Hjemmeksamen ECON3110

Kandidatnummer 6

2025-12-15

Oppgave 1

pakker

```
library(dplyr) # Databehandling
library(glmnet) # Elastic net
```

DATA

```
df <- read.csv("data/corruption.csv")

df |> dim()
```

```
## [1] 2353   43
```

```
df |>
  summarise(across(everything(), ~ sum(. == -1000)/nrow(df) * 100)) # Prosent
```

```
##   country iso3c Region Year      CPI      gdp population school_enroll     area
## 1       0     0      0    0 1.869953 3.314917   1.104972      4.972376 1.104972
##   nat_rents armed_forces mammal_threat rail_lines real_interest_rate
## 1  25.83935     36.84658      1.104972   70.54824           40.33149
##   stocks_traded poverty_gap830 pop_growth_pct pop_largest_city pop_density
## 1      61.19847       60.56099       1.147471         17.12707    8.712282
##   democracy union_freedom rule_of_law_index x1 x2 x3 no_CPI_control
## 1  3.867403      3.867403          3.867403  0  0  0              0
##   no_gdp_control no_population_control no_school_enroll_control no_area_control
## 1              0                     0                        0               0
##   no_nat_rents_control no_armed_forces_control no_mammal_threat_control
## 1                    0                       0                        0
##   no_rail_lines_control no_real_interest_rate_control no_stocks_traded_control
## 1                     0                              0                        0
##   no_poverty_gap830_control no_pop_growth_pct_control
## 1                         0                         0
##   no_pop_largest_city_control no_pop_density_control no_democracy_control
## 1                           0                       0                    0
##   no_union_freedom_control no_rule_of_law_index_control
## 1                        0                            0
```

Dimensjonen til datasettet er 2353 rader med 43 kolonner, som man kan se fra output er det mange variabler som har manglende verider. En måte å løse dette på er å fjerne alle nas.

```
df |>
  mutate(across(everything(), ~ ifelse(. == -1000, NA, .))) |>
  na.omit() |>
  dim()
```

```
## [1] 150  43
```

Da ser vi at vi mangler mange observasjoner. Derfor tenker jeg det er hensiktsmessig å variablene med over 40 prosent na's.

```
df <- df |>
  select(-real_interest_rate, -stocks_traded, -poverty_gap830, -rail_lines,) |> #Variabler med mange -1
  mutate(across(everything(), ~ ifelse(. == -1000, NA, .))) |> # Gjør -1000 om til na
  na.omit() # Fjerner NA

df |> dim()
```

```
## [1] 1230   39
```

Fjernet mye data. Hvis dette ikke hadde vært eksamen kunne det vært hensiktsmessig å kankje imputere noen nas med gjennomsnitt eller median.

a

Standardiser variablene i datasettet, og del datasettet i én treningsdel og en testdel

```
df <- df |>
  mutate(across(where(is.double), ~ scale(.))) # Skalerer alle variabler som er numeriske


split <- sample(1:nrow(df), floor(nrow(df)*0.8), replace = F) # 80 prosent av indeksen


training = df[split,]
testing = df[-split,]
```

b

Definer en funksjon for K-fold kryssvalidering av en algoritme. Funksjonen skal ta som input et datasett og antall folder (K). Funksjonen skal returnere et datasett med en ny kolonne som indikerer fold.

```
kfold <- function(data, k)
{
  n <- nrow(data) # lengden

  tall <- rep(1:k) # vektor fra 1 til k
```

```
  folds <- rep(1:k, length.out = n) # repeterer 1 til k frem til n Så langt  det lar seg gå
  data$fold <- folds # legger den til i datarammen

  return(data)
}
```

c

Bruk funksjonen fra b)på treningsdatasettet til å tune en LASSO modell som skal predikere korrupsjon(CPI). Dersom du ikke får til b)kan du bruke en alternativ metode. Du kan søke i domenet 0 til 0.1.

```
set.seed(3110)
library(glmnet) # bruker denne til Lasso

train_fold <- kfold(training, 10)

lasso.results <- c()
mse <- 1000000
lambda <-  0

for (i in unique(train_fold$fold))
{

  train <- train_fold |>
    filter(fold != i)

  train_x <- train |>
    select(-CPI, -fold) |>
    as.matrix()

  train_y <- train |>
    select(CPI) |>
    pull()

  test <- train_fold |>
    filter(fold == i)

  test_x <- test |>
    select(-CPI, -fold) |>
    as.matrix()

  test_y <- test |>
    select(CPI) |>
    pull()

  lasso.mod <- glmnet(
      train_x, train_y,
      alpha = 1, # LASSO
      )

  len <- length(lasso.mod$lambda)

  best.lambda <- lasso.mod$lambda[len]
```

```
  lasso.preds <- predict(lasso.mod, test_x, s = best.lambda)

  lasso.mse <- mean((test_y-  lasso.preds)^2)

  lasso.results <- c(lasso.results, lasso.mse)

  if (lasso.mse < mse)
  {
    mse <- lasso.mse
    lambda <- best.lambda
  }


}

paste("Beste lambda", lambda)
```

```
## [1] "Beste lambda 0.024333234311886"
```

Kjører loopen lik antall folds. Lager "nye" train og test basert på folden. Kjører en lasso modell. Tar ut den siste lambdaen. Gjør prediksjoner regner ut mse på den utelatte folden

d

Med hyperparameter-konfigurasjonen du kom frem til i c), tren modellen på hele treningsdatasettet og rapporter OOS-prediksjonsnøyaktighet.

```
set.seed(3110)
Lasso <- glmnet(
  training |> select(-CPI) |> as.matrix(),
  training |> select(CPI) |> pull(),
  alpha = 1
)

x <- testing |> select(-CPI) |> as.matrix()
Lasso.pred <- predict(Lasso,x, s = lambda)

y <- testing |> select(CPI) |> pull()
mse <- mean( (y - Lasso.pred)^2 )


paste("OOS prediksjonsnoyaktighet:", mse)
```

```
## [1] "OOS prediksjonsnoyaktighet: 69.9200595076219"
```

Lasso ned tunet parameter

e

Tolk resultatene fra LASSO modellen - hvilke variabler later til å predikere korrupsjon best?

```
coef(Lasso, s =lambda) |> round(4)
```

```
## 39 x 1 sparse Matrix of class "dgCMatrix"
##                                      s1
## (Intercept)                    719.7170
## country                          .
## iso3c                            .
## Region                           .
## Year                            -0.3447
## gdp                             10.3046
## population                       0.0000
## school_enroll                    2.6781
## area                             0.3439
## nat_rents                       -1.9063
## armed_forces                    -0.6268
## mammal_threat                   -0.0314
## pop_growth_pct                   0.7075
## pop_largest_city                 0.0000
## pop_density                      0.4543
## democracy                        3.8487
## union_freedom                    .
## rule_of_law_index                4.1518
## x1                               0.4565
## x2                               0.2300
## x3                               0.0161
## no_CPI_control                   .
## no_gdp_control                   .
## no_population_control            .
## no_school_enroll_control         .
## no_area_control                  .
## no_nat_rents_control             .
## no_armed_forces_control          .
## no_mammal_threat_control         .
## no_rail_lines_control            0.3596
## no_real_interest_rate_control    3.4599
## no_stocks_traded_control        -0.4448
## no_poverty_gap830_control        1.7902
## no_pop_growth_pct_control        .
## no_pop_largest_city_control      .
## no_pop_density_control           .
## no_democracy_control             .
## no_union_freedom_control         .
## no_rule_of_law_index_control     .
```

Variabelen med størst effekt ser ut til å være gdp. Rikere land har mindre korrupsjon. Ser også ut til at land med høy utdanning og mer demokratiske har mindre korrupsjon. Modellen satt bandt annet populasjon til null.

f

Gjenta c) og d) med Neural Net.Dersom neuralnet ikke konvergerer kan du sette stepmax = 1e+06 eller høyere

```r
set.seed(3110)
library(caret)

train_fold <- kfold(training, 10) |> select(-country, -iso3c,-Region) # Får feil med.

nnet.results <- c()
mse <- 1000000
decay <-  0
size <- 0

for (i in unique(train_fold$fold))
{

  train <- train_fold |>
    filter(fold != i)

  train_x <- train |>
    select(-CPI, -fold)

  train_y <- train |>
    select(CPI) |>
    pull()

  test <- train_fold |>
    filter(fold == i)

  test_x <- test |>
    select(-CPI, -fold)

  test_y <- test |>
    select(CPI) |>
    pull()

  grid <- expand.grid(size = c(1,2,3,4,5),
                      decay = c(0.01, 0.05, 0.1))

  nnet.mod <- train(
    train_x,
    train_y,
    tuneGrid = grid,
    method = "nnet",
    trace = F
  )

  nnet.preds <- predict(nnet.mod, test_x)

  nnet.mse <- mean((test_y -  nnet.preds)^2)

  nnet.results <- c(nnet.results, nnet.mse)

  if (nnet.mse < mse)
  {
    mse <- nnet.mse
```

```
    size <- nnet.mod$bestTune[1]
    decay <- nnet.mod$bestTune[2]
  }
}


size;decay
```

```
##    size
## 13    5
```

```
##    decay
## 13  0.01
```

NEsten akkurat samme som for Lasso bare at jeg bruker caret isteden og tuner to parametere decay og size.

```
grid <- expand.grid(size = size,
                      decay = decay)

NNET <- train(
  training |> select(-CPI, -country, -iso3c,-Region),
  training$CPI,
  method = "nnet",
  tuneGrid = data.frame(size = size, decay = decay),
  trace = F

)


x <- testing |> select(-CPI, -country, -iso3c,-Region)
NNET.pred <- predict(NNET,x)

y <- testing |> select(CPI) |> pull()
mse <- mean( (y - NNET.pred)^2 )


paste("OOS prediksjonsnoyaktighet NNET:", mse)
```

```
## [1] "OOS prediksjonsnoyaktighet NNET: 2202.67909909092"
```

Her ser det jo ut til at modellen er helt off, sammenlignet med Lasso. Veldig mye høyere MSE. Kanksje det har skjedd noe feil i algoritmen. Man vil i hvertfall foretrekke Lasso.

Har litt dårlig tid på eksamen, men jeg ville jo egnetlig kanskje gjort noe slik isteden. Brukt bootstrapping og den innebygde cv-en til caret.

nnet.results <- c() n <- 10

for (i in 1:n) { x <- bs(df) # Bootstrap indexer til test og training

training <- x[split,] testing <- x[-split,]

cv <- trainControl(method = “cv”) grid <- expand.grid(size = c(1,2,3,4,5,6,7,8), decay = c(0.01, 0.05, 0.1))

nnet.mod <- train(training[,-1], training[,1], trControl = cv, tuneGrid = grid, method = “nnet”, ntree = 500)

nnet.preds <- predict(nnet.mod, testing[,-1])

nnet.mse <- mean((testing[,1] - nnet.preds)^2)

nnet.results[i] <- nnet.mse

}

paste(“OOS mse”, mean(nnet.results)) paste(“OOS mse sd”, sd(nnet.results))

Oppgave 2

a

Det er tre krav for at en tidsrekke skal være stasjonær: konstant varians og forventning, samt at tidsperiodene ikke korrelerer sammen.

i) I dette plottet ser det ut til at vi har en konstant varians og forventning, men det ser også ut til at det er en underliggende trend, som bølger opp og ned. Brudd på siste krav. Altså ikke stasjonær.

ii)

Her ser vi tydlig en stasjonær tidsserie. Hvor alle tre kravene er oppfylt.

iii)

Her ser vi at variansen øker med med tiden. Det er ikke en stasjonær tidsrekke.

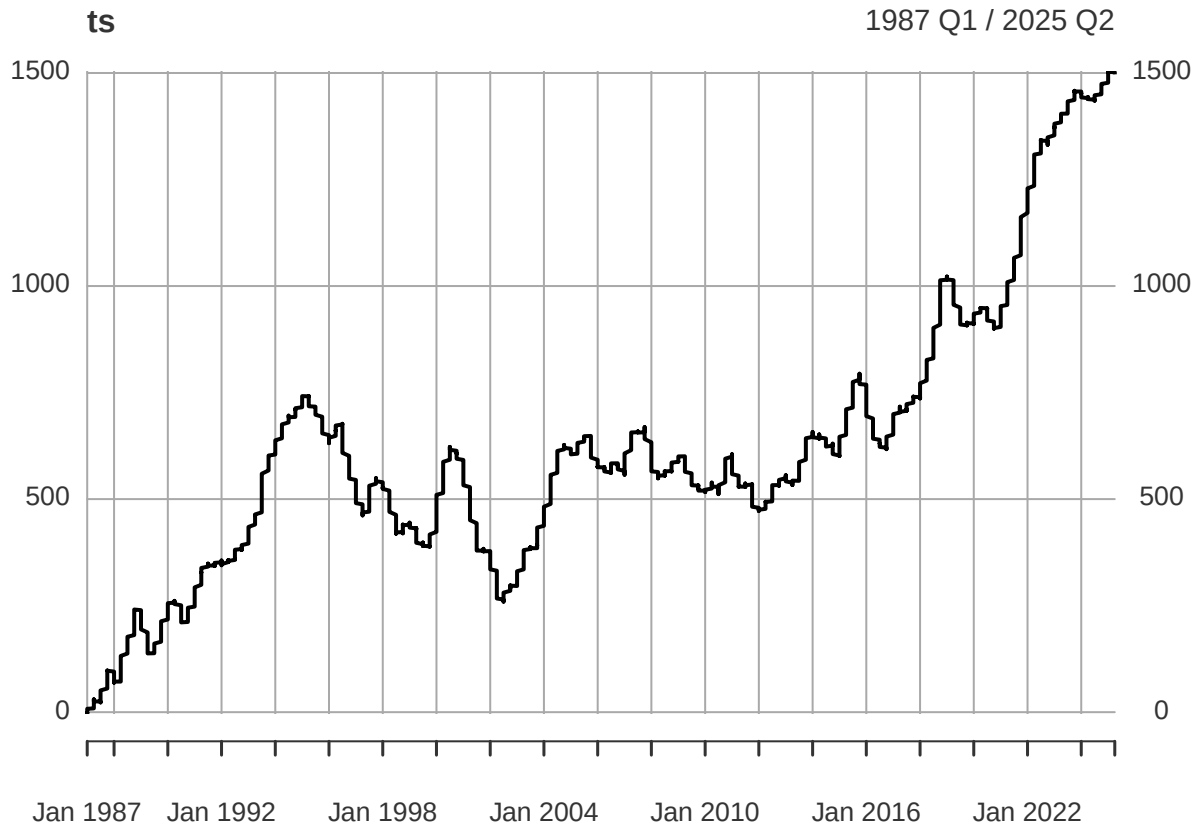
iv)

Her ser vi at forventningen ikke er konstant over tid. Ikke stasjonær.

b

Last inn og plot tidsserien time_series.csv.

```
library(tseries)
library(stats)
library(forecast)
library(quantmod)
library(lubridate)



time_series <- read.csv("data/time_series.csv")

time_series$Time <- as.yearqtr(time_series$Time, format = "%Y-%m-%d")

ts <- xts(time_series[,'Y'], order.by = time_series$Time)

plot(ts)
```



Her ser vi åpenbart at tidsserien ikke er stasjonær.

c

Test om tidsserien er stajonær, forklar hvilken test du bruker, hvordan du tolker resultatet, og hvordan du løser eventuelle utfordninger med tidsserien.

For å teste stasjonæritet bruker jeg en Dickey-Fuller test. Nullhypotesen er at serien har en unit-root, som vil si at den er ikke stasjonær mot alternativ hypotese om at den er stasjonær. Vi kan sette oss et fem prosent signifikansnivå og sier dermed at vi forkaster nullhypotesen dersom vi observerer en p-verdi under 5 prosent. Hvis tidsrekken ikke er stasjonær kan vi løse dette problemet ved å ta log og ta differansen av serien.

```
adf.test(ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts
## Dickey-Fuller = -1.7663, Lag order = 12, p-value = 0.6772
## alternative hypothesis: stationary
```

Forkaster ikke H0.

```r
ts_d1 <- ts(na.omit(diff(log(ts))))

adf.test(ts_d1)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_d1
## Dickey-Fuller = -10.018, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

Forkaster H0. Nå er tidsrekken stasjonær.

d

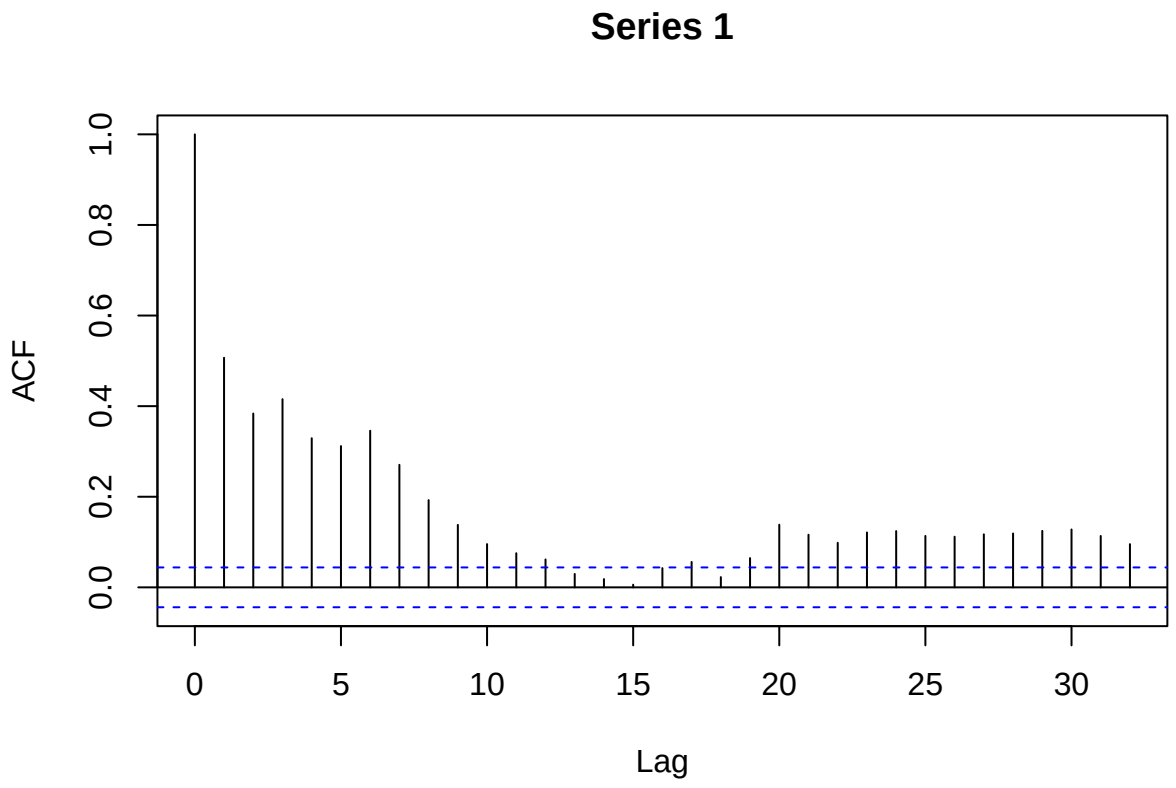
Del tidsserien opp i en treningsdel (de 80% første observasjonene) og en testdel (de 20% siste observasjonene)

```r
first_80 <- 1:floor(nrow(ts_d1)*0.8)

ts_d1_train <- ts_d1[first_80,]
ts_d1_test <- ts_d1[(floor(nrow(ts_d1)*0.8) + 1):nrow(ts_d1),]
```

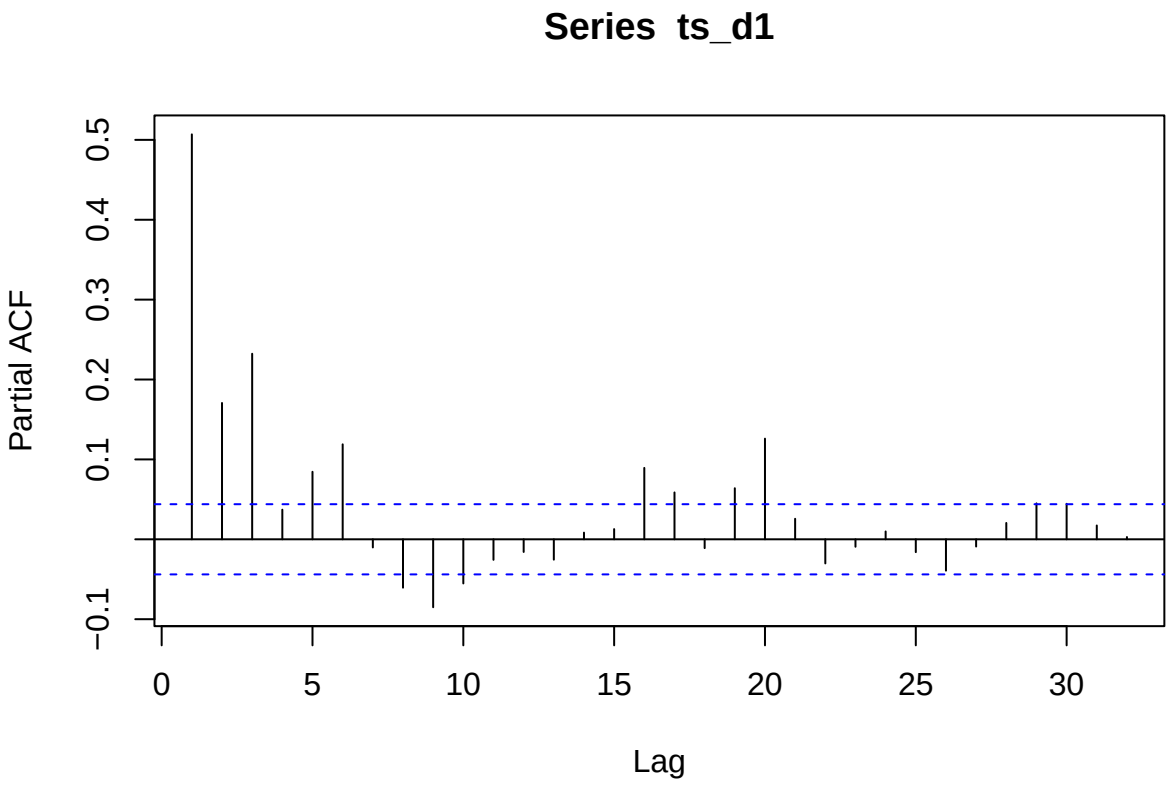
e

Lag et ACF og PACF plot for tidsserien. Estimer deretter en ARIMA(p, d, q) modell på treningsdatasettet.Du skal selv avgjøre verdiene påp,d, ogq. Husk å begrunne valget av verdienep,d, ogq.

```r
acf(ts_d1) # Får feil hvis ikke |> ts()
```




```
pacf(ts_d1) # Får feil hvis ikke |> ts()
```




```
arima1 <-  auto.arima(ts_d1_train ,
                      seasonal = F,
                      stationary = T,
                      stepwise = F)

arima1
```

```
## Series: ts_d1_train
## ARIMA(0,0,5) with non-zero mean
##
## Coefficients:
##          ma1     ma2     ma3     ma4     ma5    mean
##       1.9433  1.8538  1.8050  1.6939  0.7013  0.0064
## s.e.  0.0164  0.0310  0.0358  0.0306  0.0175  0.0031
##
## sigma^2 = 0.0001932:  log likelihood = 4547.42
## AIC=-9080.84   AICc=-9080.77   BIC=-9043.23
```

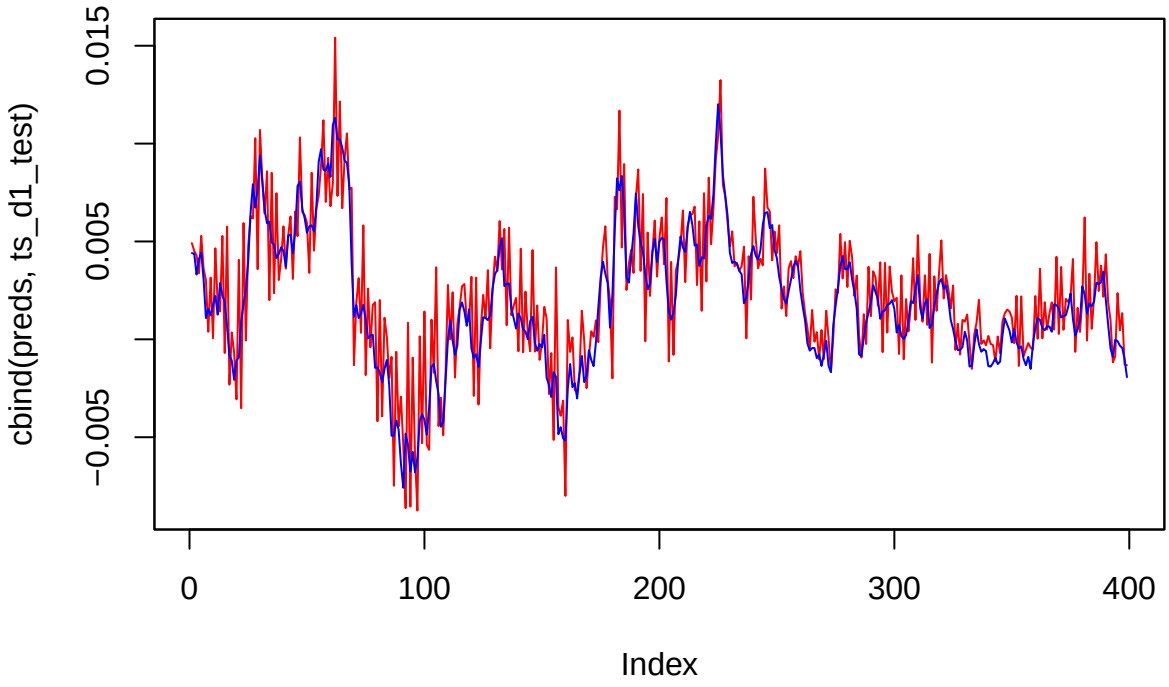
Ved å bruke auto.arima ser vi at modellen som blir valgt er en ARIMA(0,0,5) som vil si at det egentlig er en MA(5) modell.

f

Bruk den estimerte modellen fra c)til å lage 1-stegs framskrivninger ("1-step ahead forecasts") over heletest-datasettet. Det vil si at dere for hver periode t + 1 i testdatasettet skal bruke modellen fra c) til å predikere

. Lag et plot som viser både de faktiske verdiene og de predikerte verdiene.Hint: Bruk Arima() funksjonen for å predikere.

```
model <- Arima(ts_d1_test , model=arima1)
preds <- fitted(model)
plot.zoo(cbind(preds,ts_d1_test),plot.type = "single",
col = c("red", "blue"))
```



Hentet litt inspirasjon til plottet fra senosorveiledningen i fjor.

Oppgave 3

a)

For å finne effekten av Z på Y, mens man mistenker at kovariatene påvirker både Z og Y. Da kan man bruke DML. Kort sagt så deler man dataen inn i to deler. Estimerer E(y gitt x) og E(Z gitt x) med en maskinnærings algoritme på hver fold. Når de er estimert finner du residualene. Når du har funnet residualene kjører du en OLS for å finne effekten av Z på Y.

b)

Lasso er lineær modell, når den får mange variabler vil den velge å fjerne/sette til null de variablene som den ikke mener er viktige. Et problem ved dette er at hvis to variabler korrelerer så kan lasso sette en til null og den andre blir stående uten noen forklaring. Lasso vil i gjengjeld være lettere å tolke ettersom at den

er lineær i parameteren og man kan se hvor stor effekt variabler har. Neural net på den andre siden er mer black box. Vi vet ikke helt hva som skjer, men den kan i mye større grad finne komplekse forhold mellom variabler. Jeg hadde nok endt opp med å bruke neural net, fordi den i større grad finner interaksjon mellom variabler kontra den lineære modellen Lasso.

c)

Honesty betyr at vi deler dataen inn i to deler. En brukes til å lager splittene i et decision tree. Den andre brukes til å regne ut behandlingseffektene i hver gruppe. Du minimerer risikoen ved å få svar som kommer fra tilfeldighet.