Seminar 2

Eric

30 8 2020

Velkommen til seminar 2!

I dette seminaret skal vi gå igjennom litt mer data-wrangling, før vi skal gå over på å se hvordan vi kan gjøre en OLS-regresjon! Før vi kan gjøre dette må vi først se hvilke variabler vi ønsker å undersøke, og sørge for at de er i en sånn form at vi kan bruke dem i en regresjon. Vi skal også se litt på hvordan vi kan sjekke om regresjonen vår er gyldig, gjennom å undersøke forutsetningene til OLS, og hvordan vi kan tolke den.

Omkoding av variabler

Som vi snakket om forrige gang er ofte dataene i en annen form en sånn vi ønsker dem. Dette gjelder ofte også for variabler, og isåfall må vi kode dem om så de er i den formen vi ønsker! På dette seminaret tenker jeg vi kan først se på to variabler, og gjøre hendholdsvis sentrering og omkoding til en dikotom variabel. Som vanlig må vi begynne med å laste inn dataene, og pakkene vi ønsker å bruke.

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------ tidyvers
```

```
## v ggplot2 3.3.0     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

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

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

Som forrige gang er dataene nå alle individer, for alle land-runder som har vært med i ESS. Dette er ikke en dataform vi kan jobbe med, og vi må derfor hente ut en land-runde vi ønsker. I neste seminar skal vi se på paneldata, og hvordan vi faktisk kan bruke all informasjonen i datasettet. For nå tenker jeg vi kan se på Storbrittania i runde 8.

```
#Som dere husker fra igår kan vi nå bruke filter() for å hente ut den
#runden og det landet vi ønsker

UK_8 <- ESS_Data %>%
  filter(essround == 8 & Country == "GB") #Her ser dere jeg bruker "&" som betyr "and", altså må begge
                                          #kriteriene være TRUE for at den skal hente de ut.
#Ofte når vi har laget et nytt datasett ønsker vi å lagre det, sånn at det er lettere å finne
#fram igjen når vi skal bruke det igjen, eller dele det med andre.
"Dette kan vi f.eks. gjøre med saveRDS()"
```

```
## [1] "Dette kan vi f.eks. gjøre med saveRDS()"
```

```
saveRDS(UK_8, "Uk_8.rds") #Her skriver jeg først datasettet jeg vil lagre, og så hvilket
                          #navn jeg vil at filen skal ha.
                          #Når du lagrer RDS må du alltid avslutte med ".rds"
```

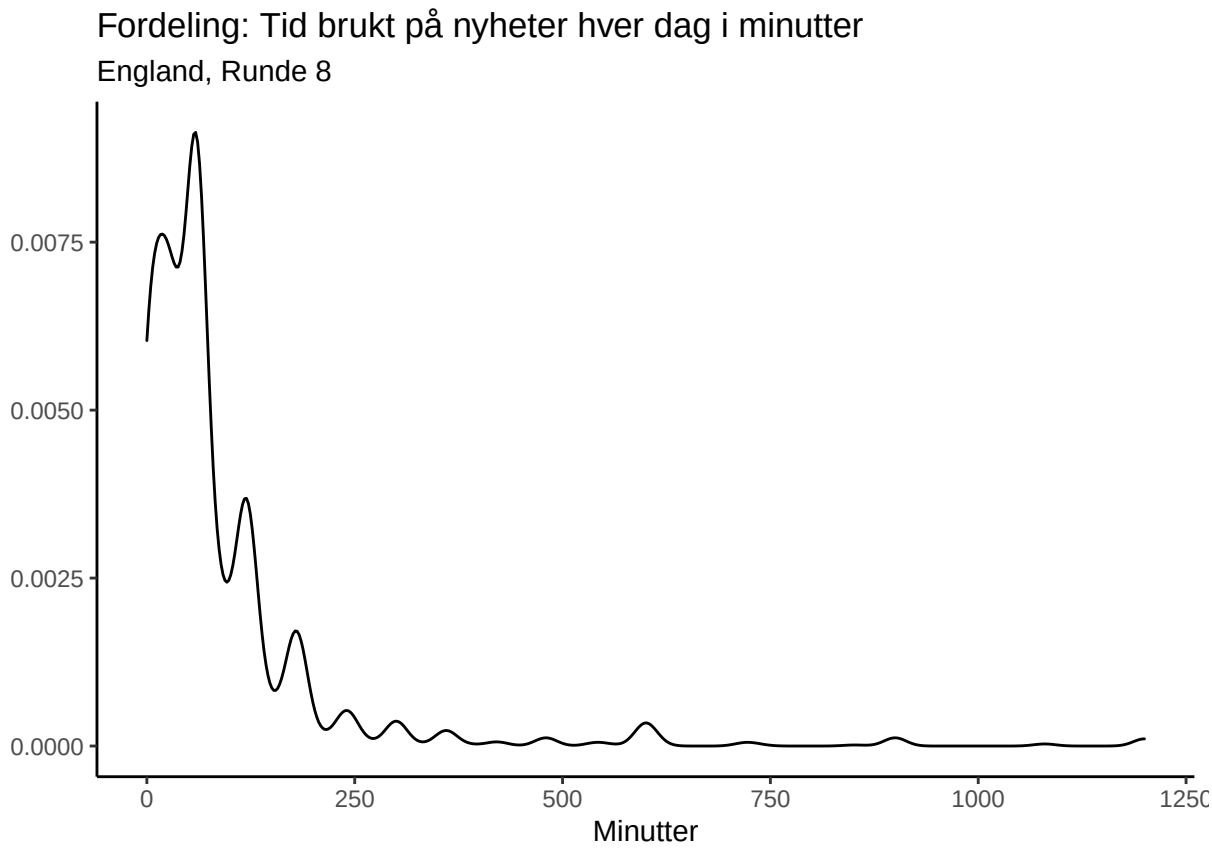
Før vi sentrerer en variabel kan det være en god idé å se på hvordan variabelen er fordelt fra før. Det er flere måter vi kan gjøre dette på, men selv synes jeg det beste er gjennom å visualisere variabelen. Vi kan først prøve å bruke summary() for å se det i tekst likevell. La oss undersøke hvor mye tid briter bruker på nyhetene.

```
summary(UK_8$Time_News)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   30.00   60.00   89.29  120.00 1200.00       4
```

Fra kodeboken husker vi at denne er målt i antall minutter pr. dag. En gjennomsnittelig brite bruker 89,29 minutter pr. dag, mens medianen er noe lavere på en time. Dette sier oss at vi har en skjevfordelt variabel. Dette er selvfølgelig mye lettere å se om vi kan visualisere variabelen, så la oss gå tilbake til ggplot. Som vanlig har vi først en linje hvor vi definere datasettet og variablene, og så en som sier hva slags plott vi ønsker.

```
ggplot(UK_8, aes(Time_News))+
  geom_density() +
  theme_classic()+ #GGplot gjør at du kan legge til en del ting for å få det penere, f.eks. et tema
  labs(title = "Fordeling: Tid brukt på nyheter hver dag i minutter", #Her legger jeg til nye titler
       subtitle = "England, Runde 8",
       y ="",
       x ="Minutter")
```



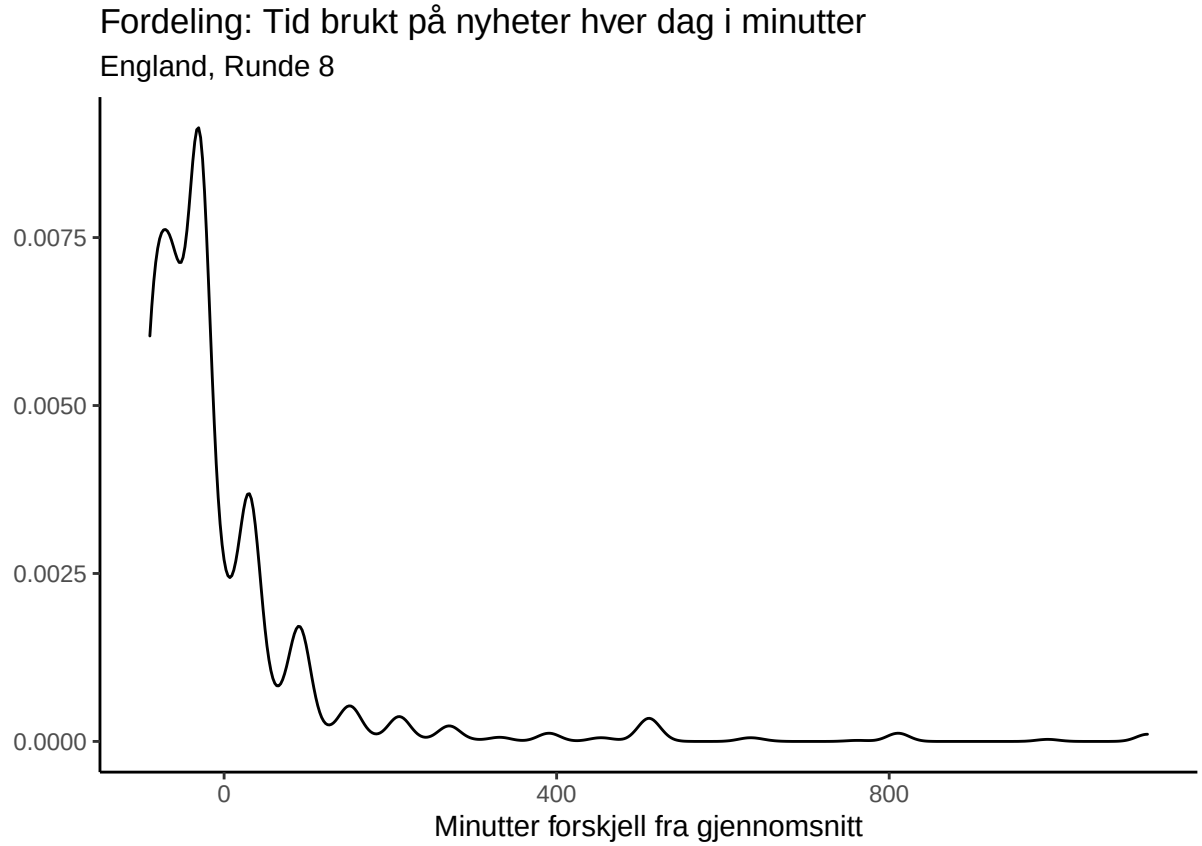
Som vi kan se fra plottet er variabelen veldig skjevfordelt. Her det viktig å huske at normalfordelte uavhenige

variabler *ikke* er en forutsetning for OLS, men å forandre noe på dem ofte kan gjøre det lettere å tolke modellen. La oss prøve å sentrere denne variabelen, altså la den nye variabelen være *X - mean(x)*

```
UK_8 <- UK_8 %>%
  mutate(Time_News_ST = Time_News - mean(Time_News, na.rm = TRUE)) #Mutate() skaper nye, eller endrer
                                                                   #variabler.
```

Vi kan nå prøve å plottet den nye variabelen for å se hva som har forandret seg.

```
ggplot(UK_8, aes(Time_News_ST))+
  geom_density() +
  theme_classic()+ #GGplot gjør at du kan legge til en del ting for å få det penere, f.eks. et tema
  labs(title = "Fordeling: Tid brukt på nyheter hver dag i minutter", #Her legger jeg til nye titler
       subtitle = "England, Runde 8",
       y ="",
       x ="Minutter forskjell fra gjennomsnitt")
```



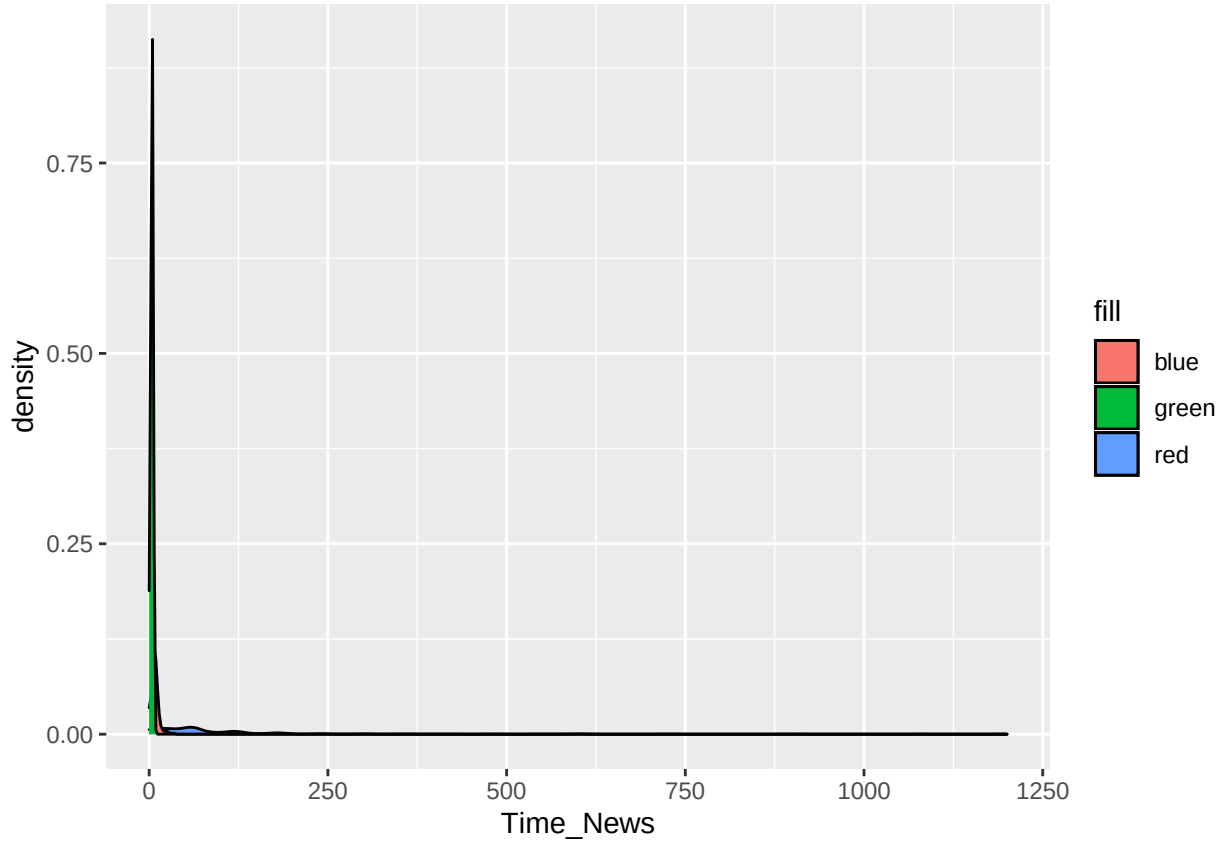
Som vi kan se har ikke formen på variabelen forandret seg! Det er fordi sentrering ikke endrer formen, men gjør at gjennomsnittet er null, og alle andre variabler nå er antall minutter forskjellig fra gjennomsnittet. Dette kan kanskje være litt lettere å tolke. Ønsker vi faktisk å endre formen kan vi prøve å gjøre transformasjon med kvadratrot ("sqrt()") eller logaritmen ("log"). For logaritmen legger jeg til 1, da log(0) gir -Inf som svar.

```
UK_8 <- UK_8 %>%
  mutate(Time_News_Sqrt = sqrt(Time_News),
         Time_News_Log = log(Time_News+1)) #Du kan lage flere variabler i samme mutate,
                                           #så lenge de er skilt med et komma
```

Nå kan vi prøve å plotte dem sammen, så kan vi lettere se forskjellene mellom dem. Leg merke til at jeg nå

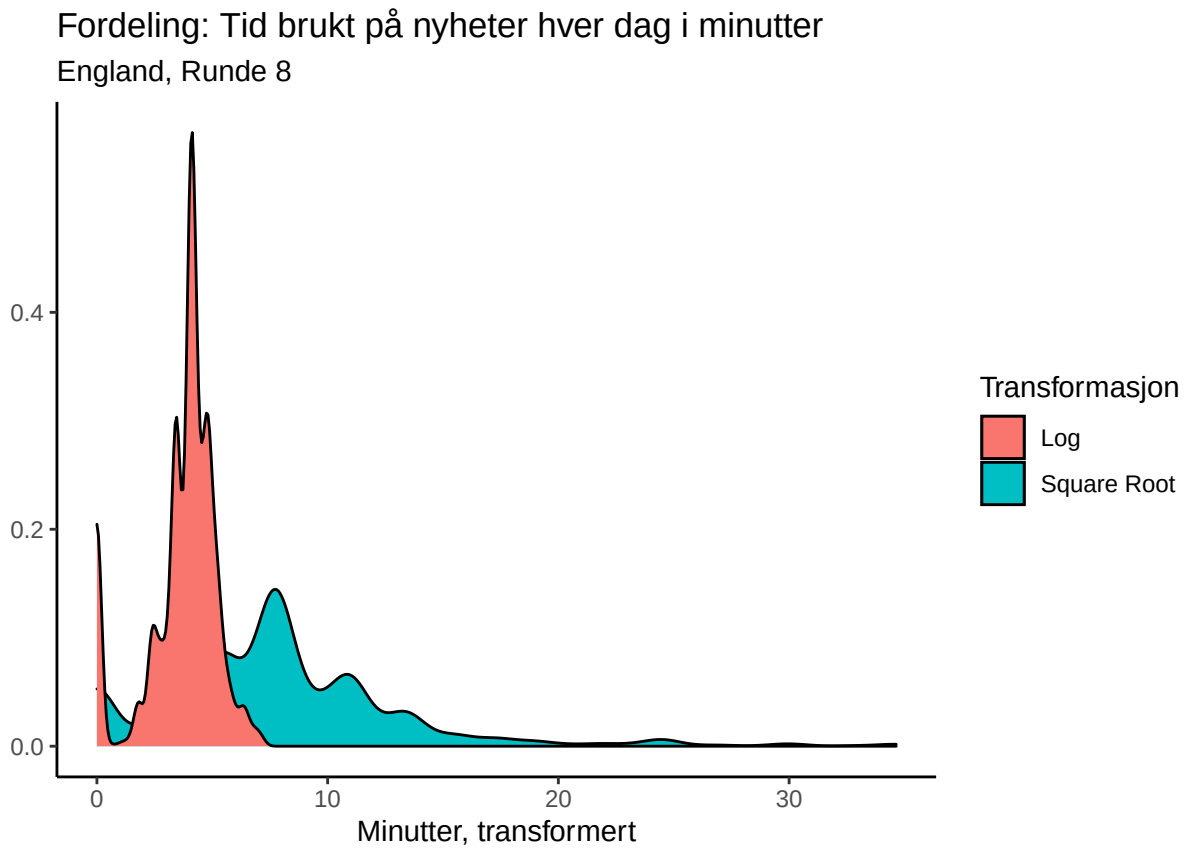
sier hvilken variabel som skal plottes i geom funksjonen, heller en aller først!

```
ggplot(UK_8) +
  geom_density(aes(Time_News, fill = "red")) +
  geom_density(aes(Time_News_Sqrt, fill = "blue")) +
  geom_density(aes(Time_News_Log, fill = "green"))
```



Her ser vi med en gang at det er vanskelig å skille dem pga. hvor lang hale det er på den orginale, kanskje er det lettere om vi dropper den? La oss prøve uten den

```
ggplot(UK_8) +
  geom_density(aes(Time_News_Sqrt, fill = "Square Root")) +
  geom_density(aes(Time_News_Log, fill = "Log")) +
  theme_classic() +
  labs(title = "Fordeling: Tid brukt på nyheter hver dag i minutter",
       subtitle = "England, Runde 8",
       y ="",
       x ="Minutter, transformert",
       fill = "Transformasjon",
       red = "Square Root")
```



Som vi kan se er logtransformasjonen nærmest en normalfordeling, men også den er veldig spiss med korte haler. Som oftest vil du aldri få til en perfekt transformasjon, så her handler det om hva som er nærmest og "godt nok." Tenk alltid over hvorfor du gjør det, og hva det hjelper deg med!

Dikotome variabler, og ifelse()

Så langt har vi sett på en del logiske tester. La oss gå litt dypere inn i hva de faktisk er for noe, all den tid dette er noe av det viktigste vi har i R! En logisk test er veldig enkelt noe som undersøker ett eller annet som kan beskrives med enten "TRUE" eller "FALSE." Et enkelt eksempel er hvorvidt 1 == 2. Legg igjen merke til de dobble likhetstegnene, som betyr at jeg vil sjekke hvorvidt de er like. La oss prøve å kjøre dette i R.

```
1 == 2
```

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

Som dere ser får vi svaret "FALSE", da dette jo er ulike tall. Om vi prøver igjen med 2 == 2 vil vi se at R gir oss svaret "TRUE."

```
2 == 2
```

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

Det finnes nærmest en uendelig mengde tester du kan gjøre, det er kun fantasien som setter grenser. Det eneste viktige er at de kan besvaret med enten "TRUE" eller "FALSE." Ting som kan være nyttig å vite er f.eks. at dere kan bruke "&" som betyr "and", og krever at flere ting får verdien TRUE, og "|" som betyr "or" og krever at minst 1 retunerer "TRUE".

```
(2 == 1) & (2 == 2)
```

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

```
(2 == 2) & (2 == 2)
```

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

```
(2 == 1) | (2 == 2)
```

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

```
(2 == 1) | (2 == 1)
```

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

Når vi brukte filter() for å hente ut kun de landene og rundene vi ønsket var det nettopp logiske tester vi gjorde! Vi ba R gi oss et datasett med kun de respondentene som får "TRUE" på om de er ifra det landet vi ønsket, og den runden vi ville ha.

En annen måte vi ofte bruker logiske tester på er i det som kalles ifelse(), dette er både en funksjon i seg selv, og en vanlig måte å strukturere kode. Vi skal se på det som en funksjon her i seminaret. I en ifelse() gir du en logisk test, og så sier hva R skal gjøre hvis den returnerer TRUE, og hva den skal gjøre hvis det er FALSE. Altså, "if this is true then do that, else do this." Dette kan bruker til utrolig mye, men la oss her se på det som en måte å skape dikotome variabler.

La oss si vi ønsker å vite hvorvidt en respondent er ung eller ikke. Først må vi definere hva vi mener er ungt, la oss si under 35 år.Så vil vi altså at variabelen skal ha verdien 1 om personen er under 35, og 0 ellers. Altså må vi først ha en logisk test som kan finne ut av dette. I R kan vi bruke "<" og ">" for å se om en ting er større enn en annen. Et siste spørsmål er, hvilken gruppe er den som er nøyaktig 35 i? Om vi ønsker at de skal være unge må vi si "mindre enn, eller lik" som kan gjøres ved å skrive "<=" eller ">=". La oss prøve dette i kode.

```
#Først kan vi prøve testen
60 < 35
```

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

```
35 < 35
```

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

```
36 <= 35
```

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

```
35 <= 35
```

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

```
#Ok, vi kan se at testen fungerer sånn vi ønsker, la oss prøve å lage en ny variabel med den!

UK_8$Young <- ifelse(UK_8$Age <=35, 1, 0) #Jeg skriver først verdien jeg vil ha hvis TRUE, og
#så hvis FALSE. La oss lage en tabell som viser verdiene på Age og Young opp mot hverandre

table(UK_8$Age, UK_8$Young)
```

```
##
##       0  1
##   15  0  4
##   16  0 17
##   17  0 13
##   18  0 24
##   19  0 10
##   20  0 20
```

```
##    21  0 14
##    22  0 21
##    23  0 17
##    24  0 18
##    25  0 28
##    26  0 27
##    27  0 20
##    28  0 24
##    29  0 25
##    30  0 27
##    31  0 31
##    32  0 40
##    33  0 26
##    34  0 30
##    35  0 37
##    36 27  0
##    37 28  0
##    38 34  0
##    39 27  0
##    40 30  0
##    41 28  0
##    42 34  0
##    43 28  0
##    44 33  0
##    45 30  0
##    46 32  0
##    47 28  0
##    48 26  0
##    49 35  0
##    50 30  0
##    51 39  0
##    52 38  0
##    53 31  0
##    54 27  0
##    55 37  0
##    56 24  0
##    57 29  0
##    58 26  0
##    59 30  0
##    60 34  0
##    61 38  0
##    62 30  0
##    63 28  0
##    64 34  0
##    65 32  0
##    66 35  0
##    67 45  0
##    68 37  0
##    69 41  0
##    70 43  0
##    71 26  0
##    72 25  0
##    73 28  0
##    74 14  0
```

```
##   75 20  0
##   76 18  0
##   77 31  0
##   78 14  0
##   79 22  0
##   80 13  0
##   81 17  0
##   82 10  0
##   83 15  0
##   84 13  0
##   85  9  0
##   86 13  0
##   87 10  0
##   88  8  0
##   89  4  0
##   90  5  0
##   91  3  0
##   92  4  0
##   93  2  0
##   94  1  0
```

Som vi kan se får alle som er 35 år og yngre verdien 1, mens de som er eldre får 0. Altså ble det akkurat som vi ønsket det. Vi kan også gjøre dette i Tidyverse med koden:

```
UK_8 <- UK_8 %>%
  mutate(Young = ifelse(Age <= 35, 1, 0))
```

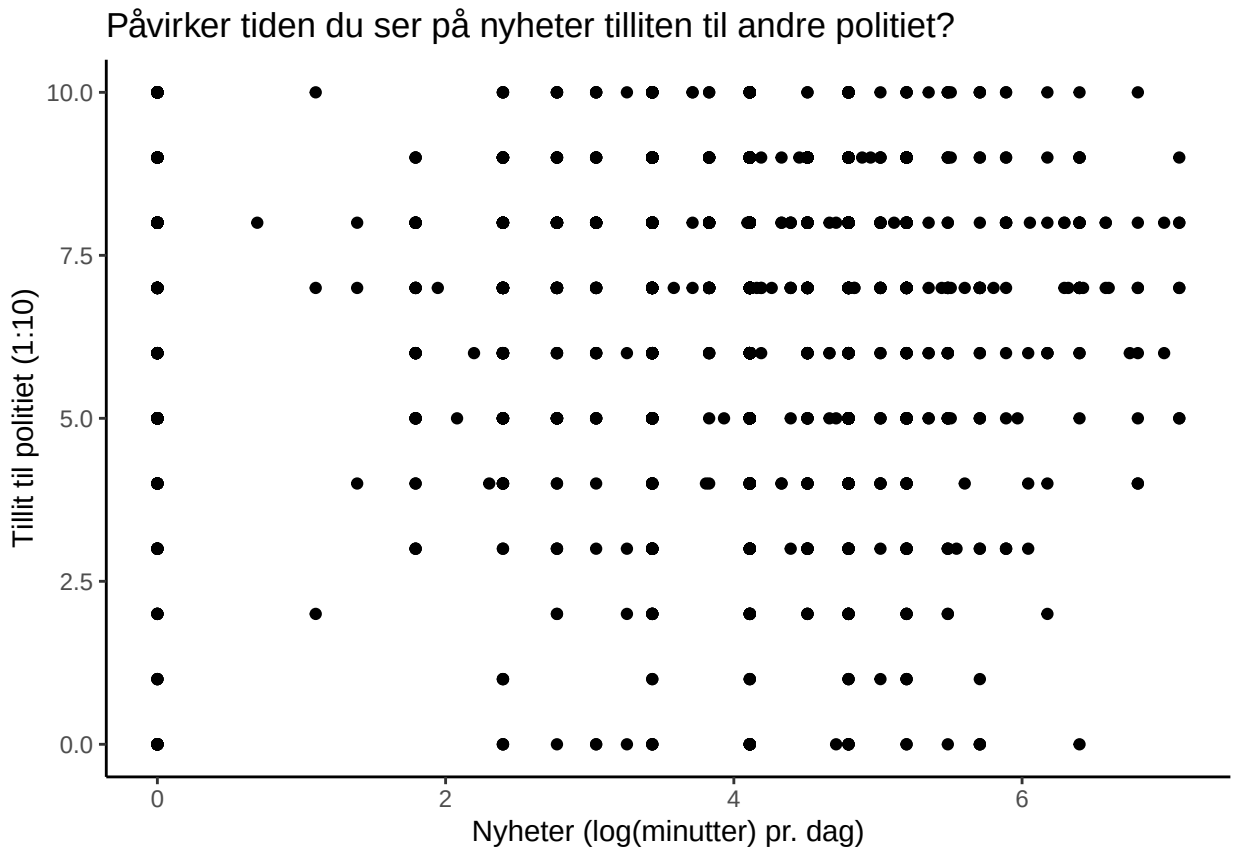
Dette gir oss det samme resultatet.

OLS: Moddelering, tolkning, og plotting!

OLS, eller ordinary least squares, er for de fleste av oss den første regresjonsmodellen vi lærer om. Derfor det jo bare passende at det også er en første vi lærer å bruke! I dag skal vi først se på en regresjons med kun uavhengig variabel, og så flere. Vi skal så på hvordan vi kan undersøke om vi oppfyller forrutsetningene til OLS, plotte en regresjon, og hvordan å presentere og tolke resultatene på best mulig måte.

Før vi kan starte en regresjon må vi først velge hva vi ønsker å finne ut av, og hva vi tror årsaksretningen er. La oss her se på hvorvidt tiden du bruker på nyheter påvirker hvorvidt du stoler på andre politiet. Vi kan først lage et scatterplot for å se om vi tror det er en sammenheng, og om denne er linær. Når vi skal lage et plot med flere variabler i ggplot skriver vi først inn X variabelen, og så y!

```
ggplot(UK_8, aes(Time_News_Log, Trust_Police)) +
  geom_point() +
  theme_classic() +
  ggtitle("Påvirker tiden du ser på nyheter tilliten til andre politiet?") +
  ylab("Tillit til politiet (1:10)") +
  xlab("Nyheter (log(minutter) pr. dag)")
```



Som vi kan se er det tegn til en linær sammenheng her, la oss nå se om vi kan modellere denne. For å gjøre dette bruker vi lm() funksjonen. Denne forventer en formel, et datasett, og hva den skal gjøre med NA. En formel skriver som "Y ~ X1 + X2 + X3 + ... Xk". Vi lagrer modellen som et objekt, hvorifra vi kan hente ut koeffisientene, og annet vi ønsker å se på.

```
Mod1 <- lm(Trust_Police ~ Time_News_Log, data =  UK_8, na.action = "na.exclude")
```

Vi kan nå bruke summary() for å se verdiene i regresjonen vår:

```
summary(Mod1)
```

```
##
## Call:
## lm(formula = Trust_Police ~ Time_News_Log, data = UK_8, na.action = "na.exclude")
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7666 -1.5877  0.3463  1.3797  3.5491
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.45086    0.13050  49.432   <2e-16 ***
## Time_News_Log  0.04935    0.03264   1.512    0.131
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.264 on 1951 degrees of freedom
##   (6 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.001171,   Adjusted R-squared:  0.0006588
## F-statistic: 2.287 on 1 and 1951 DF,  p-value: 0.1306
```

Her ser vi Time_News_Log har en stigningskoeffisient på *0,03*, men denne er ikke signifikant med en en p-verdi på *0,131*. Altså kan vi ikke si at denne koeffisienten er forskjellig fra null. Vi ser også at R^2 er svært lav, nærmest null, og modelllen forklarer derfor veldig lite av variansen i tilitt til politiet. Husk her at en modell som viser at det ikke er en sammenheng kan være like informativ som en som viser en sammenheng!

Ofte ønsker vi å ha penere modeller enn dette å vise i tekstene våre, derfor finnes det pakker som lager tabeller som er lettere å vise frem. Her kan vi se på en som heter stargazer:

```
#install.packages("stargazer") Som vanlig må vi innstalere den om vi ikke har gjort det før
library(stargazer)
```

```
##
## Please cite as:
```

```
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.
```

```
##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

```
stargazer(Mod1, type = "text", title = "Regresjon: Tid brukt på nyheter, og tilitt til politiet",
          dep.var.labels = "Tillitt til politiet",
          covariate.labels = "log(Minutter) på nyheter pr. dag")
```

```
##
## Regresjon: Tid brukt på nyheter, og tilitt til politiet
## ================================================================
##                                       Dependent variable:
##                                   ---------------------------
##                                      Tillitt til politiet
## ----------------------------------------------------------------
## log(Minutter) på nyheter pr. dag            0.049
##                                            (0.033)
##
## Constant                                  6.451***
##                                            (0.131)
##
## ----------------------------------------------------------------
## Observations                                1,953
## R2                                          0.001
## Adjusted R2                                 0.001
## Residual Std. Error                   2.264 (df = 1951)
## F Statistic                          2.287 (df = 1; 1951)
## ================================================================
## Note:                              *p<0.1; **p<0.05; ***p<0.01
```

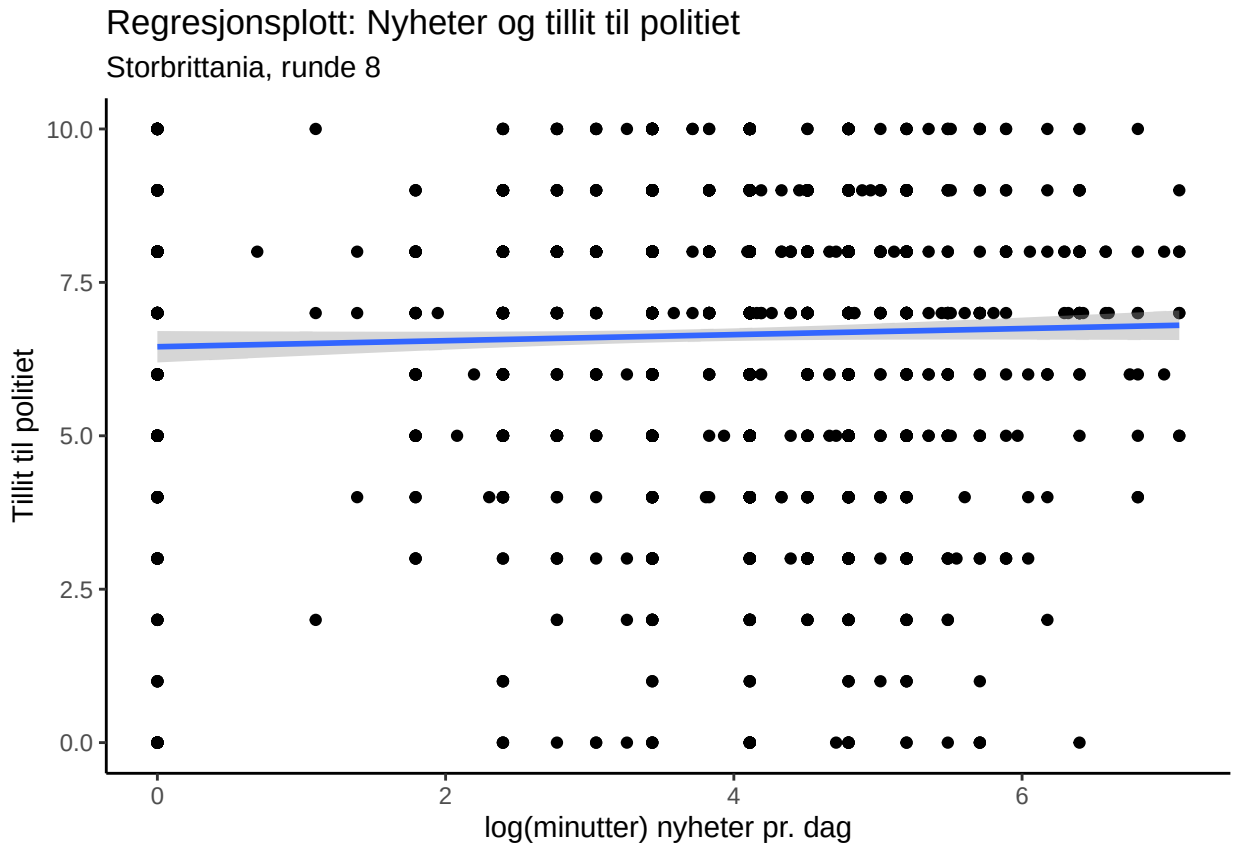
Her ser dere jeg kan endre navnene til noe som er mer forståelig, og få en tabell som ser penere ut og kan kopieres rett inn i word. Der det nå står "type ==”text"" kan dere også endre denne til "latex" om dere bruker det skriveprogrammet, eller "html" som kan være penere i word enn ren tekst.

Plotte

For å plotte regresjonen bruker vi nesten samme kode som vi i forrige seminar gjorde for å kun se på korrelasjon. Her kan det ofte være fint å legge regresjonslinjen oppå et scatterplott, så kan en samtidig se hvordan de observerte sprer seg rundt regresjonslinjen.

```
ggplot(UK_8, aes(Time_News_Log, Trust_Police)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_classic() +
  labs(title = "Regresjonsplott: Nyheter og tillit til politiet",
       subtitle = "Storbrittania, runde 8",
       x = "log(minutter) nyheter pr. dag",
       y = "Tillit til politiet")
```

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



Som vi kan se er linjen veldig flat, og med ganske stor avstand til observasjonene. Det forklarer nok hvorfor modellen ikke var signifikant, og hadde så lav R^2.

Multivariat

Å legge til flere variabler i en regresjonsvariabel er heldigvis veldig enkelt, du bare skriver den nye variabelen inn i formelen med et pluss imellom. La oss se om alder forandrer på regresjonen vår.

```
Mod2 <- lm(Trust_Police ~ Time_News_Log + Age, data =  UK_8, na.action = "na.exclude")
stargazer(Mod2, type = "text", title = "Regresjon: Tid brukt på nyheter,
          Alder, og tilitt til politiet",
          dep.var.labels = "Tillitt til politiet",
          covariate.labels = c("log(Minutter) på nyheter pr. dag",
                               "Alder"))
```

```
##
## Regresjon: Tid brukt på nyheter
```

```
## ================================================================
##                                        Dependent variable:
##                                    ----------------------------
##                                       Tillitt til politiet
## ----------------------------------------------------------------
## log(Minutter) på nyheter pr. dag             0.061*
##                                              (0.034)
##
## Alder                                       -0.009***
##                                              (0.003)
##
## Constant                                    6.850***
##                                              (0.179)
##
## ----------------------------------------------------------------
## Observations                                  1,920
## R2                                            0.006
## Adjusted R2                                   0.005
## Residual Std. Error                     2.265 (df = 1917)
## F Statistic                           5.443*** (df = 2; 1917)
## ================================================================
## Note:                               *p<0.1; **p<0.05; ***p<0.01
```

Som vi kan se er nå tiden en bruker på nyheter svakt signifikant på *0,1* nivå, mens alder er svært signifikant på *0,01* nivå om enn med en svært svak negativ effekt på *-0,009* (Gitt at denne måler enkelte år, er den egentlig så svak?. Hvor mange år må du endre for å gå ett skalapunkt ned?)

Forutsetninger for OLS

OLS har flere forutsetninger for at vi skal ha et korrekt resultat. Bl.a. må vi passe på at variablene våre er relevante, og at vi ikke har utelatt noen spesielt viktige variable (Omitted Variable Bias). Her skal vi se på hvorvidt vi har homoskedastiske restledd, korellasjon mellom de uavhengige variablene, og fravær av korrelasjon mellom predikerte variabler og residualene.

Dere kan først prøve å i konsollen trykke `plot(Mod2)`. Da får dere opp flere plott som viser alt sammen, men vi vill selvfølgelig viten hvordan vi kan lage disse på egenhånd!

La oss først sjekke hvorvidt vi har normalfordelte restledd. Å sjekke normalforeling gjorde vi istad, så dette kan vi jo!

```
ggplot(Mod2, aes(residuals)) +
  geom_density()
```

```
## Error in `$<-.data.frame`(`*tmp*`, ".hat", value = c(`1` = 0.000562995968186125, : replacement has 19
```

Her fikk vi vist en feilmelding, hva kan dette være? Ggplot forventer en data.frame som den kan hente dataene ifra, og modellen er jo ikke dette. For å få det til å fungere må vi derfor legge til residualene som en ny kollone i datene, og så plotte dem. La oss prøve det!

```
UK_8$Residuals <- Mod2$residuals
```

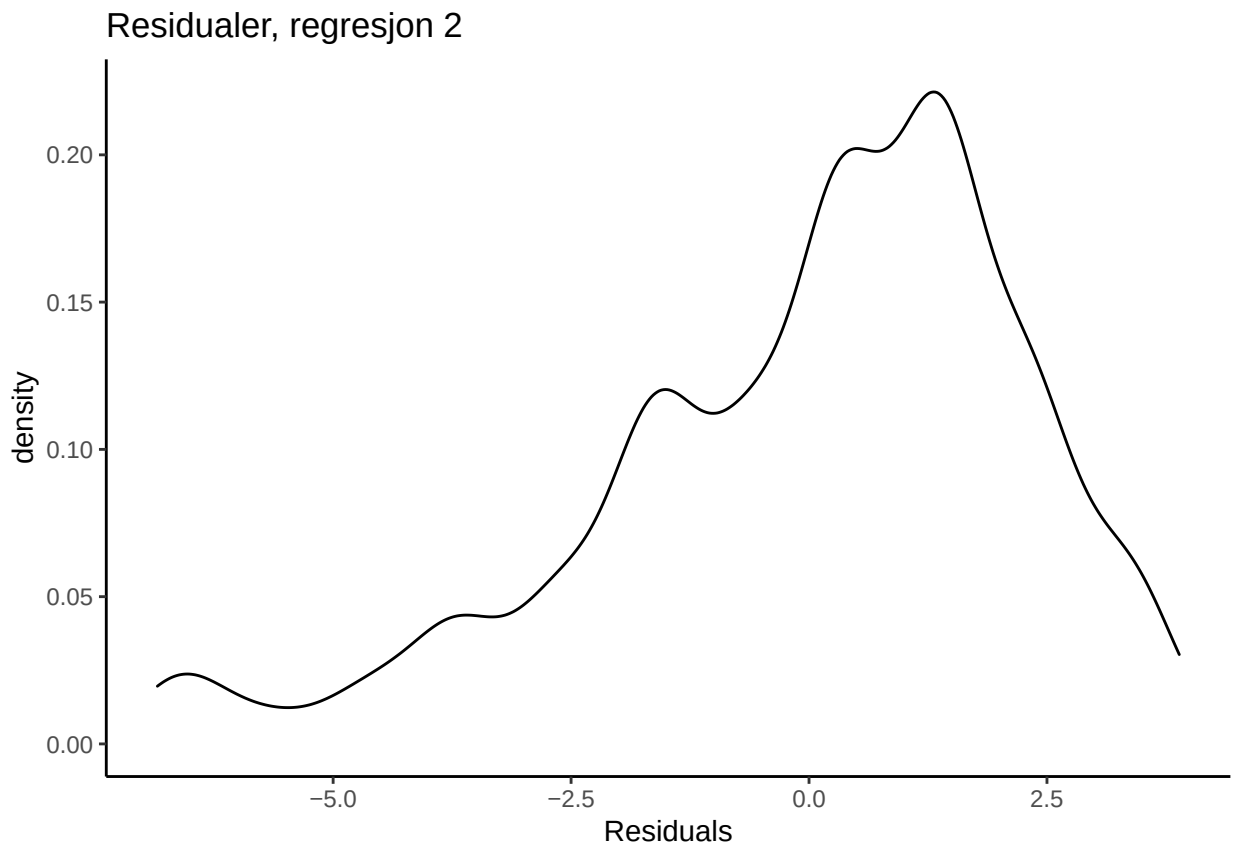
```
## Error in `$<-.data.frame`(`*tmp*`, Residuals, value = c(`1` = 3.34801613959733, : replacement has 192
```

Her ser vi at får en feil fordi dataene har forskjellig lengde. Det er færre residualer enn det respondenter i de orginale datene. Dette er fordi noen av dem har NA på en variablene, og dermed ikke ble med i analysen. For å fikse må vi gjøre at de som har NA på en variablene også får NA på residual-kollonen. Heldigvis finnes det en enkel fiks for dette, i funksjonen resid()!

```
UK_8$Residuals <- resid(Mod2)
```

Der har vi endelig residualene som en del av datasettet! Nå kan vi plotte residualene for å se om de har riktig fordeling eller ikke.

```
ggplot(UK_8, aes(Residuals)) +
  geom_density() +
  theme_classic() +
  ggtitle("Residualer, regresjon 2")
```



Som vi kan se er residualene noe høyreskjeve. Det er alikevell ikke så mye at jeg ville sett på det som et stort problem.

Korrelasjon mellom uavhengige variabler

OLS krever at vi har fravær av multikolinaritet, hvorvidt det er korrelasjon mellom dem kan vi teste med en korrelasjonstest. Husk at noe korrelasjon vil det som oftest være, og $r > 0$ er ikke nødvendigvis et problem.

```
cor.test(UK_8$Age, UK_8$Time_News_Log)
```

```
##
##  Pearson's product-moment correlation
##
## data:  UK_8$Age and UK_8$Time_News_Log
## t = 8.2379, df = 1920, p-value = 3.203e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1412222 0.2275986
```

```
## sample estimates:
##       cor
## 0.1847672
```

Korrelason mellom predikerte verdier og residualer

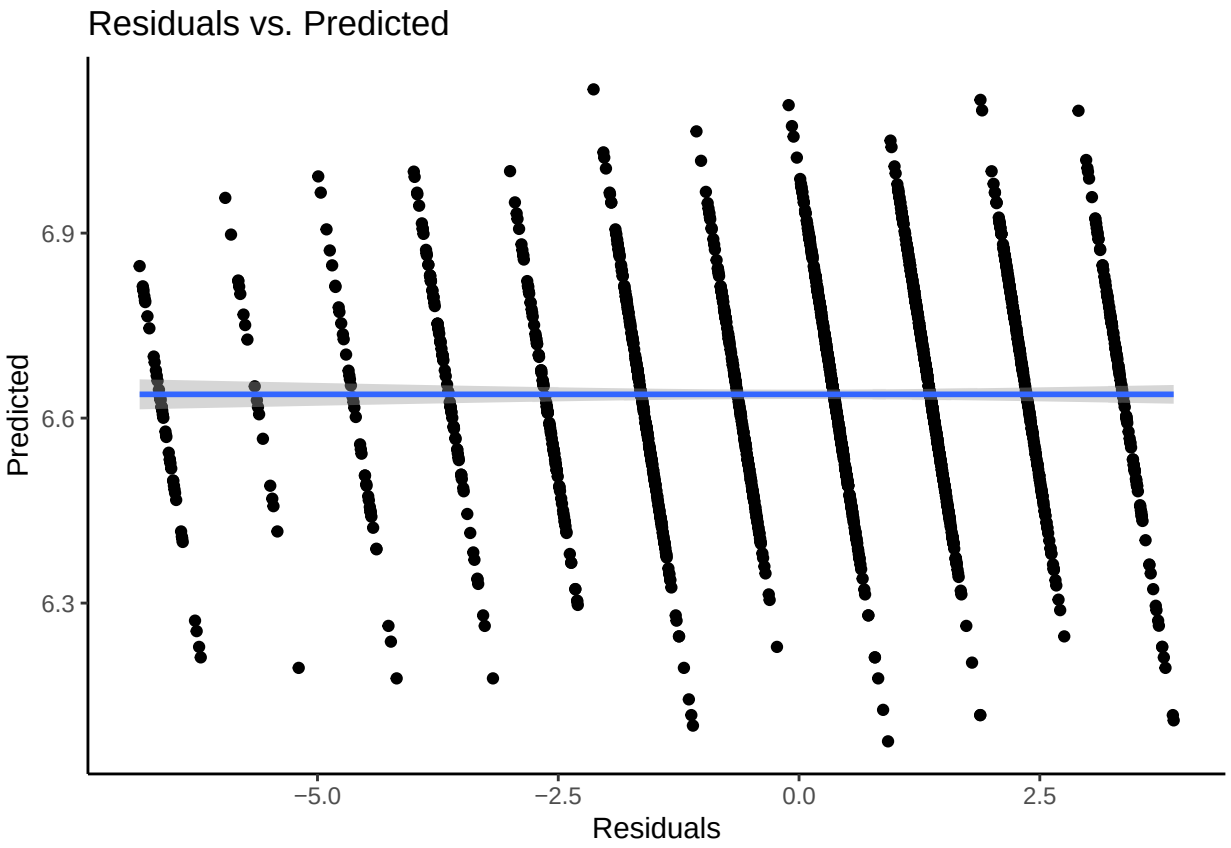
Til slutt vil vi se om det er noen sammenheng mellom de predikerte verdiene, og residualene. For å gjøre dette må vi også legge til de predikerte verdiene i datene.

```
UK_8$Pred <- predict(Mod2)
```

Så kan vi plotte dem som vanlig med et scatterplott, og en regresjonslinje.

```
ggplot(UK_8, aes(Residuals, Pred)) +
  geom_point() +
  geom_smooth(method = lm) +
  theme_classic() +
  labs(title = "Residuals vs. Predicted",
       x = "Residuals",
       y = "Predicted")
```

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



Som vi kan se er regresjonslinjen tilnærmet helt flat, med ingen tydelig sammenheng. Altså kan vi si at det ikke er noen korrelasjon mellom residualene og de predikerte verdiene!

Det var det for dennne gang, lykke til med oppgavene, og still gjerne spørsmål!