Seminar 3

Eric

1 9 2020

Velkommen tilbake!

I dag skal vi gå igjennom paneldatanalyse. Så langt har vi kunn sett på tversnittsdata, dette betyr altså at vi studere flere enheter på ett tidspunkt. Det fine med paneldata er at vi kan følge de samme enhetene, over lengre tid! På den måten er det mye lettere å se etter former for kausalitet, og du får ofte sterkere konklusjoner enn med tversnitt alene. Paneldata gjør at vi f.eks. kan sjekke om Y faktisk endrer seg **etter** X, noe som jo gir sterkere tegn til faktisk kausalitet!

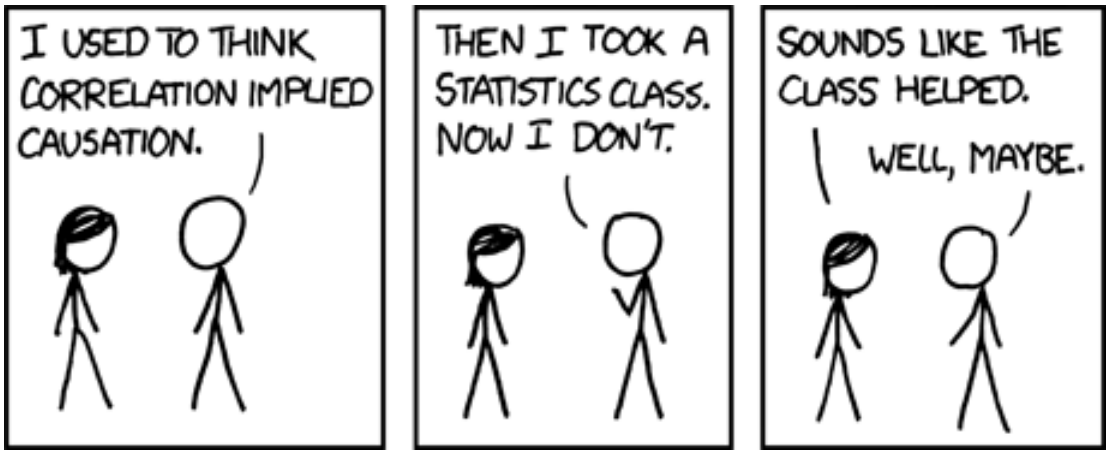


Figure 1:

Samtidig kan paneldata gi oss noen utfordringer. Dere husker kanskje fra forelesning at OLS har en forutsetning om at det ikke skal være autokorrelasjon i restleddene. I tversnittsdata vil denne forutsetningene alltid være oppfylt, men dette er *ikke* gjeldende for tidsserie data! Demokrati i t vil åpenbart være korrelert med demokrati i *t-1*, og dette vil skape problemer for oss. Derfor må vi være nøyere med hvordan vi lager modellene våre når vi har denne typen data.

Data-Wrangling og paneldata

Før vi kommer så langt som å lage modeller derimot, må vi faktisk ha noe data å modellere på! Når vi ser på de orginale dataene våre ser vi at vi har data for alle individer, i alle runder, i alle land som har vært med i ESS. I paneldata ønsker vi å følge den samme enheten over tid, så vi må derfor bestemme oss for hva denne enheten skal være, og hvordan vi skal aggregere til dette nivået. For å gjøre det enkelt kommer vi her til å skape dataene på land-runde nivå, og bruke gjennomsnittet på variablene som aggregeringsform. Skulle vi brukt dataene til mer ordentlig forskning burde vi likevell brukt mer tid på vurdere hva dette betyr, kan det være noen problemer med denne måten å aggregere på?

La oss se på hvordan vi kan gjøre dette i kode. Som vanlig må vi først laste inn dataene, og de pakkene vi ønsker å bruke.

```
ESS_Data <- read.csv("https://raw.githubusercontent.com/egen97/4020A_RSeminar/master/ESS_Selected.csv")
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
## v ggplot2 3.3.1     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Nå som vi har datene kan vi begynne å se på to funksjoner som kommer ifra tidyverse, `group_by` og `summarise`. Les gjerne hjelpefilen, og se om du kan forstå hva de gjør derifra, før du leser videre! Ofte når vi jobber med data ønsker vi å forandre noe basert på grupper de definert ut fra andre variabler. Her f.eks. ønsker vi å finne gjennomsnittet til noen variabler, men få frem et gjennomsnitt for hvert land! `group_by` gjør at vi kan definere nettopp disse gruppene, og lar andre funksjoner etter den jobbe gruppevis. Her kommer vi til å bruke denne nettopp for å sørge for at vi får ett gjennomsnitt for hver verdi pr. land-runde.

`Summarise` tar, logisk nok, og gir en oppsumering av en variabel. Her har vi flere individer som har svart på et spørsmål, og vi ønsker å finne en verdi for hele landet. Dette kan selvfølgelig oppsumeres på flere forskjellige måter, vi kunne valgt den vanligste verdien, median, gjennomsnitt, eller kanskje bare summen av alle svarene? Hvordan det er best å gjøre det vil avhenge av hva du ønsker å finne ut av, og ofte variere fra variabel til variabel. Her må du ta et aktivt valg av hva som passer best etter dine behov, og hva du ønsker å finne ut av. La oss nå prøve å gjøre dette med datene våre!

```
ESS_CY <- ESS_Data %>%
  group_by(Country, essround) %>%
  summarise_all(mean, na.rm = TRUE)
```

La oss ta en titt på de nye dataene våre, ser vi noen problemer? I all hovedsak er det tre utfordringer vi støter på i dette nye datasettet:

1. Flere variabler gir ikke nødvendigvis mening på land-runde nivå. Siden kun Norge har informasjon om hvilket parti de har stemt på, hva beyr denne variabelen nå? Hva kan vi bruke den til? Hva betyr kjønnsvariabelen nå som den har verdier mellom 1. og 2.?
2. Flere observasjoner har nå fått verdien `NaN`, hva betyr dette? `NaN` står for "Not a Number", og er en type missing verdi som kommer når R får noe annet enn et tall et sted den forventer en numerisk verdi. Her er problemet at spørsmålet ikke var med i en runde eller ett land, og derfor ender opp med *0/0* når vi prøver å regne ut gjennomsnittet. Om dere prøver dette i konsollen vil dere se at resultatet er nettopp `NaN`. For vår del ønsker vi nok at `NaN` bare skal registreres som `NA`, så vi bør derfor kode om dette.
3. Dataene våre er ikke balanserte. Altså har vi et ulikt antall runder, for hvert land som er med. Når vi har dette er det viktig å vite hvorfor noen land-runder er `NA`, og hvorvidt det er fullstending tilfeldig (MCAR), kun avhengig av variabler vi vet om (MAR), eller ikke-tilfeldig missing (MNAR). Det siste betyr at det er en variabel vi ikke vet om som **gjør** at variablene er missing, og om det er tilfellet kan vi ikke bruke dataene. Uavhengig av dette vil vi ofte ønske å balansere data sånn at vi vet hvor vi har `NA`, og dette er tydelig i datasettet.

Variabler på land-år nivå

1.

La oss først tenke over hvilke variabler vi ønsker å ha med på dette nivået. Kjønn var fra før kodet slik at 2 = dame. Når vi nå har aggregert dataene, sitter vi igjen med et tall mellom 1 og 2 for hver land-runde.

Kan vi tolke dette på noen måte? Om vi trekker ifra 1 for hver nivå, sitte vi igjen med et tall mellom 0 og 1. Dette kan vi faktisk tolke som % kvinner i hvert land-runde! Dette kan jo muligens være nyttig i senere analyser. Det samme stemmer for de ande dikotome variablene, sånn som hvorvidt de stemte.

```
ESS_CY <- ESS_CY %>%
  mutate(Gender = Gender - 1,
         vote = vote - 1)
```

Er det andre variabler som vi bør tenke over? Gjennomsnittelig inntekt er ofte ikke veldig interesant, på land-nivå har vi bedre mål for økonomi, og gjennomsnittet er ofte betydelig høyere enn det en "vanlig" person kan forvente å tjene. Dette kan vi nok dermed fjerne.

2.

Som vi kan se har flere celler fått verdien `NaN`, hva kan vi gjøre med dette? På en eller annen måte ønsker vi å gjøre slik at alle som har fått denne verdien heller får `NA`. Det gjør det lettere å bruke variablene i modeller og slikt senere. Det er flere måter vi kan gjøre det på, en mulighet er å bruke `mutate()`på alle variablene, og en logisk test med `is.nan()`for å undersøke om de er `NaN`. Samtidig vil dette gjøre at vi får veldig mye kode som gjør den samme oppgaven, og kan gjøre scriptet vårt litt rotete. Heldigvis finnes måter vi kan gjøre det lettere på! `apply()` er en gruppe funksjoner som gjør at vi kan kjøre den samme funksjonen over flere variabler. Dette kan f.eks. være kolloner i et datasett sånn som her, eller forskjelige datasett i en liste. Her kan vi ta en rask titt på funksjonen `sapply`, som er en forenklet versjon av `apply()`. For å bruke denne må vi ha datasettet, en liste over hvilke variabler den skal kjøre funksjonen på (her alle), og hva resultatet skal være. I kode vil se sånn her ut:

```
ESS_CY[sapply(ESS_CY, is.nan)] <- NA
```

Her sier vi at for alle kolloner i ESS_CY dataene skal den kjøre funksjonen `is.nan`, hvis den gir verdien TRUE vil disse cellene så får verdien `NA`. Dette er kanskje litt komplisert å forstå, så ikke stress hvis du synes det er forvirrende! Hjelpefilen forklarer ganske greit hvordan det fungerer, i tilegg til at det er masse på internett om det. I tilegg tar jo selvfølgelig jeg og svarer på spørsmål!

3. "Balansere" data

Det siste vi har lyst til å gjøre er å "balansere" dataene, altså sørge for at alle tversnittsenheter (land) har like mange tidsenheter (runder). Dette gjør det mye enklere å se hvor vi har missing, og hvorivdt det er noen problemer med ubalanseheten. Merk at jeg her sier "balansere" i hermetegn, det vi gjør nå gir oss ikke mer informasjon og dataene er derfor fortsatt ubalanserte. Det vi gjør er å sette inn missing slik at vi kan se nettopp det at det er ubalansert, og hvor vi mangler observasjoner på landene. Skulle vi ønske å reelt balansere et datasett måtte vi sett å metoder som imputering, og lignende.

For å gjøre dette skal vi ta i bruk en ny pakke som heter `plm`. Denne pakken er laget for å arbeide med paneldata, og har flere funksjoner som gjør det lettere å arbeide med disse. Som vanlig må vi først innstallere pakken.

```
#install.packages("plm")
library("plm")
```

```
##
## Attaching package: 'plm'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, lag, lead
```

Funksjonen vi skal bruke nå heter `make.pconsecutive`, denne fyller ut gaps for tversnittsenhetene, og gjør i tilegg at vi kan få det balansert. La oss se på hvordan denne fungerer.

```
ESS_Bal <- make.pconsecutive(ESS_CY, balanced = TRUE) #Legg merke til at jeg må definere
```

```
## Warning in mde(x): NAs introduced by coercion
```

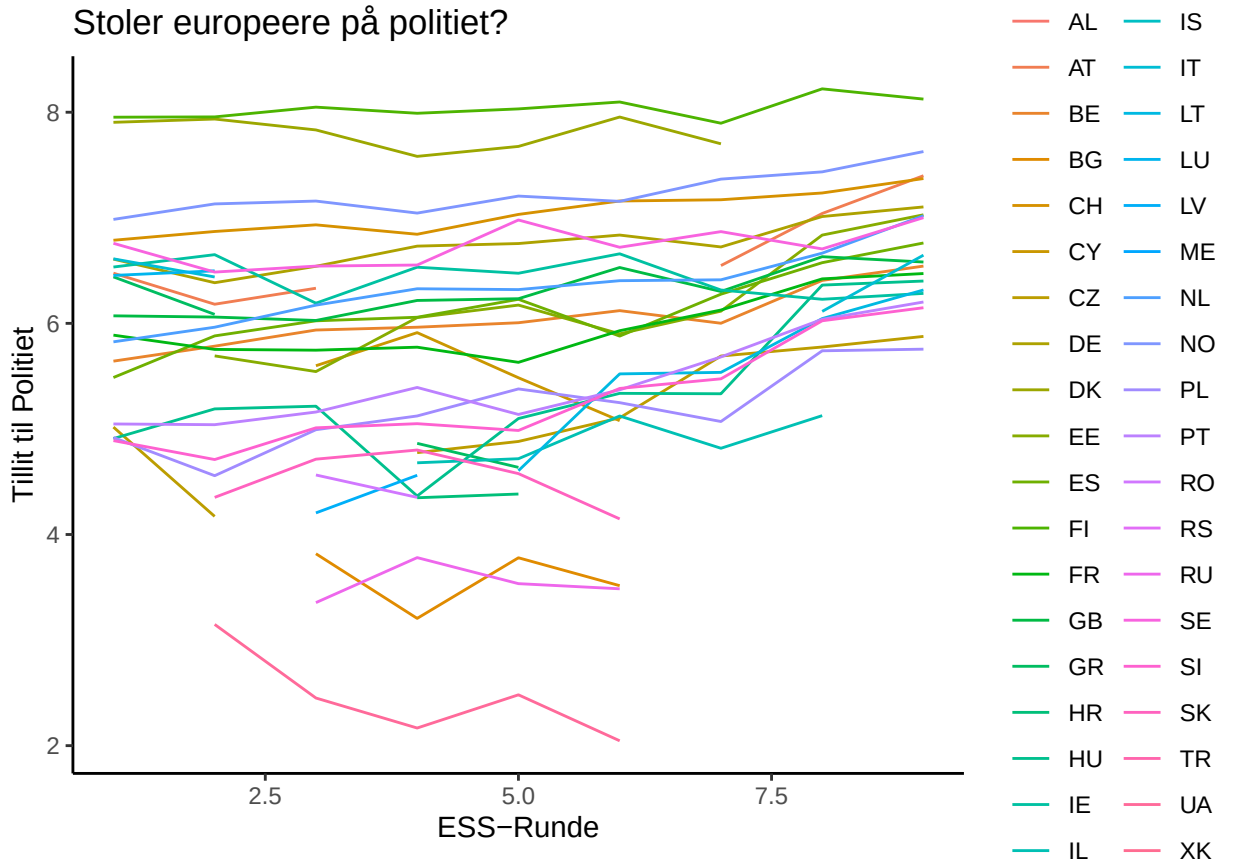
```
                                                    #at den også skal balansere
```

Som vi kan se har vi nå fått noen nye observasjoner, som kun innholder NA. Dette er altså for de rundene som det respektive landet ikke har vært med i.

Paneldata analyse: Hva kan vi gjøre med tid?

Nå som vi har tidsdimensjonen kan vi gjøre flere analyser som gjør bruk av denne. La oss først se om noen variablene forandrer seg over tid. La oss se f.eks. om tilliten til politiet har forandret seg:

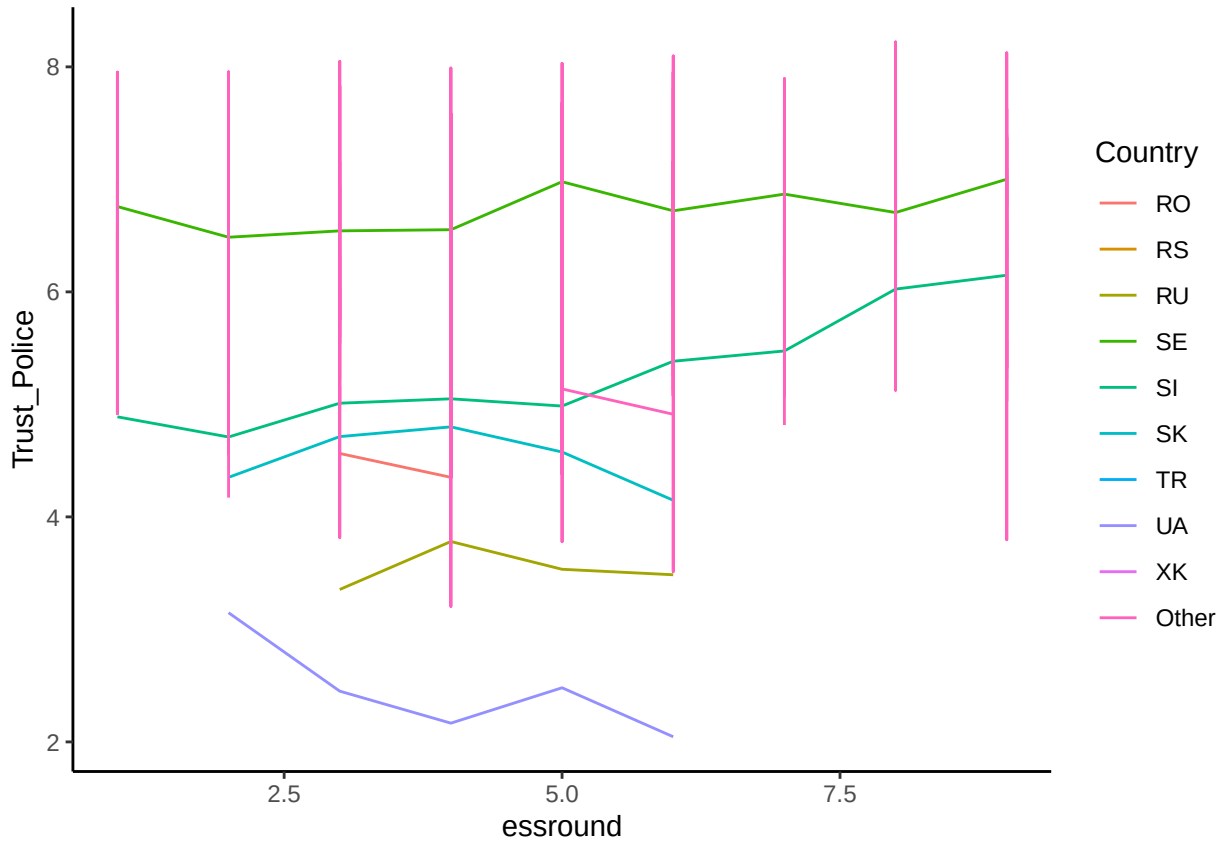
```
ggplot(ESS_Bal, aes(essround, Trust_Police, colour = Country)) +
  geom_line() +
  theme_classic() +
  labs(title = "Stoler europeere på politiet?",
       y = "Tillit til Politiet",
       x = "ESS-Runde")
```



Med en gang kan dette plottet kanskje være litt vanskelig å tolke dette plottet. Ofte er det lettere å plukke ut noen få enheter vi er innteresert i, og sammenligne disse med "andre". La oss se om vi kan få til det istedet. For å slå sammen enheter kan vi bruke `fct_lump` funksjonene.

```
ESS_Bal %>%
  mutate(Country = fct_lump_n(Country, 9, ties.method = "last")) %>%
```

```
ggplot(aes(essround, Trust_Police, colour = Country)) +
geom_line() +
theme_classic()
```

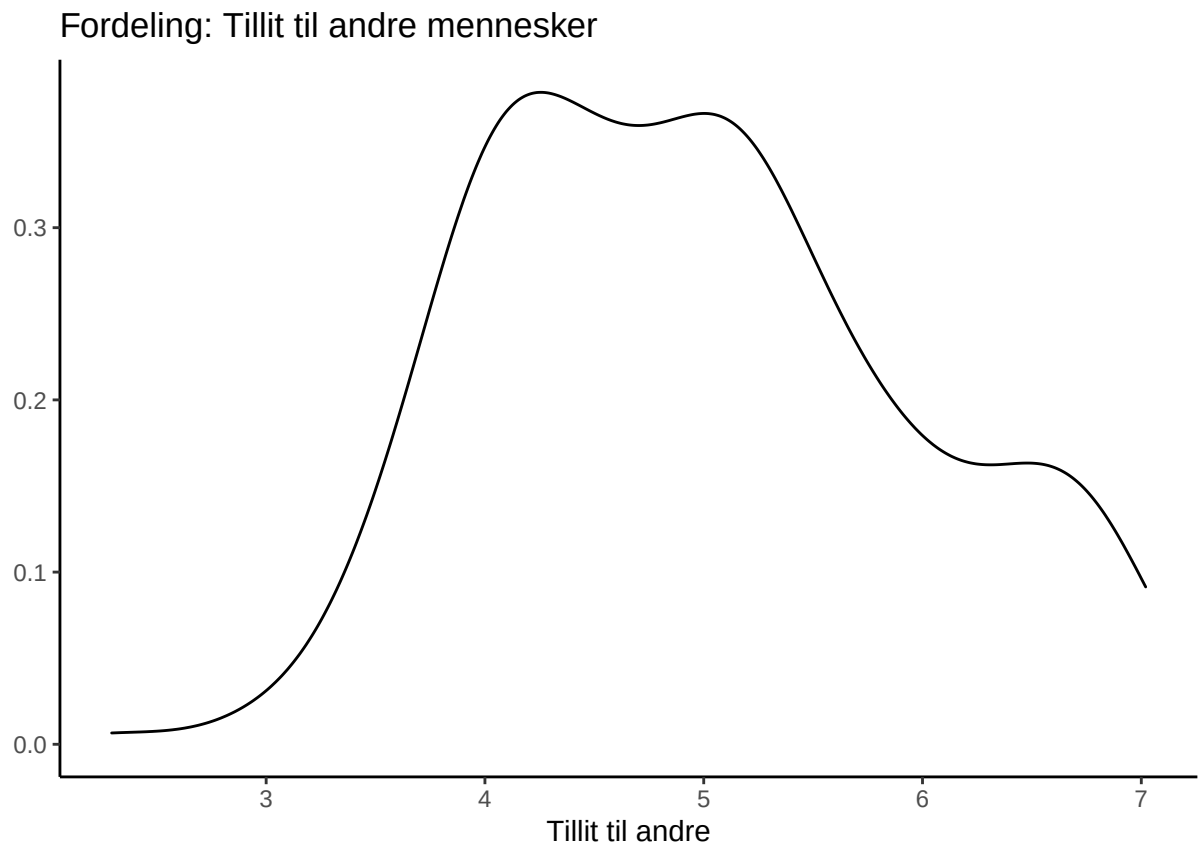


Her ser vi med en gang at det er lettere å lese plottet!

Variabler, og form

Fra før har vi jo sett på hvordan vi kan se på fordelingene til forskjellige variabler. Den letteste er jo selvfølgelig ved å plotte variablene, og se på formen selv. La oss ta en titt på tillitt til andre. Fra forrige seminar husker vi at vi kan finne formen på denne gjennom et density-plot.

```
ggplot(ESS_Bal, aes(Trust_People)) +
  geom_density() +
  theme_classic() +
  ggtitle("Fordeling: Tillit til andre mennesker") +
  xlab("Tillit til andre") +
  ylab("")
```



Samtidig kan vi ofte få lyst til å få mer nøyaktige, og tallfestede svar på hvordan en variabel er fordelt. La oss derfor se på om vi kan få ut verdien på ting som kurtose, og skjevhet! For å gjøre dette skal vi ta i bruk pakken `moments`. Som vanlig må vi først innstalere denne.

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

`Moments` gir oss to funksjoner som hendig nok heter `kurtosis`, og `skewness` som hendholdsvis gir oss pearsons mål på kurtose, og skjevheten til en variabel. Denne kan vi nå prøve på den samme variabelen!

```
skewness(ESS_Bal$Trust_People, na.rm = TRUE) #Som vanlig må vi ha et NA argument!
```

```
## [1] 0.2829227
```

```
kurtosis(ESS_Bal$Trust_People, na.rm = TRUE)
```

```
## [1] 2.424121
```

Her ser vi altså at fordelingen er noe høyreskjev, og med en høy kurtose. Begge er heldigvis innenfor et nivå hvor det ikke har spesielt mye å si for senere analyse!

Regresjoner!

Nå som vi har et datasett på land-runde nivå kan vi begynne med å analysere dette. Først kan vi prøve ut en vanlig OLS model, la oss se om vi kan finne noen sammenheng mellom tillit til andre, og politisk interesse, mens vi kontrolerer for alder.

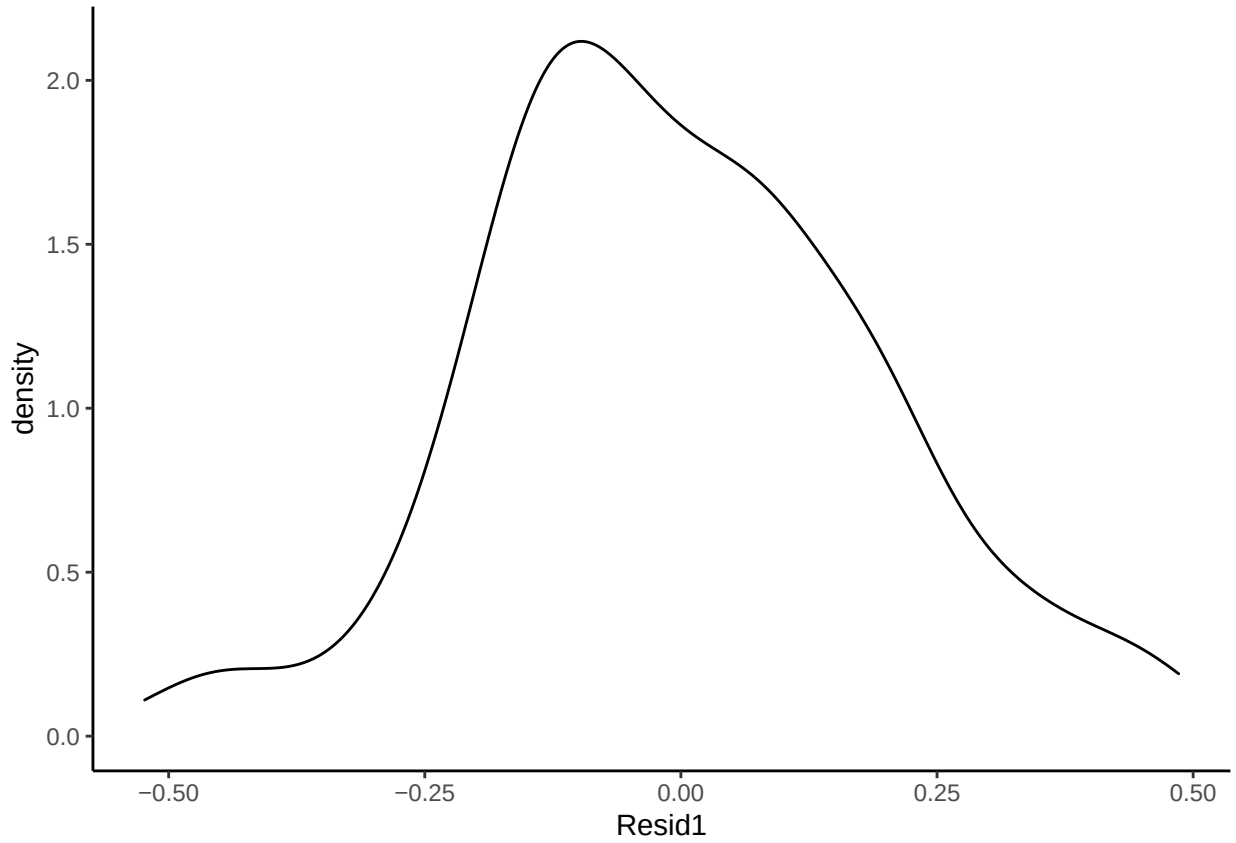
```
Mod1 <- lm(Pol_Interest ~ Trust_People + Age, na.action = "na.exclude", data = ESS_Bal)
stargazer::stargazer(Mod1, type = "text") #Her kan dere se jeg skrev stargazer:: før funksjonen.
```

```
##
## ================================================
##                          Dependent variable:
##                      ---------------------------
##                             Pol_Interest
## ------------------------------------------------
## Trust_People                 -0.171***
##                               (0.013)
##
## Age                           -0.001
##                               (0.005)
##
## Constant                     3.524***
##                               (0.236)
##
## ------------------------------------------------
## Observations                    229
## R2                             0.423
## Adjusted R2                    0.418
## Residual Std. Error      0.194 (df = 226)
## F Statistic           82.722*** (df = 2; 226)
## ================================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

```
                                        #Det kan vi gjøre istedenfor å kjøre library()
```

Hvordan kan vi tolke denne? Vel, vi har en sterkt signifikant negativ effekt, og en moderat $R\hat{}2$. Altså bør jo modellen være ganske bra. Alikevell, kan det være problemer her? La oss først prøve å undersøke resttleddene til modellen!

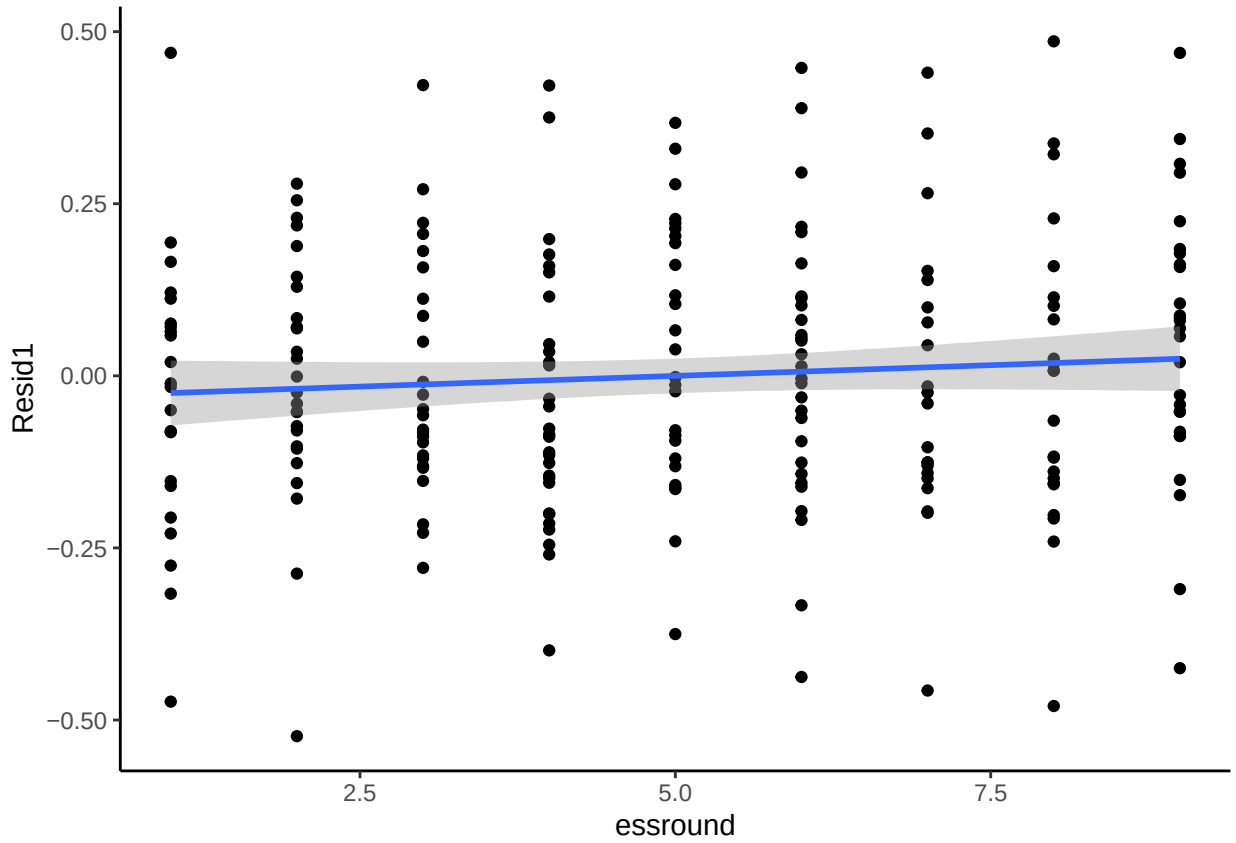
```
#Først må vi legge til restleddene i datasettet

ESS_Bal$Resid1 <- resid(Mod1)
#Så kan vi prøve å plotte disse

ggplot(ESS_Bal, aes(Resid1)) +
  geom_density() +
  theme_classic()
```



Her ser vi først at restleddene faktisk er ganske normalfordelte, hvertfall innenfor hva vi ville sagt er akseptabelt. Kanskje er modellen grei? En annen ting vi kan teste her er om vi har autokorrelasjon i restleddene, dette er et veldig vanlig problem med paneldata! For å gjøre det plotter vi restleddene opp mot tidsenheten vår.

```
ggplot(ESS_Bal, aes(essround, Resid1)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_classic()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Som vi kan se i våre data har vi faktisk ikke noe særlig til autokorrelasjon! Samtidig ser vi at vi har noen outliers, spesielt på den negative siden. Modellen ser derfor ut til å være ganske ok.

Tilbake til modellen kan vi se at vi har en *R^2* som er på ca. *0,5*, denne ønsker vi ofte å ha noe høyere. La oss se om vi kan forbedre modellen ved å legge til noen variabler, og ha FE (fixed effects) på år. Å gjøre FE er faktisk veldig enkelt i `R`! Eneste er at vi må tenke litt mer over hvordan vi tolker resultatene våre! Fra første seminar husker vi at vi har en klasse som heter `factor`, det fine med denne er at om vi legger inn en slik variabel i modellen vil R automagisk lage en dummy for hver av kategoriene, med den føste alfabetisk som referansenivået. Så fort vi gjør dette har vi altså et konstantledd pr. år. La oss legge til alder og FE som variabler, og se hvordan det hjelper modellen.

```
Mod2 <- lm(Pol_Interest ~ Trust_People + Age + as.factor(Country),
           na.action = "na.exclude", data = ESS_Bal)
stargazer::stargazer(Mod2, type = "text")
```

```
##
## ================================================
##                          Dependent variable:
##                      ---------------------------
##                              Pol_Interest
## ------------------------------------------------
## Trust_People                    -0.041
##                                 (0.037)
##
## Age                             -0.007*
##                                 (0.004)
##
```

```
## as.factor(Country)AT           -0.226
##                                (0.140)
##
## as.factor(Country)BE           -0.022
##                                (0.135)
##
## as.factor(Country)BG            0.015
##                                (0.121)
##
## as.factor(Country)CH           -0.287*
##                                (0.151)
##
## as.factor(Country)CY            0.118
##                                (0.122)
##
## as.factor(Country)CZ          0.372***
##                                (0.127)
##
## as.factor(Country)DE          -0.424***
##                                (0.133)
##
## as.factor(Country)DK          -0.405**
##                                (0.185)
##
## as.factor(Country)EE            0.028
##                                (0.146)
##
## as.factor(Country)ES            0.220
##                                (0.134)
##
## as.factor(Country)FI           -0.098
##                                (0.175)
##
## as.factor(Country)FR           -0.068
##                                (0.126)
##
## as.factor(Country)GB           -0.129
##                                (0.141)
##
## as.factor(Country)GR           0.232*
##                                (0.123)
##
## as.factor(Country)HR           0.291**
##                                (0.132)
##
## as.factor(Country)HU            0.177
##                                (0.123)
##
## as.factor(Country)IE            0.016
##                                (0.143)
##
## as.factor(Country)IL           -0.149
##                                (0.140)
##
```

```
## as.factor(Country)IS          -0.360**
##                                (0.172)
##
## as.factor(Country)IT           0.164
##                                (0.131)
##
## as.factor(Country)LT           0.281**
##                                (0.137)
##
## as.factor(Country)LU           -0.053
##                                (0.152)
##
## as.factor(Country)LV           0.111
##                                (0.132)
##
## as.factor(Country)ME          0.471***
##                                (0.152)
##
## as.factor(Country)NL           -0.287*
##                                (0.156)
##
## as.factor(Country)NO           -0.138
##                                (0.177)
##
## as.factor(Country)PL           0.010
##                                (0.118)
##
## as.factor(Country)PT           0.226*
##                                (0.119)
##
## as.factor(Country)RO           0.127
##                                (0.134)
##
## as.factor(Country)RS           0.387**
##                                (0.156)
##
## as.factor(Country)RU           -0.047
##                                (0.124)
##
## as.factor(Country)SE           -0.309*
##                                (0.164)
##
## as.factor(Country)SI           0.011
##                                (0.121)
##
## as.factor(Country)SK           0.038
##                                (0.121)
##
## as.factor(Country)TR           0.074
##                                (0.131)
##
## as.factor(Country)UA           -0.093
##                                (0.125)
##
```

```
## as.factor(Country)XK            0.171
##                                (0.152)
##
## Constant                      3.208***
##                                (0.218)
##
## ------------------------------------------------
## Observations                     229
## R2                              0.856
## Adjusted R2                     0.827
## Residual Std. Error       0.106 (df = 189)
## F Statistic           28.852*** (df = 39; 189)
## ================================================
## Note:                *p<0.1; **p<0.05; ***p<0.01
```

Her ser vi at vi får en uavhengig variabel pr. land, dette blir fort en vanskelig tabell å lese! Derfor kand det være greit å definere hvilke variabler vi faktisk ønsker å se. Heldigivs kan vi gjøre dette i stargazer.

```
stargazer::stargazer(Mod2, type = "text",
                     title = "Politisk Tillit",
                     keep = c("Trust_People", "Age"), #Her sier jeg hva jeg vil ha med
                     covariate.labels = c("Tillit til andre", "Alder"),
                     dep.var.labels = "Politisk Interesse") #Her kan jeg gi nye navn
```

```
##
## Politisk Tillit
## ================================================
##                          Dependent variable:
##                      ---------------------------
##                          Politisk Interesse
## ------------------------------------------------
## Tillit til andre              -0.041
##                               (0.037)
##
## Alder                         -0.007*
##                               (0.004)
##
## ------------------------------------------------
## Observations                    229
## R2                             0.856
## Adjusted R2                    0.827
## Residual Std. Error      0.106 (df = 189)
## F Statistic          28.852*** (df = 39; 189)
## ================================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

Som vi kan se ble modellen vår noe bedre på å predikere, med $R\hat{}2$ økt til *0,8*, men tilitt til andre er ikke lenger signifikant som en variabel. Hva kan dette bety?

En siste ting vi kan sjekke på denne er hvorvidt vi har multikollinaritet. Det betyr veldig enkelt at de uavhengige variablene korrelerer høyt med hverandre, noe som kan skape problemer for en OLS modell. For å undersøke dette kan vi bruke en funksjon fra pakken `car` som heter `vif`. Denne viser oss *variansinflasjonsfaktoren*, som viser nettopp multikolinaritet i OLS modeller. En grei tommelfingerregel er at *VIF* under 5 ikke utgjør et problem, mellom 5 og 10 er i grenseland av hva vi kan godta, og over 10 utgjør et problem. La oss teste dette på vår modell, først må vi innstalere car, og så kan vi kjøre denne funksjonen på modellen vår. Når vi har FE blir denne noe annerledes enn vanlig, for enkelhetens skyld gjør vi derfor dette på den første modellen.

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

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```r
vif(Mod1)
```

```
## Trust_People          Age
##     1.000012     1.000012
```

Som vi kan se er begge såvidt over 1, og dette utgjør derfor ikke noe større problem.

Det var det for idag!