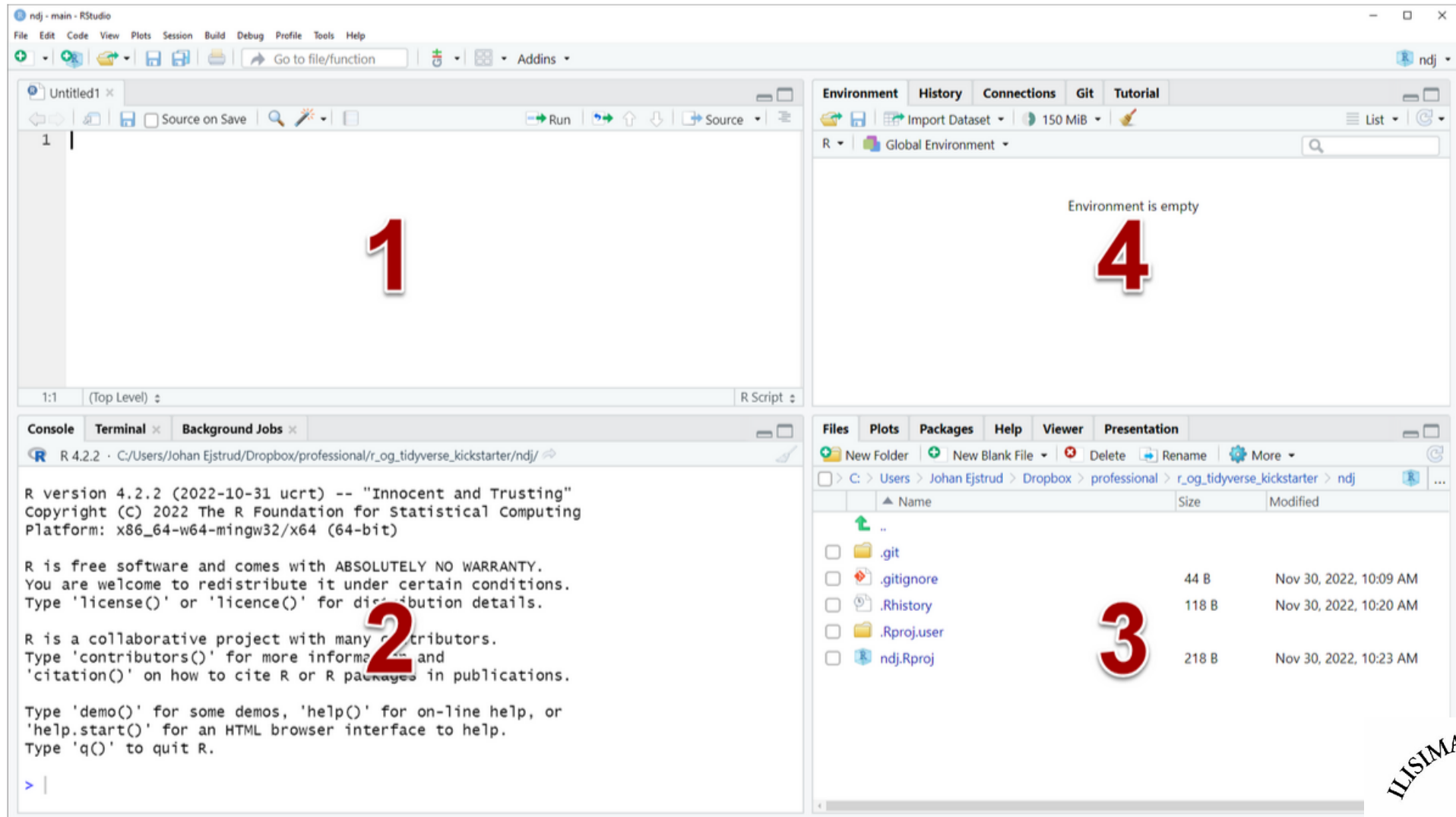
For den bedste oplevelse med R Studio skal der ændres nogle indstillinger.

Gå ind i 'Tools' → 'Global Options...' og skift indstillinger så det ser ud som markeret med gult.

Vælg evt. tema/farver under "appearance".



Rundvisning i R Studio



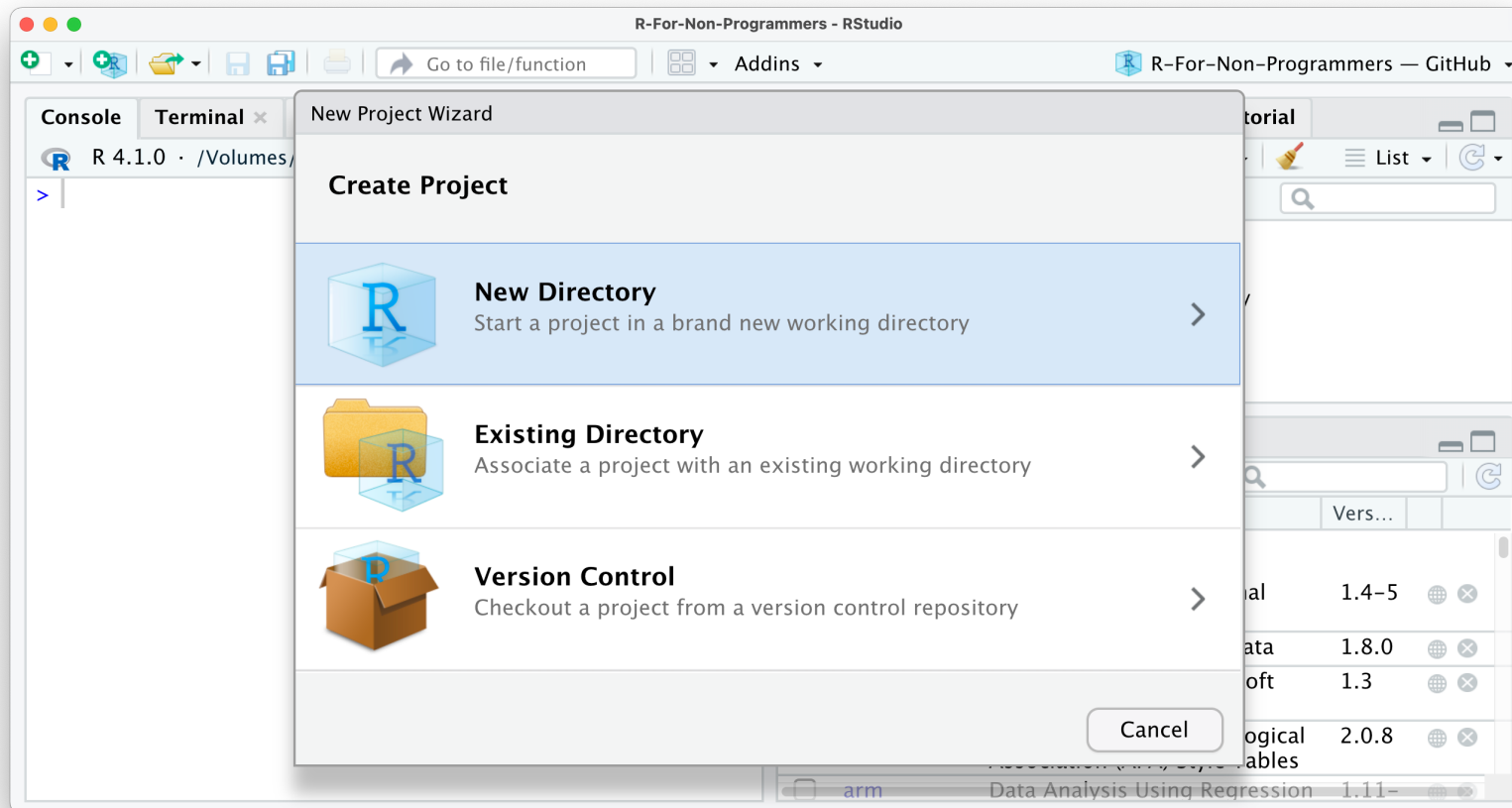
Rundvisning i R Studio

1. **Scripts** - der hvor vi skriver vores kode.
2. **R konsol** - der hvor koden fra vores scripts bliver kørt.
3. **Objekter** - de ting vi har defineret i vores scripts.
4. **Filer, plots, og hjælp** - stifinder, figurer og hjælpemenu.

Beskrivelse	Genvej
RStudio vinduer: Åbn alle 4	Ctrl+Shift+Alt+0
RStudio vinduer: Forstør 1/2/3/4	Ctrl+Shift+1/2/3/4
RStudio vinduer: Skift fokus til 1/2/3/4	Ctrl+1/2/3/4

R-projects

Når man går igang med analyser i R starter man typisk med at lave en mappe på sin computer til alle de relevante filer. I den mappe opretter man det, som kaldes et ***R-project***.



R-projects

- Et R-project er en projektfil, der hjælper dig med at navigere mellem scripts, datasæt, billeder, output etc...
- R-Studio kan som udgangspunkt ikke finde filer på din computer, som ikke ligger i samme mappe, som den fil du arbejder på (f.eks. datasæt, som du skal bruge til analyse).
- Hvis du opretter en mappe med et R project i og opbevarer alle relevante filer i samme mappe, så kan man nemt tilgå alle filerne i mappen direkte fra R Studio (vindue 4) og skifte imellem dem

Tænk på dit R projekt som “THE BIG BROTHER FILE”, der holder øje med alle de andre filer i din projektmappe.

Øvelse

- Opret en mappe på din computer med et R project, hvor du kan gemme alle de filer, som vi skal bruge
- Klik på dit R project for at åbne projektmappen
- Tjek at du er i den rigtige mappe på computeren ved at skrive `getwd()` i konsollen eller et script. Funktionen står for “get work directory” eller “fortæl mig hvilken mappe på computeren, som jeg arbejder i lige nu”.

Objekter

I R er alt, hvad du arbejder med, et objekt. Det kan være tal, tekst, lister eller datasæt. Du opretter objekter, giver dem et navn og kan bruge dem senere.

F.eks. kan vi gemme et objekt, der hedder “mit_foerste_objekt” der indholder teksten “Hello world!”. Vi definerer objektet og dets indhold med en pil

```
1 mit_foerste_objekt <- "Hello world!"
```

Vi kan printe/vise indholdet af objektet ved at køre det i R konsollen eller scriptet:

```
1 mit_foerste_objekt
```

```
[1] "Hello world!"
```

Objekter

Du kan naturligvis også gemme tal som objekter...

```
1 mit_tal <- 2
```

lave beregninger med dine objekter...

```
1 mit_tal + 2
```

```
[1] 4
```


og gemme dine resultater i objekter...

```
1 resultat <- mit_tal + 2  
2 resultat
```

```
[1] 4
```

Objekter

Du kan se dine objekter i vindue 4 under “Environment”



The screenshot shows the RStudio interface with the 'Environment' pane active. The pane has tabs for 'Environment', 'History', 'Connections', and 'Tutorial'. Below the tabs is a toolbar with icons for file operations and a search bar. The 'Global Environment' is selected, and a search bar is visible. The 'Values' section displays a table of objects in the environment.

Values	
mit_foerste_ob...	"Hello world!"
mit_tal	2
resultat	4

Øvelse

Åben et script i R og lav følgende objekter:

- Et objekt, der hedder “navn” som indeholder dit navn
- Et objekt der hedder “alder” og indeholder din alder”

Gem dit script i samme mappe som dit R-project.

Objekter

- En vigtig ting som man opbevarer i objekter er **datasæt**, som i R kaldes ***dataframes***
- Man kan indlæse mange typer af datasæt i R, f.eks.
 - Excel filer (.xlsx)
 - Csv (.csv)
 - R datasæt (.rds)
 - SPSS, STATA, SAS og mange andre filtyper...

For nu er de vigtigste filtyper, som I skal kende, **Excel** og **R** datasæt.

Funktioner

Når vi skal bearbejde objekter bruger vi ***funktioner***. Funktioner er små programmer, der kan udføre forskellige operationer for os.

F.eks. kan funktionen **max** finde den største værdi i en række af tal:

```
1 max(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
```

```
[1] 10
```

...mens andre funktioner kan indlæse datasæt!

Pakker

R har indbygget mange funktioner, mens andre skal downloades. Derfor downloader man ofte ekstra “pakker”, der indeholder smarte funktioner og analysemoduler.

Pakker installeres direkte i konsollen eller scriptet. Her indlæses f.eks. pakken tidyverse, der indeholder en masse gode data-funktioner:

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

man kan også indlæse flere pakker samtidig, f.eks. tidyverse OG janitor, som er god til bl.a. tabeller:

```
1 install.packages(c('tidyverse', 'janitor'))
```

Pakker

Pakker skal dog indlæses hver gang, at vi skal bruge dem. Det gør vi ved at indlæse dem til vores `library`.

```
1 library('tidyverse')
```

Man kan kun indlæse en enkelt pakke af gangen, med mindre man anskaffer sig en package-manager såsom `pacman`. `Pacman` er selv en pakke, der f.eks. indeholder en funktion `p_load` til at indlæse flere andre pakker på en gang:

```
1 pacman::p_load(tidyverse, janitor)
```

Indlæse datasæt

For at indlæse datasæt skal I bruge specifikke funktioner, alt efter hvilket filformat datasættet er.

Til excel-filer:

```
1 library(readxl)
2 df_excelfil <- read_xlsx("sti/til/din//excel_fil.xlsx")
```

Til R-filer:

```
1 df_Rfil <- readRDS("sti/til/din/r_fil.rds")
```

Indlæse datasæt

Lad os f.eks. prøve at indlæse et R-datasæt, som jeg har bygget og gemt i mappen med mit R-projekt. Det indeholder følgende informationer for 100 firmaer:

- industri
- antal ansatte
- omsætning
- et mål for, hvor tilfredse de er med deres salg på en skala fra 1(meget utilfreds) til 5 (meget tilfreds), hvor

Indlæse datasæt

Indlæs datasættet med funktionen `readRDS` og gem datasættet i et objekt, der hedder `df` (for dataframe):

```
1 df_firma <- readRDS("firma_data.rds")
```

Skriv datasættets navn for at printe de første 10 rækker:

```
1 df_firma
```

	navn	industri	ansatte	omsaetning	tilfredshed
1	Firm 1	Finans	305	232402.3	Tilfreds
2	Firm 2	Finans	246	966721.8	Utilfreds
3	Firm 3	Sundhed	140	912069.1	Meget tilfreds
4	Firm 4	Sundhed	490	721634.8	Meget utilfreds
5	Firm 5	Finans	397	815920.7	Meget utilfreds
6	Firm 6	Produktion	186	122152.3	Utilfreds
7	Firm 7	Detail	404	530016.4	Neutral
8	Firm 8	Teknologi	377	782613.6	Tilfreds
9	Firm 9	Sundhed	75	294767.1	Meget tilfreds
10	Firm 10	Finans	56	386362.9	Meget tilfreds

Indlæse datasæt

- Brug `View()` til at åbne datasættet som en tabel
- Brug `names()` til at printe navnene på variablene i datasættet

```
1 names(df_firma)
```

```
[1] "navn"          "industri"      "ansatte"      "omsaetning"   "tilfredshed"
```

Øvelse

- Gå ind på lectio under Dokumenter → Data og download datasættet [firma_data.rds](#)
- Gem datasættet i mappen med dit R-project
- Indlæs datasættet i det skript, som du arbejdede i før
- Undersøg datasættet og dets variable

Variable i R

Variablene i R har en *type*, vist i skrå parenteser <> efter variabelnavnet. Typerne påvirker hvad funktionerne gør med variablen - og det hænger faktisk sammen med variabelens måleniveau!

```
1 df_firma
```

	navn	industri	ansatte	omsaetning	tilfredshed
1	Firm 1	Finans	305	232402.3	Tilfreds
2	Firm 2	Finans	246	966721.8	Utilfreds
3	Firm 3	Sundhed	140	912069.1	Meget tilfreds
4	Firm 4	Sundhed	490	721634.8	Meget utilfreds
5	Firm 5	Finans	397	815920.7	Meget utilfreds
6	Firm 6	Produktion	186	122152.3	Utilfreds
7	Firm 7	Detail	404	530016.4	Neutral
8	Firm 8	Teknologi	377	782613.6	Tilfreds
9	Firm 9	Sundhed	75	294767.1	Meget tilfreds
10	Firm 10	Finans	56	386362.9	Meget tilfreds

Variable i R

Variablene i R har en *type*, vist i skrå parenteser `<>` efter variabelnavnet. Typerne påvirker hvad funktionerne gør med variablen - og det hænger faktisk sammen med variabelens måleniveau!

Kategoriske variable

- `<chr>` *character* - Nominal
- `<fct>` *factor* - Nominal
- `<ord>` *ordered factor* - Ordinal

Numeriske variable

- `<dbl>` *double* - Interval
- `<int>` *integer* - Heltal

Data wrangling

En vigtig fordel i kodebaseret data software er, at vi kan skrive kode, der transformerer vores data til at se ud, præcis som vi vil have det. Det kaldes i folkemunde *data wrangling*!

- Ændre navne på variable
- Omkode variable, dvs. gruppere indholdet i variable på nye måder
- Danne nye variable
- Optimere datastruktur, f.eks. gruppere vores datasæt fra individ niveau til gruppeniveau

Alle disse ting gør det muligt at tilpasse vores data til præcis den type af analyse, som vi gerne vil lave!

Data wrangling

- **tidyverse** er en samling af R-pakker, som er designet til at gøre dataanalyse enklere og mere konsistent.
- Pakker deler et fælles sæt af principper og gør det muligt at skrive enkel og effektiv kode
- De mest kendte pakker inkluderer bl.a. ggplot2, dplyr, tidyr, readr og purrr.



Data wrangling

En funktion som I kommer til at bruge meget er pipe operatoren: `%>%`

- Pipe operatoren `%>%` gør det muligt at “sende” resultatet af én funktion direkte videre til den næste.
- I stedet for at indlejre flere funktioner, kan du skrive dem i rækkefølge, hvilket gør koden mere læsbar.
- Genvej: `Shift + Ctrl+M` (mac: `Cmd + Shift + M`)

Data wrangling

Uden pipe operatoren:

```
1 summarize(group_by(mutate(df, nye_var = var1 + var2), gruppe_var), mean_værdi =  
  mean(nye_var))  
2 %>%
```

Med pipe operatoren:

```
1 df %>%  
2   mutate(nye_var = var1 + var2) %>%  
3   group_by(gruppe_var) %>%  
4   summarize(mean_værdi = mean(nye_var))
```


Variabel navne

Når vi indlæser et datasæt kan det ofte være smart at sørge for, at variabelenes navne er ensartede, så programmet kan læse dem. F.eks. er computere sjældent glade for mellemrum og æ,ø,å...

```
1 names(df)
```

```
[1] "køn"
```

```
"tid"
```

```
"Befolkningen.1..januar"
```

Brug `clean_names()` fra pakken `library(janitor)` til at ensarte variabelenes navne, når du loader et nyt datasæt:

```
1 library(janitor)
2 df <- clean_names(df) # rengør navnene og gem dataframen påny
3 names(df)
```

```
[1] "kon"
```

```
"tid"
```

```
"befolkningen_1_januar"
```

Variabel navne

I andre tilfælde vil vi gerne omdøbe variable manuelt...

```
1 library(tidyverse)
2 names(df)
```

```
[1] "kon"           "tid"           "befolkningen_1_januar"
```

Brug `rename()` fra pakken `library(tidyverse)` til manuelt at omdøbe variable.

```
1 df <- df %>% rename(koen = kon)
2 names(df)
```

```
[1] "koen"          "tid"           "befolkningen_1_januar"
```

Omkodning af variable

Man har ofte behov for at omkode/recode sine variable til færre/andre kategorier eller typer. Ofte vil vi gerne omkode mange variable på én gang. Derfor har pakken *dplyr* (under *tidyverse*) introduceret funktionen `mutate()`.

Logikken er: `mutate(df, ny_variabel = gammel_variabel)`

Omkodning af variable

Lad os prøve at se på nogle eksempler på et befolkningsdatasæt fra European Social Survey...

```
# A tibble: 13,799 × 5
  koen    land  alder udd                net_indkomst
  <fct> <chr> <dbl> <ord>                <dbl>
1 Kvinde DE      26 Ungdoms-/erhvervsuddannelse 20000
2 Kvinde DE      65 Ungdoms-/erhvervsuddannelse   530
3 Kvinde DE      74 Mellemlang videregående    350
4 Mand   DE      64 Ungdoms-/erhvervsuddannelse 2500
5 Kvinde DE      54 <NA>                     4000
6 Kvinde DE      20 Ungdoms-/erhvervsuddannelse   570
7 Kvinde DE      71 Ungdoms-/erhvervsuddannelse 2100
8 Mand   DE      41 Mellemlang videregående   1150
9 Mand   DE      62 Mellemlang videregående    500
10 Mand   DE      65 Ungdoms-/erhvervsuddannelse 1000
# i 13,789 more rows
```

Omkodning af variable

Her kan vi f.eks. tilføje en variabel til vores datasæt, der måler alder kvadreret:

```
1 (df_ESS <- df_ESS %>% mutate(alder2 = alder^2))
```

```
# A tibble: 13,799 × 6
  koen   land  alder udd                net_indkomst alder2
  <fct> <chr> <dbl> <ord>                <dbl>    <dbl>
1 Kvinde DE      26 Ungdoms-/erhvervsuddannelse    20000     676
2 Kvinde DE      65 Ungdoms-/erhvervsuddannelse     530    4225
3 Kvinde DE      74 Mellemlang videregående     350    5476
4 Mand   DE      64 Ungdoms-/erhvervsuddannelse    2500    4096
5 Kvinde DE      54 <NA>                4000    2916
6 Kvinde DE      20 Ungdoms-/erhvervsuddannelse     570     400
7 Kvinde DE      71 Ungdoms-/erhvervsuddannelse    2100    5041
8 Mand   DE      41 Mellemlang videregående    1150    1681
9 Mand   DE      62 Mellemlang videregående     500    3844
10 Mand   DE      65 Ungdoms-/erhvervsuddannelse    1000    4225
# i 13,789 more rows
```

Omkodning af variable

Udover at omkode variable med matematiske operationer, er det nyttigt at kunne omkode sine variable til nye kategorier på baggrund af forskellige kriterier. I kommer til at stifte bekendtskab med følgende funktioner, der skal hjælpe os med at omkode:

- `mutate()` : hovedfunktionen til at omkode variable
- `if_else()` : funktion til at implementere én betingelse
- `case_when()` : funktion til at implementere flere betingelser
- `fct_recode()` : funktion til specifikt at omkode **factor** variable

Omkodning af variable

Man bruger `mutate` til at omkode sine variable i kombination med `if_else`, hvis de kun skal omkodes på baggrund af én betingelse.

Den kan f.eks. bruges hvis man vil lave en dikotom variabel, der måler, om man er over 18 år gammel og dermed myndig:

```
1 (df_ESS <- df_ESS %>%
2   mutate(myndig = if_else(alder > 18,
3     "Myndig", # Kategori vis den binære betingelse er sand.
4     "Ikke myndig")) # Kategori hvis den binære betingelse ikke er
   sand.
```

```
# A tibble: 13,799 × 7
  koen   land  alder udd          net_indkomst alder2 myndig
  <fct> <chr> <dbl> <ord>          <dbl>   <dbl> <chr>
1 Kvinde DE      26 Ungdoms-/erhvervsuddannelse 20000    676 Myndig
2 Kvinde DE      65 Ungdoms-/erhvervsuddannelse   530   4225 Myndig
3 Kvinde DE      74 Mellemlang videregående    350   5476 Myndig
4 Mand   DE      64 Ungdoms-/erhvervsuddannelse  2500   4096 Myndig
5 Kvinde DE      54 <NA>                4000   2916 Myndig
6 Kvinde DE      20 Ungdoms-/erhvervsuddannelse    570    400 Myndig
7 Kvinde DE      71 Ungdoms-/erhvervsuddannelse  2100   5041 Myndig
8 Mand   DE      41 Mellemlang videregående   1150   1681 Myndig
9 Mand   DE      62 Mellemlang videregående    500   3844 Myndig
10 Mand  DE      65 Ungdoms-/erhvervsuddannelse  1000   4225 Myndig
# i 13,789 more rows
```

Omkodning af variable

Hvis man skal opsætte mange betingelser i sin omkodning bruger man funktionen `case_when`. `case_when` bruges også sammen med logiske operatorer, men kan også bruges med funktioner, som i eksemplet nedenfor, hvor funktionen `between()` er anvendt til at lave en variabel med alderskategorier:

```
1 (df_ESS <- df_ESS %>%
2   mutate(alder_kategori = case_when(
3     between(alder, 15, 35) ~ "15-35 år",
4     between(alder, 36, 55) ~ "36-55år ",
5     between(alder, 56, 75) ~ "56-75 år",
6     alder > 75 ~ "75 +"
7   )))
```

```
# A tibble: 13,799 × 8
  koen   land  alder udd          net_indkomst alder2 myndig alder_kategori
  <fct> <chr> <dbl> <ord>          <dbl>   <dbl> <chr>   <chr>
1 Kvinde DE      26 Ungdoms-/erhver... 20000    676 Myndig "15-35 år"
2 Kvinde DE      65 Ungdoms-/erhver...   530   4225 Myndig "56-75 år"
3 Kvinde DE      74 Mellemlang vide...   350   5476 Myndig "56-75 år"
4 Mand   DE      64 Ungdoms-/erhver... 2500   4096 Myndig "56-75 år"
5 Kvinde DE      54 <NA>          4000   2916 Myndig "36-55år "
6 Kvinde DE      20 Ungdoms-/erhver...   570    400 Myndig "15-35 år"
7 Kvinde DE      71 Ungdoms-/erhver... 2100   5041 Myndig "56-75 år"
8 Mand   DE      41 Mellemlang vide... 1150   1681 Myndig "36-55år "
9 Mand   DE      62 Mellemlang vide...   500   3844 Myndig "56-75 år"
10 Mand  DE      65 Ungdoms-/erhver... 1000   4225 Myndig "56-75 år"
# i 13,789 more rows
```


Omkodning af variable

Hvis man gerne vil omkode factor variabel kan man bruge funktionen `fct_recode()`. Denne funktion kan også bruges i kombination med `mutate`.

Logikken er: `fct_recode(variabel, ny_kategori = gammel_kategori)`

F.eks. kan lave en ny variabel for køn, oversat til engelsk med engelske kategorier:

```
1 (df_ESS <- df_ESS %>% mutate(gender = fct_recode(koen, "Male" = "Mand", "Female" =  
  "Kvinde")))
```

```
# A tibble: 13,799 × 9  
  koen    land  alder udd      net_indkomst  alder2 myndig  alder_kategori  gender  
  <fct> <chr> <dbl> <ord>      <dbl>    <dbl> <chr>    <chr>          <fct>  
1 Kvinde DE      26 Ungdoms-... 20000      676 Myndig  "15-35 år"    Female  
2 Kvinde DE      65 Ungdoms-...   530     4225 Myndig  "56-75 år"    Female  
3 Kvinde DE      74 Mellemla...   350     5476 Myndig  "56-75 år"    Female  
4 Mand   DE      64 Ungdoms-... 2500     4096 Myndig  "56-75 år"    Male  
5 Kvinde DE      54 <NA>         4000     2916 Myndig  "36-55år "    Female  
6 Kvinde DE      20 Ungdoms-...   570     400 Myndig  "15-35 år"    Female  
7 Kvinde DE      71 Ungdoms-... 2100     5041 Myndig  "56-75 år"    Female  
8 Mand   DE      41 Mellemla... 1150     1681 Myndig  "36-55år "    Male  
9 Mand   DE      62 Mellemla...   500     3844 Myndig  "56-75 år"    Male  
10 Mand  DE      65 Ungdoms-... 1000     4225 Myndig  "56-75 år"    Male  
# i 13,789 more rows
```

Subsetting

- Nogle gange er vi kun interesserede i at arbejde med særlige dele af vores datasæt. Måske er vi f.eks. kun interesseret i to variable fra vores datasæt, eller kun observationer (her firmaer) med værdien “Finans” på variablen “industri”
- I de tilfælde kan vi lave “subsets” af vores datasæt, hvor vi udvælger de specifikke informationer, som vi gerne vil beholde
- Når vi “subsetter” udvælger vi data fra vores datasæt ud for logiske udsagn. To nyttige funktioner i arbejde med datasæt/matricer er `select()` og `filter()`

Subsetting

`Select()` funktionen bruges til at udvælge kolonner/variable i datasættet. Hvis man f.eks. kun er interesseret i variablene `koen` og `net_indkomst`, kan man pipe sit dataset over i `select()`:

```
1 df_ESS %>% select(koen, net_indkomst) %>% names()
```

```
[1] "koen"          "net_indkomst"
```

Subsetting

`Select()` funktionen bruges til at udvælge kolonner/variable i datasættet. Hvis man f.eks. kun er interesseret i variablene `koen` og `net_indkomst`, kan man pipe sit dataset over i `select()`:

```
1 df_ESS %>% select(koen, net_indkomst) %>% names()
```

```
[1] "koen"          "net_indkomst"
```

På den måde kan man også gemme et nyt datasæt, der kun indeholder de ønskede variable:

```
1 (ny_df <- df_ESS %>% select(koen, net_indkomst))
```

```
# A tibble: 13,799 × 2
#   koen      net_indkomst
#   <fct>      <dbl>
1 Kvinde      20000
2 Kvinde         530
3 Kvinde         350
4 Mand        2500
5 Kvinde      4000
6 Kvinde         570
7 Kvinde      2100
8 Mand        1150
9 Mand         500
10 Mand       1000
# i 13,789 more rows
```

Subsetting

- Mens `select()` uvælger kolonner/variable, kan man bruge `filter()` til at vælge rækker/observationer på baggrund af specifikationer.
- Hvis man f.eks. kun er interesseret i resultater for kvinder i sit datasæt, kan man pipe sit dataset over i `select()` og dermed kun beholde observationer, der har værdien = “kvinde” på variabelen “koen”:

```
1 (kvinde_df <- df_ESS %>% filter(koen=="Kvinde"))
```

```
# A tibble: 7,069 × 9
  koen    land  alder udd      net_indkomst alder2 myndig alder_kategori gender
  <fct> <chr> <dbl> <ord>      <dbl>    <dbl> <chr>    <chr>          <fct>
1 Kvinde DE      26 Ungdoms-... 20000     676 Myndig  "15-35 år"    Female
2 Kvinde DE      65 Ungdoms-...   530    4225 Myndig  "56-75 år"    Female
3 Kvinde DE      74 Mellemla...   350    5476 Myndig  "56-75 år"    Female
4 Kvinde DE      54 <NA>        4000    2916 Myndig  "36-55år "    Female
5 Kvinde DE      20 Ungdoms-...   570     400 Myndig  "15-35 år"    Female
6 Kvinde DE      71 Ungdoms-...  2100    5041 Myndig  "56-75 år"    Female
7 Kvinde DE      67 <NA>         NA    4489 Myndig  "56-75 år"    Female
8 Kvinde DE      47 Ungdoms-...  1500    2209 Myndig  "36-55år "    Female
9 Kvinde DE      60 Lang vid...     0    3600 Myndig  "56-75 år"    Female
10 Kvinde DE      64 Mellemla...  1600    4096 Myndig  "56-75 år"    Female
# i 7,059 more rows
```

Øvelse

I skal nu prøve at lave følgende omkodninger på firma-datasættet:

- Lav en ny variabel, der måler omsætning i 1.000 kr
- Lav en variabel der måler, om firmaerne har erklæret sig “tilfreds” eller “meget tilfreds” med deres omsætning
- Lav et nyt datasæt kun med virksomheder fra sundhedsindustrien
- Lav et nyt datasæt kun med firmaer, der har erklæret sig “tilfreds” eller “meget tilfreds” med deres omsætning

Opsamling

- R kan kun finde filer, som vi har specificeret lokationen på computeren af. Den nemmeste måde at gøre det på er at oprette en R-project mappe og gemme alle relevante filer her.
- R er et objektbaseret kodesprog
- Vi skal installere og indlæse pakker for at bruge funktioner
- R tillader os at wrangle vores data, så vi kan strukturere den efter vores behov

Opsamling

Sidste pointe: tjek regelmæssigt i løbet af kurset, at både R og jeres pakker er opdaterede

- R update: help menu → check for updates
- Pakker: Tools menu → check for package updates