

R Markdown

2023-03-08

- 1 Projekter
- 2 Basics i R-markdown:
- 3 Lav R Markdown som en boss ... ish
- 4 For nørderne

Section 1

Projekter

Hvordan laves et nyt projekt

- Projekter gør dit liv meget nemmere!
- Klik 'File' → 'New project' → 'New directory' → 'New project' → Giv dit projekt et navn (Måske RKursus) → Vælg hvor projektmappen skal ligge på din computer (Kunne være Skrivebordet)
- Se nu øverst i højre hjørne om dit projekt fremgår.
- Alt du gemmer i denne mappe vil nu blive vist under fanen 'files' og gør det nemmere blandt andet at importere data i R.
- Lad os lave et nyt projekt!

Load data i projekter

- **Step 1** Find et hvilket som helst dataset, og gem det i din projektmappe!
- **Step 2** Find datasettet under “Files” i R-Studio
- **Step 3** Højreklik og tryk importer
- **Step 4** Kopier koden så du ikke behøver gøre det næste gang!

Vælg mellem flere projekter

- Se i øverste højre hjørne hvilket projekt du på nuværende tidspunkt er i!
- Ønsker du at skifte, vælg pilen og tryk derefter på det projekt du ønsker at åbne
- Fremgår projektet ikke, vælg “open project” og find projektet hvor du har gemt det!

Section 2

Basics i R-markdown:

Åben et R-markdown dokument

- 1 Klik på file
- 2 Vælg new file
- 3 Vælg R-markdown
- 4 Til venstre skulle feltet “document” allerede være valgt.
- 5 Giv dit dokument en titel samt skriv dit navn under Author
- 6 Default output format kommer vi til så bare tryk “ok”

Resten tager vi i R.

Brug af R-markdown

- Øverst finder i YAML koden, her kan i sætte generelle ting for hele dokumentet.
- Neden under er et markdown dokument: Her kan du blandt andet skrive text, lave overskrifter, sætte figure ind. **MEN** Du kan **ikke** skrive r-kode i dette!
- For at skrive r-kode skal vi lave en code-chunk:
 - 1 Klik der hvor du ønsker en code chunk
 - 2 tryk på +c knappen i øverste højre hjørne
 - 3 vælg R

Du kan nu skrive r kode

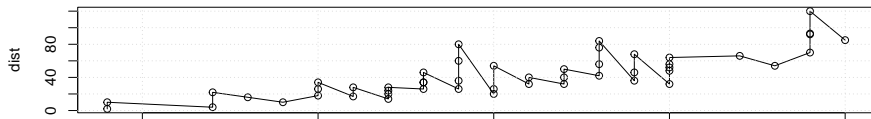
1+1

[1] 2

Brug af R-markdown

- Hvis du har Windows (gud forbyde det) er der en genvej til at lave code-chunks: **Tryk ctrl+alt+i**
- Hvis du har Mac (det eneste rigtige) er genvejen **cmd+option+i**
- Kopier følgende kode ind i din code chunk:

```
library(car)
# Car pakken har et dataset der hedder "cars"
speed=cars$speed
dist=cars$dist
plot(speed, dist); lines(speed, dist); grid()
```



Brug af R-markdown

- *Kursiv* “*”
- **Fed** “**”
- ① Overskrift: “#”
- ② Underoverskrift: “##” op til “####”

Hvis du har lavet overskrifter brug “outline” øverst i højre hjørne til at se dine sektioner

Fra R-markdown til Word, html og pdf

Word og html

- 1 Tryk nu på pilen ud fra “knit” øverst i venstre side
- 2 vælg “html”
- 3 Pøv nu “word”

PDF

```
#install.packages('tinytex')  
#tinytex::install_tinytex()
```

- Hvis det ikke virker, hjælper vi i de to timer efter pausen!!

Opsæt din R-chunk

Her er nogle af de vigtigste:

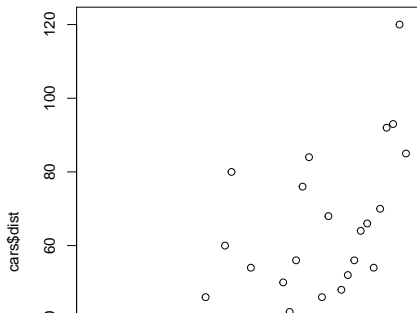
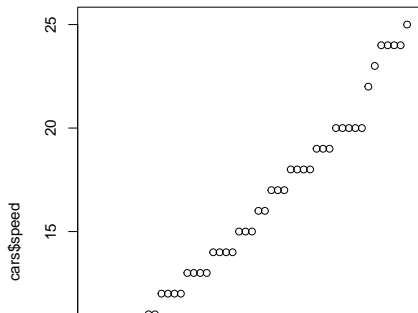
- `echo=FALSE`:
- `message=FALSE`:
- `result="FALSE"hide`:
- `fig.width= ?`
- `fig.height= ?`
- `/tiny` og `/normalsize`

Lad os se på nogle eksempler:

Opsæt din R-chunk (Eksempel 1)

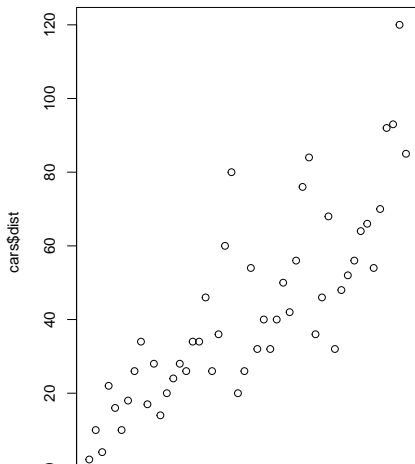
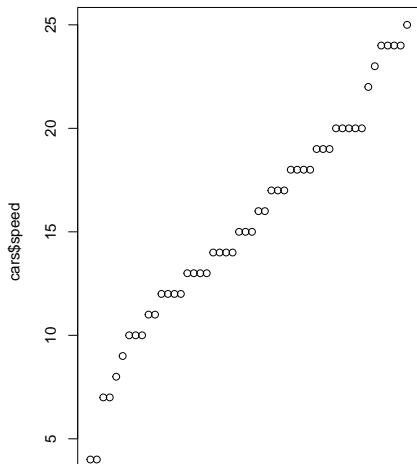
Uden jeg gør noget:

```
par(mfrow=c(1,2))  
plot(cars$speed)  
plot(cars$dist)
```



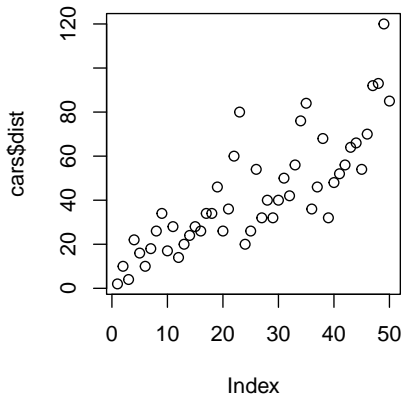
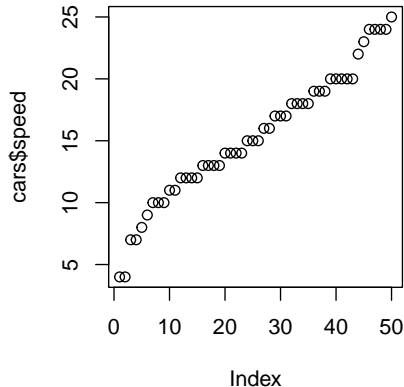
Opsæt din R-chunk (Eksempel 1)

Jeg bruger `echo= FALSE` til at fjerne kode chunken



Opsæt din R-chunk (Eksempel 1)

Jeg bruger nu også `fig.width` og `fig.height` til at ændre størrelsen på figuren



Opsæt din R-chunk (Eksempel 2)

Jeg kan have noget kode der ser helt håbløst ud når det knittes:

```
library(tidyverse)
```

```
## -- Attaching packages -----
## v ggplot2 3.4.0      v purrr  1.0.1
## v tibble  3.1.8      v dplyr  1.1.0
## v tidyr   1.3.0      v stringr 1.5.0
## v readr   2.1.3      v forcats 1.0.0
## -- Conflicts ----- tidy
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x dplyr::recode() masks car::recode()
## x purrr::some()   masks car::some()

mutate(filter(select(cars,dist),dist > 4 & dist< 20) ,sum_dist
```

Opsæt din R-chunk (Eksempel 2)

Jeg kan bruge `message=FALSE` for at fjerne beskeder når jeg bruger `library`:

```
library(tidyverse)
```

```
mutate(filter(select(cars,dist),dist > 4 & dist< 20) ,sum_dist
```

```
##    dist sum_dist
## 1    10        10
## 2    16        26
## 3    10        36
## 4    18        54
## 5    17        71
## 6    14        85
```

Opsæt din R-chunk (Eksempel 2)

Jeg kan bruge tiny og normalize til at gøre koden mindre:

```
library(tidyverse)

mutate(filter(select(cars,dist),dist > 4 & dist< 20) ,sum_dist= cumsum(dist))
```

```
##   dist sum_dist
## 1    10      10
## 2    16      26
## 3    10      36
## 4    18      54
## 5    17      71
## 6    14      85
```

Opsæt din R-chunk (Eksempel 2)

Vi kan også fjerne output hvis vi ønsker med `result='hide'`

```
library(tidyverse)

mutate(filter(select(cars,dist),dist > 4 & dist< 20) ,sum_dist= cumsum(dist))
```

Section 3

Lav R Markdown som en boss ... ish

YAML

Indeholder metadata for dokumentet → bestemmer hvilken type dokument, du laver R Markdown filen til og hvordan layout bliver, m.v.

Default YAML

```
---  
title: "Title"  
author: "Author"  
date: "`r Sys.Date()`"  
output: pdf_document  
---
```

YAML

Output

```
output:  
  pdf_document:  
    number_sections: true  
    toc: true  
    toc_depth: 3
```

YAML

Output

```
output:  
  pdf_document:  
  
    # Nummerér automatisk afsnitsoverskrifter:  
    number_sections: true  
  
    # Dan automatisk indholdsfortegnelse  
    # ud fra afsnitsoverskrifter:  
    toc: true  
  
    # Bestem hvor mange "lag" af overskrifter,  
    # der skal med i toc:  
    toc_depth: 3
```


YAML

header-includes

header-includes fungerer ligesom LaTeX' preamble - bare at vi har det hele i den samme markdownfil.

Eksempel på brugbare pakker:

```
header-includes:
```

- \usepackage{setspace}
- \onehalfspacing
- \usepackage{chngcntr}
- \counterwithin{figure}{section}
- \counterwithin{table}{section}
- \usepackage{subfig}
- \usepackage{float}

YAML

header-includes

header-includes:

```
# Pakke til at definere linjeafstand
- \usepackage{setspace}
# Vi kan bruge pakken til her at sætte til halvanden linjeafstand
- \onehalfspacing

# Pakke der definerer kommandoerne \counterwithin og \counterwithout
- \usepackage{chngcntr}
# Sætter en counter, som nulstilles ved hver ny section*:
- \counterwithin{figure}{section}
- \counterwithin{table}{section}

# Pakke der gør, at vi kan lave referencer til subfigurer indenfor et enkelt floating environment
# Bruges for at give separate titler eller referencer til subfigurer (e.g. fig 1a og fig 1b):
- \usepackage{subfig}

# Pakke der fikser floating environments' shortcomings, fx placering af figurer:
- \usepackage{float}
```

* `\counterwithout` fjerner relationen fra `counterwithin`

Ligninger

Vi kan næsten ikke undgå at bruge ligninger, når vi skriver projekter. Det gøres nemt:

Inline ligning

$$Y = \alpha K^\beta$$

OBS! Hvis \$ skal bruges i teksten skrives det efter en backslash for at omgå Markdowns default brug heraf.

Ligning på selvstændig linje

$$Y = \alpha K^\beta$$

Nummereret ligning

Hvis vi vil have nummererede ligninger må vi bruge LaTeX format og give vores ligninger et *label*.¹ Tilgængæld kan vi så også referere til dem.

```
\begin{equation}\label{y}  
Y=AL^{\beta} K^{\alpha}  
\end{equation}
```

¹OBS! Hvis labels er ens kan der ske fejl med at knitte dokumentet

LaTeX math symbols

\leq	<code>\leq</code>	\propto	<code>\propto</code>	$\$$	<code>\\$</code>	α	<code>\alpha</code>	∞	<code>\infty</code>
\geq	<code>\geq</code>	\cdots	<code>\cdots</code>	$\&$	<code>\&</code>	β	<code>\beta</code>	f'	<code>f'</code>
\neq	<code>\neq</code>	\dots	<code>\dots</code>	$\%$	<code>\%</code>	ϵ	<code>\epsilon</code>	\int	<code>\int</code>
\nleq	<code>\nleq</code>	\because	<code>\because</code>	\backslash	<code>\backslash</code>	ζ	<code>\zeta</code>	\oint	<code>\oint</code>
\ngeq	<code>\ngeq</code>	\therefore	<code>\therefore</code>	\sharp	<code>\sharp</code>	η	<code>\eta</code>	\mathbb{Z}	<code>\mathbb{Z}</code>
\cong	<code>\cong</code>	\forall	<code>\forall</code>	∂	<code>\partial</code>	κ	<code>\kappa</code>	\mathbb{R}	<code>\mathbb{R}</code>
\equiv	<code>\equiv</code>	\exists	<code>\exists</code>	90°	<code>90^\circ</code>	λ	<code>\lambda</code>	\mathbb{Q}	<code>\mathbb{Q}</code>
\sim	<code>\sim</code>	\in	<code>\in</code>	\parallel	<code>\parallel</code>	μ	<code>\mu</code>	$\sqrt[3]{2}$	<code>\sqrt[3]{2}</code>
\approx	<code>\approx</code>	\subset	<code>\subset</code>	\perp	<code>\perp</code>	ξ	<code>\xi</code>	$\frac{2}{3}$	<code>\frac{2}{3}</code>
\doteqdot	<code>\doteqdot</code>	\subseteq	<code>\subseteq</code>	\triangle	<code>\triangle</code>	ρ	<code>\rho</code>	$\lceil x \rceil$	<code>\lceil x \rceil</code>
\times	<code>\times</code>	\emptyset	<code>\emptyset</code>	∇	<code>\nabla</code>	τ	<code>\tau</code>	$\lfloor x \rfloor$	<code>\lfloor x \rfloor</code>
\cdot	<code>\cdot</code>	\cap	<code>\cap</code>	\square	<code>\square</code>	ϕ	<code>\phi</code>	$\{x\}$	<code>\{x\}</code>
\ast	<code>\ast</code>	\cup	<code>\cup</code>	\angle	<code>\angle</code>	ψ	<code>\psi</code>	\widehat{p}	<code>\widehat{p}</code>
\div	<code>\div</code>	\setminus	<code>\setminus</code>	Π	<code>\Pi</code>	π	<code>\pi</code>	\overline{AB}	<code>\overline{AB}</code>
\pm	<code>\pm</code>	\wedge	<code>\wedge</code>	Θ	<code>\Theta</code>	θ	<code>\theta</code>	\overleftrightarrow{AB}	<code>\overleftrightarrow{AB}</code>
\mp	<code>\mp</code>	\vee	<code>\vee</code>	Γ	<code>\Gamma</code>	γ	<code>\gamma</code>	\overline{AB}	<code>\overline{AB}</code>
\bigcirc	<code>\bigcirc</code>	\Rightarrow	<code>\Rightarrow</code>	Δ	<code>\Delta</code>	δ	<code>\delta</code>	\overline{AB}	<code>\overline{AB}</code>
\oplus	<code>\oplus</code>	\rightarrow	<code>\rightarrow</code>	Ω	<code>\Omega</code>	ω	<code>\omega</code>	\overline{AB}	<code>\overline{AB}</code>
\otimes	<code>\otimes</code>	\mapsto	<code>\mapsto</code>	Σ	<code>\Sigma</code>	σ	<code>\sigma</code>	\overline{AB}	<code>\overline{AB}</code>

Figure 1: LaTeX math symbols

Global options

Global options i starten af dokumentet gør, at du ikke skal skrive det i starten af hver code chunk og øger projektets consistency.

```
knitr::opts_chunk$set(echo = FALSE,  
                      message = FALSE,  
                      warning = FALSE,  
                      fig.align = "center",  
                      fig.width = 5,  
                      fig.pos = "H",  
                      fig.path = "figures/",  
                      comment = NA  
                      )
```

Referencer

I YAML

```
# Indsæt bib fil, hvor alle referencer er
bibliography: refs.bib

# Bestem referencestil
csl: apa.csl
```

I projektet

- Referer til kilde fra litteraturlisten:
 - @bibtag: Keynes (1936)
 - [@bibtag]: (Keynes, 1936)
 - [@bibtag, pp. 4]: (Keynes, 1936, s. 4)
- Referer til elementer i dit dokument:
 - Giv elementet et label (e.g. # Introduktion {#sec-intro})
 - \ref{sec-intro}: 1

Section 4

For nørderne

Child dokumenter

- Brugbart ved længere projekter, eller hvis forskellige personer skriver forskellige afsnit
- Fx indledning i et dokument, analyse i et andet osv.

```
# Et child dokument:
```

```
{r, child="child.Rmd"}
```

```
# Flere child dokumenter:
```

```
{r, child=c("firstchild.Rmd", "secondchild.Rmd")}
```

Parametre

Parametre er nyttige, hvis vi vil lave den samme undersøgelse/rapport for fx to forskellige perioder. Med parametre kan vi nemlig let ændre på hele dokumentet uden at bruge tid på at gå hele dokumentet slavisk igennem

I YAML indsættes:

```
params:  
  parameter: værdi
```

Eksempel:

```
params:  
  year: 1970  
  country: "Denmark"
```

Parametre

Parametre i teksten, eksempel

Det var i året ``r params$year`` at Simon overgav sig til Mac

Parametre i code chunks, eksempel

```
gapminder %>%  
  filter(year == params$year) %>%  
  ggplot(aes(x = gdpPerCap, y = lifeExp,  
             size = pop, color = continent)) +  
  geom_point() +  
  labs(title = paste("Life Expectancy and GDP for",  
                    params$year),  
       x = "GPD Per Capita", y = "Life Expectancy")
```