Seminar 1

Lise Rødland

2020-08-20

Seminaropplegget

De fleste seminarer vil bestå av to deler:

1. En introduksjon om dagens emner.
2. Oppgaver relatert til dagens emner. For å lære seg R så er det *helt* nødvendig å anvende kunnskapen.

I år har vi fire introduksjonsseminarer som vil bli etterfulgt av en digital prøve mandag 14. august. Prøven er gjøre tilgjengelig i Canvaskl 9 og må leveres senest kl 15, men det bør ikke ta så mye mer enn to-tre timer å løse den. Vi gir dere god tid så vi kan ta høyde for tekniske problemer e.l. Dere har lov til å bruke alle hjelpemidler, men dere skal ikke samarbeide. Dere skal levere et kjørbart script som svarer på oppgavene på prøven.

På introduksjonsseminarene skal vi lære alt fra å laste inn datasett til å kjøre egne analyser for så å plotte resultatene. Vi skal lære ting som vil være anvendbare langt utover statistisk analyse. Neste uke tilbyr vi fordypningsseminarer for de av dere som vil jobbe videre med R. Disse anbefales spesielt til dere som skal skrive en statistisk oppgave, men det er selvfølgelig ikke et krav for å delta. På fordypningsseminarene vil vi gå nærmere inn på R-kode for flernivå-analyse, faktoranalyse og paneldata. Vi håper så klart å se mange av dere der også. Det blir ikke noen prøve om innholdet i fordypningsseminarene.

Hjemmeoppgaven

Om du ikke allerede har bestemt deg så begyn å tenk på hjemmeoppgaven nå. Om du skal skrive en statistisk oppgave så er det fint å ha hjemmeoppgaven i bakhodet når vi går gjennom ting på seminar. Ta gjerne en titt på Harvard dataverse. Mange forskere bruker Harvard dataverse til å publisere replikasjonsdata når de skriver artikler. Du kan søke på temaer du synes er interessante eller en spesiell artikkel du syns er spennende.

Om du har sett deg ut en artikkel du ikke finner på Harvard dataverse så kan dere sjekke om det står noe om hvor data er tilgjengelig i artikkelen. Noen forskere tilgjengeliggjør også data i tidsskriftet artikkelen ble publisert i eller på egne hjemmesider, mens andre (dessverre) ikke publiserer data i det hele tatt.

Om å lære R

Det er noen ting som er kjekt å vite om R:

1. Øvelse gjør mester.
2. Å lære R er en prosess der eksponering er det beste læringsverktøyet.
3. Mye kan gjøres på flere måter.
4. Samarbeid gjør det hele mye enklere (og morsommere).

Det betyr at for å få mest mulig ut av seminarene så må dere jobbe med oppgavene vi legger ut mellom seminarene - gjerne sammen med noen andre. I begynnelsen kan det være vanskelig å vite hvorfor en kode ikke vil kjøre, men det å søke opp løsninger på nett og i R sine hjelpefiler er en viktig del av å lære seg R. Noen av svarene dere finner er kanskje ikke prikk like på det vi har gjennomgått i seminar, men det betyr

ikke at det ikke er en fin måte å løse det på. På prøven leverer dere et script for å vise at dere har brukt R for å finne svarene, men det finnes ikke noen **riktig** eller **feil** kode om svaret du kommer frem til er riktig.

I tillegg oppfordrer vi til å stille masse spørsmål - både til medstudenter og oss seminarledere. Bruk medstudenter, nettressurser og R sine hjelpefiler aktivt, samarbeid og hjelp hverandre når dere løser oppgaver så vil prøven gå veldig fint.

Jeg vil lenke til undervisningsmateriell på Canvas. Selve opplegget blir lastet opp på github. Der finner dere også opplegget for tidligere år, men det er mappen H20-seminarer som gjelder. Opplegget i år er en del annerledes enn tidligere år selv om vi skal gjennom mye av det samme.

Læringsressurser og nyttige R-lenker

- Gratis innføringsbok på nett - R for Data Science
- **Hjelpefilene i R:** Det krever litt trening å lære seg å forstå hjelpefiler, men det er en av de beste investeringene du kan gjøre - finn all informasjonen du trenger ved hjelp av `?` og `??` i R
- R-ladies Oslo og Oslo UseR! group inviterer til spennende R-temamøter
- Guide til ggplot2
- Facebookgruppe for R for statsvitere på UiO - bruk denne!
- Stilguide for R
- Bruke prosjekter i R
- Quick-R - et bra sted å søke etter ting, lett å forstå for nybegynnere
- Interaktiv R-intro fra datacamp
- Interaktiv tidyverse-intro fra datacamp
- Stackoverflow - har svar på det R-problemet du sliter med
- R-bloggers - har gode, kortfattede tutorials

Du kan også finne videoer, flere gratis bøker m.m. hvis du googler/søker på stackoverflow/søker på youtube. Noe av det aller viktigste er nok imidlertid å lære seg å lese og forstå hjelpefiler i R og stackoverflow. For å bli god i R er det lurt å huske noen grunnbegreper (dvs. en god del av det som du lærer i seminarene og **R for Data Science**), men ut over det bruker man hjelpefiler i R/stackoverflow som ordbøker hele tiden.

Første seminaropplegg

Mål for i dag:

1. Bli kjent med hverandre
2. Få R og Rstudio til å virke på egen PC
3. Forstå hvordan Rstudio er satt opp og hvordan man kjører kode
4. Forstå grunnleggende konsepter i R: indeksering, objekter, funksjoner og pakker
5. Løse noen oppgaver på egenhånd

Dersom du enda ikke har gjennomført installasjonsguiden[**sett inn oppdatert lenke**], ta en titt på den etterpå. I mellomtiden, finn deg en person som har R og Rstudio installert, og sett deg sammen med den personen med ok korona-avstand.

Dersom du har gjennomført installasjonsguiden, uten å ha fått Rstudio til å virke, så ser vi på dette etterpå.

Det første vi skal gjøre er å lage et lite datasett. Snakk med personen ved siden av deg og finn ut:

- Hva hen heter
- Hvor gammel hen er
- Hvor hen tok bacheloren sin
- En funfact :)

Etterpå tar vi en runde og da skal du presentere personen ved siden av deg. Informasjonen skal vi bruke til å lage et lite datasett. Bruk tre minutter.

Rstudio

Først litt helt grunnleggende om R og Rstudio. I Rstudio skriver vi kode i script. Scriptene kan lagres og gjør arbeidet vårt reproduserbart. Etter at vi har skrevet kode i script, sender vi koden til console for evaluering med *ctrl/cmd + enter*. Dersom resultet av koden ikke er et objekt eller et plot, vises resultatet i console (vinduet under script).

Objekter

For å bli kjent med R så skal vi bruke informasjonen fra bli-kjent-runden til å lage et lite datasett. Det første vi gjør er å lage objekter av en type som heter vektor. Vektorene skal inneholder informasjon om navn, alder og hvor dere tok bachelorgraden deres. I R kan en vektor defineres som en ordnet lise av verdier, dvs. at man lagrer verdier en bestemt rekkefølge. Når verdiene er i en bestemt rekkefølge så kan vi slå vektorene sammen for å lage et datasett. I dette kurset kan dere for det meste tenke på vektorer som variabler fra et datasett. Vi lager objekter ved hjelp av assignment operatoren ´<-´.

```
# Her ser dere et eksempel på R-kode
# Jeg bruker # for å skrive inn en kommentar som ikke skal evalueres av R

# Her lager vi vektoren Thea, Ole, Mari
# Vi bruker ctrl/cmd + enter for å kjøre koden
c("Thea", "Ole", "Mari")
```

```
## [1] "Thea" "Ole"  "Mari"
```

```
# Her lagrer vi vektoren Thea, Ole, Mari som et objekt vi kaller navn:
navn <- c("Thea", "Ole", "Mari")
```

Når vi lagrer objekter, returnerer ikke R en verdi i Console. I stedet dukker objektet opp i vinduet Environment (til høyre for scriptet).

```
alder <- c(23, 20, 25)
alder
```

```
## [1] 23 20 25
```

```
bachelor <- c("UIO", "UIB", "UIS")
bachelor
```

```
## [1] "UIO" "UIB" "UIS"
```

```
data <- data.frame(navn, alder, bachelor) # Her slår vi de tre vektorene sammen for å lage et datasett
data
```

```
##   navn alder bachelor
## 1 Thea    23      UIO
## 2  Ole    20      UIB
## 3 Mari    25      UIS
```

Kan du finne din rad?

Det finnes flere typer objekter i R. Vektor og datasett er de to typene vi skal fokusere på idag. Vi skal straks tilbake til datasettet vi har laget, men først skal vi ta en kort intro til R og Rstudio.

Intro til R

I R skriver vi kode i script. Du åpner et nytt script ved å trykke på den hvite firkanten med en grønn sirkel og et plusstegn oppe til venstre i Rstudio. Etter at vi har skrevet kode i scriptet, sender vi koden til console

for evaluering med ctrl/cmd + enter. Dersom output fra koden ikke er et objekt eller plot, vises resultatet i console (vinduet under script). La oss forsøke med koden `"Hello World!"`

```
"Hello world!"
```

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

R Kan også brukes som en kalkulator:

```
1 + 1  # addisjon
```

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

```
2 - 3  # subtraksjon
```

```
## [1] -1
```

```
4/2    # divisjon
```

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

```
2 * 2  # multiplikasjon
```

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

```
2^3    # potens
```

```
## [1] 8
```

```
exp(2) # eksponentiering
```

```
## [1] 7.389056
```

```
log(2) # logaritme (default er naturlig logaritme)
```

```
## [1] 0.6931472
```

```
2 * (4-2)/(4-2) # Parentesregler fungerer som i vanlig algebra: den innerste parentesen regnes ut først
```

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

Logiske tester

R kan evaluere logiske utsdagn og bedømme om de er ´TRUE´ eller ´FALSE´.

```
1 == 2                                  # tester om 1 er lik 2
```

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

```
2 == 2                                  # tester om 2 er lik 2
```

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

```
"Statsvitenskap" == "statsvitenskap"  # Logiske tester kan også brukes på tekst
```

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

```
"statsvitenskap" == "statsvitenskap"  # R er imidlertid sensitivt til store og små bokstaver
```

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

```
1 <= 2                                  # Tester om 1 er mindre enn eller lik 2
```

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

```r
1 >= 2                                  # Tester om 1 er større enn eller lik 2
```

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

```r
1 != 2                                  # Tester om 1 er ulik 2
```

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

```r
1 == 2 | 1 == 1                         # Tester om en av de to påstandene 1 er lik 2 eller 1 er lik 1 er
```

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

```r
1 == 2 & 1 == 1                         # Tester om begge de to påstandene 1 er lik 2 og 1 er lik 1 er sa
```

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

Oversikt over logiske operatorer

Oversikt over logiske operatorer:

Operator	Betydning
==	er lik
<	mindre enn
>	større enn
<=	mindre eller lik
>=	større eller lik
!=	ikke lik
!x	ikke x
|	eller
&	og

Vi kommer til å bruke disse opearterne mye, spesielt når vi gjør endringer i datasett som å lage nye variabler. Det er derfor viktig at dere forstår hvordan disse fungerer. Den beste måten å få denne forestående på er å øve.

Pakker

R er open source og mange flinke utviklere bidrar til å gjøre R bedre. Disse utviklerne deler kode (i hovedsak funksjoner) gjennom pakker. For å kunne kode fra en pakke må vi kjøre følgende kodelinjer:

```r
install.packages("pakkenavn") # Laster ned filene pakken består av fra nett til PC - må bare gjøres en
library(pakkenavn)            # Tilgjengeliggjør pakken i R-sesjonen, må gjøres hver gang du vil bruke
```

Vi skal for det meste bruke kode som kommer fra pakker i tidyverse. La oss installere disse pakkene:

```r
install.packages("tidyverse") # Fjern hashtag på starten av denne og neste linje!
install.packages("haven") # legg merke til at vi bruker "" i install.packages(), men ikke i library()
library(tidyverse)
```

Indeksering

Nå skal vi tilbake til datasettet vi lagde basert på bli-kjent-runden. Dersom vi ønsker å hente ut informasjon om en person så kan vi bruke indeksering. Indeksering lar oss spesifisere et eller flere elementer i et objekt. Det er flere måter å indeksere på og dette er nøye beskrevet i kapittel 1 i **Lær deg R** som er anbefalt pensum.

Dersom vi for eksempel er interessert i å hente ut all informasjonen om Thea så kan det gjøres på mange måter:

```r
# Vi kan bruke base R (som beskrevet i Lær deg R)
data[navn == "Thea", ]      # vi ønsker alle kolonner/variabler for observasjone/radene med verdien "Th
```

```
##   navn alder bachelor
## 1 Thea    23      UIO
```

```r
data[data$navn == "Thea", ] # vi ønsker alle kolonner/variabler for observasjone/radene med verdien "Th
```

```
##   navn alder bachelor
## 1 Thea    23      UIO
```

```r
data[1, ]                   # Thea er den første raden (observasjonen) i datasettet
```

```
##   navn alder bachelor
## 1 Thea    23      UIO
```

```r
# Vi kan også bruke tidyverse-pakken dplyr:
data %>%
  filter(navn == "Thea")
```

```
##   navn alder bachelor
## 1 Thea    23      UIO
```

Vi kan også hente ut informasjon i en variabel/kolonne:

```r
# Ved hjelp av base R
data$alder
```

```
## [1] 23 20 25
```

```r
data[, "alder"]
```

```
## [1] 23 20 25
```

```r
data[, 2]                    # Alder er kolonne nr to fra venstre
```

```
## [1] 23 20 25
```

```r
# Ved hjelp av dplyr
data %>%
  select(alder)
```

```
##   alder
## 1    23
## 2    20
## 3    25
```

Til slutt så kan vi hente ut informasjon om en variabel for en observasjon:

```r
# Ved hjelp av base R
data[navn == "Thea", "alder"]
```

```
## [1] 23
```

```r
data[data$navn == "Thea", "alder"]
```

```
## [1] 23
```

```r
# Ved hjelp av dplyr
data %>%
  filter(navn == "Thea") %>%
  select(alder)
```

```
##   alder
## 1    23
```

Ulike typer vektorer i R

Det finnes flere ulike objekter i R. Til nå har vi blitt introdusert for to av dem: datasett og vektorer. Hva slags objekt det er har noe å si for hva du kan gjøre med det. I tillegg finnes det ulike typer, eller klasser som vi kaller det, av vektorer. I tabellen under finner dere en grov inndeling av ulike typer vektorer i R (se også s. 42-47 i Lær deg R):

Atomic vector	List
numeric	blanding
integer	
character	
factor	
logical	

En hyppig årsak til at en funksjon ikke fungerer, er at en vektor/variabel ikke er i det formatet vi forventet. Tabellen gir en oversikt over variabeltypene vi skal jobbe med. Atomic vector har kun verdier av en type, mens lister kan ha flere typer verdier, samt bestå av flere variabler.

Hvilket format tror du navn, alder og bachelor har?

Det kan vi sjekke med funksjonen `class()`.

```
class(data$navn)
```

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

```
class(data$alder)
```

```
## [1] "numeric"
```

```
class(data$bachelor)
```

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

Som dere ser er alder numeric, mens navn og bachelor er character. Her er det hva objektet er lagret som som teller, ikke hvordan det ser ut i datasettet. Selv om noe ser ut som tall i datasettet så er det ikke sikkert det er registrert som tall av R. Heldigvis kan dette ofte løses ved hjelp av en funksjoner som `as.numeric()`, `as.character()` og `as.factor()`.

```
# Her lager vi en ny variabel alder2 der vi ber R lagre alder som character
data$alder_ch <- as.character(data$alder)
```

Om vi ser på datasettet etter at vi har laget den nye variabelen så ser vi at alder og alder2 ser helt like ut. Begge fremstår som tall vi kan gjøre regneoperasjoner på, men bare en av dem er det. Prøv gjerne selv å kjøre funksjonen `mean()` som regner ut gjennomsnittet på alder2.

Hva skjer om du først omkoder alder til en faktor ved hjelp av `as.factor()` og så omkoder faktorvariabelen til en numerisk variabel med `as.numeric()`?

Funksjoner i R

Dersom vi ønsker å gjøre noe i R, får vi som regel en funksjon til å gjøre jobben for oss. En funksjon tar i mot verdi(er), gjerne lagret i form av et R-objekt, utfører operasjoner basert på input, og produserer nye verdier. En typisk R-funksjon har følgende syntaks:

```r
aFunction(x = "R-objekt", arg = "alternativ for figurens oppførsel")
## Merk: dette er ikke en faktisk funksjon i R. Funksjoner kan også ha andre syntakser.
```

Vi har allerede vært innom flere funksjoner i dag; `c()`, `class()`, `filter()`, `select()` og `mean()`. Dersom vi vil lære mer om en funksjon, kan vi spørre R om hjelp med `?`. Kjør `?c()` og `mean()`

En funksjon er en kodesnutt der vi spesiferer en input. Funksjonen utfører operasjoner basert på input, og produserer deretter et resultat (eller output). Det finnes funksjoner for å løse de fleste tenkelige og utenkelige problemer. Noen funksjoner er det lurt å kunne, fordi du får bruk for dem hele tiden, men det er altfor mange funksjoner til å lære seg alle. En av de viktigste ferdighetene i R, er derfor å lære seg hvordan man kan finne og anvende funksjoner for å løse utfordringer man står overfor. Dette skal vi øve på i R-undervisningen.

Noen funksjoner er base R og vi trenger ikke gjøre noe for å bruke dem. Andre må vi laste ned og installere pakker for å bruke som med `select()` og `filter()` fra `tidyverse`.

Vi skal nå bruke noen funksjoner til å finne ut mer informasjon fra datasettet vårt:

```r
summary(data)
```

```
##      navn               alder         bachelor           alder_ch
##  Length:3           Min.   :20.00   Length:3           Length:3
##  Class :character   1st Qu.:21.50   Class :character   Class :character
##  Mode  :character   Median :23.00   Mode  :character   Mode  :character
##                     Mean   :22.67
##                     3rd Qu.:24.00
##                     Max.   :25.00
```

```r
# Finner minimumsverdi (den laveste alderen)
min(data$alder, na.rm = TRUE) # na.rm = TRUE sier at missing skal droppes i beregningen
```

```
## [1] 20
```

```r
# Finner maksimumsveriden (den høyeste alderen)
max(data$alder, na.rm = TRUE)
```

```
## [1] 25
```

```r
# Finner gjennomsnittalder
mean(data$alder, na.rm = TRUE)
```

```
## [1] 22.66667
```

```r
# Finner medianalderen
median(data$alder, na.rm = TRUE)
```

```
## [1] 23
```

```r
# Finner standardavviker
sd(data$alder, na.rm = TRUE)
```

```
## [1] 2.516611
```

```r
# Finner varians
var(data$alder, na.rm = TRUE)
```

```
## [1] 6.333333
```

```r
# Finner kvantilverdiene
quantile(data$alder, na.rm = TRUE)
```

```
##   0%  25%  50%  75% 100%
## 20.0 21.5 23.0 24.0 25.0
```

```r
# Finner forskjellig deskriptiv statistikk for en variabel
summary(data$alder)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20.00   21.50   23.00   22.67   24.00   25.00
```

```r
# Finner forskjellig deskriptiv statiskk for alle variabler i datasettet
summary(data)
```

```
##      navn                alder         bachelor           alder_ch
##  Length:3           Min.   :20.00   Length:3           Length:3
##  Class :character   1st Qu.:21.50   Class :character   Class :character
##  Mode  :character   Median :23.00   Mode  :character   Mode  :character
##                     Mean   :22.67
##                     3rd Qu.:24.00
##                     Max.   :25.00
```

```r
# install.packages("moments")
library(moments)

# Finner skjevhet ved hjelp av moments-pakken
skewness(data$alder, na.rm = TRUE)
```

```
## [1] -0.2390631
```

```r
# Finner kurtose ved hjelp av moments-pakken
kurtosis(data$alder, na.rm = TRUE)
```

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

```r
# Får en tabell med de ulike studiestedene
table(data$bachelor, useNA = "always") # useNA = "always" betyr at vi også vil ha med antall "missing"
```

```
##
##  UIB  UIO  UIS <NA>
##    1    1    1    0
```

Laste inn data

Til slutt så skal vi laste inn dataene dere skal bruke i oppgavene i etterpå. Dersom dere skal gjøre statistisk analyse, er som regel den første seksjonen import og forberedelse av data. En styrke ved R, er at det er mulig å importere mange ulike filtyper, både fra en mappe på pcen din og fra en url på internett. Jeg går gjennom import av filer fra excel, stata, spss og R, men vit at det finnes mange andre muligheter.

Når du skal laste inn eller lagre noe lokalt på pc-en så vil R til enhver tid forvente at filnavn du refererer til befinner seg i **working directory**, som er en mappe på pcen din. For å sjekke hva nåværende **working directory** er, og hvilke filer som finnes i den mappen, kan du skrive følgende kode (jeg har gjemt egen output):

```r
getwd()
list.files()
```

For å endre **working directory**, bruker dere følgende kode:

```r
setwd("C:/Users/Navn/R/der/du/vil/jobbe/fra")   # For windows
setwd("~/R/der/du/vil/jobbe/fra")               # For mac/linux
# Merk at R bruker / for å skille mellom mappenivåer, mens filutforsker i Windows bruker \
# Kopierer du mappebanen fra filutforsker må du derfor huske å snu skråstrekene i R
```

Et annet alternativ er å starte et nytt prosjekt i R, ved å trykke på *File* og deretter *New Project* i menyen øverst til venstre i Rstudio. Da får du muligheten til å lage en ny mappe på pcen din som blir satt til working directory. Senere kan du sette denne mappen til working directory med Open project fra menyen (jeg anbefaler å bruke prosjekter, særlig til hjemmeoppgaven).

Datasett kommer i mange ulike filformater. Noen vanlige formater er csv, dta (Stata-datasett), sav (SPSS-datasett) og Rdata. Hvilket format dataene dine har bestemmer hvilken funksjon du må bruke for å laste inn datasettet. Her er eksempler på noen funksjoner for å laste inn data:

```
library(tidyverse) # read_funksjoner fra readr i tidyusere
datasett <- read_filtype("filnavn.filtype") # Laster inn og lagrer datasettet som et objekt
read_csv("filnavn.csv") # for .csv, sjekk også read.table
load("") # For filer i R-format.

library(haven) # Fra haven-pakken - dette skal vi se på i senere seminar
read_spss("filnavn.sav")  # for .sav-filer fra spss
read_dta("filnavn.dta") # for .dta-filer fra stata
```

Nå skal vi laste inn datasettet vi skal bruke til å løse oppgaver i neste time.