Module 3: Time Series Regression Models I - Distributed lag models - By using dLagM R-package

MATH1307 Forecasting

Dr. Haydar Demirhan

Introduction

In time series analysis, we mainly focus on modelling the correlation structure of the variable of interest, which is created by the time dependence of the variable. However, in some cases, it is useful to include some additional predictors to explain the matter of interest in a more accurate way.

One of the approaches to include predictors in time series analysis is to use several lags of a predictor variable in the model as explanatory variables. In this case, each lag of the predictor variable will explain some of the overall variation and correlation structure in the dependent variable.

This kind of regression approaches is called distributed lag models. If the number of lags is known, the model is called **finite distributed lag model**. If it is undetermined at the beginning of the analysis, it is **infinite distributed lag model**. We will use the abbreviation **DLM** to denote the distributed lag models throughout the course. DLMs are generally used in the analysis of econometric data. But they are also successfully applied in other fields where appropriate exploratory series is available.

In this module, we will

- describe the finite and infinite linear and nonlinear DLMs,
- discuss their parameter estimation,
- diagnostic checking,
- model selection,
- forecasting with DLMs.

We will conclude the module by a practical application of DLMs to a real dataset.

In this module, we will utilize the R packages `ggplot2, AER, Hmisc, car, dynlm` for most of the tasks. The R package `dLagM` is specifically deveoped to implement distributed lag models in this course.

Distributed-Lag Models

When a decision made on a variable, some of the related variables would be effected through time. For example, when the income tax rate is increased, this would reduce expenditures of consumers on goods and services, which reduces profits of suppliers, which reduces the demand for productive inputs, which reduces the profits of the input suppliers, and so on (Judge and Griffiths, 2000). These effects occur over the future periods; hence, they are *distributed* across time.

In a distributed-lag model, the effect of an independent variable X on a dependent variable Y occurs over the time. Therefore, DLMs are dynamic models. A linear infinite DLM with one independent variable is written as follows:

$$Y_t = \alpha + \sum_{s=0}^{\infty} \beta_s X_{t-s} + \epsilon_t, \tag{1}$$

where ϵ_t is a *stationary* error term with $E(\epsilon_t) = 0, Var(\epsilon_t) = \sigma^2, Cov(\epsilon_t, \epsilon_s) = 0$ As you can observe, the model in Eq. (1) is like a simple multiple linear regression model. The difference between classical regression models and DLMs is obviously in the time dependency of the X (dependent) variables. The coefficient α can be perceived as the general mean term or a constant term. The β_s coefficients are called *lag weights*. They specify the pattern of the relationship between dependent and independent *series* $\{Y_t\}$ and $\{X_t\}$ by formatting a *lag distribution*.

Notice that the number of regression coefficients is infinite in the model in (1). It is practically impossible to implement. Thus, the DLM is an infinite DLM. A finite DLM, including q lags is as follows:

$$Y_t = \alpha + \sum_{s=0}^{q} \beta_s X_{t-s} + \epsilon_t. \tag{2}$$

Notice that if we have n observations of the pairs (Y_t, X_t), then because we loose q observations, $X_{t-1}, X_{t-2}, \ldots, X_{t-q}$, only $n - q$ complete observations are available of estimation of coefficients.

It is not possible to estimate an infinite number of β parameters. One of the approaches to deal with this situation is to truncate the number of lags to a constant q. Another approach is to use a functional form that allows the lag distribution to decay gradually to zero. The choice of an appropriate number of lags is one of the common challenges of DLMs.

In DLMs, the parameter β_t measures the **dynamic effect** of changes in past values of dependent variable, δX_{t-s} , on the expected value of outcome variable $\delta E(Y_t)$ given that all other things are constant:

$$\frac{\partial E(Y_t)}{\partial X_{t-s}} = \frac{\partial E(Y_{t+s})}{\partial X_t} = \beta_s \tag{3}$$

In these kinds of models, collinearity due to the inclusion of the same independent variable in the model is a serious problem. Least-squares estimates of the coefficients suffer from this collinearity. Mostly, they have inflated variance estimates.

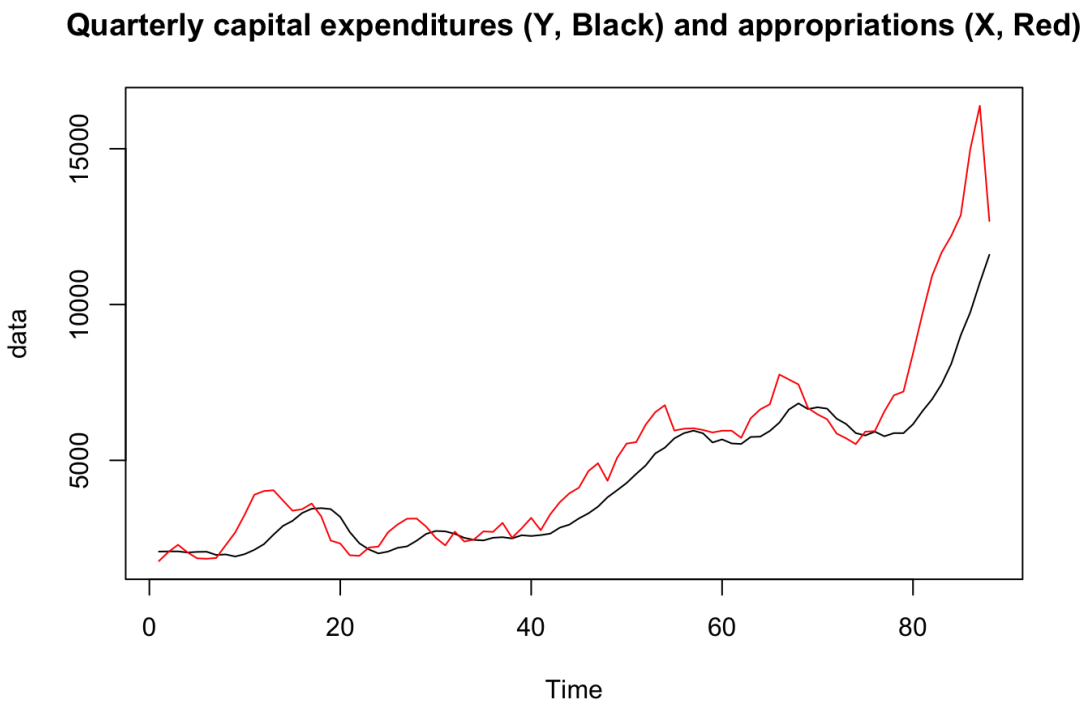
Let's see the interpretation of coefficients over a numerical example. In this example, we are interested in quarterly capital expenditures and appropriations (approved funds for expenditure) for US Manufacturing firms for a given period of 88 years.

The following code chunk displays time series plot for Y: Quarterly capital expenditures and X: Appropriations. Notice that if you change the argument `plot.type` to `"m"` , you get a multi-panel plot.

```
head(firms) # firms is the name of dataset
```

```
##   t    y    x
## 1 1 2072 1767
## 2 2 2077 2061
## 3 3 2078 2289
## 4 4 2043 2047
## 5 5 2062 1856
## 6 6 2067 1842
```

```
y = ts(firms$y)
x = ts(firms$x)
data = ts(firms[,2:3])
plot(data, plot.type="s",col = c("black", "red"), main = "Quarterly capital expendit
  ures (Y, Black) and appropriations (X, Red)")
```



X and Y series follow each other closely. We expect to have a strong correlation between these series:

```
cor(y,x) # Calculate correlation between numerical values in x and y
```

```
## [1] 0.9401348
```

This gives us an impression that we can explain the dependent variable Y more precisely if we use the information come from the independent variable X.

To apply a finite DLM to this data set, first, we need to create the design matrix and specify the number of lags. For now, let's assume that the number of lags q is 8 and make the required arrangements for dependent variables. The corresponding DLM is

$$Y_t = \alpha + \beta_0 X_t + \beta_1 X_{t-1} + \beta_2 X_{t-2} + \cdots + \beta_8 X_{t-8} + \epsilon_t. \tag{4}$$

where $t = 9, \ldots, n$. Because only $n - q$ observations are available for estimation, we will use the observations indexed by $t = q + 1, \ldots, n$ So,

- for X_t we will use $X_{q+1}, X_2, \ldots, X_n$
- for X_{t-1} we will use $X_q, X_{q+1}, \ldots, X_{n-1}$
- for X_{t-2} we will use $X_{q-1}, X_q, \ldots, X_{n-2}$
- for X_{t-3} we will use $X_{q-2}, X_{q-1}, \ldots, X_{n-3}$

- ...
- for X_{t-q} we will use $X_1, X_2, \ldots, X_{n-q}$

and

- for Y_t we will use $Y_{q+1}, Y_2, \ldots, Y_n$

For our example, we write $q = 8$ and $n = 88$ in place and create the independent X-variables with the following code chunk. This code chunk directly utilises the recently developed package `dLagM`.

```
library(dLagM)
```

```
## Loading required package: nardl
```

```
## Loading required package: dynlm
```

```
##
## Attaching package: 'dLagM'
```

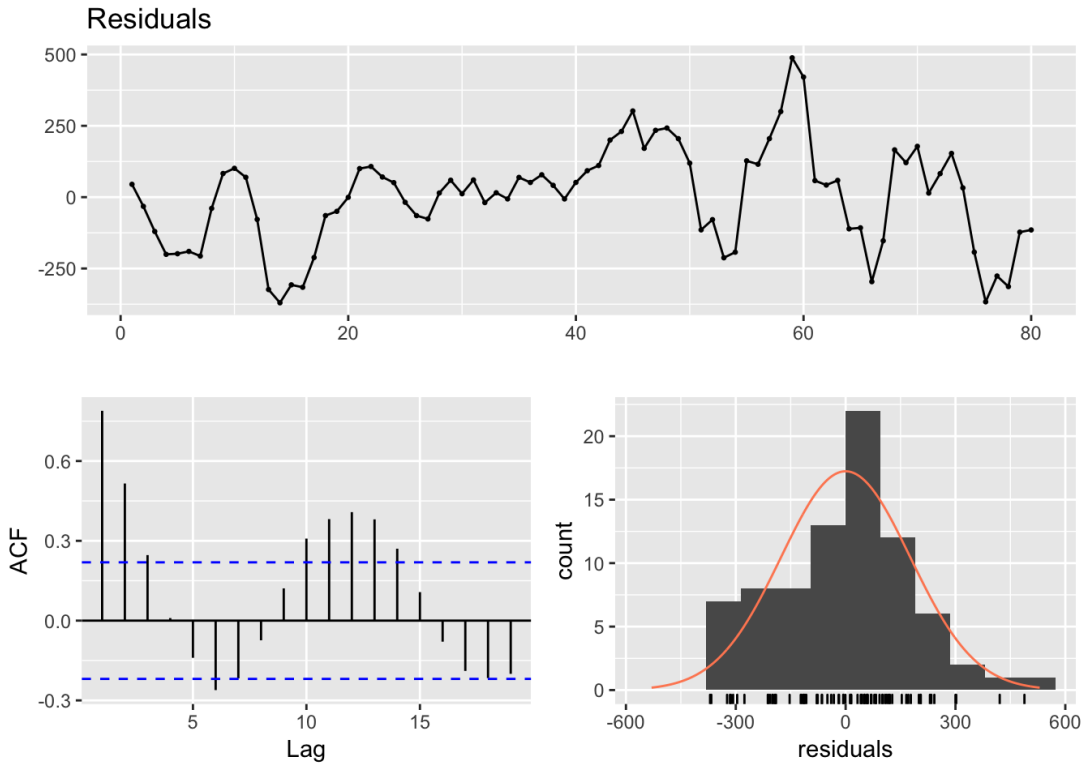
```
## The following object is masked from 'package:forecast':
##
##     forecast
```

```
model1 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 8)
summary(model1)
```

```
##
## Call:
## lm(formula = model.formula, data = design)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -370.20 -114.78   23.97  102.65  488.34
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.41477   53.70858   0.622   0.5359
## x.t          0.03838    0.03467   1.107   0.2721
## x.1          0.06720    0.06851   0.981   0.3300
## x.2          0.18124    0.08936   2.028   0.0463 *
## x.3          0.19443    0.09254   2.101   0.0392 *
## x.4          0.16989    0.09312   1.824   0.0723 .
## x.5          0.05236    0.09177   0.571   0.5701
## x.6          0.05246    0.09385   0.559   0.5780
## x.7          0.05618    0.09415   0.597   0.5526
## x.8          0.12708    0.05983   2.124   0.0372 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 187.7 on 70 degrees of freedom
## Multiple R-squared:  0.9934, Adjusted R-squared:  0.9926
## F-statistic:  1175 on 9 and 70 DF,  p-value: < 2.2e-16
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 1075.884 1102.086
```

We can apply a diagnostic check using `checkresiduals()` function from the `forecast` package. Notice that here we call the model object using `model1$model` instead of calling only `model1` .

```
checkresiduals(model1$model$residuals)
```




```
bgtest(model1$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model1$model
## LM test = 51.026, df = 1, p-value = 9.116e-13
```

```
# Send the residuals directly instead of a model object
```

In this output, the Breusch-Godfrey test is displayed to test the existence of serial correlation up to the displayed order. According to this test and ACF plot, we can conclude that the serial correlation left in residuals is highly significant. Also, from the time series plot and histogram of residuals, there is an obvious non-random pattern and very high residual values that violate general assumptions.

However, we got a very high R-squared value (0.99) for this model. We need to be careful about multicollinearity with this model due to the inclusion of the same variable even partly. The overall model is significant at 5% level of significance with a p-value of $2.2 \cdot 10^{-16}$. So the model fits the data well even we have some insignificant coefficients. Looking at the values of coefficients, the highest lag effect is seen for the lag of 3. However, we need to consider the effect of collinearity on these results. To inspect this issue, we will display variance inflation factors (VIFs). If the value of VIF is greater than 10, we can conclude that the effect of multicollinearity is high.

```
VIF.model1 = vif(model1$model) # variance inflation factors
# Notice again we are getting the model object out of the model fitting
# by using "$model" in addition to "model1"
VIF.model1
```

```
##       x.t       x.1       x.2       x.3       x.4       x.5       x.6
##  25.08859  91.90100 130.11827 117.51813 105.19779  90.33317  83.12416
##       x.7       x.8
##  74.53415  27.84059
```

```
VIF.model1 > 10
```

```
##  x.t  x.1  x.2  x.3  x.4  x.5  x.6  x.7  x.8
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

From the VIF values, it is obvious that the estimates of the finite DLM coefficients are suffering from the multicollinearity. To deal with this issue, we can use the restricted least squares method to find parameter estimates. In this approach, some restrictions are placed on the model parameters to reduce the variances of the estimators. In the context of DLMs, we translate the pattern of time effects into the restrictions on parameters. In the next section, we will use polynomial curves to restrict lag weights.

Polynomial Distributed Lags

To reduce the harmful effect of multicollinearity, we will impose a polynomial shape on the lag distribution. Suppose, lag weights follow a smooth polynomial pattern. Because this idea first introduced by Shirley Almon, the resulting model is called Almon Distributed Lag Model or Polynomial Distributed Lag model.

Imposing a polynomial pattern on the lag distribution is equivalent to representing β parameters with another kth order polynomial model of time. So, the effect of change in X_{t-s} on the expected value of Y_t is represented as follows:

$$\frac{\partial E(Y_t)}{\partial X_{t-s}} = \beta_s = \gamma_0 + \gamma_1 s + \gamma_2 s^2 + \cdots + \gamma_k s^k \tag{5}$$

where $s = 0, \ldots, q$

Let's turn back to our numerical example and apply the idea over the DLM in Eq. (5). When we impose 2nd order polynomial restrictions, we have

$$\begin{aligned}
\beta_0 &= \gamma_0 & s = 0 \\
\beta_1 &= \gamma_0 + \gamma_1 + \gamma_2 & s = 1 \\
\beta_2 &= \gamma_0 + 2\gamma_1 + 4\gamma_2 & s = 2 \\
\beta_3 &= \gamma_0 + 3\gamma_1 + 9\gamma_2 & s = 3 \\
\beta_4 &= \gamma_0 + 4\gamma_1 + 16\gamma_2 & s = 4 \\
\beta_5 &= \gamma_0 + 5\gamma_1 + 25\gamma_2 & s = 5 \\
\beta_6 &= \gamma_0 + 6\gamma_1 + 36\gamma_2 & s = 6 \\
\beta_7 &= \gamma_0 + 7\gamma_1 + 49\gamma_2 & s = 7 \\
\beta_8 &= \gamma_0 + 8\gamma_1 + 64\gamma_2 & s = 8
\end{aligned} \tag{6}$$

Then we will substitute the parameters into the DLM in Eq. (6).

$$\begin{aligned}
Y_t = \alpha \quad & +\gamma_0 X_t + (\gamma_0 + \gamma_1 + \gamma_2) X_{t-1} + (\gamma_0 + 2\gamma_1 + 4\gamma_2) X_{t-2} \\
& +(\gamma_0 + 3\gamma_1 + 9\gamma_2) X_{t-3} + (\gamma_0 + 4\gamma_1 + 16\gamma_2) X_{t-4} \\
& +(\gamma_0 + 5\gamma_1 + 25\gamma_2) X_{t-5} + (\gamma_0 + 6\gamma_1 + 36\gamma_2) X_{t-6} \\
& +(\gamma_0 + 7\gamma_1 + 49\gamma_2) X_{t-7} + (\gamma_0 + 8\gamma_1 + 64\gamma_2) X_{t-8} \\
& +\epsilon_t.
\end{aligned} \tag{7}$$

To simplify the model, we find X-variables associated with each γ-parameter and rewrite them as new variables:

$$
\begin{aligned}
Z_{t0} &= X_t + X_{t-1} + X_{t-2} + X_{t-3} + X_{t-4} + X_{t-5} + X_{t-6} + X_{t-7} + X_{t-8} \\
Z_{t1} &= X_{t-1} + 2X_{t-2} + 3X_{t-3} + 4X_{t-4} + 5X_{t-5} + 6X_{t-6} + 7X_{t-7} + 8X_{t-8} \\
Z_{t2} &= X_{t-1} + 4X_{t-2} + 9X_{t-3} + 16X_{t-4} + 25X_{t-5} + 36X_{t-6} + 49X_{t-7} + 64X_{t-8}
\end{aligned} \tag{8}
$$

and the model becomes:

$$
Y_t = \alpha + \gamma_0 Z_{t0} + \gamma_1 Z_{t1} + \gamma_2 Z_{t2} + \epsilon_t. \tag{9}
$$

Once the model is fitted and the estimates of γ coefficients, namely $\hat{\gamma}_k$ are found, we can go back and find the estimates of β coefficients, $\hat{\beta}_k$ using Eq. (7).

The following code chunk applies the polynomial distributed lag model.

```
model2 = polyDlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 8 ,
                 k = 2, show.beta = FALSE )
summary(model2)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -399.46 -123.35    4.78  118.15  516.09
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 51.572529  53.164239   0.970  0.33509
## z.t0         0.067168   0.015227   4.411 3.34e-05 ***
## z.t1         0.038180   0.012795   2.984  0.00383 **
## z.t2        -0.005128   0.001625  -3.156  0.00229 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 188.8 on 76 degrees of freedom
## Multiple R-squared:  0.9928, Adjusted R-squared:  0.9925
## F-statistic:  3481 on 3 and 76 DF,  p-value: < 2.2e-16
```

Now, the model and all parameters are significant at 5% level of significance. Because we used a 2nd order polynomial, we will use the following equation to find estimates of original β parameters

$$
\hat{\beta}_s = \gamma_0 + s\gamma_1 + s^2\gamma_2 \tag{10}
$$

where $s = 0, \ldots, q$ Also, we need to find standard errors of β parameters. The following code chunk uses `polyDlm()` function with `show.beta` argument is set to `TRUE` to generate estimates of lag weights (β parameters) and their standard deviations using standard regression methodology.

```
model2 = polyDlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 8 ,
                 k = 2, show.beta = TRUE )
```

```
## Estimates and t-tests for beta coefficients:
##        Estimate Std. Error t value  P(>|t|)
## beta.0   0.0672    0.01520    4.41 3.53e-05
## beta.1   0.1000    0.00511   19.60 1.41e-30
## beta.2   0.1230    0.00541   22.70 1.37e-34
## beta.3   0.1360    0.00941   14.40 6.83e-23
## beta.4   0.1380    0.01070   12.90 2.52e-20
## beta.5   0.1300    0.00908   14.30 9.76e-23
## beta.6   0.1120    0.00534   20.90 2.55e-32
## beta.7   0.0832    0.00735   11.30 1.19e-17
## beta.8   0.0444    0.01800    2.47 1.58e-02
```

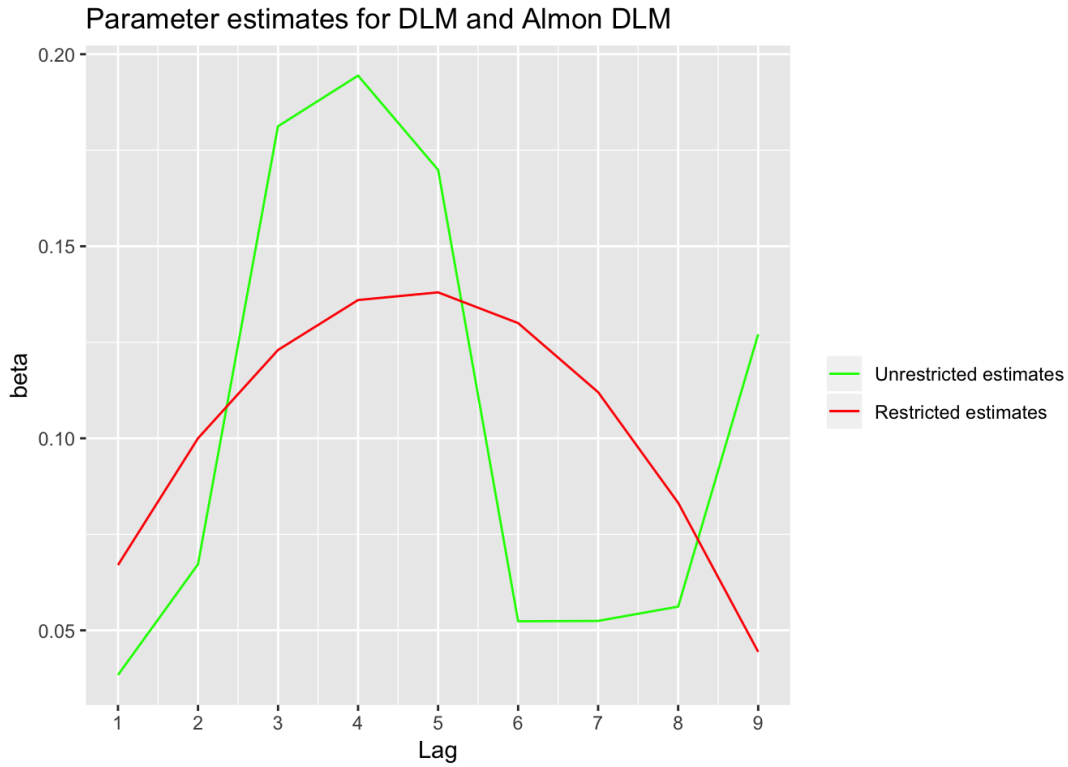
```
summary(model2)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -399.46 -123.35    4.78  118.15  516.09
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 51.572529  53.164239   0.970  0.33509
## z.t0         0.067168   0.015227   4.411 3.34e-05 ***
## z.t1         0.038180   0.012795   2.984  0.00383 **
## z.t2        -0.005128   0.001625  -3.156  0.00229 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 188.8 on 76 degrees of freedom
## Multiple R-squared:  0.9928, Adjusted R-squared:  0.9925
## F-statistic:  3481 on 3 and 76 DF,  p-value: < 2.2e-16
```

Now, all the distributed lag weights are significant at 5% level of significance. Also, notice that the standard errors of estimators are much less than their unconstrained counterparts. This implies that we have more precise estimates with polynomial DLM.

Also, we plot both sets of estimates for distributed lag weights from unrestricted and restricted models below:

```
beta.model1 = model1$model$coefficients[2:10]
beta.model2 = c(0.067, 0.1, 0.123, 0.136, 0.138, 0.13, 0.112, 0.0832, 0.0444)
xx = 1:9
df = data.frame(xx,beta.model1,beta.model2)
ggplot(df, aes(xx)) +
  geom_line(aes(y=beta.model1,colour = "Unrestricted estimates")) +
  geom_line(aes(y=beta.model2, colour = "Restricted estimates")) +
  ylab("beta") + xlab("Lag") + labs(title = "Parameter estimates for DLM and Almon D
  LM") +
  scale_colour_manual("",breaks = c("Unrestricted estimates", "Restricted estimates"
  ), values = c("red", "green")) +
  scale_x_continuous(breaks = seq(1,9,1))
```



While the restricted estimates are going upward and the downward, unrestricted ones have a polynomial shape following the restrictions created by using a 2nd order polynomial. We can infer that the effect of change in capital appropriations X_t at time t leads to an increase in capital expenditures in the current period Y_t by a relatively small amount. However, the expenditures arising from the appropriation decision increase during the next four quarters, before the effects begin to taper off.

Specification of Finite Lag Lenght

Specification of an optimal lag length of q is the most crucial part of DLMs. Under- or over-specification can lead to biased inferences. There are different approaches proposed in the literature for this task. We will focus on a simple approach that is based on the maximization of goodness-of-fit.

In the first approach,

1. select a *maximum* lag length Q,
2. find the value of q that minimizes AIC or BIC.

For the quarterly capital expenditures and appropriations series, let's start with the maximum lag length of $Q = 12$ and apply the approach mentioned above. To apply the approach, first, we will fit the models with $q = 1, 2, \ldots, 12$ and then compute their ACI and BIC values and sort the models according to these goodness-of-fit measures. The following code chunk implements this approach.

```
model.12 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 12 )$model
model.11 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 11 )$model
model.10 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 10 )$model
model.9 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 9 )$model
model.8 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 8 )$model
model.7 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 7 )$model
model.6 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 6 )$model
model.5 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 5 )$model
model.4 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 4 )$model
model.3 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 3 )$model
model.2 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 2 )$model
model.1 = dlm(x = as.vector(firms$x) , y = as.vector(firms$y) , q = 1 )$model
# "$model" is added here to get only the model component

models.AIC = AIC(model.12,model.11,model.10,model.9,model.8,model.7,model.6,model.5,
  model.4,model.3,model.2,model.1)
models.BIC = BIC(model.12,model.11,model.10,model.9,model.8,model.7,model.6,model.5,
  model.4,model.3,model.2,model.1)

sort.score(models.AIC, "aic")
```

```
##          df      AIC
## model.12 15 1030.257
## model.11 14 1041.821
## model.10 13 1052.525
## model.9  12 1063.907
## model.8  11 1075.884
## model.7  10 1091.676
## model.6   9 1113.934
## model.5   8 1144.220
## model.4   7 1181.259
## model.3   6 1231.763
## model.2   5 1294.976
## model.1   4 1360.432
```

```
sort.score(models.BIC,  "bic")
```

```
##          df      BIC
## model.12 15 1065.218
## model.11 14 1074.634
## model.10 13 1083.162
## model.9  12 1092.341
## model.8  11 1102.086
## model.7  10 1115.620
## model.6   9 1135.594
## model.5   8 1163.571
## model.4   7 1198.275
## model.3   6 1246.419
## model.2   5 1307.248
## model.1   4 1370.295
```

Both ACI and BIC measures suggest the use of lag length 12. This approach could also be applied over the Almon DLMs. But in this case, computations would be more complex due to transformations to create Z variables.

The Geometric Lag Model

In this model, lag weights are all positive and decline *geometrically*. This approach is also used with infinite DLMs. In this model, the transformation of the parameters is

$$\beta_s = \beta\phi^s \tag{11}$$

where $|\phi| < 1, \beta$ is a scaling parameter. The β_s value gets closer to zero as s gets larger. As a result of this, the most recent part of the model is more heavily weighted than the more distant part. Substituting Eq. (11) into the infinite DLM in Eq. (1), we get

$$\begin{aligned} Y_t &= \alpha + \beta_0 X_t + \beta_1 X_{t-1} + \beta_2 X_{t-2} + \cdots + \epsilon_t \\ &= \alpha + \beta(X_t + \phi X_{t-1} + \phi^2 X_{t-2} + \phi^3 X_{t-3} + \cdots) + \epsilon_t. \end{aligned} \tag{12}$$

This model is called infinite geometric DLM. In this model, we have three parameters: an intercept α, a scale factor β, and the rate at which the weights decline ϕ. The effect of one unit change in X_{t-s} on $E(Y_t)$ is

$$\frac{\partial E(Y_t)}{\partial X_{t-s}} = \beta_s = \beta\phi^s \tag{13}$$

Implementation of this model seems to be impossible. The following transformation helps to implement this infinite geometric DLM.

The Koyck Transformation

The geometric DLM is nonlinear in terms of its parameters. One way to deal with this infinite DLM is to use Koyck transformation. To apply Koyck transformation, we will take the first lag of geometric DLM in Eq. (12), multiply by ϕ, and subtract the result from Eq. (12). So we get the following:

$$\begin{aligned} Y_t - \phi Y_{t-1} &= [\alpha + \beta(X_t + \phi X_{t-1} + \phi^2 X_{t-2} + \phi^3 X_{t-3} + \cdots) + \epsilon_t] \\ &\quad -\phi[\alpha + \beta(X_{t-1} + \phi X_{t-2} + \phi^2 X_{t-3} + \phi^3 X_{t-4} + \cdots) + \epsilon_{t-1}] \\ &= \alpha(1-\phi) + \beta X_t + (\epsilon_t - \phi\epsilon_{t-1}). \end{aligned} \tag{14}$$

When we solve Eq. (14) for Y_t, we obtain Koyck form of the geometric DLM as follows:

$$\begin{aligned} Y_t &= \alpha(1-\phi) + \phi Y_{t-1} + \beta X_t + (\epsilon_t - \phi\epsilon_{t-1}) \\ &= \delta_1 + \delta_2 Y_{t-1} + \delta_3 X_t + \nu_t. \end{aligned} \tag{15}$$

where $\delta_1 = \alpha(1-\phi), \delta_2 = \phi, \delta_3 = \beta$and the random error after the transformation is $\nu_t = (\epsilon_t - \phi\epsilon_{t-1})$.

Notice that the model obtained in Eq. (15) is in the form of a simple multiple linear regression model. The differences between Koyck models and an ordinary multiple regression model are

- one of the explanatory variables is the first lag of the dependent variable, namely Y_{t-1}
- the error term ν_t depends on both ϵ_t and ϵ_{t-1}.

The consequence of these differences is that Y_{t-1} and the error term ν_t will be correlated to each other. This makes the least-squares estimates biased and inconsistent. So, the use of least squares method is not recommended with the Koyck model.

We will use another technique called **instrumental variables estimator**. Because Y_{t-1} is correlated with the error terms, this variable creates the problem in the estimation process. We will represent this variable with an appropriate instrument X_{t-1}, which is correlated with Y_{t-1} but uncorrelated with the error term ν_t. Because of the correlation between Y_{t-1} and X_{t-1}, X_{t-1} contains most of the information carried by Y_{t-1}. So use of X_{t-1} in place of Y_{t-1} is suitable. Then we will use two-staged least squares to get parameter estimates.

1. Find estimates of Y_{t-1}, namely $\hat{Y}_{t-1}$ by regressing it on X_{t-1} through the simple linear regression model

$$Y_{t-1} = a_0 + a_1 X_{t-1} + error. \tag{16}$$

So, we will use the estimation equation

$$\hat{Y}_{t-1} = \hat{a}_0 + \hat{a}_1 X_{t-1}. \tag{17}$$

2. Then, fit the following model with instrumental variable using least squares method:

$$Y_t = \delta_1 + \delta_2 \hat{Y}_{t-1} + \delta_3 X_t + error. \tag{18}$$

Here X_{t-1} is an instrument for Y_{t-1} and X_t is an instrument for itself.

Notice that this approach will not produce a standard error of parameter estimators and hence we will not be able to carry on t-tests for the significance of parameters. Therefore, we need to use packages specifically designed to implement two-staged least squares method.

The `koyckDlm()` function utilises the `ivreg()` (https://www.rdocumentation.org/packages/AER/versions/1.2-5/topics/ivreg) function from the package `AER` to implement the two-staged least squares method.

Let's turn back to our numerical example on the quarterly US capital expenditures and appropriations series and apply this approach over Koyck DLM.

The following code chunk implements Koyck DLM with the quarterly US capital expenditures and appropriations series.

```
model4 = koyckDlm(x = as.vector(firms$x) , y = as.vector(firms$y) )
summary(model4)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -346.243  -95.894   -1.308   96.646  612.698
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -28.90734   36.66470  -0.788    0.433
## Y.1           0.81532    0.02064  39.500   <2e-16 ***
## X.t           0.18028    0.01422  12.680   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 145.2 on 84 degrees of freedom
## Multiple R-Squared: 0.9957,  Adjusted R-squared: 0.9956
## Wald test:  9807 on 2 and 84 DF,  p-value: < 2.2e-16
##
##                             alpha      beta       phi
## Geometric coefficients:  -156.5277 0.1802848 0.8153212
```

```
summary(model4$model, diagnostics = TRUE)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -346.243  -95.894   -1.308   96.646  612.698
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -28.90734   36.66470  -0.788    0.433
## Y.1           0.81532    0.02064  39.500   <2e-16 ***
## X.t           0.18028    0.01422  12.680   <2e-16 ***
##
## Diagnostic tests:
##                  df1 df2 statistic  p-value
## Weak instruments   1  84    252.43  < 2e-16 ***
## Wu-Hausman         1  83     44.16 2.97e-09 ***
## Sargan             0  NA        NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 145.2 on 84 degrees of freedom
## Multiple R-Squared: 0.9957,  Adjusted R-squared: 0.9956
## Wald test:  9807 on 2 and 84 DF,  p-value: < 2.2e-16
```

```
# If set to TRUE, diagnostics argument shows three tests:
# 1. an F test of the first stage regression for weak instruments
# 2. the Wu-Hausman test for the significance of the correlation between an explanat
  ory variable and the error term (endogeneity)
# 3. the Sargan test of overidentifying restrictions (only if there are more instrum
  ents than regressors).
```

From the `Weak Instruments` line, we conclude that the model at the first stage of the least-squares fitting is significant at 5% level of significance. In this model, both δ_2 and δ_3 are significant at 5% level. Using the coefficients, we find the estimates of original parameters such that $\hat{\beta} = 0.18$ and $\hat{\phi} = 0.815$. So the weights will decline quickly at the rate of 0.815. Thus, the first lags of appropriations will have a more strong impact on expenditures than the latter ones.

In the Koyck DLM, we cannot be sure whether the error term is correlated with the lagged dependent variable or not. So, we will test the error term ν_t for autocorrelation using the **Wu-Hausman test**. The Wu-Hausman test compares the consistency of instrumental variable estimates to ordinary least squares estimates.

If the endogeneity if present, the instrumental variable estimator is concluded to be consistent while the ordinary least squares estimator is inconsistent.

From the Wu-Hausman test result in the model output, we reject the null hypothesis that *the correlation between explanatory variable and the error term is zero (There is no endogeneity)* at 5% level. So, there is a significant correlation between the explanatory variable and the error term at 5% level.

But notice that this test should be used cautiously with small samples (like $n < 100$) due to its asymptotic validity.

Autoregressive Distributed Lag Model

Autoregressive DLMs are useful when we cannot find suitable solutions with neither polynomial nor Kyock DLMs. Actually, the autoregressive DLM is a flexible and parsimonious infinite DLM. For example,

$$Y_t = \mu + \beta_0 X_t + \beta_1 X_{t-1} + \gamma_1 Y_{t-1} + e_t \tag{19}$$

is an autoregressive DLM. Similar to the Koyck DLM, it is possible to write this model as an infinite DLM. This form of the model is denoted by ARDL(1,1). In general, the model is denoted as ARDL(p, q) and the model equation is as follows:

$$Y_t = \mu + \beta_0 X_t + \beta_1 X_{t-1} + \cdots + \beta_p X_{t-p} + \gamma_1 Y_{t-1} + \cdots + \gamma_q Y_{t-q} + e_t. \tag{20}$$

With sufficiently large values of p and q, the model is flexible enough to approximate an infinite lag distribution of any shape rather than a polynomial or geometric shape.

Let's revisit the capital expenditures series to illustrate the model. We will fit the ARDL models with $p = q = 1, 2, 3$. To fit the model we will use `ardlDlm()` function from `dLagM` package. This package provides a very flexible implementation for the model including differences, lags, seasonal and harmonic components.

```
model.11 = ardlDlm(x = as.vector(firms$x) , y = as.vector(firms$y) , p = 1 ,
                   q = 1 )$model
summary(model.11)
```

```
##
## Time series regression with "ts" data:
## Start = 2, End = 88
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -281.42  -59.02   14.44   76.16  214.85
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.40362   28.95567  -0.083   0.9340
## X.t          0.03478    0.01897   1.834   0.0703 .
## X.1          0.15619    0.02350   6.645 2.97e-09 ***
## Y.1          0.80080    0.01701  47.080  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 111.8 on 83 degrees of freedom
## Multiple R-squared:  0.9975, Adjusted R-squared:  0.9974
## F-statistic: 1.104e+04 on 3 and 83 DF,  p-value: < 2.2e-16
```

```
formula = y ~ x
model.22 = ardlDlm(formula = formula, data = firms, p = 2 ,
                   q = 2 )$model
summary(model.22)
```

```
##
## Time series regression with "ts" data:
## Start = 3, End = 88
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -249.915  -52.520    9.437   69.408  231.196
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.52056   26.90555   0.280  0.78057
## x.t          0.04377    0.01758   2.491  0.01482 *
## x.1          0.04948    0.03545   1.396  0.16670
## x.2          0.09110    0.03796   2.400  0.01873 *
## y.1          1.03777    0.10064  10.312 2.39e-16 ***
## y.2         -0.23385    0.08258  -2.832  0.00586 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 102.3 on 80 degrees of freedom
## Multiple R-squared:  0.998,  Adjusted R-squared:  0.9978
## F-statistic:  7808 on 5 and 80 DF,  p-value: < 2.2e-16
```

```
model.33 = ardlDlm(x = as.vector(firms$x) , y = as.vector(firms$y) , p = 3 ,
                   q = 3 )$model
summary(model.33)
```

```
##
## Time series regression with "ts" data:
## Start = 4, End = 88
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -276.279  -53.633    0.812   63.560  232.205
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.78350   27.44020   0.393   0.6954
## X.t          0.04064    0.01831   2.220   0.0294 *
## X.1          0.06164    0.03718   1.658   0.1014
## X.2          0.05224    0.04707   1.110   0.2705
## X.3          0.05236    0.04059   1.290   0.2010
## Y.1          0.96362    0.11235   8.577 7.86e-13 ***
## Y.2         -0.13152    0.15564  -0.845   0.4007
## Y.3         -0.05304    0.08842  -0.600   0.5503
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 102.6 on 77 degrees of freedom
## Multiple R-squared:  0.998,  Adjusted R-squared:  0.9978
## F-statistic:  5474 on 7 and 77 DF,  p-value: < 2.2e-16
```

```
models.AIC = AIC(model.11,model.22,model.33)
models.BIC = BIC(model.11,model.22,model.33)
sort.score(models.AIC, "aic")
```

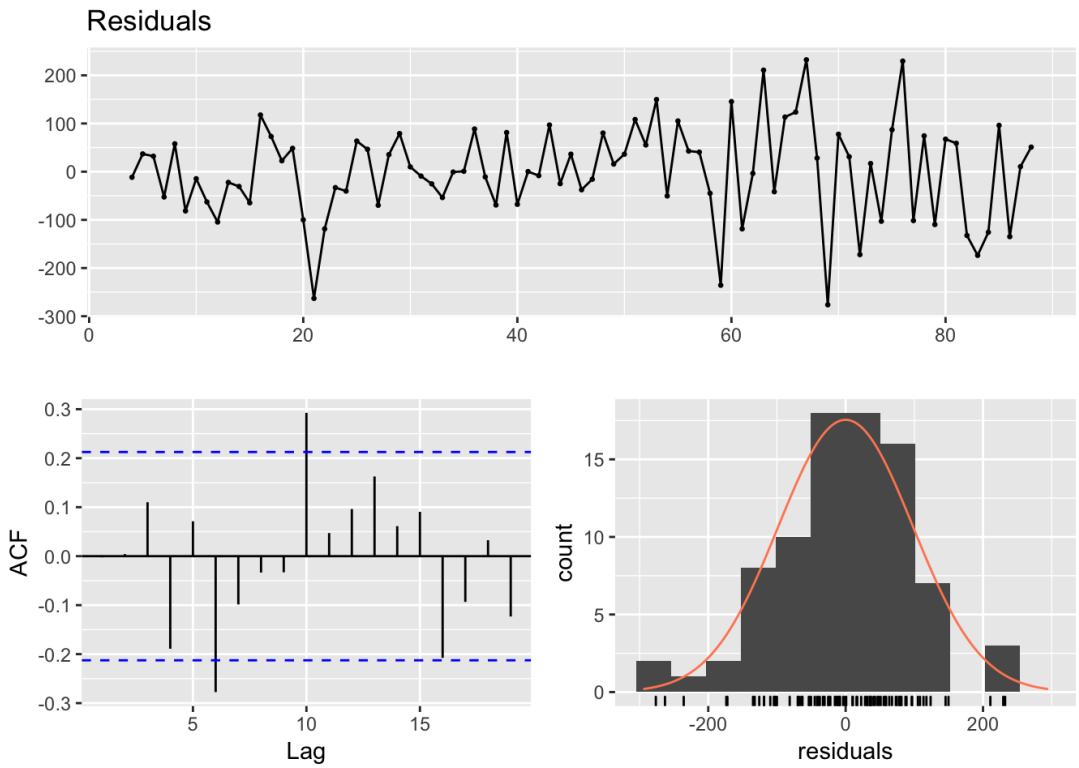
```
##          df      AIC
## model.33  9 1038.106
## model.22  7 1047.898
## model.11  5 1073.452
```

```
sort.score(models.BIC,  "bic")
```

```
##          df      BIC
## model.33  9 1060.090
## model.22  7 1065.079
## model.11  5 1085.782
```

According to both AIC and BIC, ARDL(3,3) model is the most accurate one. We will display diagnostic plots of residuals from ARDL(3,3) to examine residuals.

```
checkresiduals(model.33$residuals)
```




```
bgtest(model.33)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model.33
## LM test = 0.017265, df = 1, p-value = 0.8955
```

From the time series plot of standardised residuals, we can infer that standardised residuals are distributed around zero mean level as desired. From the histogram and QQ plots, we can say that standardised residuals are distributed according to standard normal distribution within $(-3, 3)$ range; hence, we do not have an

influential point. ACF and PACF of the standardised residuals do not suggest the existence of autocorrelation within error terms. This is an important and desired property for the fitted ARDL models.

Forecasting

Forecasting with DLMs is a straightforward task as in simple linear regression analysis. We will directly use model and parameter estimates to calculate predictions for the future values of the dependent variable. For this aim, the package `dLagM` provides a bunch of forecasting functions for DLMs.

Finite distributed lag model

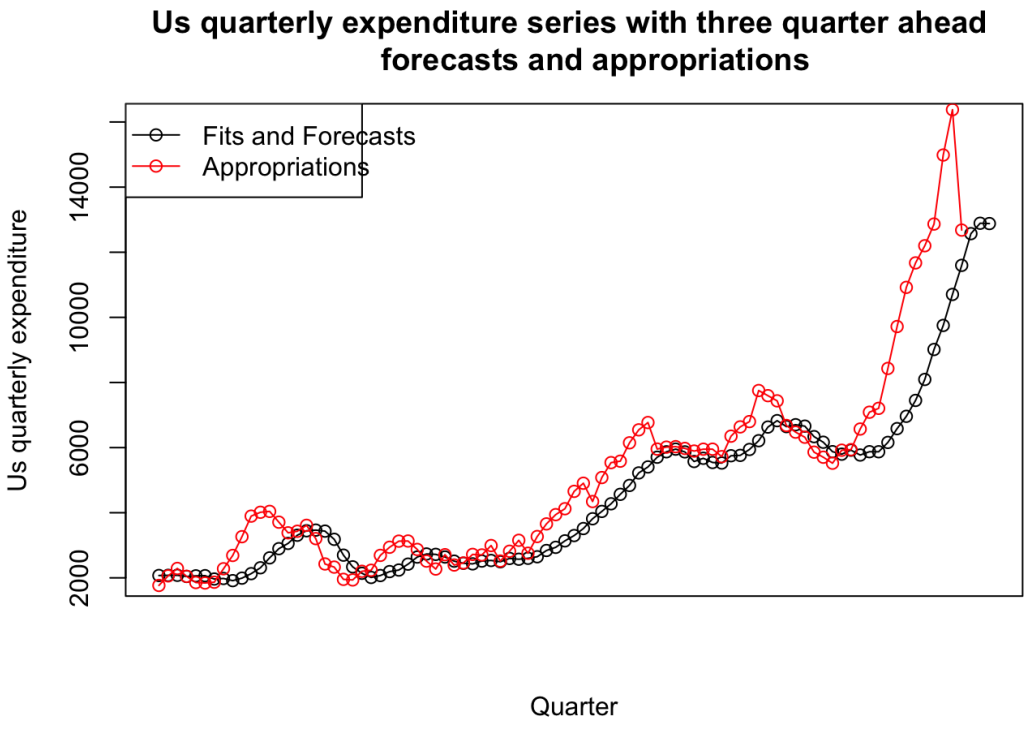
For our numerical example, let's calculate 3 quarters ahead forecasts. So, we will find $\hat{Y}_{89}$, $\hat{Y}_{90}$, and $\hat{Y}_{91}$. Suppose the given values of appropriations for there periods are $X_{89} = 13500$, $X_{90} = 14700$, $_{91} = 13980$, respectively. To get forecasts for the finite distributed lag model with $q = 8$, we will use `dlmForecast()` function from the `dLagM` package.

```
forecasts.dlm = dLagM::forecast(model = model1 ,
                                x = c(13500 , 14700 , 13980) ,
                                h = 3)$forecasts
forecasts.dlm
```

```
## [1] 12570.83 12890.35 12880.61
```

```
y.extended = c(firms$y , forecasts.dlm)

plot(ts(y.extended),type="o",xaxt="n", ylim= c(2000, 16000),
     ylab = "Us quarterly expenditure", xlab = "Quarter",
     main="Us quarterly expenditure series with three quarter ahead
     forecasts and appropriations")
lines(firms$x,col="Red",type="o")
legend("topleft",lty=1, pch = 1, text.width = 16, col=c("black","red"),
       c("Fits and Forecasts", "Appropriations"))
```



Ploynomial distributed lag model

For the polynomial DLM, recall that we rewrite the model with a polynomial transformation. For example, for the second-order polynomial model, we have the following prediction model

$$\hat{Y}_s = \hat{\alpha} + \hat{\gamma}_0 Z_{s0} + \hat{\gamma}_1 Z_{s1} + \hat{\gamma}_2 Z_{s2}. \tag{21}$$

The forecast for the time $s > n$, we first need to compute the values of Z variables Z_{s1}, Z_{s1}, Z_{s2} using the Eq. (8) with t is replaced by s and write the calculated values in place in Eq. (21).

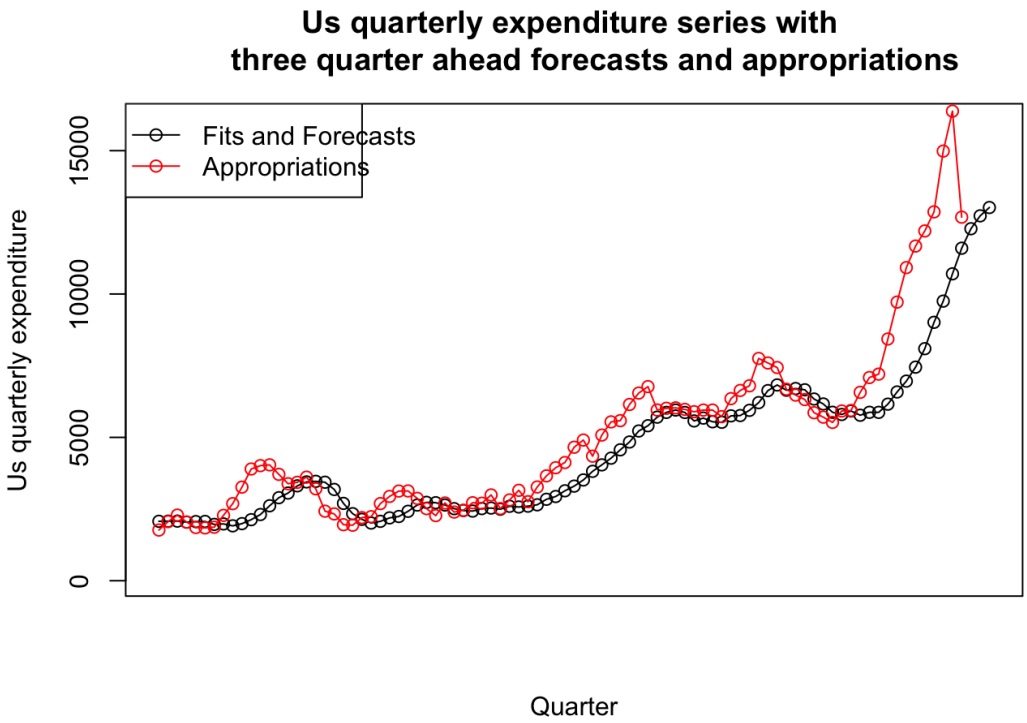
For our numerical example, lets calculate 3 quarters ahead forecasts. So, we will find $\hat{Y}_{89}, \hat{Y}_{90}$, and $\hat{Y}_{91}$. Suppose the given values of appropriations for there periods are $X_{89} = 13500, X_{90} = 14700, {}_{91} = 13980$, respectively. The following code chunk computes Z variables:

```
forecasts.polydlm = dLagM::forecast(model = model2 , x = c(13500 , 14700 , 13980) ,
  h = 3)$forecasts
forecasts.polydlm
```

```
## [1] 12276.48 12726.68 13017.09
```

```
y.extended = c(firms$y , forecasts.polydlm)

{# Put these curlies to put all displays into the same plot
plot(ts(y.extended),type="o",xaxt="n", ylim= c(100, 16000),
ylab = "Us quarterly expenditure", xlab = "Quarter",
main="Us quarterly expenditure series with
     three quarter ahead forecasts and appropriations")
par(new=TRUE)
lines(firms$x,col="Red",type="o")
legend("topleft",lty=1, pch = 1, text.width = 16,
       col=c("black","red"), c("Fits and Forecasts", "Appropriations"))
}
```



Koyck distributed lag model

For the Koyck DLM, we will use the usual `predict()` function to generate forecasts. The prediction equation for Koyck DLM is

$$\hat{Y}_s = \hat{\delta}_1 + \hat{\delta}_2 \hat{Y}_{s-1} + \hat{\delta}_3 X_s. \tag{22}$$

For one step ahead forecast $s = t + 1$, we have

$$\hat{Y}_{t+1} = \hat{\delta}_1 + \hat{\delta}_2 \hat{Y}_t + \hat{\delta}_3 X_{t+1}. \tag{23}$$

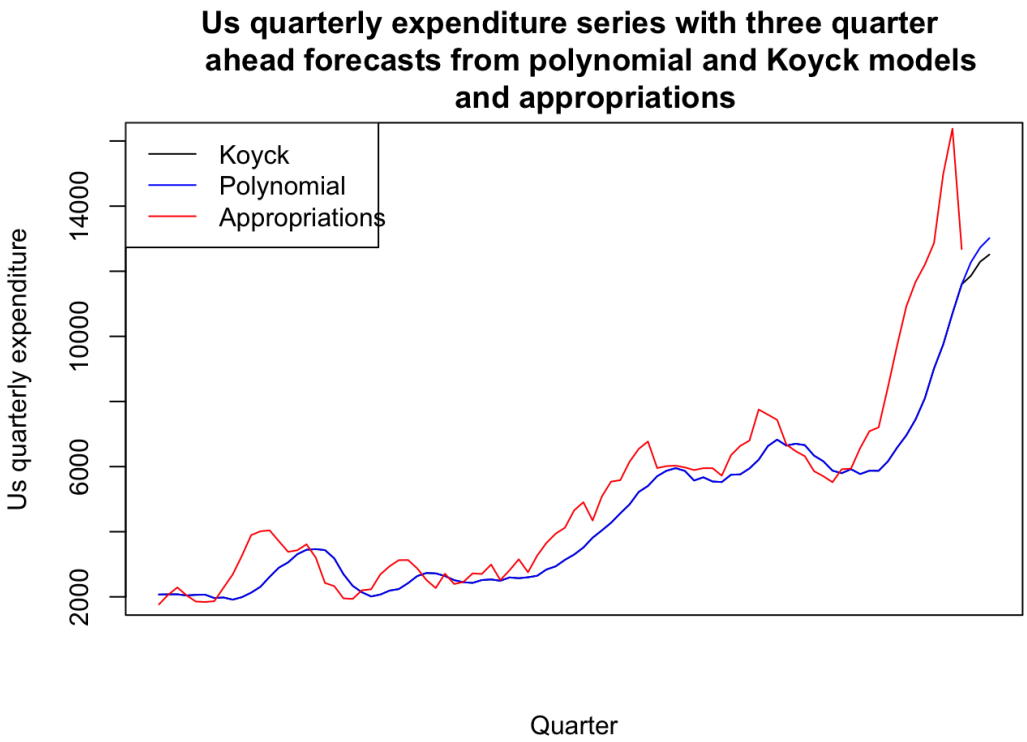
So, we will use the last fitted $\hat{Y}_t$ and new value of independent variable X_{t+1}. For $\hat{Y}_{89}$, we will use $\hat{Y}_{88} = 10984.302$ and $X_{89} = 13500$ such that

```
forecasts.koyckdlm = dLagM::forecast(model = model4 , x = c(13500 , 14700 , 13980) ,
   h = 3)$forecasts
forecasts.koyckdlm
```

```
## [1] 11860.22 12291.17 12512.72
```

```
y.extended2 = c(firms$y , forecasts.koyckdlm)

plot(ts(y.extended2),type="l",xaxt="n", ylim= c(2000, 16000),
ylab = "Us quarterly expenditure", xlab = "Quarter",
main="Us quarterly expenditure series with three quarter
     ahead forecasts from polynomial and Koyck models
     and appropriations")
lines(y.extended,col="Blue",type="l")
lines(firms$x,col="Red",type="l")
legend("topleft",lty=1, text.width = 16,
       col=c("black","blue","red"),
       c("Koyck", "Polynomial", "Appropriations"))
```



Fits and forecasts from polynomial and Koyck models are close to each other.

Autorregressive distributed lag model

The way we proceed to generate forecasts for ARDL models is the same as the one with Koyck DLM. We will compute the forecasts starting with the last fitted value.

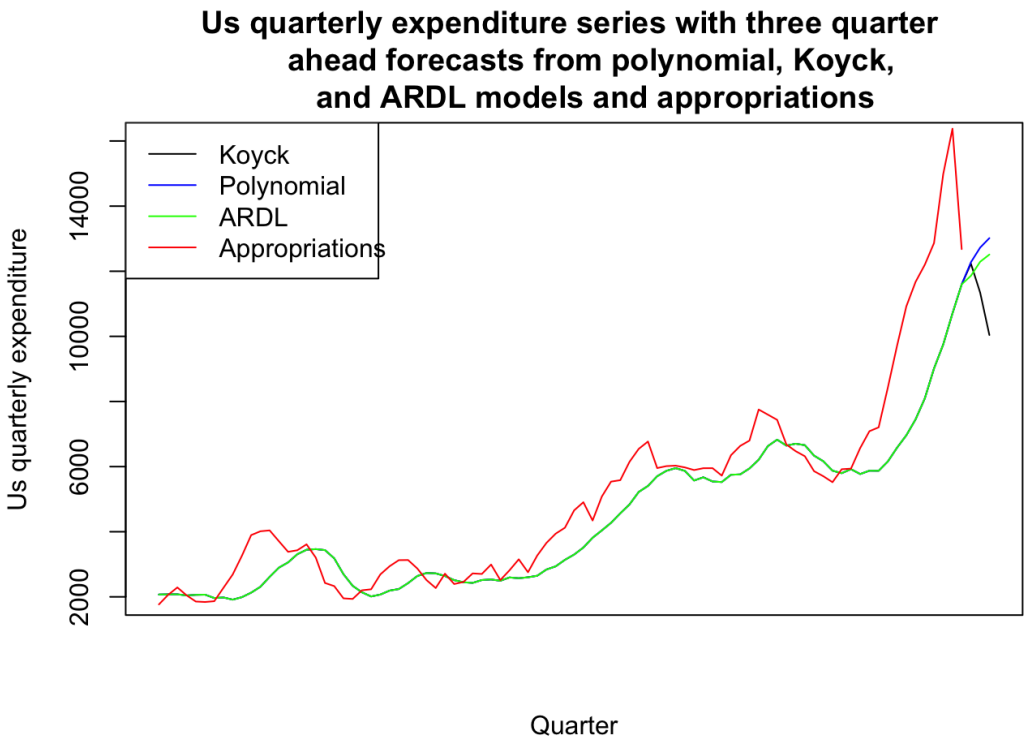
Let's find 3 times ahead forecasts for the US quarterly expenditures series with previously given values of future appropriations.

```
model.33 = ardlDlm(x = as.vector(firms$x) , y = as.vector(firms$y) , p = 3 , q = 3)
# Model is fitted again without "$model" to get all outputs
forecasts.ardldlm = dLagM::forecast(model = model.33 ,  x = c(13500 , 14700 , 13980)
   , h = 3)$forecasts
forecasts.ardldlm
```

```
## [1] 12231.29 11343.52 10041.10
```

```
y.extended3 = c(firms$y , forecasts.ardldlm)

{
plot(ts(y.extended3),type="l",xaxt="n", ylim= c(2000, 16000),
ylab = "Us quarterly expenditure", xlab = "Quarter",
main="Us quarterly expenditure series with three quarter
     ahead forecasts from polynomial, Koyck,
     and ARDL models and appropriations")
lines(y.extended,col="Blue",type="l")
lines(y.extended2,col="Green",type="l")
lines(firms$x,col="Red",type="l")
legend("topleft",lty=1, text.width = 16,
       col=c("black","blue","green", "red"),
       c("Koyck", "Polynomial", "ARDL", "Appropriations"))
}
```



Summary

In this module, we worked on the inclusion of an independent variable into the time series analysis through different regression approaches. We focused on

- finite distributed lag model, where we need to choose the lag length and deal with multicollinearity;
- polynomial distributed lag model, which addresses the collinearity by changing lag weights smoothly;
- the infinite distributed lag model removes the need to specify lag length but it is impossible to fit the model in its straightforward form;
- to be able to fit the model, we used geometric lag weights and converted the model to a form in which we can fit the model;
- to find the parameter estimated we transformed the geometrically distributed lag model into Koyck form and came up with a finite number of parameters to be estimated.
- Lastly, we discussed the calculation of forecasts for the considered distributed lag models.

References

G.G. Judge, W.E. Griffiths. Undergraduate Econometrics. Wiley, 2000.