Self directed activities 7

Josh Copeland (SID: 1444772)

2025-09-15

```
#Packages used:

# library(urca)
# library(forecast)
# library(tidyverse)
```

Q1

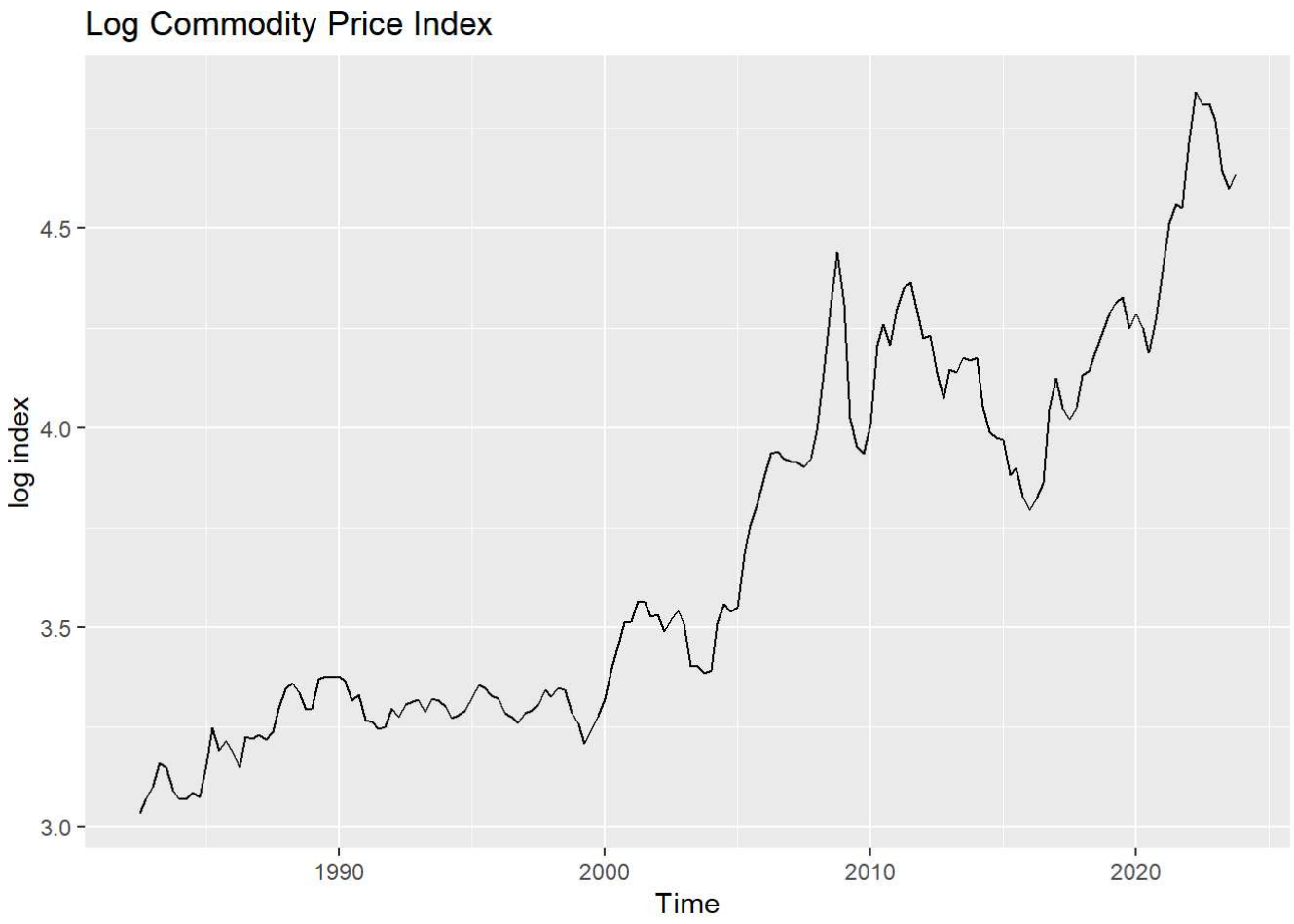
Read in the data file and set up time series as below to implement this forecasting exercise.

```
# Data read

dt <- read.csv("QtrlyCommodityPriceIndex.csv")

# log of commodity price index, 1982q3 to 2024q4
P <- ts(dt$CommodityPriceIndex, start=c(1982,3),
        end=c(2024,4), frequency=4)
Y <- log(P)
DY <- diff(Y)

# 2024 observations for forecast evaluation
Pf <- window(P, start=c(2024,1), end=c(2024,4))
Yf <- window(Y, start=c(2024,1), end=c(2024,4))
DYf <- window(DY, start=c(2024,1), end=c(2024,4))

# Estimation sample ends 2023q4
Y <- window(Y, end=c(2023,4))
DY <- window(DY, end=c(2023,4))

par(mfrow=c(2,1))
```

Make time series plots of Y and DY and carry out ADF tests to decide whether / how much to difference Y prior to AR modelling.

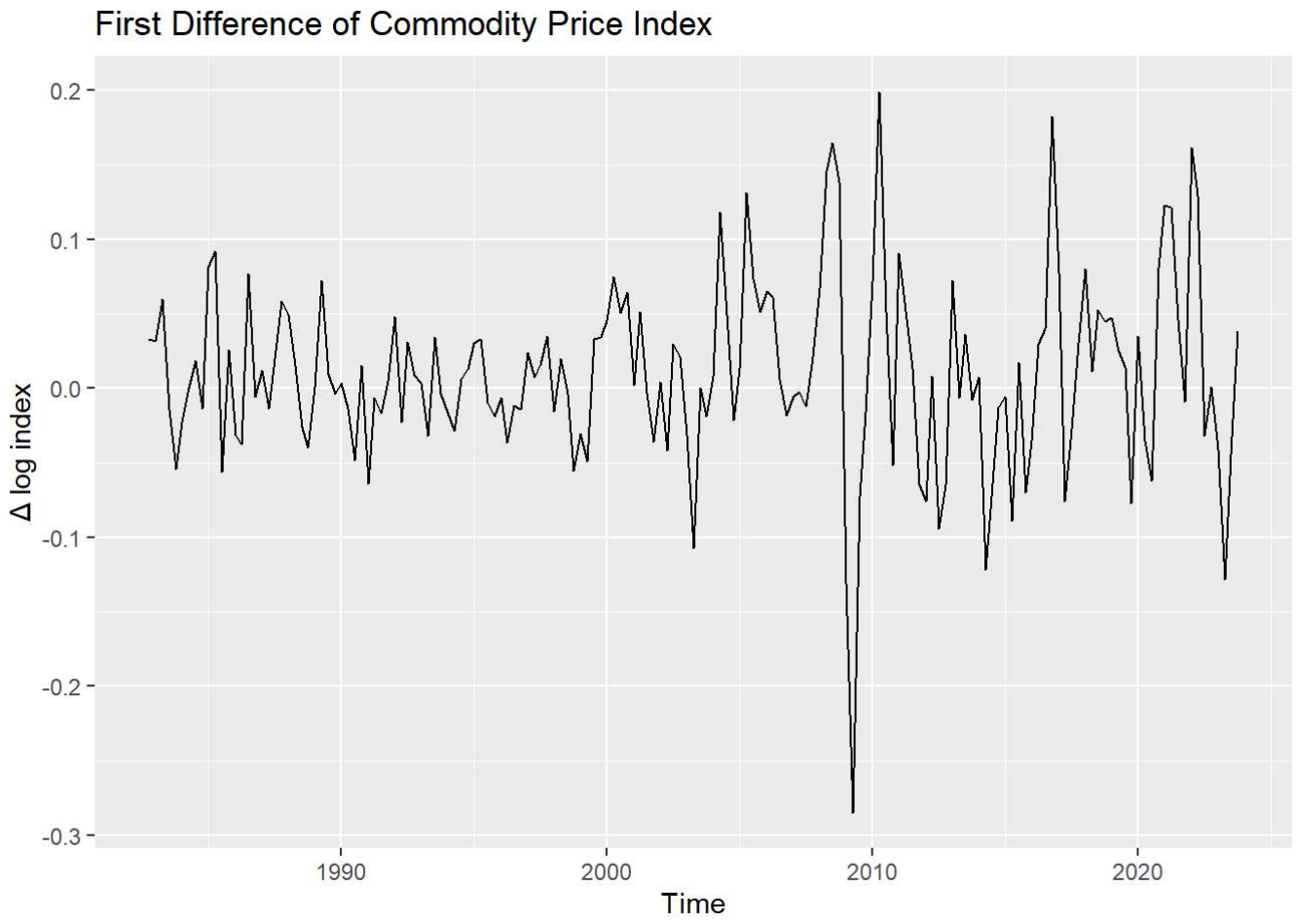
```
# Plot Y and DY
autoplot(Y, ts.colour="blue") +
  ggtitle("Log Commodity Price Index") +
  ylab("log index")
```

```
## Warning in ggplot2::geom_line(na.rm = TRUE, ...): Ignoring unknown parameters:
## `ts.colour`
```




```
autoplot(DY, ts.colour="red") +
  ggtitle("First Difference of Commodity Price Index") +
  ylab("Δ log index")
```

```
## Warning in ggplot2::geom_line(na.rm = TRUE, ...): Ignoring unknown parameters:
## `ts.colour`
```




```
# ADF test on Y (with trend)
ADF_Y <- ur.df(Y, type="trend", lags=9, selectlags="AIC")
summary(ADF_Y)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.172254 -0.031826 -0.000636  0.033756  0.183248
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.1530861  0.0781072   1.960   0.0519 .
## z.lag.1     -0.0513383  0.0266246  -1.928   0.0557 .
## tt           0.0005397  0.0002630   2.052   0.0419 *
## z.diff.lag1  0.4635813  0.0793213   5.844 3.09e-08 ***
## z.diff.lag2 -0.2028338  0.0879762  -2.306   0.0225 *
## z.diff.lag3  0.0928911  0.0881354   1.054   0.2936
## z.diff.lag4 -0.1952702  0.0825338  -2.366   0.0193 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05617 on 149 degrees of freedom
## Multiple R-squared:  0.2444, Adjusted R-squared:  0.214
## F-statistic: 8.034 on 6 and 149 DF,  p-value: 1.598e-07
##
##
## Value of test-statistic is: -1.9282 2.87 2.1085
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
```

```
# ADF test on DY (with drift)
ADF_DY <- ur.df(DY, type="none", lags=9, selectlags="AIC")
summary(ADF_DY)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.175908 -0.025432  0.007806  0.043225  0.190978
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## z.lag.1     -0.87294    0.11822  -7.384 9.63e-12 ***
## z.diff.lag1  0.34454    0.10586   3.255   0.0014 **
## z.diff.lag2  0.11686    0.09104   1.284   0.2012
## z.diff.lag3  0.20252    0.08015   2.527   0.0125 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05695 on 151 degrees of freedom
## Multiple R-squared:  0.3575, Adjusted R-squared:  0.3405
## F-statistic: 21.01 on 4 and 151 DF,  p-value: 8.689e-14
##
##
## Value of test-statistic is: -7.3841
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

Reject the null hypothesis on the ADF test on the differenced series. So difference it once.

Q2

In the usual way, select an appropriate AR(p) model for DY.

```
pmax <- 9
Stats <- matrix(
  NA_real_,
  nrow = pmax + 1,
  ncol = 2,
  dimnames = list(paste0("p=", 0:pmax), c("LBp", "AICc"))
)

for (p in 0:pmax) {
  eq <- forecast::Arima(DY, order = c(p, 0, 0))
  Stats[p + 1, "LBp"]  <- stats::Box.test(eq$residuals, lag = p + 4,
                                          type = "Ljung-Box", fitdf = p)$p.value
  Stats[p + 1, "AICc"] <- eq$aicc
}

# Convert to data.frame for easier sorting
Stats_df <- as.data.frame(Stats)
Stats_df$p <- 0:pmax

# Order by AICc ascending (most negative at top)
Stats_sorted <- Stats_df[order(Stats_df$AICc), ]

print(round(Stats_sorted, 4))
```

```
##        LBp      AICc p
## p=4 0.9969 -478.6385 4
## p=5 0.8900 -476.4728 5
## p=6 0.8155 -474.5138 6
## p=2 0.0727 -474.4984 2
## p=3 0.0716 -472.5139 3
## p=7 0.4532 -472.3617 7
## p=8 0.5688 -470.6327 8
## p=1 0.0094 -469.4291 1
## p=9 0.5759 -468.7845 9
## p=0 0.0000 -445.7938 0
```

The preferred p value is 4, which is supported by being unable to reject the null hypothesis of the LB test - no evidence of autocorrelation in its residuals.

Now to calculate the forecasts:

```
# Preferred AR(p) order
p <- 4

# Maximum forecast horizon:
hmax <- length(DYf)

# Estimate selected model:
eq <- Arima(DY, order=c(p,0,0))

# Recursive forecasts out to horizon hmax:
DYf1 <- forecast(eq, h=hmax)$mean
```

Q3

Implement the direct forecasting approach to compute forecasts for DY for each of the four quarters of 2024

```
# Direct forecasting - model selection

# Storage for AICc and Ljung-Box p-values
AICc <- matrix(nrow=1+pmax, ncol=hmax)
rownames(AICc) <- paste0("p=",0:pmax)
colnames(AICc) <- paste0("h=",1:hmax)
LBp <- AICc

# Loop over all forecast horizons
for (h in 1:hmax){

  # Loop over all possible AR lag orders
  for (p in c(0,h:pmax)){

    # Estimate AR(p)
    # zeros has same length as parameters in AR(p)+intercept
    zeros <- rep(NA,p+1)
    # coefficients on first h-1 lags set to zero
    if (p>0 & h>1){
      zeros[1:(h-1)] <- 0
    }
    eq <- Arima(DY, order=c(p,0,0), fixed=zeros)
    AICc[1+p,h] <- eq$aicc

    # Ljung-Box statistic, include 4 lags beyond AR order
    LB <- Box.test(eq$residuals, lag=p+4, type="Ljung-Box")$statistic
    # Remove first h-1 lags from LB statistic
    if (h>1){
      LB <- LB-Box.test(eq$residuals, lag=h-1, type="Ljung-Box")$statistic
    }
    # p-value for LB test
    LBp[1+p,h] <- pchisq(LB, df=4, lower.tail=FALSE)

  }
}
print(round(LBp,4))
```

```
##        h=1    h=2    h=3    h=4
## p=0 0.0000 0.0195 0.0209 0.0519
## p=1 0.0094     NA     NA     NA
## p=2 0.0727 0.0027     NA     NA
## p=3 0.0716 0.0193 0.0168     NA
## p=4 0.9969 0.6560 0.7005 0.7527
## p=5 0.8900 0.8823 0.8734 0.8601
## p=6 0.8155 0.7801 0.7640 0.7459
## p=7 0.4532 0.3484 0.3344 0.3226
## p=8 0.5688 0.4443 0.4420 0.4359
## p=9 0.5759 0.3765 0.3925 0.3949
```

```
print(round(AICc,2))
```

```
##         h=1     h=2     h=3     h=4
## p=0 -445.79 -445.79 -445.79 -445.79
## p=1 -469.43      NA      NA      NA
## p=2 -474.50 -443.87      NA      NA
## p=3 -472.51 -443.81 -445.88      NA
## p=4 -478.64 -449.33 -451.31 -453.26
## p=5 -476.47 -448.36 -450.35 -452.14
## p=6 -474.51 -446.34 -448.28 -450.08
## p=7 -472.36 -444.38 -446.36 -448.07
## p=8 -470.63 -442.81 -444.78 -446.55
## p=9 -468.78 -441.35 -443.31 -445.06
```

```
##### Automate selecting the optimal p for each direct forecast

# Convert matrices to long tibble
sel_tbl <- as.data.frame(AICc) |>
  tibble::rownames_to_column("p") |>
  tidyr::pivot_longer(-p, names_to = "h", values_to = "AICc") |>
  mutate(LBp = as.vector(LBp),
         p = as.integer(sub("p=", "", p)),
         h = as.integer(sub("h=", "", h)))

# For each horizon, pick the p with min AICc among LBp > 0.05;
# if none pass, take the global min AICc
ph_tbl <- sel_tbl |>
  group_by(h) |>
  summarise(
    ph = if (any(LBp > 0.05)) {
      p[which.min(AICc + ifelse(LBp > 0.05, 0, Inf))]
    } else {
      p[which.min(AICc)]
    },
    .groups = "drop"
  )

# As in the recursive case, all models select AR(4)
ph <- ph_tbl$ph
print(ph_tbl)
```

```
## # A tibble: 4 × 2
##       h    ph
##   <int> <int>
## 1     1     4
## 2     2     4
## 3     3     4
## 4     4     4
```

```
zeros <- rep(NA,p+1)

p <- 4
h <- 3

zeros <- rep(NA,p+1)
zeros[1:(h-1)] <- 0
print(zeros)
```

```
## [1]  0  0 NA NA NA
```

```
# Ljung-Box statistic, include 4 lags beyond AR order
LB <- Box.test(eq$residuals, lag=p+4, type="Ljung-Box")$statistic

# Remove first h-1 lags from LB statistic
if (h>1){
  LB <- LB-Box.test(eq$residuals, lag=h-1, type="Ljung-Box")$statistic
}
# p-value for LB test
LBp[1+p,h] <- pchisq(LB, df=4, lower.tail=FALSE)

# Direct forecasting - forecasts
# ph contains the AR lag order for each h


# ONce the AR lag order are selected for each h, compute direct forecasts for each quarter of
2024.

DYf2 <- DYf1
for (h in 1:hmax){
  zeros <- c(rep(0,h-1),rep(NA,ph[h]-h+2))
  eq <- Arima(DY, order=c(ph[h],0,0), fixed=zeros)
  DYf2[h] <- forecast(eq, h=h)$mean[h]
}
```

Q4

Calculate RMSEs to compare the accuracy of the recursive and indirect forecasts of DY over the four quarters of 2024.

It shows the first method is slightly more accurate - but not super matierally.

```
RMSE_DY_Recursive <- sqrt(mean((DYf-DYf1)^2))
RMSE_DY_Direct <- sqrt(mean((DYf-DYf2)^2))

print(RMSE_DY_Recursive)
```

```
## [1] 0.07577184
```

```
print(RMSE_DY_Direct)
```

```
## [1] 0.07345554
```

Q5

Now convert these growth rates to forecast levels via cumulative sums. Comparing outputs shows direct forecasts always incur a lower RMSE.

```
# Last observed Y (log index)
n <- length(Y)

# Convert forecasts of ΔY into forecasts of Y
Yf1 <- Y[n] + cumsum(DYf1)   # recursive
Yf2 <- Y[n] + cumsum(DYf2)   # direct

# Convert forecasts of Y into forecasts of P
Pf1 <- exp(Yf1)
Pf2 <- exp(Yf2)

# 1. RMSEs in differences (ΔY)
RMSE_DY_Recursive <- sqrt(mean((DYf - DYf1)^2))
RMSE_DY_Direct    <- sqrt(mean((DYf - DYf2)^2))

# 2. RMSEs in log-levels (Y)
RMSE_Y_Recursive  <- sqrt(mean((Yf - Yf1)^2))
RMSE_Y_Direct     <- sqrt(mean((Yf - Yf2)^2))

# 3. RMSEs in original levels (P)
RMSE_P_Recursive  <- sqrt(mean((Pf - Pf1)^2))
RMSE_P_Direct     <- sqrt(mean((Pf - Pf2)^2))

# 4. Combine into a tidy table
rmse_tbl <- tibble(
  Variable = c("differences", "log-levels", "index levels"),
  Recursive = c(RMSE_DY_Recursive, RMSE_Y_Recursive, RMSE_P_Recursive),
  Direct    = c(RMSE_DY_Direct,    RMSE_Y_Direct,    RMSE_P_Direct)
)

# 5. Add column showing which method is more accurate
rmse_tbl <- rmse_tbl %>%
  mutate(More_Accurate = ifelse(Recursive < Direct, "Recursive", "Direct"))

print(rmse_tbl)
```

```
## # A tibble: 3 × 4
##   Variable     Recursive  Direct More_Accurate
##   <chr>            <dbl>   <dbl> <chr>
## 1 differences     0.0758  0.0735 Direct
## 2 log-levels      0.182   0.173  Direct
## 3 index levels   18.8    17.8    Direct
```