Laster inn datasettet. Her bruker jeg funksjonen read_dta fra haven pakken. Velg den funksjonen som passer til fil-formatet til datasettet. Er det en CSV-fil, så bruker vi eksempelvis read_csv.

ESS8SE <- read_dta("ESS8SE.dta")

Alltid lurt å ta en titt på datasettet for å få en oversikt over hvordan datamaterialet er organisert. Dette gjør vi for å se hva slags struktur datasettet vårt har, er det tverrsnittsdata, tidsserie eller paneldata? Dersom vi jobber i store datasett er det lurt å bruke slike funksjoner istedenfor å «trykke» på datasettet da dette krever mye fra pc'en. Dersom vi ønsker å få en oversikt over alle variablene i datasettet er det viktig/lurt å bruke kodeboken til dette.

head(ESS8SE)

Vi har mange variabler som ikke gir substansiell mening, eksempelvis denne variabelen. For å titte på enkelt variabler bruker vi syntaksen datasett$variabel. Det kan også være praktisk å se denne informasjonen i en tabell. Da kan vi bruker table(datasett$variabel).

ESS8SE$prtvtdes

Vi lager et subset av det opprinnelige ESS datasettet, da tar vi kun med de variablene vi vil ha til resten av analysen. Dette er ikke nødvendig i streng forstand, men vi gjør det for å gjøre datasettet litt mer oversiktlig.

Vi bruker select() fra dplyr for å velge ut variablene vi vil ha med i datasettet.

Husk også at det er lurt å lagre det nye datasettet i et nytt objekt, slik at vi ikke overskriver det originale datasettet.

ess <- ESS8SE %>%

select(gndr, agea, eduyrs, nwspol, stfdem, vote)

Det er alltid lurt å sjekke hvor mye missing vi har i data. Da bruker vi table() og complete.cases(), FALSE viser til hvilke enheter som har NA på minst en variabel, TRUE viser til enhetene som har verdier på alle variabler

table(complete.cases(ess))

Vi bruker funksjonen na.omit() til å kaste ut enheter med missing verdier. Å kaste ut missing er en mulig strategi, men dette bør vi alltid tenke igjennom. Dette gjelder spesielt om data har mye missing verdier.

ess <- na.omit(ess)

Vi tar en titt på avhengig variabel til regresjonsanalysen. Denne variabelen måler hvor fornøyd respondenten er med demokratiet målt på en skala fra 0-10.

str(ess$stfdem)

summary(ess$stfdem)

Videre tar vi en titt på noen av de uavhengige variablene.

Variabelen gndr er kodet 1 = mann og 2 = kvinne.

table(ess$gndr)

Vi vil kode om denne slik at mann = 0 og kvinne = 1. Dette er en noe mer intuitiv koding av denne variabelen. Det kan også være lurt å tenke over hvilke forventninger vi har til variabelen, her forventer vi for eksempel at kvinner er mer fornøyde med demokratiet enn menn. Da kvinne skal kodes som 1 vil vi i så fall få en positiv koeffisient dersom forventningen innfris.

Vi bruker mutate til å opprette en ny variabel for kjønn, transmute vil erstatte den gamle variabelen med den nye omkodede. Jeg vil anbefale å bruke mutate som hovedregel, da slipper vi å gå tilbake til originaldata dersom noe går galt.

```
ess <- ess %>%
  mutate(kjønn = if_else(gndr == 1, 0, 1))
```

Vi bruker if_else inne i mutate for å kode om variabelen. Koden ovenfor sier: dersom gndr har verdier 1, skal den nye variabelen få verdien 0, dersom gndr har andre verdier får disse verdien 1 i den nye variabelen.

Vi sjekker hvordan vote variabelen er kodet. Denne variabelen har informasjon om respondenten stemte ved forrige valg, og er kodet ja, nei og ikke stemmeberettiget.

table(ess$vote)

Vi vil ha en dikotom variabel som fanger hvorvidt personen stemte ved siste valg eller ikke. Vi vil har de som ikke var stemmeberettiget sammen med de som ikke stemte i forrige valg. Dvs, kodet som nei.

```
ess <- ess %>%
  mutate(vote2 = if_else(vote >= 2, 0, 1))
```

Sjekk ?if_else() for hjelpefilen til denne funksjonen

Koden sier at respondentene som har verdien 2 eller høyere på vote skal ha verdien 0 i den nye variabelen. De som har verdier under 2 får verdien 1 på den nye variabelen.

Sjekk omkodingen med table(), vi kan også sammenligne den omkodede variablene med den originale variabelen ved å bruke table(ess$vote). På den måten kan vi se at omkodingen er riktig.

table(ess$vote2)

Vi endrer navnene på variablene, for å gjøre det hele mer oversiktlig. Dette er helt valgfritt, men dere vil ofte oppleve at variablene i diverse datasett har navn som er lite intuitive.

Vi bruker rename funksjonen som er en enkelt kode for å endre navn. Sjekk ?rename for mer informasjon.

```
ess <- ess %>%
  rename(alder = agea,
         utdanning = eduyrs,
         nyheter = nwspol,
         demokrati = stfdem)
```

Vi kan sjekke at omkodningen og navnebyttene er korrekte ved å bruke head funksjonen

head(ess)

Kjøre OLS regresjonen, vi bruker funksjonen lm for å kjøre en OLS i R

For å legge til uavhengige variabler bruker vi +, avhengig variabler kommer først og spesifiseres med ~

Vi forteller R hvilket datasett vi vil bruke med data =

Vi lagrer også modellen i et objekt, slik at vi kan bruke verdiene senere

```
mod1 <- lm(demokrati ~ vote2 + nyheter + utdanning +
```

```
        alder + kjønn, data = ess)
```

Vi bruker summary for å se resultatene av modellen vår.

```
summary(mod1)
```

Her har jeg lagt til ett samspillsledd mellom stemmegivning og konsum av politiske nyheter.

En måte å legge til samspill på er ved å bruke * mellom samspillsvariablene. Det er også mulig å opprette en ny variabel med mutate hvor man spesifiserer samspillsleddet.

```
mod2 <- lm(demokrati ~ vote2 * nyheter + utdanning +
        alder + kjønn, data = ess)
```

Legg merke til at vi for koeffisienter for både samspillsleddet og koeffisienter for variablene i samspillet individuelt. Dette er viktig informasjon når vi skal tolke resultatene fra en modell med samspill.

Her plotter vi regresjonslinjen for utdannings variabelen vår. Det er viktig å velge plot etter hvilke variabler vi er interesserte i å visualisere. Her har vi to kontinuerlige variabler som er rimelig enkelt å plotte. Dersom vi vil plott dummyvariabler eller kategoriske variabler må vi finne andre plots. En jukselapp til ggplot finnes her: https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

```
ggplot(ess,
    aes(x = utdanning,
        y = demokrati)) +
 stat_smooth(method = "lm", col = "red")
```

Videre skal vi se på noen grafiske verktøy for å vurdere om enkelte forutsetninger for OLS er oppfylt

Da må vi først lagre restleddene og verdiene fra modellen vår i datasettet

Dette kan også gjøres med mutate, men her velger vi å bruke en enkelt kode fra base.

Vi bruker resid funksjonen for å trekke ut restleddene fra modell 1. fitted.values ligger allerede innbakt i modellobjektet, derfor kan vi enkelt trekke disse ut med $ og lagre disse i datasettet vårt.

ess$mod1Resid <- resid(mod1)

ess$mod1Fitted <- mod1$fitted.values

Vi vil så vurdere restleddenes fordeling med et histogram, er restleddene våre normalfordelte?

Her bruker vi restleddene fra mod 1 som vi la inn i datasettet vårt med koden ovenfor.

```
ggplot(ess,
    aes(x = mod1Resid)) +
 geom_histogram()
```

Nå skal vi forsøke å lage en figur som plotter restleddene mot modellens verdier

Dette gjør vi for å vurdere eventuell hetroskedastisitet

Vi brukerverdiene som i lagret i datasettet på x aksen og modellens restledd på y aksen. Vi legger til de ulike enhetene med geom_point, så trekker vi en linje gjennom punktene med geom_smooth.

```
ggplot(ess,
    aes(x = mod1Fitted,
       y = mod1Resid)) +
 geom_point() +
 geom_smooth()
```

Slike plott kan være noe vanskelig å tolke, hvor enkel eller vanskelig tolkningen blir avhenger ofte av hvordan variablene våre er kodet.

Vi kan også bruke plot() for å få ulike figurer for diagnosistikk, vi vil få 4 ulike plot med denne funksjonen. Disse plottene er ikke like fine som de vi lager i ggplot, men de kan hjelpe oss med å få en rask oversikt over ulike diagnosistikk

plot(mod1)

Vi lager en fin regresjonstabell med begge modellene våre side ved side. Stargazer pakken kan bruken til svært mye i R. Husk å sjekke hjelepfilen ?stargazer. Vi bruker type = til å spesifisere hvilket format vi vil ha tabellen i. covariate.labels bruker til å legge til nye navn til de uavhengige variablene i modellen. Det er viktig å legge inn navnene i samme rekkefølge som i regresjonsmodellen, vist ikke risikerer vi å gi feil navn til variablene. Dep.var.labels bruker til å gi navn til avhengig variabel.

```
stargazer(mod1, mod2,
        type = "text",
        title = c("Modeller"),
        covariate.labels = c("Evne til politisk deltakelse",
                             "Politiske nyheter",
                             "Utdanning",
                             "Alder",
                             "Kjønn",
                             "Samspill"),
        dep.var.labels = c("Tilfredshet med demokratiet"))
```

Vi ønsker ofte å bruke tabellene våre i eksempelvis word

Vi kan lagre tabellen som en html-fil og deretter bruke den i eksempelvis word

Dette gjør vi ved å sette type = "html",

Deretter legger vi til argumentet out = "navnpåtabell.htm"

```
stargazer(mod1, mod2,
        type = "html",
        title = c("Modeller"),
        covariate.labels = c("Evne til politisk deltakelse",
                             "Politiske nyheter",
                             "Utdanning",
                             "Alder",
```

```
            "Kjønn",
            "Samspill"),
    dep.var.labels = c("Tilfredshet med demokratiet"),
    out = "regtabell.htm")
```

Tabellen vi da lagers i working directory. Vi kan da bruke «åpne i» i mappen hvor filen ligger – og velge åpne i word.