HW6

2023-05-23

```r
# Load packages
pacman::p_load(dynlm, sandwich, lmtest, here)
```

2)

```r
# Set parameters
mu = 1
phi = 0.75
sigma = 1
n = 500

# Set seed and simulate process
set.seed(101)
y = mu + arima.sim(model = list(ar = phi), n = n)

# Using formula derived in handwritten notes, calculate long run variance
(LRV = sigma^2/(1-phi)^2)
```

```
## [1] 16
```

```r
# Calculate standard error
(se = sqrt(LRV/n))
```

```
## [1] 0.1788854
```

3)

```r
# Fit AR(1) model
ar1 = dynlm(y ~ L(y,1))
summary(ar1)
```

```
##
## Time series regression with "ts" data:
## Start = 2, End = 500
##
## Call:
## dynlm(formula = y ~ L(y, 1))
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -3.11444 -0.65392 -0.00975  0.61632  2.65135
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.18392    0.04817   3.818 0.000152 ***
## L(y, 1)      0.75143    0.02958  25.406  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9599 on 497 degrees of freedom
## Multiple R-squared:  0.565,  Adjusted R-squared:  0.5641
## F-statistic: 645.5 on 1 and 497 DF,  p-value: < 2.2e-16
```

```
# Calculate fitted parameters
phi_hat = ar1$coefficients[2]
sigma_hat = mean(ar1$residuals^2)

# Use these to calculate LRV
(LRV_hat = sigma_hat^2/(1-phi)^2)
```

```
## [1] 13.47752
```

```
# Calculate standard error
(se_hat = sqrt(LRV/n))
```

```
## [1] 0.1788854
```

5)

```
# Regress y on a constant to calculate newey-west standard errors
new = dynlm(y ~ 1)

# Calculate newey-west se
(newey = coeftest(new, vcov=NeweyWest(new, prewhite=FALSE)))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  0.73516    0.14888  4.9379 1.079e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(se_new = newey[2])
```

```
## [1] 0.1488822
```

```r
# Calculate LRV using these ses
(LRV_new = 500*se_new^2)
```

```
## [1] 11.08296
```

Hansen 14.20

a

```r
# Load data
fred_df = haven::read_dta(here("HW6", "FRED-MD.dta"))
# Make unemployment rate convenient time series
unrate = ts(fred_df$unrate, frequency=12, start=1959)

# Function for calculating various lengths of AR models and their AICs
ar_sim = function(p){
  mod = dynlm(unrate ~ L(unrate, 1:p), start=c(1960,1)) # p is number of lags
  aic = AIC(mod)
  return(list(mod, aic))
}

# Calculate for 1 through 8 lags
(models = lapply(1:8, ar_sim))
```

```
## [[1]]
## [[1]][[1]]
##
## Time series regression with "ts" data:
## Start = 1960(1), End = 2017(12)
##
## Call:
## dynlm(formula = unrate ~ L(unrate, 1:p), start = c(1960, 1))
##
## Coefficients:
##    (Intercept)  L(unrate, 1:p)
##        0.03113         0.99456
##
##
## [[1]][[2]]
## [1] -420.3042
##
##
## [[2]]
## [[2]][[1]]
##
## Time series regression with "ts" data:
## Start = 1960(1), End = 2017(12)
##
## Call:
## dynlm(formula = unrate ~ L(unrate, 1:p), start = c(1960, 1))
##
```

```
## Coefficients:
##     (Intercept)  L(unrate, 1:p)1  L(unrate, 1:p)2
##         0.03672          1.11070         -0.11702
##
##
## [[2]][[2]]
## [1] -427.9882
##
##
## [[3]]
## [[3]][[1]]
##
## Time series regression with "ts" data:
## Start = 1960(1), End = 2017(12)
##
## Call:
## dynlm(formula = unrate ~ L(unrate, 1:p), start = c(1960, 1))
##
## Coefficients:
##     (Intercept)  L(unrate, 1:p)1  L(unrate, 1:p)2  L(unrate, 1:p)3
##         0.05035          1.08020          0.17781         -0.26649
##
##
## [[3]][[2]]
## [1] -477.8072
##
##
## [[4]]
## [[4]][[1]]
##
## Time series regression with "ts" data:
## Start = 1960(1), End = 2017(12)
##
## Call:
## dynlm(formula = unrate ~ L(unrate, 1:p), start = c(1960, 1))
##
## Coefficients:
##     (Intercept)  L(unrate, 1:p)1  L(unrate, 1:p)2  L(unrate, 1:p)3
##         0.06106          1.03292          0.20937         -0.07172
## L(unrate, 1:p)4
##        -0.18079
##
##
## [[4]][[2]]
## [1] -499.298
##
##
## [[5]]
## [[5]][[1]]
##
## Time series regression with "ts" data:
## Start = 1960(1), End = 2017(12)
##
## Call:
```

```
## dynlm(formula = unrate ~ L(unrate, 1:p), start = c(1960, 1))
##
## Coefficients:
##     (Intercept)  L(unrate, 1:p)1  L(unrate, 1:p)2  L(unrate, 1:p)3
##        0.072216         1.003344         0.196562        -0.036887
## L(unrate, 1:p)4  L(unrate, 1:p)5
##       -0.004773        -0.170310
##
##
## [[5]][[2]]
## [1] -518.2109
##
##
## [[6]]
## [[6]][[1]]
##
## Time series regression with "ts" data:
## Start = 1960(1), End = 2017(12)
##
## Call:
## dynlm(formula = unrate ~ L(unrate, 1:p), start = c(1960, 1))
##
## Coefficients:
##     (Intercept)  L(unrate, 1:p)1  L(unrate, 1:p)2  L(unrate, 1:p)3
##         0.07933          0.98759          0.19671         -0.04076
## L(unrate, 1:p)4  L(unrate, 1:p)5  L(unrate, 1:p)6
##         0.01298         -0.07677         -0.09300
##
##
## [[6]][[2]]
## [1] -522.3805
##
##
## [[7]]
## [[7]][[1]]
##
## Time series regression with "ts" data:
## Start = 1960(1), End = 2017(12)
##
## Call:
## dynlm(formula = unrate ~ L(unrate, 1:p), start = c(1960, 1))
##
## Coefficients:
##     (Intercept)  L(unrate, 1:p)1  L(unrate, 1:p)2  L(unrate, 1:p)3
##         0.08413          0.98224          0.19235         -0.03967
## L(unrate, 1:p)4  L(unrate, 1:p)5  L(unrate, 1:p)6  L(unrate, 1:p)7
##         0.01036         -0.06572         -0.03616         -0.05745
##
##
## [[7]][[2]]
## [1] -522.7277
##
##
## [[8]]
## [[8]][[1]]
##
## Time series regression with "ts" data:
## Start = 1960(1), End = 2017(12)
##
## Call:
## dynlm(formula = unrate ~ L(unrate, 1:p), start = c(1960, 1))
##
## Coefficients:
##     (Intercept)  L(unrate, 1:p)1  L(unrate, 1:p)2  L(unrate, 1:p)3
##       0.0841171        0.9822538        0.1923565       -0.0396568
## L(unrate, 1:p)4  L(unrate, 1:p)5  L(unrate, 1:p)6  L(unrate, 1:p)7
##       0.0103606       -0.0657103       -0.0361923       -0.0576041
## L(unrate, 1:p)8
##       0.0001603
##
##
## [[8]][[2]]
## [1] -520.7277
```

b

```
# Print AICs
(aic = sapply(1:8, function(x){models[[x]][[2]]}))
```

```
## [1] -420.3042 -427.9882 -477.8072 -499.2980 -518.2109 -522.3805 -522.7277
## [8] -520.7277
```

c

```
# Find minimum AIC
which.min(aic) # the AR(7) model has the lowest AIC
```

```
## [1] 7
```

d

```
summary(dynlm(unrate ~ L(unrate, 1:7), start=c(1960,1)))
```

```
##
## Time series regression with "ts" data:
## Start = 1960(1), End = 2017(12)
##
## Call:
## dynlm(formula = unrate ~ L(unrate, 1:7), start = c(1960, 1))
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max 
## -0.56971 -0.10061 -0.00689  0.09910  0.72160 
## 
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)    
## (Intercept)      0.08413    0.02541   3.311 0.000977 ***
## L(unrate, 1:7)1  0.98224    0.03769  26.064  < 2e-16 ***
## L(unrate, 1:7)2  0.19235    0.05285   3.640 0.000294 ***
## L(unrate, 1:7)3 -0.03967    0.05327  -0.745 0.456728    
## L(unrate, 1:7)4  0.01036    0.05329   0.194 0.845852    
## L(unrate, 1:7)5 -0.06572    0.05331  -1.233 0.218051    
## L(unrate, 1:7)6 -0.03616    0.05289  -0.684 0.494375    
## L(unrate, 1:7)7 -0.05745    0.03768  -1.524 0.127845    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.165 on 688 degrees of freedom
## Multiple R-squared:  0.9892, Adjusted R-squared:  0.9891 
## F-statistic:  9043 on 7 and 688 DF,  p-value: < 2.2e-16
```