Seminar 2

Plan for seminaret:

1. Pakker
2. Laste inn data
3. Organisering av arbeidet
4. Målenivå
5. Klasser og målenivå
6. Utforske data
7. Plotting

0. Working directory og datasett

Når vi skal begynne å jobbe i R, setter vi først working directory til den mappen vi ønsker å hente og lagre filer til.

Her kan man enten bruke "setwd"-funksjonen eller så kan man trykke "Session" i verktøy-linjen - "Set Working Directory" - "Choose Directory" for å velge en mappe.

```
setwd("~/Desktop/STV1020")
```

I dette seminaret skal vi bruke et datasett fra European Social Survey Round 9 (2018), og dettedatasettet inneholder svarene fra norske respondenter. Filen heter "ESS9NO.dta" og ligger i Canvas.

Pass på at du har lagret datasettet vi skal bruke i dag i samme mappe som den du har satt som working directory.

1. Pakker

R-pakker er utvidelser til programmeringsspråket R. De inneholder kode, data, og dokumentasjon som gir oss tilgang til funksjoner som løser ulike problemer og gjør koding enklere. Første gang man skal bruke en pakke må man installere den.

Så må vi "hente den fra biblioteket" for å fortelle R at vi ønsker å bruke pakken, dette må vi gjøre hver gang vi åpner R på nytt og ønsker å bruke pakken.

```
install.packages("tidyverse")

library(tidyverse)
```

2. Laste inn data

Datasettene som man vil kunne ønske å bruke i R vil ikke alltid være i det samme formatet, og man bruker ulike funksjoner for å laste disse ulike filtypene inn i R. Noen av funksjonene krever at vi først har installert en pakke. Hvis man lurer på hvordan man skal laste inn en bestemt filtype og har glemt hvordan man gjør det, så er dette veldig lett å Google.

Her kommer noen eksempler på hvordan man kan laste inn ulike filtyper inn i R.

For eksempel så er datasettet vi skal bruke i dag en STATA-fil, den kan lastes inn på følgende måte:

```
install.packages("foreign")
library(foreign)

df <- read.dta("ESS9NO.dta")
```

For å laste inn en excel-fil:

```
install.packages("readxl")
library(readxl)
df <- read_excel("file")
```

For å laste inn en CSV (Comma-separated values)-fil:

read.csv() er brukt for filer hvor komma er brukt til å separere verdiene i dokumentet (f.eks. tall) og read.csv2() er brukt for filer hvor semikolon er brukt til å separere verdiene.

```
df <- read.csv("file")
df <- read.csv2("file")
```

3. Organisering av arbeidet

Overskrifter og tekst
Hvordan man organiserer et R-script kommer an på hva man selv synes er mest oversiktlig, men det er viktig at man klarer å holde oversikt over hva man har kodet og forstår hva man har gjort når man kommer tilbake til et script.

F.eks. kan det være lurt å lage overskrifter og underoverskrifter til oppgaver eller elementer av arbeidet, som tydeliggjør hvorfor du har inkludert akkurat denne koden og hva du tenker at den skal gjøre.

Det kan også være lurt å inkludere #tekst som forklarer hva du gjør og hvorfor. Tekst som ikke skal leses av R skriver man etter emneknagg (#).

Organisering av data
Når man bruker større datasett som ESS, så inneholder datasettet ofte mange flere variabler enn de vi ønsker å bruke i våre analyser og variablene har navn som kan være vanskelig å huske, f.eks. nwspol.

Derfor kan det være lurt å fjerne de variablene vi ikke skal bruke og gi variablene navn som er lette for oss å forstå og huske.

Her endrer vi navnet til variablene til noe mer intuitivt. Vi bruker en pipe (%>%), som tar outputen til et utsagn og gjør det til inputen til det neste utsagnet. Pipen kan sees på som ordet

"så". Rename-funksjonen lar oss forandre navnet til variabler og bruker syntaksen nytt_navn = gammelt_navn.

```
df <- df %>% rename(
news = nwspol,
interest = polintr,
age = yrbrn)
```

Vi kan forandre "age" til å vise alder og ikke hvilket år respondenten er født.

```
df$age <- 2018-df$age # Undersøkelsen er fra 2018.
```

Her velger vi hvilke variabler vi vil ha med oss videre i datasettet. Da det blir mer oversiktlig, ettersom dette er et stort datasett med veldig mange variabler. Select- funksjonen lar oss velge variabler i et datasett. I dette tilfellet velger vi variabler ved å vise til navnene deres.

```
df <- df %>% select(news, interest, vote, age)
```

4. Målenivå

Nominalnivå
Når variabler er på nominalnivå, kan egenskapen deles i to eller flere gjensidig utelukkende kategorier.

F.eks. variabelen "vote", man har enten stemt, ikke stemt, eller så er man ikke berettiget til å stemme.

Hvis man har to gjensidig utelukkende kategorier så kan man bruke disse i binomisk regresjonsanalyse, så i dette tilfellet kunne man vurdert å fjerne kategorien ikke berettiget til å stemme.

Ordinalnivå

Når variabler er på ordinalnivå kan de deles i to eller flere kategorier som kan rangordnes. Så dette at verdiene kan rangordnes er det som skiller ordinalnivå fra nominalnivå.

F.eks. variabelen "interest", man kan være ikke interessert, lite interessert, ganske interessert, eller veldig interessert.

Intervallnivå

Når variabler er intervallnivå kan verdiene graderes på en skala med et tilfeldig nullpunkt, der en skalaenhet utgjør like mye av den underliggende egenskapen over hele skalaen.

F.eks. temperatur eller år.

Forholdstallsnivå

Når variabler er på forholdstallsnivå kan egenskapen graderes på en skala med et absolutt nullpunkt, der en skalaenhet utgjør like mye av den underliggende egenskapen over hele skalaen. Så forskjellen mellom intervallnivå og forholdstallsnivå er at variabler på forholdstall har et definert nullpunkt.

F.eks. variabelen "news", som viser hvor mye tid en respondent bruker på nyheter hver dag, og "age", som viser alderen til respondentene.

5. Klasser og målenivå

Vi kan sjekke hvilken klasse en variabel har med class-funksjonen, f.eks. for å sjekke politisk interesse: class(df$interest).

Variabler på nominalnivå og ordinalnivå vil være av klassen "factor".

Variabler på intervallnivå vil være av klassen "numeric" eller "integer", det samme gjelder forholdstallsnivå.

6. Utforske data

Det er mange ulike måter å utforske data på. Vi skal se på funksjonene: summary, str, levels, head, og tail.

For å få et deskriptivt sammendrag av et objekt kan vi bruke summary-funksjonen.

```
summary(df$vote)
```

Et alternativ til summary-funksjonen er str-funksjonen, som viser det interne strukturen til et R-objekt.

```
str(df)
```

For å undersøke nivåene til en variable kan man bruke levels-funksjonen, outputen man får er nivåene til variabelen og verdiene deres.

```
levels(df$interest)
```

Hvis man vil se de første eller siste radene i et datasett, kan man bruke henholdsvis head- og tail-funksjonene. Man kan også velge for eksempel å bare se på de første eller siste verdiene til en bestemt variabel.

```
head(df$interest)
tail(df$interest)
```

7. Plotting

Vi skal kort introdusere hvordan man kan visualisere data i dette seminaret, og så vil dere få en mer grundig gjennomgang neste seminar.

Det er gøy å kunne visualisere dataene våre, både for vår egen del, men også for de som skal lese oppgavene våre. For å få fine grafer kan man bruke pakken ggplot.

Søylediagram med en variabel
Hvordan kan vi visualisere hvordan fordelingen av politisk interesse er? Her kan vi bruke geom_bar for å lage et søylediagram. Et søylediagram viser antallet enheter innenfor hver verdi.

```
ggplot(data = df, aes(x = interest)) + geom_bar()
```

Søylediagram med to variabler
Hvor mange innenfor hvert nivå av politisk interesse stemte? Vi kan bruke geom_bar igjen, men vi sier at vi også vil se fordelingen av hvordan respondentene stemte innenfor hvert nivå av politisk interesse med (aes(fill=vote)). Så sier vi at vi vil at det skal være en søyle for de ulike alternativene for vote med position = dodge.

```
ggplot(data = df, aes(x = interest)) + geom_bar(aes(fill=vote),
                                position = "dodge")
```

Histogram
Hvordan fordeler respondentenes alder og tiden de bruker på nyheter seg? Disse variablene er kontinuerlige, så vi kan bruke geom_histogram for å lage et histogram for hver variabel.

```
ggplot(data = df, aes(x = news)) +
  geom_histogram(bins = 5)

ggplot(data = df, aes(x = age)) +
  geom_histogram(binwidth = 10)
```

Et histogram viser hvor mange enheter det er i hver kategori. Vi kan enten spesifisere hvor mange søyler vi vil ha (bins) eller hvor stor hver søyle skal være (bindwidth).

Spredningsplott
Hvordan fordeler tiden man bruker på nyheter på alder? Her kan vi bruke geom_point for å lage et spredningsplott.

```
ggplot(data = df, aes(x = age, y = news)) +
 geom_point(alpha = 0.2)
```

I et spredningsplott viser hvert punkt verdiene til en observasjon for de to variablene som blir presentert. Man kan bruke spredningsplott for å vise sammenhenger og hvordan dataene grupperer seg i datasettet.

Boksplott
Hvordan fordeler alder seg på interesse? Vi kan lage et boksplott med geom_boxplot. Et boksplott kan vise hvordan en kontinuerlig variabel (alder) er fordelt innenfor en annen kategorisk variabel (politisk interesse). Boksen i midten representerer spennet til de 50% vanligste verdiene (andre og tredje kvartil), mens strekene viser spennet til verdiene i nedre og øvre kvartil.

```
ggplot(data = df, aes(x = interest, y = age)) +
 geom_boxplot()
```

Hvis dere vil utforske hvordan man kan tilpasse de ulike diagrammene vi har sett på og mange andre, kan denne siden være nyttig: https://www.r-graph-gallery.com/index.html