Visualisering af biologiske datasæt

Sarah Rennie

2022-02-21

Contents

Chapter 1

Grundlæggende R

1.1 Inledning til kapitel

Jeg har prøvet at opsummere nogle grundlæggende ting, som I skal vide før vi går videre i kurset. Derefter er der nogle problemstillinger som jeg anbefaler at I arbejder igennem, for at tjekke jeres forståelse af koncepterne og udfylder eventuelle huller i jeres viden.

Det er helt fint, hvis I ikke har set de hele før. Skrive til mig gerne, hvis I har nogle spørgsmål, som vi kan diskutere i vores næste lektion.

1.2 Rstudio

Det allerførste man skulle gør, hvis man ikke har brugt RStudio før, er at downloade den gratis på nettet:

https://www.rstudio.com/products/rstudio/download/#download

Vi kommer fremadrettet til at være meget afhængig af nogle af dets funktionalitet til at lave blandt andet R Markdown dokumenter. R Markdown bliver præsenteret i vores næste lektion.

1.2.1 De forskellige vinduer i RStudio

Hvis man ikke har et kendskab til RStudio, kan man tjekke det her for at lære de forskellige vinduer at kende:

https://bookdown.org/ndphillips/YaRrr/the-four-rstudio-windows.html

- Man skrive kode i Source (øverst til venstre)
- Man kører kode ved at tryk CMD+ENTER (eller WIN-KEY+ENTER)

- Kode køres ind i Console (som plejer at være nederst til venstre, selvom det er øverst til højere i billedet).
- Environment - her kan man se, alle objekter i workspace-en.

1.3 Sætte working directory

Når man arbejde på et projekt, er det nyttigt at vide, den *working directory* som R arbejder fra - det er det sted, hvor R forsøger at åbne eller gemme filer, medmindre man angiver et andet path.

```
getwd() #se nuværende working directory
list.dirs(path = ".", recursive = FALSE) #se mappe indenfor working directory
setwd("~/Documents/") #sætte en ny working directory (C:/Users/myname/Documents hvis m
```

1.4 R pakker

R pakker er simpelthen en samling af funktioner (eller datasæt), som udvider hvad er tilgængelige i base-R (den R man få, uden at indlæse nogle som helst pakke). I R er der mange tusind R pakke (faktisk op mod 100,000), som plejer at være tilgængelige på **CRAN** (https://cran.r-project.org/). Indenfor biologiske fag er der også mange flere på **Bioconductor** (https://www.bioconductor.org/), og nogle gange er R-pakke også installeret direkte fra **Github**.

I dette kursus arbejder vi rigtig meget med en pakke der hedder **tidyverse**. Før man indlæse det, skal man først sikre sig, at pakken er installeret på systemet:

```
install.packages("tidyverse")
```

Alle pakker på **CRAN** er installeret på samme måde. Når man bruger en R pakke, skal man først indlæse den ved at bruge `library()`:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

I dette tilfælde kan man se, at **tidyverse** er faktisk en samling af otte andre pakke, som blev indlæste. Vi kommer til at arbejde med disse pakker fra kapitel tre (vi starter med **ggplot2** og så nogle af de andre pakke fra **tidyverse** fra kapitel fire)

1.5 Hvor kommer data fra?

De data, vi arbejde med i kurset stammer fra forskellige steder.

1.5.1 Indbygget datasæt

I R er der mange indbygget datasæt som er meget brugbare for at vise eller lege med koncepter, som især gøre dem populære for undervisningsmateriale. Indbygget datasæt kan være tilgænglige indenfor mange pakke, men `library(datasets)` er den mest brugt (der er også mange indenfor `library(ggplot2)`. For eksempel, for at indlæse datasættet, der hedder 'iris', kan man bruge `data()`:

```
library(datasets)
data(iris)
```

Så er en *dataramme* tilgængelige som en *objekt* i *workspacen* - se "Environment" fane på højere side i RStudio, eller indtaste `ls()`, så bør du kunne se en objekt med navn 'iris'. Man kan kun arbejde med objekter som er en del af workspacen.

1.5.2 Importering af data fra .txt fil

Det er meget hyppigt, at man har sin data i formen af en .txt fil eller .xlsx fil på sin computer. Den nemmeste måde at få åbnet en .txt fil er ved at bruge `read.table()`, som i nedenstående:

```
data <- read.table("mydata.txt") #indlæse data filen mydata.txt som er i working directory
head(data)
```

Huske at hvis data har kolonner navne, så skal man bruge `header=T` for at undgå, at den første række i data bliver disse navne i stedet for virkelige observationer.

```
data <- read.table("mydata.txt",header=T) #indlæse data filen mydata.txt som er i working directo
head(data)
```

1.5.3 Importering af data fra Excel

Der findes også en hjælpsom pakke, der kan indlæse Excel-ark direkte ind i R:

```
library(readxl)
data <- read_excel("data.xlsx")
data
```

1.6 Dataramme koncepter

http://www.r-tutor.com/r-introduction/data-frame

Mange af de ting, som vi laver i R tager udgangspunkten i datarammer.

```
mydf <- data.frame("person ID"=1:5, "height"=c(140,187,154,132,165), "age"=c(34,31,25,4
mydf
```

```
##   person.ID height age
## 1         1    140  34
## 2         2    187  31
## 3         3    154  25
## 4         4    132  43
## 5         5    165  29
```

Huske, at vores `data.frame`, ligesom et matrix (i R: `matrix()`) har to dimensioner - række og kolonner Forskellen mellem en matrix og en dataramme er, at datarammer kan indeholde mange forskellige data typer (herunder numeriske, faktorer, karakterer osv.), men matrix indeholder kun numeriske data. For eksempel

```
mydf$colour <- c("red","blue","green","orange","purple")
mydf
```

```
##   person.ID height age colour
## 1         1    140  34    red
## 2         2    187  31   blue
## 3         3    154  25  green
## 4         4    132  43 orange
## 5         5    165  29 purple
```

er en dataramme med forskellige data type men følgende er en matrix

```
matrix(c(1, 2, 3, 4, 5, 6),
    nrow=3,
    ncol=2)
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

med kun numeriske data, som kan bruges til matematik operationer (matrix multiplikation osv.). I dette kursus beskæftiger os primært med datarammer.

Man kan kigge på en subset af rækkerne i de data ved at

```
mydf[mydf$height>=165,] #alle rækker i datarammen med height = 165 eller over
```

```
##   person.ID height age colour
## 2         2    187  31   blue
## 5         5    165  29 purple
```

Her i kurset arbejder vi på noget der hedder et 'tibble'. Det får vi at vide mere om senere.

```
mydf_tibble <- tibble(mydf)
mydf_tibble
```

```
## # A tibble: 5 x 4
##   person.ID height   age colour
##       <int>  <dbl> <dbl> <chr>
## 1         1    140    34 red
## 2         2    187    31 blue
## 3         3    154    25 green
## 4         4    132    43 orange
## 5         5    165    29 purple
```

1.7 Descriptive statistics

1.7.1 Simulere data fra distributions

Se for eksempel:

http://www.r-tutor.com/elementary-statistics/probability-distributions/normal-distribution

Man kan nemt lave sin egne 'fake' data ved at simulere det fra nogle distributioner. Det vil typiske være den normale distribution, idet den normale distribution opstå mest hyppigt i den virkelige verden (huske den klassiske klokke-form). I R kan man bruge funktionen `rnorm` til at simulere data - først angiver man, hvor mange observationer man vil have, og så den mean og standard deviation, som er de to parametre som er nødvendige til at beskrive en normal distribution.

```
x <- rnorm(25,mean=0,sd=1) #standard normal distribution
x #så har vi 25 værdier fra en normal distribution med mean=0 og standard deviation=1.
```

```
##  [1]  0.66705324 -1.28047497  0.54987317  0.51085239 -1.14952601  0.07188865
##  [7]  0.61563390  0.31238346 -0.29262857 -0.27051935  0.22063605  1.48022276
## [13]  0.14809493 -0.05053194  0.05219344  1.48701663 -1.58118898 -0.77298599
## [19]  0.11534840 -0.58103809  1.16270069  1.31932455 -0.06933115 -0.45496956
## [25]  0.67189632
```

Her har vi kun 25 værdier, men hvis dataen er store, måske vil vi hellere kun kigge på de første (eller sidste) værdier:

```
head(x) #første 6
```

```
## [1]  0.66705324 -1.28047497  0.54987317  0.51085239 -1.14952601  0.07188865
```

```
tail(x) #sidste 6
```

```
## [1] -0.58103809  1.16270069  1.31932455 -0.06933115 -0.45496956  0.67189632
```

```
x[1] #første værdi
```

```
## [1] 0.6670532
```

```
x[length(x)] #sidste data point
```

```
## [1] 0.6718963
```

Bemærk, at til forskellen af Python og mange andre programmering sprog, R bruger 1-baserende indicer - det betyder, at den første værdi er `x[1]` og **ikke** `x[0]` som i Python.

1.7.2 Measures of central tendency

function	Description
`mean()`	mean $\bar{x}_i = \frac{1}{n}\sum_{i=1}^{n} x_i$
`median()`	median value
`max()`	maximum value
`min()`	minimum value
`var()`	variance $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x}_i)^2$
`sd()`	standard deviation s

Lad os afprøve dem på vores simulerede data:

```
my_mean <- mean(x)
my_median <- median(x)
my_max <- max(x)
my_min <- min(x)
my_var <- var(x)
my_sd <- sd(x)
c(my_mean,my_median,my_max,my_min,my_var,my_sd)
```

```
## [1]  0.1152770  0.1153484  1.4870166 -1.5811890  0.6650062  0.8154791
```

Man kan også lave et summary af dataen, som bestå af mange af de statistiker navnt ovenpå:

```
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.5812 -0.2926  0.1153  0.1153  0.6156  1.4870
```

1.7.3 `tapply()`

En meget brugbart funktion, som er værd at vide, er `tapply()`.

```
data(iris)
tapply(iris$Sepal.Length,iris$Species,mean) # ovenstående i kun en linje
```

```
##     setosa versicolor  virginica
##      5.006      5.936      6.588
```

Her tager vi en variabel der hedder `Sepal.Length`, opdele den i henhold til `Species`, og beregner `mean` for enhver af de tre `Species` (setosa, versicolor og virginica). Man kan opnå det samme resultat ved at beregne `mean` for de tre `Species` hver for sig:

```
# gennemsnit Sepal Length for Species setosa
mean_setosa <- mean(iris$Sepal.Length[iris$Species=="setosa"])

# gennemsnit Sepal Length for Species versicolor
mean_versi <- mean(iris$Sepal.Length[iris$Species=="versicolor"])

# gennemsnit Sepal Length for Species virginica
mean_virgin <- mean(iris$Sepal.Length[iris$Species=="virginica"])

c(mean_setosa,mean_versi,mean_virgin)
```

```
## [1] 5.006 5.936 6.588
```

Hvis der er mange grupper, så er der en stor fordele ved at brug `tapply` her. Det er også værd at ved koncepten, fordi vi kommer til lære en lignende koncept i **tidyverse** (med `group_by` og `summarise`).

1.8 Statistike analyse

Det er bare en kort oversigt over nogle af de grundlæggende analyser man kan lave i R, som man kan referere til.

1.8.1 1 sample t-test

Første simulere vi noget data fra normal distribution med mean = 3.

```
set.seed(290223) # bare for at få den samme resultat hver gang
x <- rnorm(10,3,1)
```

Nullhypotsen vs alternativ hypotesen:

- $H_0 : \mu = 3$, VS
- $H_1 : \mu \neq 3$

Lave test i R:

```
t.test(x,mu = 3)
```

```
##
##  One Sample t-test
##
## data:  x
## t = -1.1448, df = 9, p-value = 0.2818
## alternative hypothesis: true mean is not equal to 3
## 95 percent confidence interval:
##  2.169968 3.272231
## sample estimates:
## mean of x
##  2.721099
```

p-værdien er 0.2818, som er > 0.05, så forkaster vi ikke nullhypotesen, og konkluderer at $mu = 3$.

1.8.2 2-sample t-test

Samme variance:

```
x <- rnorm(10,3,1)
y <- rnorm(10,5,1)

t.test(x,y,var.equal = T)
```

```
##
##  Two Sample t-test
##
## data:  x and y
## t = -5.4258, df = 18, p-value = 3.729e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.700858 -1.193081
## sample estimates:
## mean of x mean of y
##  2.783056  4.730025
```

Variance forskellige:

```
x <- rnorm(10,3,1)
y <- rnorm(10,5,3)

t.test(x,y,var.equal = F) #var.equal=F er 'default' så man behøver ikke at specifere
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = -2.0238, df = 11.77, p-value = 0.0663
```

```
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.9077927  0.1483728
## sample estimates:
## mean of x mean of y
##  2.757436  4.637146
```

1.8.3 Paired t-test

```
before <- rnorm(10,3,1)
after <- rnorm(10,5,3)

t.test(before,after,paired=T)
```

```
##
##  Paired t-test
##
## data:  before and after
## t = -0.89463, df = 9, p-value = 0.3943
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.900384  1.689647
## sample estimates:
## mean of the differences
##               -1.105368
```

1.8.4 ANOVA

Her har man flere grupper i stedet for to.

- Grupper må være normale fordelt
- Variancen må være de samme i alle grupper

For k grupper, er nul/alternativhypotese:

- $H_0 : \mu_1 = \mu_2 = ... = \mu_k$
- H_1 : ikke alle middelværdier er enes

```
group1 <- rnorm(50,10,3)
group2 <- rnorm(55,10,3)
group3 <- rnorm(48,5,3)

#data må være i en dataramme, med den ene kolon = vores værdier, og den anden kolon = grupper
y <- c(group1,group2,group3)
x <- c(rep("G1",50),rep("G2",55),rep("G3",48))
mydf <- data.frame("group"=x,"values"=y)
```

```
mylm <- lm(values~group,data=mydf)
anova(mylm)
```

```
## Analysis of Variance Table
##
## Response: values
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## group       2  896.42  448.21  46.703 < 2.2e-16 ***
## Residuals 150 1439.56    9.60
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

P-værdien er 1.877e-15 (<0.05), så nulhypotesen er forkastet til fordele af alternativhypotesen. Bemærke at det er til trods af, at to af de tre grupper kommer fra en normal fordeling med præcis de samme middelværdier (det er nok, at den trejde gruppe har en anden middelværdi).

1.8.5 Correlations

Måler sammenhængen mellem to variabler:

- > 0 betyder, at der er en postiv sammenhæng
- < 0 betyder, at der er en negativ sammenhæng
- $= 0$ betyder, at der er ingen sammenhængen mellem de to variabler

```
data(cars)
cor(cars$speed, cars$dist)
```

```
## [1] 0.8068949
```

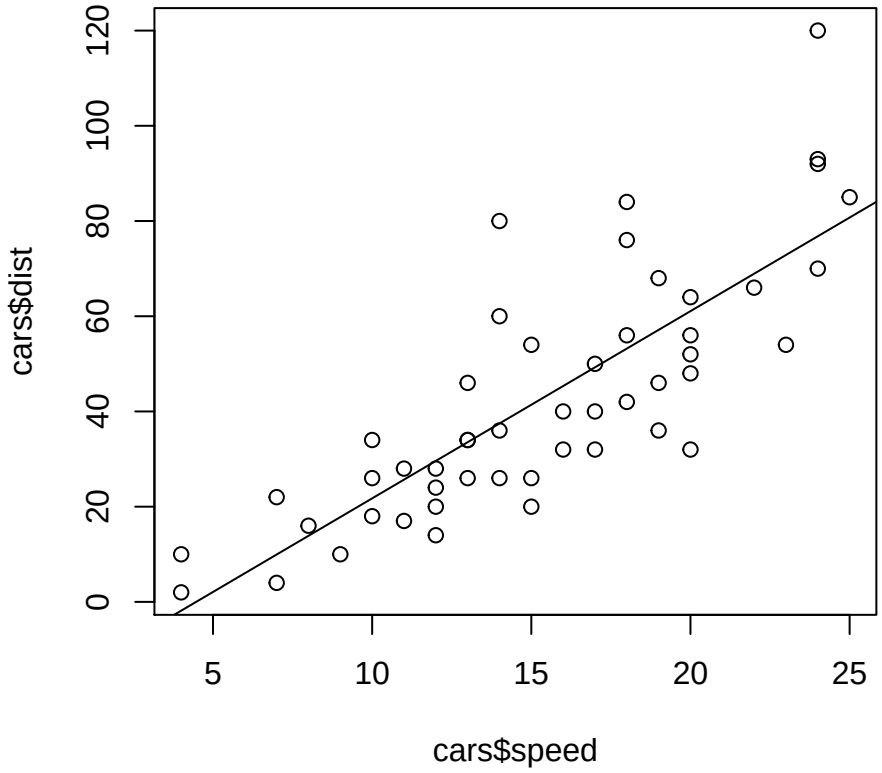
```
cor.test(cars$speed, cars$dist)
```

```
##
##  Pearson's product-moment correlation
##
## data:  cars$speed and cars$dist
## t = 9.464, df = 48, p-value = 1.49e-12
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6816422 0.8862036
## sample estimates:
##       cor
## 0.8068949
```

1.8.6 Linær regression

Formål: finde den bedste rette linje:

```
plot(cars$speed,cars$dist)
abline(lm(dist ~ speed, data=cars) )
```



Man bruger `lm()` og huske at det skrives som `lm(y~x,data=mydata)`, hvor i dette tilfælde er x `speed` (x-aksen i ovenstående plot) og y er `dist` (y-aksen i ovenståen plot), for `mydata` = `cars`.

```
mylm <- lm(dist ~ speed, data=cars)  # build linear regression model
mylm
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Coefficients:
## (Intercept)        speed
##     -17.579        3.932
```

1.9 Problemstillinger

1) Jeg har lavet en quiz i Absalon, hedder "Quiz - Basics" - som I kan starte med.

2) Åben en ny fil i Rstudio, ved at trykke på "File" > "New File" > "R script". Leger lidt - fk. nogle muligheder:

```
2+2
x <- 4
x <- x+2
sqrt(x)
sqrt(x)^2
rnorm(10,2,2)
log10(100)
```

Huske at den nemmeste måde at kører kode er ved at trykke CMD+ENTER (Mac) eller WIN-KEY+ENTER (Windows).

3) Åbne op og kigge på nogle af de indbygget datasæt som vi bruger i kurset. Prøve `head()`, `summary()` osv. Prøve også fk. `?cars` for at se en beskrivelse.

```
data(iris)
data(cars)
data(sleep)
data(PlantGrowth)
head(chickwts)
#se her for andre:
library(help = "datasets")
```

4) Lave en `data.frame` (dataramme) med tre kolonner som hedder "navn", "alder" og "farve". Sørge for, at den har 8 rækker.

```
mydf <- data.frame("navn"= ...) #not run, slette "..." og skrive data
dim(mydf) # otte række og tre kolonner
mydf
```

5) Lads os lave plotter i 'baseR'. Vi kommer til at omdefinere hvordan vi laver plotter, når vi arbejder med `ggplot2`, men det er nyttigt at have et kendskab til baseR plotter. Jeg giver nogle muligheder for datasættet, der hedder "iris" - afprøve dem for nogle af de andre ovenstående indbygget datasæt, som I kiggede på.

```
plot(iris$Sepal.Length,iris$Sepal.Width)
hist(iris$Sepal.Width)
boxplot(iris$Sepal.Length~iris$Species)
```

Man kan også gøre plotterne lidt pænere ved at give dem en titel/aksen-navne osv. Prøve `?plot` for at se nogle muligheder, og tilføje `ylab`, `xlab`, `main` (titel)

i én af plotterne. Lege også med `col` (farver).

6) Øve med at åbne en fil, der sidder i Absalon og hedder "reactions.txt". Kopiere filen ind i egen working directory (fk. Downloads or Documents - huske man kan sætte en working directory `setwd("~/Documents/")`) og bruge `read.table()` (giv objektet et navn, e.g. `data`). Huske at tjekke, om filen har en 'header' og bruge således `header=T` hvis nødvendigt.

7) Kolonner navne "subject" og "condition" indlæses som data type 'int' (heltal) men skulle hellere være 'factorer' (fordi de er kvalitatitiv). Gøre dem til facktorer, fk.

```
data$subject <- as.factor(data$subject) #gøre subject til en faktor
## gøre den samme her for condition:
data$...
```

8) Bruge `mean` til at beregne den gennemsnitlige reaktion tid (RT) for hver af de to konditioner. Nu prøve at anvende `tapply` til at gøre den samme.

9) Bemærke at vores data er 'paired' - det er den samme sæt "subject"s for hver af de to "conditions"s (altså "subject" = 1 har en værdi for både "condition" = 1 og "condition" = 2).

Lave en scatter-plot af reaktioner tider mellem de to "condition"s.

10) Lave en paired-t-test `t.test(x,y,paired=T)`. Hvad er p-værdien? Hvad er nulhypotesen og alternativ hypotesen? Er der en forskel mellem de to konditioner?

OBS: "reactions.txt" er emnet af video 2 i det næste kapitel om R Markdown.

Chapter 2

Introduktion til R Markdown

2.1 Hvad er R Markdown?

R Markdown er ligesom R, gratis og 'open source' og kan bruges som en bekvemt måde at arbejde med R til projektor på. Man kan nemlig anvende R Markdown til følgende:

- Skrive, gemme og køre R kode
- Få direkte adgang til datasæt
- Lave rapporter som kan dels med andre
- En nem tilgang til at forstå andres kode, fk. gennem eksemplerne og opgaverne fra dette kursus

Vi kommer til at bruge R Markdown hele vejen igennem kurset. Man kan installere R Markdown indenfor R ved brugen af den følgende kommando:

Her er en 'quick tour' som kan være nyttig (valgfri) https://rmarkdown.rstudio.com/authoring_quick_tour.html

2.2 Video demonstrationer

Jeg har lavet to korte videoer som kan ses her.

Video 1:

- Jeg viser hvordan man laver et nyt dokument i R Markdown
- Jeg viser hvordan man skriver tekst ind i dokumentet
- Jeg viser hvordan man bruger "knit" til at lave en HTML-rapport
- Jeg viser hvordan man oprette og køre kode chunks

Link her hvis det ikke virker nedenunder: https://vimeo.com/541035944

Video 2:

- Jeg viser en kort analyse med et datasæt
- Jeg indlæser de data og laver en forløbig undersøgelse
- Jeg laver en paired t-test og skriver konklusioner
- Jeg tjekker antagelserne omkring den normal fordeling

Link her hvis det ikke virker nedenunder: https://vimeo.com/541232690

2.3 Lave et nyt dokument i R Markdown

```
install.packages("rmarkdown")
```

Huske at indlæse pakken i RStudio:

```
library(rmarkdown)
```

Man åbne et nyt rmarkdown dokument ved at trykke "New" > "New File" > "New R Markdown…". Man kan også trykke på den "+" knap øverst til venstre i hjørnet.

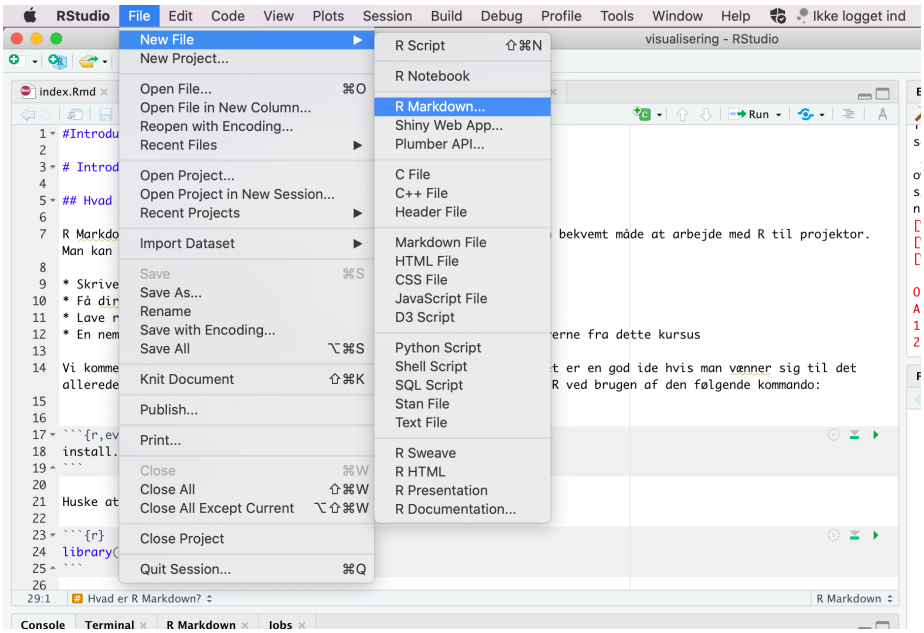


Figure 2.1: Hvordan man åbne et nyt R Markdown dokument

I de fleste tilfælde arbejder vi med HTLM dokumenter, men man har også andre muligheder (PDF/Word/Shiny osv…).

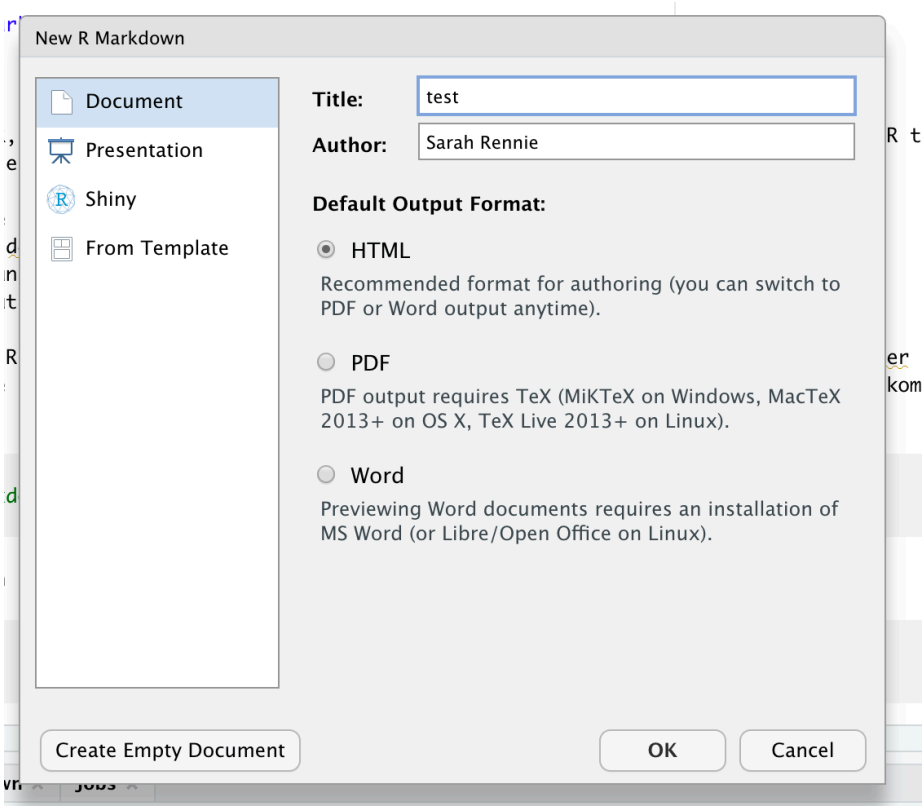
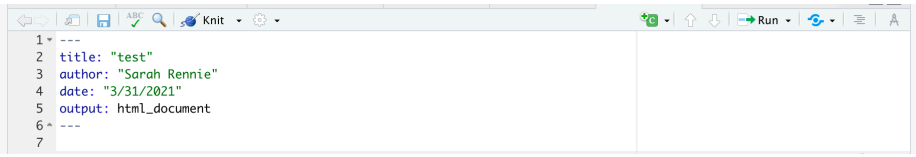


Figure 2.2: Hvordan man åbne et nyt R Markdown dokument

2.3.1 YAML

Den første sektion hedder 'YAML'. (Dette står for 'YAML Ain't Markup Language').



Det indeholder oplysninger om dokumentet, og her kan man specificere forskellige muligheder - fk. titel, forfatter, output-type (fks. HTML eller PDF), datoen, osv.

I de fleste tilfælde, kan vi bare nøjes med at bruge standard indstillinger, men hvis man gerne vil lære mere om de forskellige muligheder med YAML, kan man læse her:

https://bookdown.org/yihui/rmarkdown/html-document.html

eller se en liste muligheder her på den her cheatsheet:

https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

2.3.2 Globale options

Bemærke, at der også er nogle tekst som ser ud som følgende:

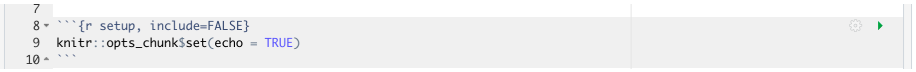


Figure 2.3: Hvordan man åbner et nyt R Markdown dokument

Med funktionen `opts_chunk$set()` kan man specificere de globale indstillinger, man gerne vil have, som styrer hvordan det færdige dokument ser ud. I dette tilfælde, er de fleste parametre angivet som 'default' (da de ikke er nævnt eksplicit), og `echo` er den eneste der har noget andet angivet. Hvis `echo` er `TRUE`, så betyder det, at når man kører sine kode og kompilerer dokumentet, så kan man også se den kode, der blev kørte, ligesom dens output, som en del af den færdige HTML, der fremvises.

2.4 Skrive baserende tekst

Her er nogle nyttige måder, man kan skrive tekste på, i opgaverne eller rapporter.

```
*italic*   **bold**

_italic_   __bold__
```

italic **bold**

italic **bold**

2.4.1 Headers

Man kan også lave sektioner:

```
# Header 1

## Header 2

### Header 3
```

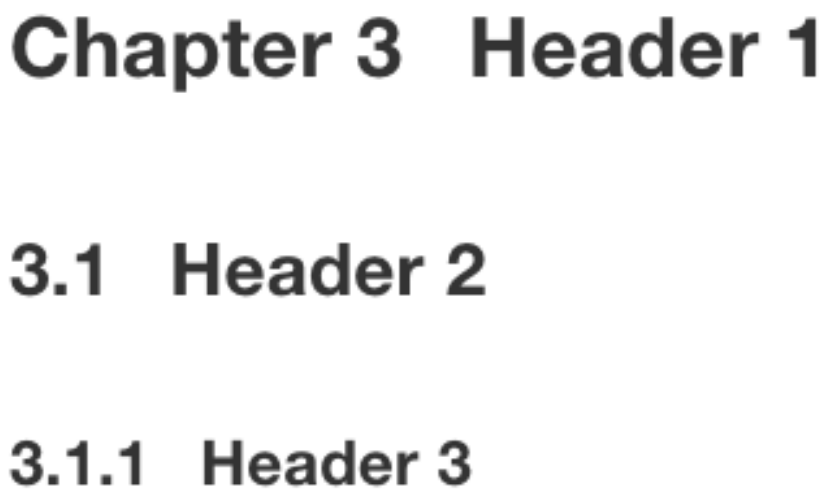


Figure 2.4: Caption for the picture.

2.4.2 liste

```
* Item 1
* Item 2
    + Item 2a
    + Item 2b
```

- Item 1

- Item 2
  - Item 2a
  - Item 2b

2.5 Kode indenfor teksten ('Inline chunks')

De fleste vores kode skrives indenfor såkaldte 'chunks' som vi kommer til nedenfor. Men nogle gange kan det være nyttigt at skrive kode direkte indenfor teksten. Dette gøres ved at skrive

```
Her er nogle `kode`
```

som ser sådan ud indenfor teksten:

Her er nogle `kode`

I dette tilfælde, bliver koden ikke kørte. Hvis man vil køre koden indenfor teksten, kan man skrive (for eksempel):

```
De gennemsnitlige antal af observationer er `r mean(c(5,7,4,6,3,3))`
```

som ser sådan ud indenfor teksten:

De gennemsnitlige antal af observationer er 4.6666667

Og hvis man dropper de 'r', så bliver kodet ikke kørte:

```
De gennemsnitlige antal af observationer er `mean(c(5,7,4,6,3,3))`
```

giver:

De gennemsnitlige antal af observationer er `mean(c(5,7,4,6,3,3))`

2.6 Kode chunks

Man kan oprette en ny chunk, enten ved at trykke på de *Insert a new code chunk* knap ovenpå, eller ved at trykke *Cmd+Option+I* på tastaturet (hvis man bruger MAC) eller *Ctrl+Alt+I* (hvis man bruger Windows).

Tryk på den grønne pile, der hedder *Run current chunk* for at køre hele den chunk. Resultatet kan ses lige nedenunder. Man kan også trykke på den grønne pile som ligger øverst til højre i den kode chunk.

Det er ofte hurtigere at bruge *Run current chunk* i stedet for at *Knit* (se nedenfor) hver gang man vil køre kode, fordi her kører man kun den enkele chunk, man er interesset i, i stedet for det hele dokument (som er tilfældet med *Knit*).

2.6.1 Chunk options

For eksempel, en chunk med en 'option' nævnt ser sådan ud (fjerne # symbol)

```
#```{r,eval=FALSE}
#
#```
```

Her er nogle muligheder (sektionen "Embed code with knitr syntax"):

https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf

Her er seks populær muligheder som jeg har kopiret fra nettet:

- `include = FALSE`
  - prevents code and results from appearing in the finished file. R Markdown still runs the code in the chunk, and the results can be used by other chunks.
- `echo = FALSE`
  - prevents code, but not the results from appearing in the finished file. This is a useful way to embed figures.
- `message = FALSE`
  - prevents messages that are generated by code from appearing in the finished file.
- `warning = FALSE`
  - prevents warnings that are generated by code from appearing in the finished.
- `fig.cap = "..."`
  - adds a caption to graphical results.
- `eval = FALSE`
  - does not evaluate the code

2.7 Knit kode

Man bruger *Knit* for at gengive filen i HTML form. Når man trykke på *Knit*, alle kode i filen bliver kørte og en HTML fremviser.

Man kan også bruge *Preview*, som kører ikke kode chunks, men bruger koden som blev kørt før.

2.8 Matematik

For example, the inline code `$\int_0^5 x^2 dx$` will typeset as $\int_0^5 x^2 dx$:

2.9 Problemstillinger

1) Jeg har lavet en kort **quiz** i Absalon, som hedder "Quiz - R Markdown".

2) Lave et nyt R Markdown dokument i R Studio. Prøve at lave en list og nogle overskrifter af forskellige størrelser.

3) Tryk på `Knit` og tjekke at et html-dokument fremvises som forventet.

4) Tilføj en ny R chunk, med noget kode. e.g.

```
x <- rnorm(20,1,2) #make a sample of normally distributed data
plot(x)
```

Øve med at trykke enten på den grønne pile, eller på knit.

5) Bruge `$ $` til at skrive formulen til 'mean' ind i teksten, $\bar{x}$. Hint: `$\sum_{i=1}^{n}$` and `${x_{i}}$`.

2.10 Opgaven

Huske at sende mig eventuelle spørgsmål, som jeg kan svare på i Zoom rummet (sende gerne via mail eller chat).

Ind på Absalon har jeg lagt en R Markdown fil som hedder "R Markdown opgave", som I kan bruge til at starte med at arbejde lidt med R Markdown baserede opgaver. Det kombinerer koncepter fra de forudgående kapitel om de grundlæggende ting i R og statistik.

2.11 Slut for ugen

Næste gang begynder vi at arbejde vi med `ggplot2`.

2.12 Ekstra links

https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf?_ga=2.49295910.1034302809.1602760608-739985330.1601281773