Macroeconomic Forecasting with Error Correction and Distributed Lag Models: A Practial Guide

Sebastian Krantz

January 28, 2021

Abstract

This short paper introduces and evaluates an error correction model that successfully forecasts quarterly revenues in Uganda conditional on the GDP forecast. It also lays down a general methodology with accompanying R code to speficy, estimate and evaluate models of the distributed lag (DLM) or error correction (ECM) class that forecast one economic time series based on one or more forecasted series. The principles of forecast evaluation demonstrated in this paper are general and also apply to univariate time series models of the ARIMA or ETS class as well as ARIMAX models not covered in this paper.

Introduction

A first step in approaching any forecasting problem is to decide on an appropriate methodology to tackle the problem. For tax revenues there are many suggested approaches. The IMF *Financial Programming Manual* reviews 3 of them: (i) the effective tax rate approach; (ii) the elasticity approach; and (iii) the regression approach. Approach (iii) typically results in the most accurate short-term forecasts. The simple regression approach stipulates a distributed lag model (DLM) where tax revenue is regressed on its own lags and GDP with some lags.

In the absence of large abrupt shifts in the tax base, domestic revenue can be assumed to have a linear relationship with GDP. Since however both revenue and GDP are typically non-stationary series, this relationship often takes the form of cointegration. The correct way to deal with cointegrated variables is to specify and error correction model (ECM). There are in fact 3 possible models depending on the relationship of revenue and GDP: If both are tend-stationary, a DLM in levels is the appropriate model. If one of the series is non-stationary or both are non-stationary but not cointegrated, a DLM in first-differences is appropriate. If however both are non-stationary and cointegrated, an ECM is the right choice.

Examining the Data

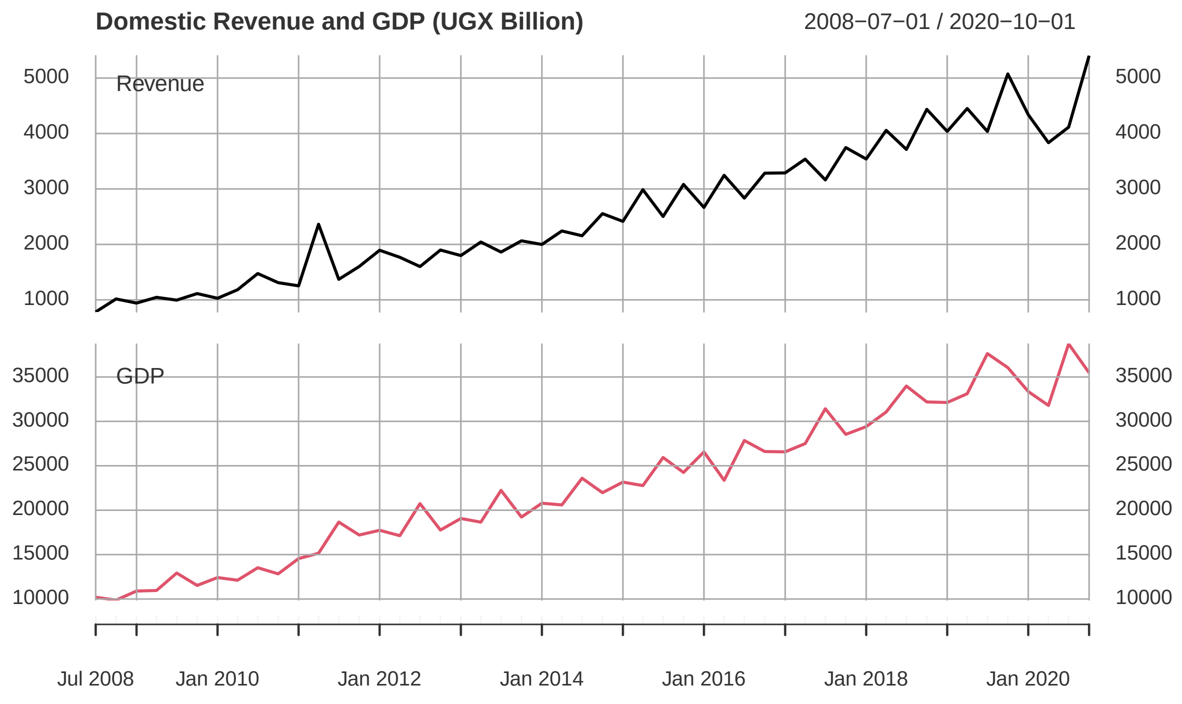
It is thus necessary to start by examining the the data and running some tests in order to determine the appropriate modeling approach.

```r
library(readxl)      # Import from Excel
library(collapse)    # Data transformation and time series operators
library(magrittr)    # Pipe operators %>%, %$%
library(tseries)     # Time series tests
library(lmtest)      # Linear model tests
library(sandwich)    # Robust standard errors
library(jtools)      # Enhanced regression summary
library(xts)         # Extensible time series + pretty plots

# Set working directory
setwd("C:/Users/Sebastian Krantz/Dropbox/MoFPED/Revenue Forecasting/")
```

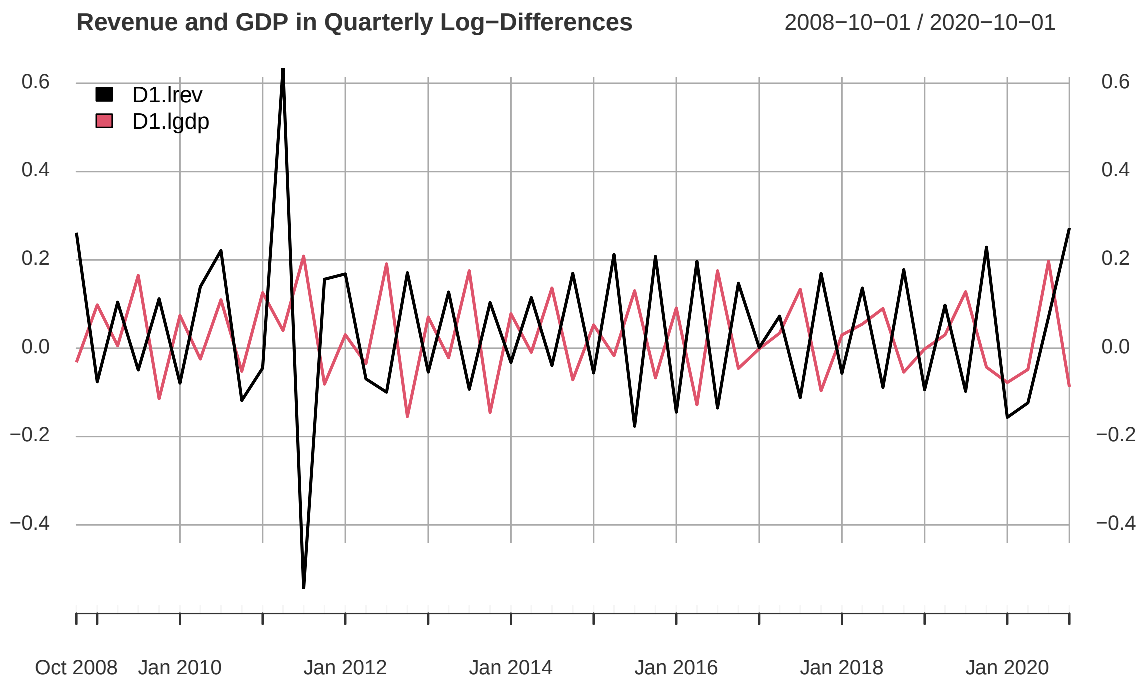
```r
# Load data from a prepared excel file
data <- read_xlsx("Quarterly Revenue Forecast 2020-21/GDP_REV_Q.xlsx")

# Show the first lines of the data
head(data, 3)
## # A tibble: 3 x 22
##   Date                FY    Period Type    REV GDP_KPR GDP_CUPR DEFL1617
##   <dttm>              <chr> <chr>  <chr> <dbl>   <dbl>    <dbl>    <dbl>
## 1 2008-07-01 00:00:00 2008~ Q1     OLD    783.  19228.   10201.     53.2
## 2 2008-10-01 00:00:00 2008~ Q2     OLD   1017.  17752.    9881.     55.9
## 3 2009-01-01 00:00:00 2008~ Q3     OLD    942.  17308.   10900.     58.4
## # ... with 14 more variables: GDP_KP_CPR <dbl>, G_DEFL1617 <dbl>,
## #   G_DEF_YOY <dbl>, IFL_Food_YOY <dbl>, IFL_EFU_YOY <dbl>, IFL_Core_YOY <dbl>,
## #   IFL_HL_YOY <dbl>, CPI_Food <dbl>, CPI_EFU <dbl>, CPI_Core <dbl>,
## #   CPI_HL <dbl>, DEFL1617_pred <dbl>, DEFL1617_comb <dbl>, GDP_CP_comb <dbl>
```

The following R code creates a time series matrix `X` containing the natural log of quarterly revenue and a combined quarterly GDP series at current prices starting in Q3 2008 with outcomes until Q4 2020, and forecasted GDP until Q1 2025. The series are cast in logs as this helps reduce the effects of changes in the variance of series over time on the forecast (leading to heteroskedastic error terms).

```r
# Creating an xts time series matrix X
X <- data %$% xts(cbind(lrev = log(REV),
                        lgdp = log(GDP_CP_comb)),
                  order.by = as.Date(Date),
                  frequency = 4L)

# Plotting the raw data
na.omit(X) %>% exp %>% setColnames(.c(Revenue, GDP)) %>%
plot(multi.panel = TRUE, yaxis.same = FALSE,
     main = "Domestic Revenue and GDP (UGX Billion)",
     major.ticks = "years", grid.ticks.on = "years")
```



```r
# Plotting the log-differenced data
plot(na.omit(D(X)), legend.loc = "topleft",
     main = "Revenue and GDP in Quarterly Log-Differences",
     major.ticks = "years", grid.ticks.on = "years")
```



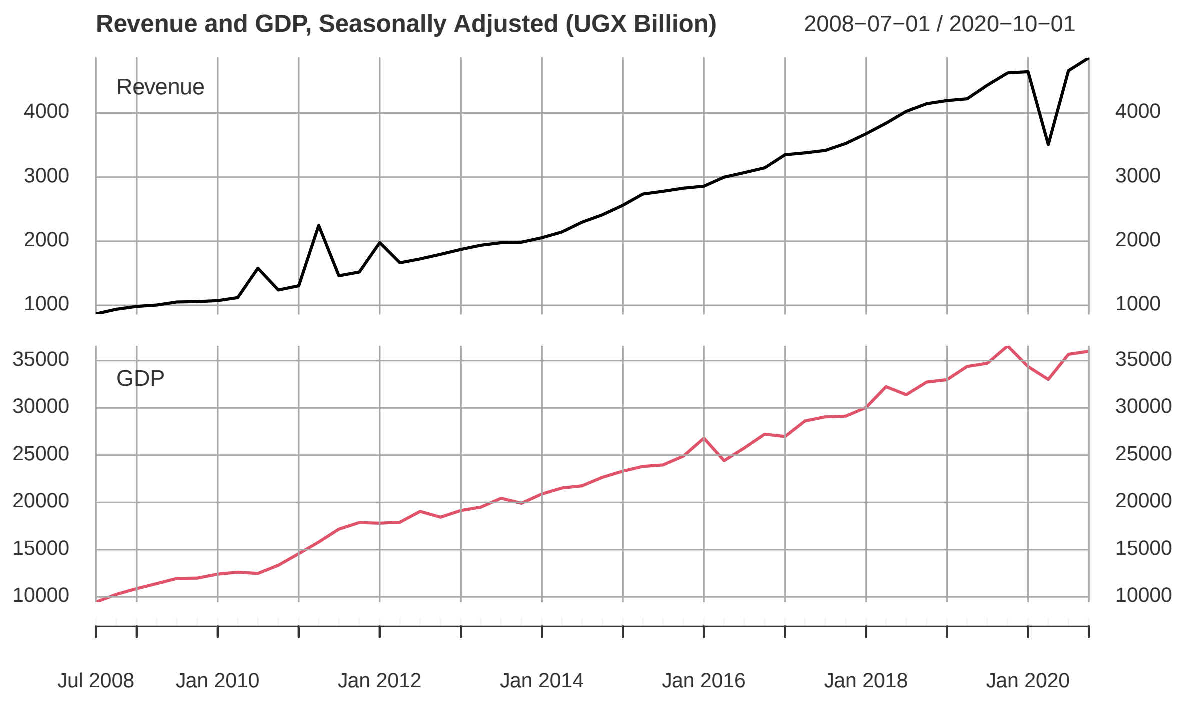
The plots suggest that both series exhibit seasonal patterns. The `seastests` package provides several seasonality tests, including the Webel-Ollech overall seasonality test that combines results from different seasonality tests.

```r
library(seastests)
# Check for seasonality based on Webel-Ollech overall seasonality test
isSeasonal(X[, "lrev"])
## [1] TRUE
isSeasonal(X[, "lgdp"])
## [1] TRUE
```

Since both series exhibit seasonal patterns, it is better to keep the seasonality in the series and potentially estimate a model with additional seasonal lags, than to remove seasonality beforehand using a statistical procedure. In cases where only one of the series used exhibits seasonality, and in particular if it is a covariate series and the variable to be forecasted does not exhibit seasonal patterns, it may be preferable to deseasonalize that series beforehand. For completeness I provide the code to deseasonalize a series using R's interface to the X-13-ARIMA-SEATS seasonal adjustment software by the US Census Bureau available through the `seasonal` package. The software by default uses automatic ARIMA model search and outlier detection to model a time series and then decomposes the model to remove the seasonal component. It seems to produce better seasonal adjustment results than software currently used in Uganda Bureau of Statistics.

```r
library(seasonal) # Seasonal adjustment using X-13 ARIMA SEATS
X_sa <- dapply(X, function(y) { # Need to specify time series parameters:
          y <- ts(y, start = c(2008L, 3L), frequency = 4L)
          y[!is.na(y)] <- predict(seas(y)) # seas() removes missing values
          return(y)
        })
```

```
# Seasonally adjusted data
na.omit(exp(X_sa)) %>% setColnames(.c(Revenue, GDP)) %>%
plot(multi.panel = TRUE, yaxis.same = FALSE,
     main = "Revenue and GDP, Seasonally Adjusted (UGX Billion)",
     major.ticks = "years", grid.ticks.on = "years")
```



Again since both series exhibit seasonal patterns, we move ahead with the unadjusted series. The next step is to investigate other properties of the series such as stationarity and cointegration. To test for stationarity we can employ two tests: The Augmented Dickey Fuller (ADF) test tests the null of non-stationarity against the alternative of trend-stationarity. The Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test tests the null of trend-stationarity. Since both tests have low statistical power, results where both tests point in the same direction provide a good indication about the stationarity or non-stationarity of a series.

```
# ADF test of the Null of a unit root
adf.test(na.omit(X[, "lrev"]))
##
##  Augmented Dickey-Fuller Test
##
## data:  na.omit(X[, "lrev"])
## Dickey-Fuller = -0.84745, Lag order = 3, p-value = 0.9517
## alternative hypothesis: stationary

adf.test(na.omit(X[, "lgdp"]))
##
##  Augmented Dickey-Fuller Test
##
## data:  na.omit(X[, "lgdp"])
## Dickey-Fuller = -2.2619, Lag order = 4, p-value = 0.4688
## alternative hypothesis: stationary
```

```
# KPSS test of the null of trend-stationarity
kpss.test(na.omit(X[, "lrev"]), null = "Trend")
##
##  KPSS Test for Trend Stationarity
##
## data:  na.omit(X[, "lrev"])
## KPSS Trend = 0.19022, Truncation lag parameter = 3, p-value = 0.01967

kpss.test(na.omit(X[, "lgdp"]), null = "Trend")
##
##  KPSS Test for Trend Stationarity
##
## data:  na.omit(X[, "lgdp"])
## KPSS Trend = 0.29783, Truncation lag parameter = 3, p-value = 0.01
```

In this case both tests indicate that both series are non-stationary. In many cases however these tests may disagree with each other and then it is necessary to explore further or make an informed judgment. As both series are non-stationary, it is also necessary to test for cointegration to decide whether an ECM should be specified, or a DLM in first differences is appropriate.

The Phillips-Ouliaris (PO) test tests the null hypothesis that the variables are not cointegrated. Strictly speaking we would first have to ascertain that both series are integrated of the same order before testing for cointegration, but with very few exceptions (such as price or stock market series), economic time series are I(1), so we refrain here from further testing and assume the differenced series are stationary.

```
# Phillips-Ouliaris Test of the Null of No Cointegration
po.test(X)
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  X
## Phillips-Ouliaris demeaned = -74.245, Truncation lag parameter = 0,
## p-value = 0.01
```

The test rejects the null of no cointegration, thus the ECM appears to be the correct model in this case. For completeness sake I note that the PO test follows the 2-Step approach of Engele & Granger by regressing the first series on the other(s) and testing the residuals for a unit root. The test outcome thus depends on which is the first series in the dataset. The more sophisticated Johannsen test, which is preferable especially in multivariate settings is available in the `urca` package via the `ca.jo()` function.

Estimating an ECM

For illustration purposes this section presents a simple ECM estimated using Engele & Granger's 2-step approach, whereas the next section estimates and compares a series of different ECM specifications estimated in one step, of which the best model will be used for forecasting. In the 2-step approach first the cointegration equation is estimated.

```
# This estimates the cointegration equation
cieq <- lm(lrev ~ lgdp, X)

# Summarizing with heteroskedasticity and autocorrelation consistent (HAC) errors
summ(cieq, digits = 4L, vcov = vcovHAC(cieq))
```

Then the residuals are obtained, and included in the regression in first differences.

```
# Plot residuals of cointegration equation
res <- resid(cieq) %>% as.xts(dateFormat = "Date")
```

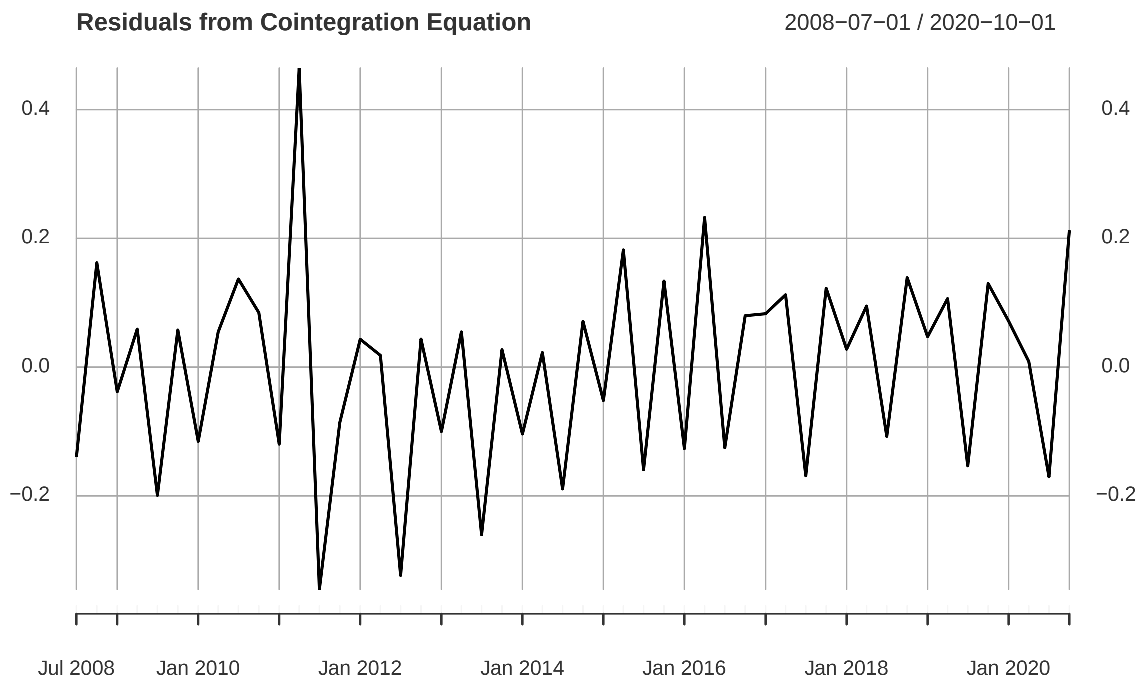
Observations	50 (18 missing obs. deleted)
Dependent variable	lrev
Type	OLS linear regression

F(1,48)	498.0640
R^2	0.9121
Adj. R^2	0.9103

	Est.	S.E.	t val.	p
(Intercept)	-4.8951	0.2979	-16.4313	0.0000
lgdp	1.2673	0.0297	42.6291	0.0000

Standard errors: User-specified

```r
plot(res, main = "Residuals from Cointegration Equation",
     major.ticks = "years", grid.ticks.on = "years")
```



```r
# Testing residuals: Stationary (same conclusion as po.test)
kpss.test(res)
##
##  KPSS Test for Level Stationarity
##
## data:  res
## KPSS Level = 0.3292, Truncation lag parameter = 3, p-value = 0.1
```

The plot of the cointegration residuals suggests that they contain a significant portion of the mismatch in seasonal variation in the two series, which is visible in the second graph of the paper visualizing the two series in log-differences.

Apart from a cointegration relationship which governs the medium-term relationship of revenue and GDP, and, in this case, the mismatch in seasonal variation, revenue may also be affected by past revenue collection and short-term fluctuations in GDP. The general form of a bivariate ECM is:

$$A(L)\Delta r_t = \gamma + B(L)\Delta y_t + \alpha(r_{t-t} - \beta_0 - \beta_1 y_{t-1}) + v_t, \tag{1}$$

where

$$A(L) = 1 - \sum_{i=1}^{p} L^i = 1 - L - L^2 - \cdots - L^p,$$
$$B(L) = \sum_{i=0}^{q} L^i = 1 + L + L^2 + \cdots + L^q$$

are polynomials in the lag operator L of order p and q, respectively. The term $r_{t-t} - \beta_0 - \beta_1 y_{t-1}$ represents the lagged residual from the cointegration equation $r_t = \beta_0 + \beta_1 y_t + \epsilon_t$ computed above, which can be interpreted as the deviation of r_t from it's long-term equilibrium relationship with GDP in the previous period. The parameter α measures the speed of adjustment. We expect $\alpha < 0$, as a positive residual represents a positive deviation of revenue from it's long term relationship with GDP in the previous period, and we would expect revenue to decrease back to it's normal level of GDP after such a deviation. p and q need to be selected for optimal forecasting performance which is done in the next section. Below a simple ECM with $p = 1$ and $q = 1$ is estimated:

```r
# Estimating error correction model
ecm <- lm(D(lrev) ~ D(lgdp) + L(res), merge(X, res))

summ(ecm, digits = 4L, vcov = vcovHAC(ecm))
```

Observations	49 (19 missing obs. deleted)
Dependent variable	D(lrev)
Type	OLS linear regression

F(2,46)	38.2681
R^2	0.6246
Adj. R^2	0.6083

	Est.	S.E.	t val.	p
(Intercept)	0.0304	0.0135	2.2595	0.0286
D(lgdp)	0.1819	0.2288	0.7950	0.4307
L(res)	-1.0121	0.2043	-4.9530	0.0000

Standard errors: User-specified

The large negative coefficient on the error correction term is likely due to the seasonal component captured in it. Curiously, changes in revenue in the current quarter do not seem to be very strongly related to changes in GDP in the current quarter, which could also be accounted for by data being published with a lag. We can visualize regression diagnostic plots using `plot(ecm)` and test the residuals for heteroskedasticity and autocorrelation[1] as follows:

```r
# No heteroskedasticity (null of homoskedasticity not rejected)
bptest(ecm)
##
##  studentized Breusch-Pagan test
##
## data:  ecm
```

[1]If there are lagged dependent variables on the right-hand side of the regression, the classical Durbin-Watson test, implemented in R in `lmtest::dwtest`, is no longer valid. In this case, it is appropriate to use a Breusch-Godfrey Lagrange Multiplier test for general, higher order serial correlation (Godfrey, L. G., 1988).

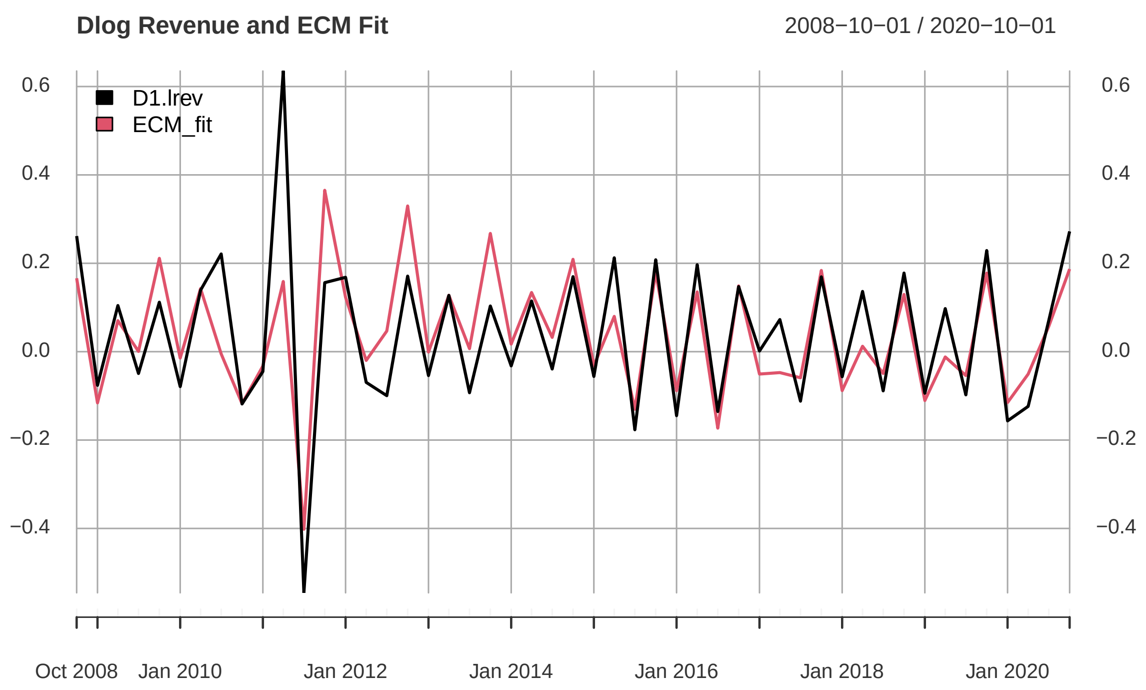
```
## BP = 3.2813, df = 2, p-value = 0.1939

# No autocorrelation up to 4 lags
bgtest(ecm, order = 4)
##
##  Breusch-Godfrey test for serial correlation of order up to 4
##
## data:  ecm
## LM test = 6.61, df = 4, p-value = 0.158

# Test residuals for iid-ness: seem to be iid.
Box.test(resid(ecm), lag = 4, type = "Ljung-Box")
##
##  Box-Ljung test
##
## data:  resid(ecm)
## X-squared = 5.8138, df = 4, p-value = 0.2135
```

The statistical properties of the equation are acceptable. Errors are homoskedastic and serially uncorrelated at the 5% level. For convenience and further use, the function below can also be used to examine the autocorrelation and partial autocorrelation of the residuals at higher lags orders.

```
# Function to compute a correlogram of a time series
corrgram <- function(x, lag.max = NULL, plot = FALSE,
                     na.action = function(x) if(anyNA(unclass(x)))
                      x[!is.na(x)] else x) {
  if(plot) {
    oldpar <- par(mfrow = c(1, 2))
    on.exit(par(oldpar))
  }
  ac <- eval(substitute(acf(x, lag.max, plot = plot, na.action = na.action)))
  acf <- drop(ac[[1L]])[-1]
  pacf <- drop(eval(substitute(pacf(x, lag.max, plot = plot,
                                    na.action = na.action)))[[1L]])
  n <- ac$n.used
  qstat <- n * (n + 2) * cumsum(acf^2 / seq.int(n - 1, n - length(acf)))
  lags <- seq_along(acf)
  res <- cbind(Lag = lags, AC = acf, PAC = pacf,
               Q = qstat, `Pr(>Q)` = pchisq(qstat, lags, lower.tail = FALSE))
  class(res) <- "corrgram"
  res
}

# Print method
print.corrgram <- function(x, digits = 3, ...) {
  xx <- cbind(format(x[, 1L, drop = FALSE], drop0trailing = TRUE),
              format(round(x[, -1L], digits)))
  print.default(`rownames<-`(xx, rep(" ", nrow(xx))), quote = FALSE, right = TRUE)
}

# Compute correlogram of the residuals
corrgram(resid(ecm), lag.max = 6)
##   Lag     AC    PAC     Q Pr(>Q)
##     1  0.014  0.014 0.010  0.920
##     2 -0.140 -0.141 1.057  0.589
##     3  0.269  0.278 4.979  0.173
##     4  0.123  0.092 5.814  0.213
##     5 -0.150 -0.090  7.090  0.214
##     6 -0.114 -0.170  7.845  0.250
```

It appears that the residuals are indeed white noise serially uncorrelated and homoskedastic. We can also compute and plot the fitted values of the model:

```
# Get ECM fitted values
ECM_fit <- fitted(ecm) %>% as.xts(dateFormat = "Date")

# Plot together with revenue
plot(D(X[, "lrev"]) %>% merge(ECM_fit) %>% na.omit,
     main = "Dlog Revenue and ECM Fit",
     legend.loc = "topleft", major.ticks = "years", grid.ticks.on = "years")
```



The plot suggests an acceptable but but far from perfect fit. So adding some higher or seasonal lag orders p and q could possibly help both to increase the fit of the model as well as to remove some of the seasonal variation from the error correction term.

Model Selection

To ease the search of the optimal model as well as the forecasting process, we will specify different unrestricted ECM's estimated in one step. Stock (1987) showed that this is asymptotically equivalent to estimating the ECM in two steps and may even be desirable if the cointegration relationship is influenced by the lagged terms. In our case, the introduction of lagged differences $p > 1$ and $q > 1$ may capture seasonal variation in the series and give a cleaner estimate of the cointegrating vector. Mathematically, rewriting Eq. 1 yields:

$$A(L)\Delta r_t = (\gamma - \alpha\beta_0) + B(L)\Delta y_t + \alpha r_{t-t} - \alpha\beta_1 y_{t-1} + v_t, \quad (2)$$

such that α, the coefficient on r_{t-t} is still the adjustment speed parameter and β_1 can easily be obtained from the coefficient on y_{t-t} through dividing by α and negating.

Below a list of 1-step ECM's is estimated. ECM0 is the baseline model examined above. ECM's 1-4 add additional lags p and q to the equation. Then there are a set of seasonal ECM's which add the 4th lagged difference reflecting the quarterly frequency of the data. As we are now interested in finding the best model to forecast the recent period, the sample is restricted to years following 2014, which still gives 28 quarterly observations to estimate these models.

```r
# Restricting the sample to years after 2014
XR <- X["2014/"]
# 28 observations in the restricted sample
nrow(na.omit(XR))
## [1] 28

# Now estimating a battery of models
modlist <- list(
# Non-Seasonal Models
ECM0 = lm(D(lrev) ~ D(lgdp) + L(lrev) + L(lgdp), XR),
ECM1 = lm(D(lrev) ~ L(D(lrev)) + D(lgdp) + L(lrev) + L(lgdp), XR),
ECM2 = lm(D(lrev) ~ L(D(lgdp), 0:1) + L(lrev) + L(lgdp), XR),
ECM3 = lm(D(lrev) ~ L(D(lrev)) + L(D(lgdp), 0:1) + L(lrev) + L(lgdp), XR),
ECM4 = lm(D(lrev) ~ L(D(lrev), 1:2) + L(D(lgdp), 0:1) + L(lrev) + L(lgdp), XR),
# Seasonal Models
SECM0 = lm(D(lrev) ~ L(D(lgdp), c(0,4)) + L(lrev) + L(lgdp), XR),
SECM1 = lm(D(lrev) ~ L(D(lrev)) + L(D(lgdp), c(0,4)) + L(lrev) + L(lgdp), XR),
SECM2 = lm(D(lrev) ~ L(D(lrev), 4) + L(D(lgdp), c(0,4)) + L(lrev) + L(lgdp), XR),
SECM3 = lm(D(lrev) ~ L(D(lrev), c(1,4)) + D(lgdp) + L(lrev) + L(lgdp), XR),
SECM4 = lm(D(lrev) ~ L(D(lrev), c(1,4)) + L(D(lgdp),c(0,4)) + L(lrev) + L(lgdp),XR),
SECM5 = lm(D(lrev) ~ L(D(lrev), c(1,4)) + L(D(lgdp),c(0:1,4)) + L(lrev) + L(lgdp),XR)
)
```

We can compactly examine the statistical properties of this list of models using `sapply` and only retaining the p-values of the Breusch-Pagan heteroskedasticity test, the Breusch-Godfrey residual autocorrelation test and the Ljung-Box white noise test applied to the residuals.

```r
sapply(modlist, function(x) c(bptest(x)$p.value,
    BG = bgtest(x, order = 4)$p.value,
    LB = Box.test(resid(x), lag = 4, type = "Ljung-Box")$p.value)) %>% round(2)
##    ECM0 ECM1 ECM2 ECM3 ECM4 SECM0 SECM1 SECM2 SECM3 SECM4 SECM5
## BP 0.78 0.10 0.88 0.16 0.54  0.90  0.62  0.31  0.28  0.34  0.43
## BG 0.56 0.29 0.45 0.29 0.20  0.39  0.39  0.58  0.61  0.58  0.76
## LB 0.43 0.24 0.31 0.24 0.26  0.33  0.34  0.68  0.75  0.72  0.91
```

The p-values of all tests are > 0.05, suggesting that all models provide an acceptable fit of the data. To examine the fit of these models, a first step is to compute and compare their information criteria. R naively offers the Akaike's (AIC) and Schwarz's Bayesian (BIC) information criteria which penalize the fit of the models by the number of fitted parameters. A lower IC indicates a better model.

```r
# Computing AIC and BIC of the models
ICs <- sapply(modlist, function(x) c(AIC = AIC(x), BIC = BIC(x)))
round(ICs, 1)
##      ECM0  ECM1  ECM2  ECM3  ECM4 SECM0 SECM1 SECM2 SECM3 SECM4 SECM5
## AIC -66.0 -62.0 -60.6 -60.0 -56.8 -57.7 -55.8 -57.2 -56.8 -55.2 -54.9
## BIC -59.5 -54.5 -53.1 -51.2 -47.0 -50.9 -47.9 -49.2 -48.8 -46.2 -44.7

# IC ranking of models
dapply(ICs, rank, MARGIN = 1)
##     ECM0 ECM1 ECM2 ECM3 ECM4 SECM0 SECM1 SECM2 SECM3 SECM4 SECM5
## AIC    1    2    3    4    8     5     9     6     7    10    11
## BIC    1    2    3    4    9     5     8     6     7    10    11
```

The IC ranking favors the more parsimonious models without seasonal components, suggesting that they do not add much to the fit. Whether they also don't add much to the forecasting performance is a different matter investigated below.

In-Sample Forecasting

Forecast a in-sample and then computing statistics that evaluate the forecast against the outcome is a direct and useful approach to evaluate the forecasting performance of a model. The function below takes a regression model and data as input and forecasts n periods in-sample using all available data i.e. it performs expanding window one-period ahead forecasts where the model is re-estimated for each new period being forecasted using all available data up to that period.

```r
# Function to forecast n periods in-sample for forecast evaluation
forecast_is <- function(model, data, n = 1L) {
  data <- qDF(na_omit(data))
  N <- nrow(data)
  s <- (N-n):(N-1L)
  form <- terms(model)
  y <- attr(form, "variables")[1:2]
  # Dependent variable and forecast are returned as in the model (e.g. differenced)
  ydat <- qDF(eval(y, data))
  names(ydat) <- deparse(y[[2L]])
  # Forecasting with expanding window
  fc <- numeric(0L)
  for(i in s) {
    # This re-estimates the model on the selected sample
    reest <- lm(form, ss(data, seq_len(i)))
    # This forecasts one period ahead
    fc <- c(fc, flast(predict(reest, newdata = ss(data, seq_len(i+1L)))))
  }
  ydat[s + 1L, "fc"] <- unattrib(fc)
  return(ydat)
}
```

This approach is optimal to evaluate which model gives us the best 1-period ahead forecasts. If we are primarily interested in longer forecast horizons, the function must be modified to perform dynamic in-sample forecasting. Below the function used to forecast 8 quarters in-sample using data running from 2012.

```r
# Forecast 8 quarters one-step ahead out of sample, on data running from 2012:
fcl <-  lapply(modlist, forecast_is, X["2012/"], n = 8L)

# Create a data.frame of those forecasts
fcl <- do.call(cbind, c(unname(fcl[1L]), lapply(fcl[-1L], `[`, "fc")))
names(fcl)[-1L] <- names(modlist)

# This shows the forecasts
tail(fcl, 8L)
##        D(lrev)         ECM0        ECM1         ECM2        ECM3         ECM4
## 29 -0.09406650 -0.002138135 -0.02042251  0.006023654 -0.01819213 -0.057155092
## 30  0.09747392  0.033111230  0.11031568  0.034251215  0.12541708  0.153481050
## 31 -0.09767896 -0.011431121 -0.07291684 -0.017930055 -0.07266627 -0.085597312
## 32  0.22885575  0.200574852  0.12616669  0.201153061  0.14483320  0.155986438
## 33 -0.15656256  0.021554005 -0.02643604  0.029657107 -0.02828628 -0.067585722
## 34 -0.12377822  0.005583935  0.17846099  0.019652186  0.20960480  0.222547039
## 35  0.07052492  0.055146900  0.07235850  0.056704584  0.07033496  0.003906892
## 36  0.27219746  0.221063469  0.14261738  0.220099422  0.14277425  0.110709251
##             SECM0       SECM1       SECM2       SECM3       SECM4       SECM5
## 29 -0.0172864447 -0.06690639 -0.03685734 -0.06527336 -0.06536938 -0.06649749
## 30  0.0163269411  0.12729872  0.05481922  0.13321086  0.12572325  0.15028239
## 31 -0.0168172213 -0.06394362 -0.05353726 -0.07062308 -0.06079545 -0.06329783
## 32  0.1948660787  0.12158202  0.19147726  0.12123312  0.12001609  0.14141791
## 33  0.0007727251 -0.09206726 -0.05169576 -0.09198236 -0.09316286 -0.10121555
## 34 -0.0208601122  0.17835137  0.02695647  0.17959986  0.17513689  0.21757206
## 35  0.0688575433  0.10777059  0.02851532  0.05408194  0.07476784  0.08274568
## 36  0.2063460366  0.11219138  0.22681185  0.17434257  0.15089254  0.15144641
```

Forecast Evaluation

Having computed forecasts from the various methods, we want to compare them in a rigorous way using an extended set of forecast evaluation metrics. The following function computes a set of well established forecast evaluation statistics. The `fc` argument to the function may be a vector, matrix or data frame of forecasts for the outcome `y`. By default also a 'naive' forecast is added which is simply `y` lagged `n.ahead` times.

```r
eval_forecasts <- function(y, fc, add.naive = TRUE, n.ahead = 1) {
  # eval substitute to get the name of the forecast if only a vector is passed
  mfc <- eval(substitute(qDF(fc)))
  lagy <- flag(y, n.ahead)
  if (add.naive) mfc <- c(list(Naive = lagy), mfc)
  if (!all(length(y) == lengths(mfc)))
    stop("All supplied quantities must be of equal length")
  res <- vapply(mfc, function(fcy) {
    # Preparation
    cc <- complete.cases(y, fcy)
    y <- y[cc]
    fcy <- fcy[cc]
    lagycc <- lagy[cc]
    n <- sum(cc)
    # Undo Bessel's correction: (n-1) instead of n in denominator
    nobessel <- sqrt((n - 1) / n)
    sdy <- sd(y) * nobessel
    sdfcy <- sd(fcy) * nobessel
    diff <- fcy - y
    # Calculate Measures
    bias <- sum(diff) / n          # Bias
    MSE <- sum(diff^2) / n         # Mean Squared Error
    BP <- bias^2 / MSE             # Bias Proportion
    VP <- (sdy - sdfcy)^2 / MSE    # Variance Proportion
    CP <- 1 - BP - VP              # Covariance Proportion
    # CP <- 2 * (1 - cor(y, fcy)) * sdy * sdfcy / MSE
    RMSE <- sqrt(MSE)              # Root MSE
    R2 <- 1 - MSE / sdy^2          # R-Squared
    SE <- sd(diff) * nobessel      # Standard Forecast Error
    MAE <- sum(abs(diff)) / n      # Mean Absolute Error
    MPE <- sum(diff / y) / n * 100 # Mean Percentage Error
    MAPE <- sum(abs(diff / y)) / n * 100 # Mean Absolute Percentage Error
    U1 <- RMSE / (sqrt(sum(y^2) / n) + sqrt(sum(fcy^2) / n))   # Theils U1
    U2 <- sqrt(fmean((diff/lagycc)^2) / fmean((y/lagycc-1)^2)) # Theils U2
    # Output
    return(c(Bias = bias, MSE = MSE, RMSE = RMSE, `R-Squared` = R2, SE = SE,
             MAE = MAE, MPE = MPE, MAPE = MAPE, U1 = U1, U2 = U2,
             `Bias Prop.` = BP, `Var. Prop.` = VP, `Cov. Prop.` = CP))
  }, numeric(13))
  attr(res, "naive.added") <- add.naive
  attr(res, "n.ahead") <- n.ahead
  attr(res, "call") <- match.call()
  class(res) <- "eval_forecasts"
  return(res)
}

# Print method
print.eval_forecasts <- function(x, digits = 3, ...) print.table(round(x, digits))
```

These metrics can be used to obtain a full picture of the quality of the forecast and fulfill different purposes: a good forecast will have a low bias and MPE (which gives the bias in percentage terms). This is essential to ensure that we are not systematically over or under predicting the outcome. The bias proportion (BP) gives the share of MSE which is attributable to the bias. It should be close to 0. The MSE, RMSE and MAE measure the deviation of forecast and outcome in absolute terms. The key difference between the RMSE and the MAE (which are often comparable in magnitude) is that the MAE considers all deviations of the forecast from the outcome equally, whereas the RMSE, computed from squared forecast errors, penalizes large forecast errors disproportionately. Thus a forecast that performs well on all years except for one year where it gives a large error, will likely have a higher RMSE than MAE.

The MAPE represents the MAE as a percentage of the outcome variable. Theil's U1 index also provides a normalization of the RMSE bounded between 0 and 1. U1 is however not a very well behaved statistic (lower does not always imply a better model) and should be interpreted with caution. A better statistic, U2, compares each forecast to the naive model. It is computed as the square root of the ration of the mean squared percentage error of the forecast to the naive forecast. The naive forecast will have a U2 index of 1, and forecasts with U2 $<$ 1 are better than the naive one.

Another more familiar normalization of the MSE is the r-squared, which provides the proportion of the variance in the outcome variable explained by the forecast. Note that it is possible that the MSE is greater than the variance of y, so the r-squared can be negative and should also be interpreted with caution. Another important metric to look at is the variance proportion (VP) which gives the proportion of MSE that can be attributed to a mismatch of the variance of the forecast with the variance of outcome. The VP should ideally be close to 0, along with the BP, such that most of the MSE is made up of unsystematic forecast errors contained in the covariance proportion (CP).

To summarize: A good forecast has a low bias / MPE, indicating it does not systematically over or under-predict the outcome, a low MSE / RMSE and MAE / MAPE indicating small forecast errors in absolute value, and low BP and VP (close to 0), indicating that the forecasted series has similar statistical properties in terms of its mean and variance to the outcome series, and that most of the forecast error is contained in the CP (close to 1). The forecast should also be better than the naive forecast (U2 $<$ 1).

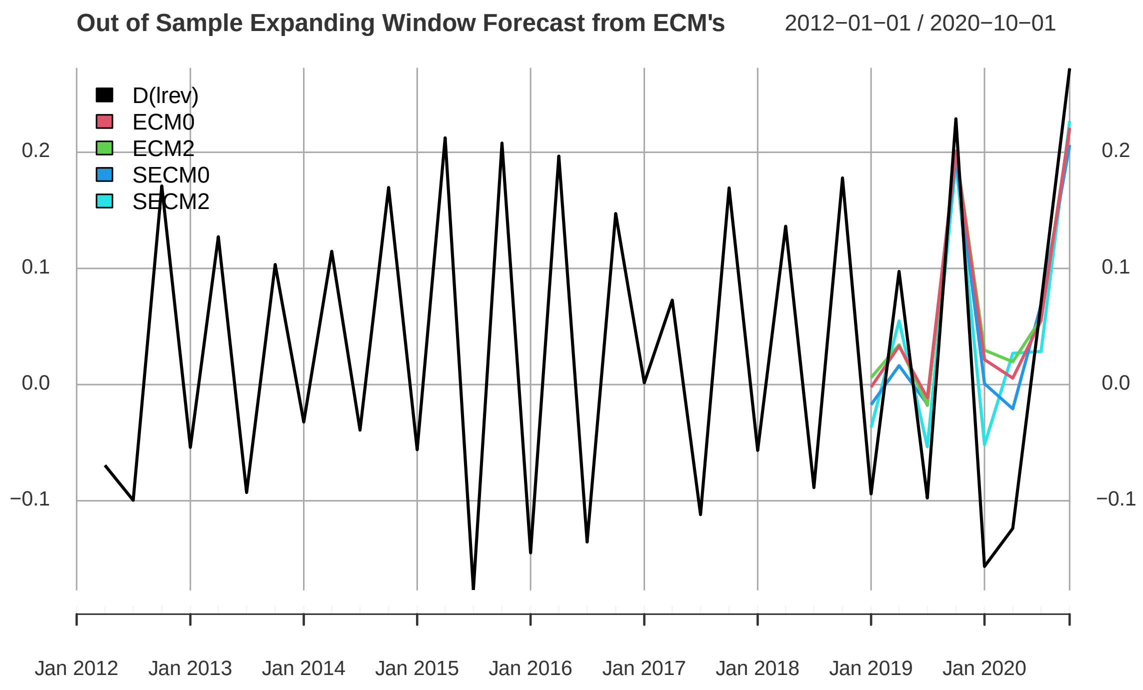
It is very easy to apply this function to our data frame of forecasts and also compute a ranking according to different evaluation metrics:

```r
fc_stat <- eval_forecasts(fcl[[1L]], fcl[-1L])
fc_stat
##              Naive    ECM0    ECM1    ECM2    ECM3    ECM4  SECM0   SECM1
## Bias        -0.010   0.041   0.039   0.044   0.047   0.030  0.029   0.028
## MSE          0.060   0.009   0.018   0.010   0.020   0.021  0.007   0.017
## RMSE         0.245   0.095   0.133   0.100   0.141   0.145  0.087   0.131
## R-Squared   -2.316   0.629   0.270   0.592   0.184   0.134  0.692   0.296
## SE           0.245   0.086   0.127   0.089   0.133   0.142  0.081   0.128
## MAE          0.228   0.081   0.097   0.083   0.101   0.105  0.075   0.095
## MPE         87.920 -65.410 -63.454 -67.321 -64.175 -64.551 -59.088 -46.368
## MAPE       489.377  65.410  67.398  67.321  71.342  78.915  59.088  67.220
## U1           0.909   0.357   0.502   0.373   0.510   0.513  0.330   0.481
## U2            1.000   0.312   0.500   0.319   0.514   0.573   0.316   0.546
## Bias Prop.    0.002   0.185   0.086   0.196   0.112   0.043   0.116   0.047
## Var. Prop.    0.001   0.536   0.274   0.513   0.191   0.090   0.599   0.176
## Cov. Prop.    0.998   0.279   0.640   0.291   0.697   0.868   0.285   0.776
##               SECM2   SECM3   SECM4   SECM5
## Bias          0.024   0.030   0.029   0.039
## MSE           0.006   0.015   0.015   0.018
## RMSE          0.076   0.123   0.124   0.135
## R-Squared     0.763   0.380   0.367   0.245
## SE            0.072   0.119   0.121   0.130
## MAE           0.066   0.085   0.086   0.091
## MPE         -53.888 -51.785 -50.923 -48.337
## MAPE         53.888  60.951  59.672  66.213
## U1            0.281   0.441   0.454   0.468
## U2            0.257   0.454   0.482   0.506
## Bias Prop.    0.098   0.059   0.054   0.085
## Var. Prop.    0.532   0.156   0.189   0.092
## Cov. Prop.    0.370   0.785   0.757   0.824
```

```
dapply(fc_stat, rank, MARGIN = 1)
##             Naive ECM0 ECM1 ECM2 ECM3 ECM4 SECM0 SECM1 SECM2 SECM3 SECM4 SECM5
## Bias            1   10    8   11   12    7     5     3     2     6     4     9
## MSE            12    3    8    4   10   11     2     7     1     5     6     9
## RMSE           12    3    8    4   10   11     2     7     1     5     6     9
## R-Squared       1   10    5    9    3    2    11     6    12     8     7     4
## SE             12    3    7    4   10   11     2     8     1     5     6     9
## MAE            12    3    9    4   10   11     2     8     1     5     6     7
## MPE            12    2    5    1    4    3     6    11     7     8     9    10
## MAPE           12    5    9    8   10   11     2     7     1     4     3     6
## U1             12    3    9    4   10   11     2     8     1     5     6     7
## U2             12    2    7    4    9   11     3    10     1     5     6     8
## Bias Prop.      1   11    7   12    9    2    10     3     8     5     4     6
## Var. Prop.      1   11    8    9    7    2    12     5    10     4     6     3
## Cov. Prop.     12    1    5    3    6   11     2     8     4     9     7    10
```

All of the most important metrics (Bias, MSE, MAE) favor the simple seasonal model given by SECM2, which includes one seasonal lagged difference of revenue and GDP in the baseline ECM specification (ECM0). Competing models are SECM0 in the second place, followed by ECM0 and ECM2. We can also visualize the in-sample forecasts from these models.

```
# Plotting in-sample forecasts from the best 4 models against the outcome
fcl[, .c("D(lrev)", ECM0, ECM2, SECM0, SECM2)] %>%
  qM %>% copyMostAttrib(na.omit(X["2012/"])) %>%
plot(main = "Out of Sample Expanding Window Forecast from ECM's",
     legend.loc = "topleft", major.ticks = "years", grid.ticks.on = "years")
```



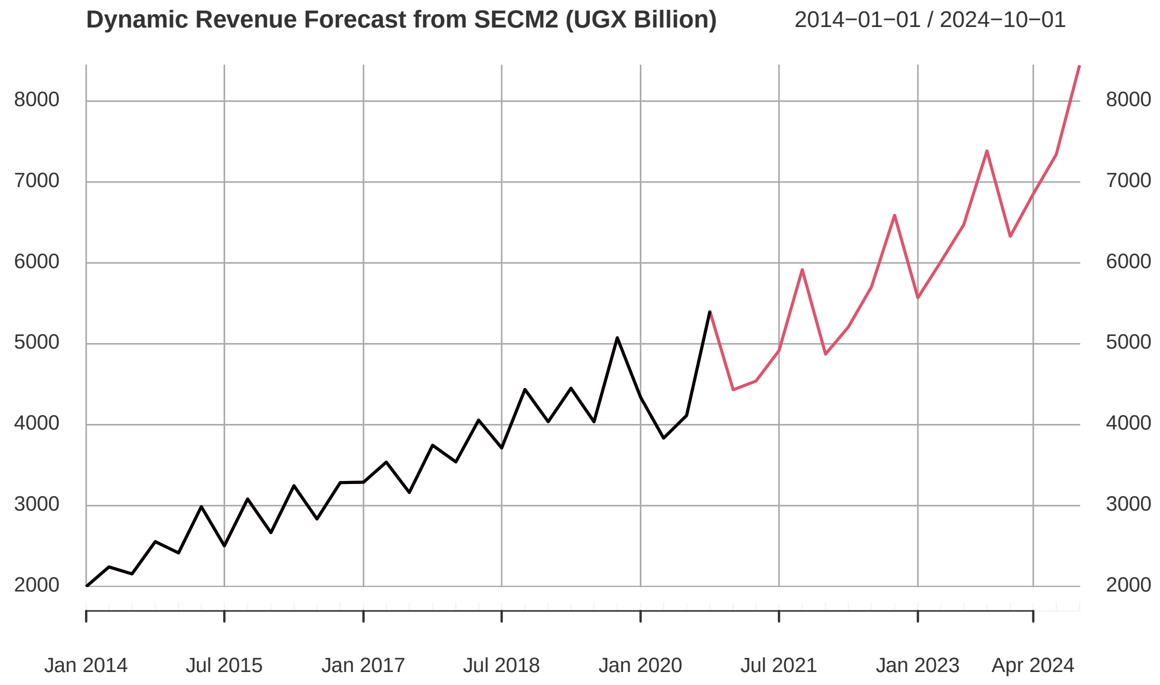
It is evident that SECM2 performs quite a bit better than the other models as it follows the movements of the outcome series much more closely. None of the models were however able to fully account for the drop in revenue due to COVID-19 in early 2020, suggesting, as the IMF Fiscal Affairs Department pointed out in a policy statement from April 2020, that buoyancy or macro elasticities based empirical forecasting approaches underestimate the true revenue shortfall caused by COVID-19 due to the asymmetric nature of the shock across sectors of the economy and size of businesses.

Having found the optimal model, it remains to use it for actual forecasting into the future. The function below generates dynamic forecasts for all periods where forecasted covariates are available, being provided a dynamic model and a dataset containing the outcome (with missing values for future periods) and forecasted predictors. It assumes that all past missing values were removed from the data beforehand.

```r
# Function to forecast dynamically from a dynamic model (with forecasted predictors)
forecast_dyn <- function(mod, data, y.diff = TRUE) {
  miss <- is.na(data)
  data[miss] <- 0
  # If the dependent variable is first-differenced in the model ...
  if(y.diff) { # .. results at original scale
    for(i in which(rowSums(miss) > 0)) {
      last <- data[i-1L, miss[i, ]]
      data[i, miss[i, ]] <- last + flast(predict(mod, newdata = ss(data, seq_len(i))))
    }
  } else {
    for(i in which(rowSums(miss) > 0))
      data[i, miss[i, ]] <- flast(predict(mod, newdata = ss(data, seq_len(i))))
  }
  data
}
```

It is again very easy to use this function to forecast revenue into the future, using the quarterly GDP projections for Uganda through 2024.

```
# Dynamic forecast through 2024, using quarterly GDP projection
ecm_fc = forecast_dyn(modlist$SECM2, X["2014/2024"]) %>%
         exp %>% setColnames(.c(Revenue, GDP))

# Plot together with outcome series
X["2014/2024", "lrev"] %>% exp %>% merge(ecm_fc[, "Revenue"]) %>%
  plot(main = "Dynamic Revenue Forecast from SECM2 (UGX Billion)")
```



Having obtained the level forecast, we can view the forecasted revenue for the remaining quarters of the financial year 2020/21 and obtain the total revenue forecast for the year as the sum of the two outcome and the two forecasted quarters.

```
# Forecasted values for the end of the financial year
ecm_fc["2020-07/2021-04"]
##             Revenue      GDP
## 2020-07-01 4114.513 38718.84
## 2020-10-01 5401.723 35476.29
## 2021-01-01 4432.129 35625.16
## 2021-04-01 4539.506 37168.90

# Sum of revenue and GDP forecasts for the financial year
colSums(ecm_fc["2020-07/2021-04"])
##   Revenue       GDP
##  18487.87 146989.19
```

The forecasted revenue of 18,487 billion UGX is on the lower end of what has been projected so far for the financial year. As a final exercise, we may be interested to see how this projection is impacted by our choice of optimal model. The code below first forecasts revenue using all models considered above, and then computes the annual revenue forecast for 2020/21 from each model.

```
# Dynamic forecasting with all models
ecm_fcl <- lapply(modlist, function(mod)
                    forecast_dyn(mod, X["2014/2024"]) %>%
                    exp %>% setColnames(.c(Revenue, GDP)))
```

```r
# Revenue forecast 2020/21 from all models
fc_all <- sapply(ecm_fcl, function(x) sum(x["2020-07/2021-04", "Revenue"]))
fc_all
##     ECM0     ECM1     ECM2     ECM3     ECM4    SECM0    SECM1    SECM2
## 18725.70 18783.63 18704.58 18783.05 19104.55 18888.04 18914.19 18487.87
##    SECM3    SECM4    SECM5
## 18141.23 18426.46 18180.01

# Weighted average of these forecasts, weighted by their inverse-MSE
fmean(fc_all, w = 1 / fc_stat["MSE", -1L])
## [1] 18644.4
```

The forecasts for 2020/21 from these models range from 18,100 to 19,100, so the 18,487 from the benchmark model is towards the lower end. In cases where we are not very confident in using the forecast from a particular model, we could also resort to an ensemble forecast computed as a weighted average of the forecasts from different models or forecasting approaches. We could weight these forecasts using information criteria or forecast evaluation metrics (such as MSE, MAE, U2). Another option, if we have sufficient data to forecast in-sample, is to regress the outcome on the various forecasts, and then combine dynamic forecasts using that coefficient vector. With many different forecasts to consider, a possible extensions to the regression method is to shrink the coefficients of the unimportant forecasts using a LASSO regression[2]. Another alternative is to opt for more complex non-linear combinations of forecasts, for example by using a Random Forest model[3] to predict the outcome from various in-sample forecasts and using this trained model to combine dynamic forecasts. In short, there are lots of possibilities for ensemble forecasting, but in this particular case I am quite confident that SECM2 provides a better forecast than the ensemble.

Conclusion

For many economic variables including tax revenues, effective short-term forecasting models can be developed based on empirical relationships captured in ECM's and DLM's. It is however very important to specify a model appropriate to the specific nature of the data at hand: cointegrated data should be forecasted using ECM's, otherwise DLM's in levels or differences are appropriate. Seasonality in one series can be dealt with through appropriate seasonal adjustment, whereas seasonality in the dependent variable or in all series should generally be dealt with by including seasonal lags in the model. A second important aspect is to test and rigorously evaluate a broad set of models. This paper provided a solid theoretical foundation as well as R code and functions to conduct such empirical forecasting exercises with relative ease. In light of the COVID-19 pandemic, the viability of forecasts based on past empirical relationships must however be examined critically, and disaggregated forecasts should be adopted wherever data allows it. Informed judgement is also necessary, both to evaluate forecasts and to gauge the viability and usefulness of an empirical forecasting model in light of the political and economic circumstances and the available data.

[2] Available in the `glmnet` R package.
[3] Available in the `randomForest` R package.

References

IMF Fiscal Affairs Department (2020). *Challenges in Forecasting Tax Revenue.* Special Series on Fiscal Policies to Respond to COVID-19.

IMF Institute for Capacity Development. *Financial Programming and Policies: Volume I.* Retrieved from: https://courses.edx.org/asset-v1:IMFx+FPP.1x+1T2017+type@asset+block@FPP1x_Manua l.pdf

Engle, Robert, and Clive Granger. 1987. *Co-integration and Error Correction: Representation, Estimation and Testing.* Econometrica 55 (2): 251–76.

Johansen, Søren (1991). *Estimation and Hypothesis Testing of Cointegration Vectors in Gaussian Vector Autoregressive Models.* Econometrica. 59 (6): 1551–1580. JSTOR 2938278.

Stock, J. H. (1987). *Asymptotic properties of least squares estimators of cointegrating vectors.* Econometrica. 55 (5): 1551–1580.

Enders, Walter (2010). *Applied Econometric Time Series (Third ed.).* New York: John Wiley & Sons. pp. 272–355. ISBN 978-0-470-50539-7.

Lütkepohl, Helmut (2006). *New Introduction to Multiple Time Series Analysis.* Berlin: Springer. pp. 237–352. ISBN 978-3-540-26239-8.

Alogoskoufis, G., & Smith, R. (1991). *On error correction models: specification, interpretation, estimation.* Journal of Economic Surveys, 5(1), 97-128.

https://davegiles.blogspot.com/2016/05/forecasting-from-error-correction-model.html

https://en.wikipedia.org/wiki/Error_correction_model

https://www.econometrics-with-r.org/16-3-cointegration.html

https://bookdown.org/ccolonescu/RPoE4/time-series-nonstationarity.html#the-error-correction-model

https://www.youtube.com/watch?v=wYQ_v_0tk_c