Lecture 4, September 22, 2025

Estimation of ADL models

Marta Boczon

Department of Economics

Copnehagen Business School

mbo.eco@cbs.dk

In [1]:
```
#install.packages("quantmod")
#install.packages("fredr")
#install.packages("ggfortify")
#install.packages('urca')
#install.packages("tseries")
#install.packages("forecast")
#install.packages("dynlm")
#install.packages("stargazer")
#install.packages("pracma")
#install.packages("dLagM")
#install.packages("gets")
#install.packages("astsa")
options(warn=-1)
```

Roadmap

Today's lecture focuses on learning by doing with a real data example — we will estimate an **ADL (Autoregressive Distributed Lag) model** to explore whether **hours worked in the Canadian manufacturing sector** help explain the **growth rate of Canadian GDP**.

Data Overview

1. Canadian GDP

1. **Data:** Real Gross Domestic Product for Canada
2. **Source:** International Monetary Fund via FRED®
3. **Frequency:** Quarterly
4. **Units:** Millions of domestic currency
5. **Seasonal adjustment:** Seasonally adjusted

https://fred.stlouisfed.org/series/NGDPRSAXDCCAQ

2. Hours Worked (Manufacturing Sector)

1. **Data:** Hours Worked: Manufacturing: Weekly for Canada (HOHWMN02CAM065N)
2. **Source:** Organization for Economic Co-operation and Development via FRED®
3. **Frequency:** Quarterly (aggregation method: Average)
4. **Units:** Hours
5. **Seasonal adjustment:** Not seasonally adjusted

https://fred.stlouisfed.org/series/HOHWMN02CAM065N

Steps for Estimating an ADL Model (Assuming Stationarity)

1. Create a joint time series object of at least two stationary series with the same frequency.
2. Find the best-fitting ADL model.
3. Estimate and interpret the best-fitting ADL model.

Before we begin, we need to ensure that all data are stationary. This requires revisiting the **five steps from last week's lecture**, now applied to the *Hours Worked* series:

1. **Import** the dataset into the software.
2. **Create** a time series object.
3. **Plot** the data to examine general patterns.
4. **Test** whether the series has a unit root.
5. **Transform** the data, if necessary, to remove non-stationarity, and then re-test.

Revisiting the ARIMA Model

Before estimating the ADL model, we will briefly revisit whether **ARIMA(0,0)** — essentially white noise — is indeed the best-fitting model for Canadian GDP. To check this, we will loop over multiple ARIMA specifications and compare them using information criteria (AIC and BIC). This will show whether any more complex model provides a significantly better fit. Once we confirm the appropriate ARIMA structure, we will move on to extending the framework by adding explanatory variables — leading us naturally to the ADL and ARIMAX models.

In [2]:
```
# Import the dataset from a CSV file downloaded from FRED
gdp =read.csv("NGDPRSAXDCCAQ.csv")
# Convert the second column of the dataset (the actual GDP values) into a time series object
tsgdp = ts(gdp[, 2], frequency = 4, start = c(1961, 1))
# Transform the series into (approximate) percentage changes
dgdp = diff(log(tsgdp))
```

Fitting ARIMA Model with auto.arima

In [21]:
```
library(forecast)   # Load the forecast package for auto.arima()

auto.arima(dgdp,
           max.d = 0,    # no differencing allowed (series already made stationary)
           max.p = 4,    # maximum AR order (p)
           max.q = 4,    # maximum MA order (q)
           max.D = 0,    # no seasonal differencing
           max.P = 0,    # no seasonal AR terms
           max.Q = 0,    # no seasonal MA terms
           ic = "aic",   # choose the model that minimizes AIC
           stepwise = FALSE,     # search over all models (not just stepwise path)
           approximation = FALSE # use exact maximum likelihood (slower but more accurate)
)
```

```
Series: dgdp
ARIMA(1,0,1) with non-zero mean

Coefficients:
        ar1      ma1    mean
      0.990  -0.9636  0.0080
s.e.  0.015   0.0240  0.0023

sigma^2 = 0.0001549:  log likelihood = 760.94
AIC=-1513.87   AICc=-1513.71   BIC=-1499.69
```

In [4]:
```
library(forecast)   # Load the forecast package for auto.arima()

auto.arima(dgdp,
           max.d = 0,    # no differencing allowed (series already made stationary)
           max.p = 4,    # maximum AR order (p)
           max.q = 4,    # maximum MA order (q)
           max.D = 0,    # no seasonal differencing
           max.P = 0,    # no seasonal AR terms
           max.Q = 0,    # no seasonal MA terms
           ic = "bic",   # choose the model that minimizes BIC
           stepwise = FALSE,     # search over all models (not just stepwise path)
           approximation = FALSE # use exact maximum likelihood (slower but more accurate)
)
```

```
Series: dgdp
ARIMA(0,0,0) with non-zero mean

Coefficients:
        mean
      0.0075
s.e.  0.0008

sigma^2 = 0.000157:  log likelihood = 758.46
AIC=-1512.91   AICc=-1512.87   BIC=-1505.82
```

Behind the Hood of auto.arima

In [5]:
```
p = 4       # Maximum AR order
q = 4       # Maximum MA order
pq = (p+1) * (q+1)   # Total number of (p,q) combinations including 0
resAIC = rep(0, pq)  # Storage for AIC values
resBIC = rep(0, pq)  # Storage for BIC values
pdf = rep(0, pq)     # Storage for AR order (p)
qdf = rep(0, pq)     # Storage for MA order (q)

k = 0  # Counter for results
for (i in 0:p) {
    for (j in 0:q) {
        k = k + 1
        # Estimate ARMA(p,q) model for stationary series dy
        model = arima(dgdp, order = c(i,0,j))
        # Store information criteria
        resAIC[k] = AIC(model)
        resBIC[k] = BIC(model)

        # Store corresponding lag orders
        pdf[k] = i
        qdf[k] = j
    }
}

# Combine and display results: p, q, AIC, and BIC
results = cbind(pdf, qdf, data.frame(resAIC), data.frame(resBIC))
results

# Find the model with the lowest AIC
bestAIC = results[which.min(results$resAIC), ]   # Row with min AIC

# Find the model with the lowest BIC
bestBIC = results[which.min(results$resBIC), ]   # Row with min BIC

# Display them
print("According to the AIC information criterion, the best model is")
bestAIC

print("According to the BIC information criterion, the best model is")
bestBIC
```

A data.frame: 25 × 4

pdf	qdf	resAIC	resBIC
<dbl>	<dbl>	<dbl>	<dbl>
0	0	-1512.913	-1505.822
0	1	-1511.087	-1500.451
0	2	-1509.389	-1495.209
0	3	-1507.523	-1489.798
0	4	-1507.440	-1486.169
1	0	-1511.100	-1500.464
1	1	-1513.873	-1499.692
1	2	-1511.968	-1494.242
1	3	-1509.978	-1488.707
1	4	-1508.028	-1483.211
2	0	-1509.462	-1495.281
2	1	-1511.967	-1494.241
2	2	-1510.473	-1489.202
2	3	-1508.501	-1483.685
2	4	-1506.164	-1477.803
3	0	-1507.657	-1489.931
3	1	-1509.977	-1488.706
3	2	-1508.018	-1483.202
3	3	-1510.732	-1482.370
3	4	-1509.083	-1477.177
4	0	-1507.326	-1486.055
4	1	-1505.598	-1480.782
4	2	-1506.646	-1478.284
4	3	-1509.206	-1477.300
4	4	-1506.980	-1471.529

```
[1] "According to the AIC information criterion, the best model is"
```

A data.frame: 1 × 4

	pdf	qdf	resAIC	resBIC
	<dbl>	<dbl>	<dbl>	<dbl>
7	1	1	-1513.873	-1499.692

```
[1] "According to the BIC information criterion, the best model is"
```

A data.frame: 1 × 4

	pdf	qdf	resAIC	resBIC
	<dbl>	<dbl>	<dbl>	<dbl>
1	0	0	-1512.913	-1505.822

Interpretation

The manual loop we implemented — where we estimated many ARMA(p,q) models and compared their AIC and BIC values — is essentially what the automatic selection procedure does under the hood. It systematically evaluates a range of models and then chooses the one that minimizes the chosen information criterion.

Because of this, it is not correct in a report or synopsis to simply state that *"the best fitting model was chosen by auto.arima"*. We need to know what the automatic procedure is doing and explain that process clearly. This means avoiding direct references to R functions and instead describing the model selection procedure in plain language:

1. What maximum lag orders (p and q) were allowed in the search.
2. Which information criterion was used for model comparison (AIC or BIC).
3. Whether the search considered all models exactly or followed a faster stepwise strategy.
4. Whether likelihood estimation was exact or approximate, and if approximate, how much time was saved.

In other words, we describe how the model was chosen under the hood rather than pointing to the function name. This shows our understanding of both the mechanics and the reasoning behind the selected model.

For example:

We evaluated autoregressive moving average (ARMA) models with up to 4 lags in both the autoregressive and moving average terms. This meant that a total of 25 models (all combinations of p = 0,...,4 and q = 0,...,4) were estimated. The choice of 4 lags reflects a balance between allowing enough dynamics to capture quarterly fluctuations in GDP and keeping the model space manageable for interpretation.

The models were compared using the Akaike Information Criterion (AIC), and the search considered all possible specifications within this range (an exact search rather than a stepwise approximation). Likelihood estimation was carried out without approximation. Based on this procedure, the ARMA(0,0) model — which corresponds to white noise — was selected as the best fitting model.

Finding Best-Fitting Subset-ARMA Model Using Information Criteria

Auto.arima automates model selection by fitting a grid of ARMA(p,q) candidates and choosing the one that minimizes an information criterion (typically AIC or BIC), often within user-set bounds (e.g., $p, q \leq 4 \Rightarrow 5 \times 5 = 25$ models).

Next, we move to a **subset-ARMA search** that is more flexible: instead of enforcing consecutive lags up to p and q, we allow selected lags only by fixing some coefficients to zero and estimating others.

With four AR slots and four MA slots, each coefficient can be either fixed (0) or free (NA), giving $2^8 = 256$ possible specifications. We'll evaluate each feasible specification (skipping non-estimable ones) and compare AIC/BIC — trading off fit and parsimony while revealing which specific lags (e.g., AR3, MA4) actually matter.

In [6]:
```
##############################################
## Exhaustive ARIMA Subset Search with AIC/BIC
##############################################
p = 4                        # number of AR coefficients considered (AR(1) .. AR(4))
q = 4                        # number of MA coefficients considered (MA(1) .. MA(4))
pq = p + q                   # total number of non-seasonal AR and MA parameters

# Build a data frame of all 2^(p+q) combinations of {0, NA} for AR1..AR4 and MA1..MA4
# Each row is a pattern telling arima() which coefficients are fixed (0) vs free (NA).
indvar = do.call(expand.grid, replicate(pq, c(0, NA), simplify = FALSE))

## Give column names to aid interpretation (AR first, then MA)
colnames(indvar) = c("ar1","ar2","ar3","ar4","ma1","ma2","ma3","ma4")

res_list = list()           # container to store per-model results (AIC/BIC and names)

iter = 0  # Counter for results (row counter in res_list)

# Loop over every 0/NA pattern (i.e., every model specification)
for (i in 1:nrow(indvar)) {
        iter = iter + 1
        # Identify which AR/MA parameters are FREE under this pattern:
        # NA entries indicate parameters to estimate.
        active_idx  = which(is.na(indvar[i, ]))  # indices of NA among ar*/ma* slots

        # Map those indices for reporting.
        active_names = colnames(indvar)[active_idx]

        # Fit ARIMA(4,0,4) with this fixed pattern.
        # 'fixed' must contain AR1..AR4, MA1..MA4, and then the mean.
        # Appending NA at the end frees the intercept/mean in every spec.
        model = try(suppressWarnings(arima(dgdp, order = c(p,0,q), fixed =c(indvar[i,], NA))), silent = TRUE)

        # If estimation fails (e.g., non-invertible MA, singular information matrix),
        # skip this specification and continue to the next pattern.
        if (inherits(model, "try-error")) next

        # Save a compact summary row:
        # - 'regressors' lists the FREE (estimated) AR/MA parameter names for this pattern
        # - AIC/BIC are the information criteria for model comparison
        res_list[[iter]] = data.frame(
         regressors = paste(active_names, collapse = " + "),
         AIC        = AIC(model),
         BIC        = BIC(model)
        )
}

## ---- 8) Bind results into a single data frame -----------------------------
# Stack all rows from res_list into a single data frame
results = do.call(rbind, res_list)

## Quick look at the first and last few rows of the results table
head(results)
tail(results)

## ---- 9) Identify the best models by AIC and BIC --------------------------
# Find the row indices of the minimum AIC and BIC
best_by_AIC = results[which.min(results$AIC), ]
best_by_BIC = results[which.min(results$BIC), ]

## Display the best specs under each criterion
best_by_AIC
best_by_BIC
```

A data.frame: 6 × 3

	regressors	AIC	BIC
	<chr>	<dbl>	<dbl>
1		-1512.913	-1505.822
2	ar1	-1511.100	-1500.464
3	ar2	-1511.285	-1500.649
4	ar1 + ar2	-1509.462	-1495.281
5	ar3	-1511.130	-1500.494
6	ar1 + ar3	-1509.303	-1495.123

A data.frame: 6 × 3

	regressors	AIC	BIC
	<chr>	<dbl>	<dbl>
250	ar2 + ar4 + ma1 + ma2 + ma3 + ma4	-1507.974	-1479.613
251	ar1 + ar2 + ar4 + ma1 + ma2 + ma3 + ma4	-1507.324	-1475.418
252	ar3 + ar4 + ma1 + ma2 + ma3 + ma4	-1508.496	-1480.135
253	ar1 + ar3 + ar4 + ma1 + ma2 + ma3 + ma4	-1506.500	-1474.593
254	ar2 + ar3 + ar4 + ma1 + ma2 + ma3 + ma4	-1509.232	-1477.326
255	ar1 + ar2 + ar3 + ar4 + ma1 + ma2 + ma3 + ma4	-1506.980	-1471.529

A data.frame: 1 × 3

	regressors	AIC	BIC
	<chr>	<dbl>	<dbl>
196	ar3 + ma3 + ma4	-1515.354	-1497.628

A data.frame: 1 × 3

	regressors	AIC	BIC
	<chr>	<dbl>	<dbl>
1		-1512.913	-1505.822

Subset-ARMA Selection Results

Using the subset-ARMA search (256 specifications with selected lags allowed), the **BIC** criterion selected the **same model as before—white noise (ARMA(0,0))**, confirming our earlier conclusion that additional dynamics are not warranted once we penalize complexity strongly.

In contrast, **AIC**—which penalizes extra parameters less—prefers a richer specification with **AR3 + MA3 + MA4**.

In short: **BIC ⇒ white noise**, **AIC ⇒ AR3 + MA3 + MA4**.

In [18]:
```
library(stargazer)
model = arima(dgdp, order = c(4,0,4), fixed=c(0, 0, NA, 0, 0, 0, NA, NA, NA))
stargazer(model, type = "text")
```

```
=============================================
                     Dependent variable:
                  ---------------------------
                            dgdp
---------------------------------------------
ar1                          0.000


ar2                          0.000


ar3                        0.929***
                            (0.037)

ar4                          0.000


ma1                          0.000


ma2                          0.000


ma3                        -0.939***
                            (0.031)

ma4                         0.107***
                            (0.022)

intercept                    0.008
                            (0.002)

---------------------------------------------
Observations                  256
Log Likelihood              762.677
sigma2                      0.0001
Akaike Inf. Crit.         -1,515.354
=============================================
Note:             *p<0.1; **p<0.05; ***p<0.01
```

ARIMAX / ADL Models

ARIMAX: Autoregressive Moving Average Model with Explanatory Variable

$$y_t = a_0 + \sum_{i=1}^{p} a_i y_{t-i} + \sum_{i=1}^{q} b_i \epsilon_{t-i} + \sum_{j=1}^{r} c_j z_{t-j} + \epsilon_t$$

where y_t is the dependent variable, y_{t-i} are its lagged values, ϵ_{t-i} are lagged error terms, z_{t-j} are exogenous regressors with their lags, a_i, b_i, and c_j are coefficients to be estimated, and ϵ_t is the white noise error term.

Estimation

The ARIMAX model cannot be estimated using OLS. Instead, one needs to use Maximum Likelihood (ML) estimation.

ADL: Autoregressive Distributed Lag Model

$$y_t = a_0 + \sum_{i=1}^{p} a_i y_{t-i} + \sum_{j=1}^{r} c_j z_{t-j} + \epsilon_t$$

where y_t is the dependent variable, y_{t-i} are its lagged values, z_{t-j} are exogenous regressors with their lags, a_i and c_j are coefficients to be estimated, and ϵ_t is the white noise error term.

Examples

1. $y_t = a_0 + a_1 y_{t-1} + \mathbf{c_0 z_t} + \epsilon_t$
2. $y_t = a_0 + a_1 y_{t-1} + \mathbf{c_0 z_t} + \mathbf{c_1 z_{t-1}} + \epsilon_t$
3. $y_t = a_0 + a_1 y_{t-1} + \mathbf{c_1 z_{t-1}} + \epsilon_t$

In an ADL model, if you include a contemporaneous regressor (say, z_t on the right-hand side of y_t), then:

In theory, z_t might be endogenous, because z_t and y_t could be determined simultaneously, or because both respond to the same shock in the error term ϵ_t. This means the OLS estimate of the coefficient on z_t could be biased and inconsistent.

That's why the classic ADL setup usually emphasizes lagged values of regressors ($z_{t-1}, z_{t-2}, \ldots$). Using lags reduces the risk of contemporaneous feedback and makes it more plausible that the regressors are predetermined relative to y_t.

Contemporaneous Terms in ADL Models

When specifying an ADL model, it may be tempting to include the contemporaneous value of a regressor (e.g. x_t on the right-hand side of y_t).

Important rule:

- In general, we should **not include contemporaneous regressors** in an ADL model.
- The reason is that z_t may be **endogenous** — it might move together with y_t because of common shocks or simultaneous determination. In this case, the error term ϵ_t is correlated with z_t, and OLS estimates will be biased and inconsistent.

If we believe the contemporaneous regressor is **strictly exogenous**, we must **argue clearly why**. For example:

- If z_t is a policy variable set independently of current GDP shocks (e.g., a predetermined tax rule).
- If z_t reflects external conditions (e.g., world oil prices), where feedback from Canadian GDP is negligible.

In such cases, we can justify including z_t, but the exogeneity assumption must be stated explicitly.

Estimation

The ADL model can be estimated using OLS.

Exercise 1

Download the data on hours worked and, following Steps 1–5 listed above, transform the series into a stationary dataset.

Solution to Exercise 1

In [8]:
```r
# Read the CSV file "HOHWMN02CAM065N.csv" into a data frame called pce
hours = read.csv("HOHWMN02CAM065N.csv")

# Display the first 6 rows of the dataset to check its structure
head(hours)

# Display the last 6 rows of the dataset to see the most recent observations
tail(hours)

# Convert the second column of pce into a time series object
# frequency = 4 means quarterly data
# start = c(1960, 1) sets the beginning of the series in Q1 1960
tshours = ts(hours[, 2], frequency = 4, start = c(1960, 1))
```

A data.frame: 6 × 2

	observation_date	HOHWMN02CAM065N
	<chr>	<dbl>
1	1960-01-01	40.5
2	1960-04-01	40.3
3	1960-07-01	40.7
4	1960-10-01	40.0
5	1961-01-01	40.3
6	1961-04-01	40.7

A data.frame: 6 × 2

	observation_date	HOHWMN02CAM065N
	<chr>	<dbl>
250	2022-04-01	36.8
251	2022-07-01	37.0
252	2022-10-01	36.9
253	2023-01-01	37.3
254	2023-04-01	37.5
255	2023-07-01	37.6

In [9]:
```r
# Calculate the number of observations in the time series object
nd = length(tshours)

# Print a message along with the number of observations
cat("There are", nd, "observations in our data set.")
```

```
There are 255 observations in our data set.
```

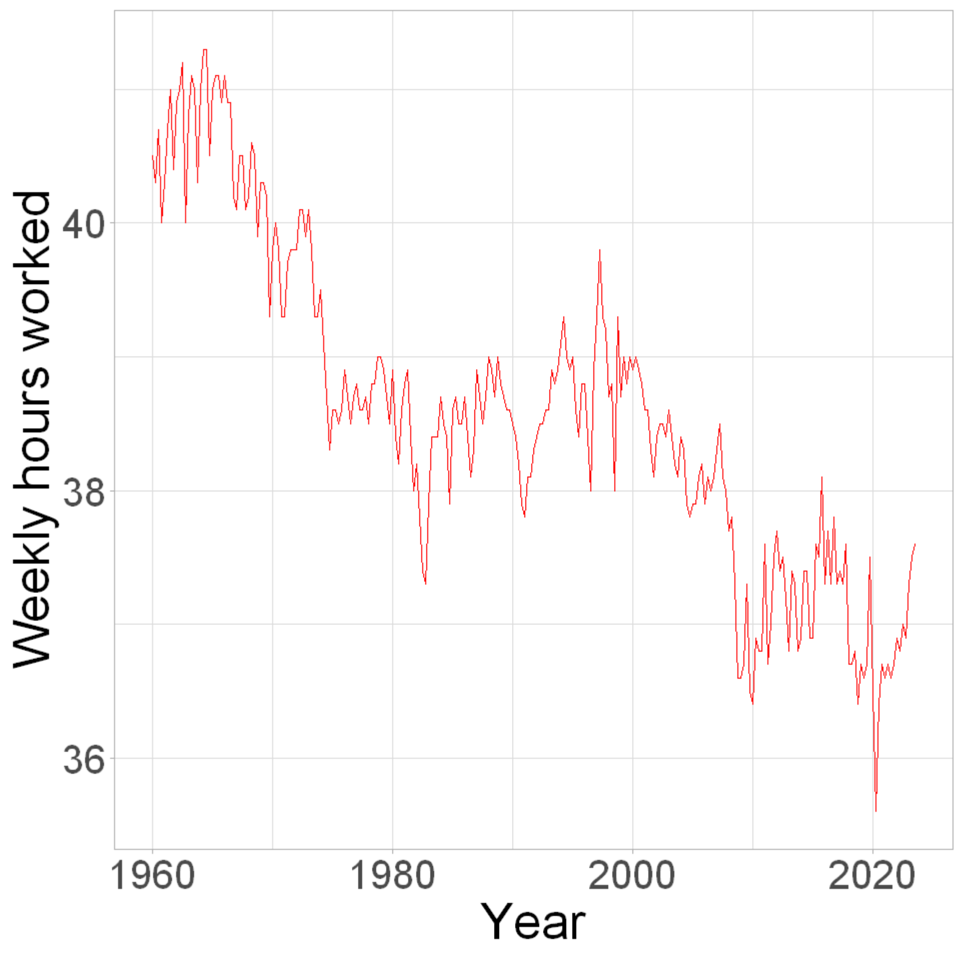
In [19]:
```r
# Load required libraries
library(ggplot2)   # for plotting
library(ggfortify) # for autoplot of time series objects

# Set default figure size (width = 8, height = 8)
options(repr.plot.width = 8, repr.plot.height = 8)
```

```
# Create an initial time series plot of tspce in red
fig = autoplot(tshours, colour = 'red')

# Customize the plot
fig = fig +
  theme(aspect.ratio = 1) +                          # make plot square-shaped
  theme_light() +                                    # use a light theme
  theme(aspect.ratio = 1) +                          # reinforce square aspect ratio
  theme(plot.margin = ggplot2::margin(0.2, 0.2, 0.2, 0.2, "cm")) + # adjust margins
  theme(text = element_text(size = 30)) +            # set large font size for readability
  labs(x = "Year") +                                 # label x-axis
  labs(y = "Weekly hours worked")                    # label y-axis

# Display the plot
fig
```



In [11]:

```
library(urca)
summary(ur.df(tshours, type='trend', lags=8, selectlags="AIC"))
```

```
###############################################
# 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
-0.90865 -0.18990  0.02053  0.20121  1.04538

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.3057929  1.6595773   2.595  0.01007 *
z.lag.1     -0.1076452  0.0409813  -2.627  0.00919 **
tt          -0.0013600  0.0006615  -2.056  0.04090 *
z.diff.lag1 -0.1862935  0.0711183  -2.619  0.00938 **
z.diff.lag2 -0.2070523  0.0708395  -2.923  0.00381 **
z.diff.lag3 -0.1015793  0.0719109  -1.413  0.15910
z.diff.lag4  0.1818924  0.0707114   2.572  0.01072 *
z.diff.lag5 -0.0809457  0.0716118  -1.130  0.25949
z.diff.lag6 -0.0050747  0.0702730  -0.072  0.94249
z.diff.lag7 -0.1039312  0.0670651  -1.550  0.12256
z.diff.lag8  0.1031624  0.0643336   1.604  0.11016
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3196 on 235 degrees of freedom
Multiple R-squared:  0.2636,	Adjusted R-squared:  0.2323
F-statistic: 8.414 on 10 and 235 DF,  p-value: 1.111e-11


Value of test-statistic is: -2.6267 2.9474 3.7048

Critical values for test statistics:
      1pct  5pct 10pct
tau3 -3.98 -3.42 -3.13
phi2  6.15  4.71  4.05
phi3  8.34  6.30  5.36
```

Interpretation of Unit Root Test

The reported test statistics are:

- **tau3 = -2.6267**
- **phi2 = 2.9474**
- **phi3 = 3.7048**

The corresponding critical values are:

- **tau3:** -3.98 (1%), -3.42 (5%), -3.13 (10%)
- **phi2:** 6.15 (1%), 4.71 (5%), 4.05 (10%)
- **phi3:** 8.34 (1%), 6.30 (5%), 5.36 (10%)

The **phi2 (2.9474)** and **phi3 (3.7048)** statistics are **below their respective critical values**, so we **cannot reject the null hypothesis** that the constant is zero or that the trend is zero.

Conclusion

We cannot reject that the constant and trend are zero. Therefore, the test should be **re-run with a simpler specification** (without constant and trend) to properly test for stationarity.

In [12]:
```
library(urca)
summary(ur.df(tshours, type='none', lags=8, selectlags="AIC"))
```

```
###############################################
# 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
-0.8648 -0.2042  0.0012  0.2149  1.0881

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
z.lag.1     -0.0007340  0.0005426  -1.353  0.17739
z.diff.lag1 -0.2795002  0.0637327  -4.386 1.74e-05 ***
z.diff.lag2 -0.2802193  0.0663361  -4.224 3.42e-05 ***
z.diff.lag3 -0.1726109  0.0681863  -2.531  0.01200 *
z.diff.lag4  0.1432734  0.0681844   2.101  0.03667 *
z.diff.lag5 -0.1531833  0.0678833  -2.257  0.02494 *
z.diff.lag6 -0.0759378  0.0659625  -1.151  0.25079
z.diff.lag7 -0.1660033  0.0633163  -2.622  0.00931 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3235 on 238 degrees of freedom
Multiple R-squared:  0.2373,    Adjusted R-squared:  0.2116
F-statistic: 9.254 on 8 and 238 DF,  p-value: 4.301e-11


Value of test-statistic is: -1.3528

Critical values for test statistics:
      1pct  5pct 10pct
tau1 -2.58 -1.95 -1.62
```

Interpretation of Unit Root Test

The reported test statistic is:

- **tau1 = -1.3528**

The corresponding critical values are:

- **1%**: -2.58
- **5%**: -1.95
- **10%**: -1.62

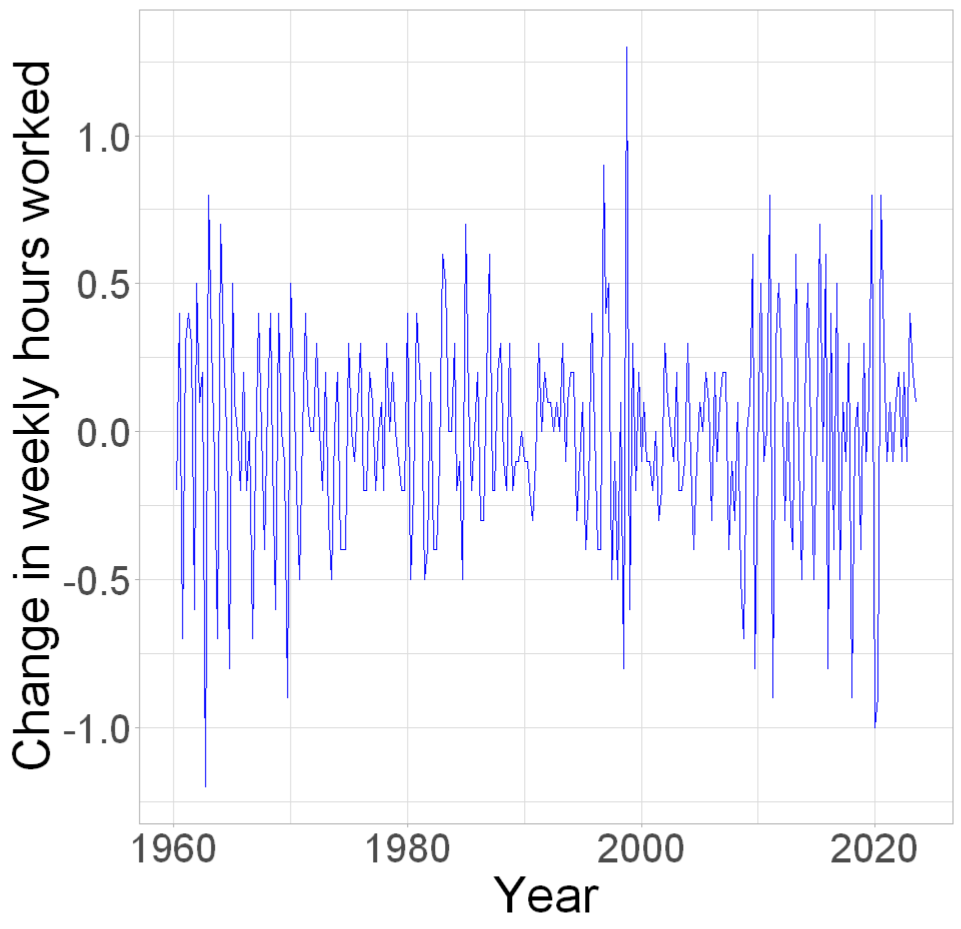
Since **-1.3528 > -1.62**, the test statistic is above even the 10% critical value. Therefore, we **cannot reject the null hypothesis of a unit root** at any conventional significance level.

Conclusion

The *Hours Worked* series is **non-stationary** in levels and must be **transformed** (e.g., first differenced) before proceeding with ADL estimation.

In [13]:
```
dhours = diff(tshours)

options(repr.plot.width=8, repr.plot.height=8)
fig = autoplot(dhours, colour = 'blue')
fig = fig + theme(aspect.ratio=1) +
theme_light() +
theme(aspect.ratio=1) +
theme(plot.margin = ggplot2::margin(0.2, 0.2, 0.2, 0.2, "cm")) +
theme(text=element_text(size=30)) +
labs(x = "Year") +
labs(y = "Change in weekly hours worked")
fig
```



In [14]:
```
library(urca)
summary(ur.df(dhours, type='none', lags=8, selectlags="AIC"))
```

```
###############################################
# 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 
-0.89333 -0.23352 -0.02465  0.18460  1.06413 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
z.lag.1     -1.93500    0.25016  -7.735 2.92e-13 ***
z.diff.lag1  0.65919    0.23021   2.863  0.00457 ** 
z.diff.lag2  0.38705    0.20613   1.878  0.06164 .  
z.diff.lag3  0.22176    0.18079   1.227  0.22118    
z.diff.lag4  0.37383    0.14245   2.624  0.00924 ** 
z.diff.lag5  0.22801    0.10322   2.209  0.02814 *  
z.diff.lag6  0.16099    0.06345   2.537  0.01181 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3246 on 238 degrees of freedom
Multiple R-squared:  0.6878,	Adjusted R-squared:  0.6786 
F-statistic: 74.89 on 7 and 238 DF,  p-value: < 2.2e-16


Value of test-statistic is: -7.7351 

Critical values for test statistics: 
      1pct  5pct 10pct
tau1 -2.58 -1.95 -1.62
```

Interpretation of Unit Root Test

The reported test statistic is:

- **tau1 = -7.7351**

The corresponding critical values are:

- **1%**: -2.58
- **5%**: -1.95
- **10%**: -1.62

Since **-7.7351 < -2.58**, the test statistic is far below the 1% critical value. We can therefore **reject the null hypothesis of a unit root** at the 1% significance level.

Conclusion

By retesting the *change in hours worked*, we find that the differenced series is **stationary**. This means there is **no second unit root**, and the series is suitable for use in ADL estimation.

ADL Step 1:

Create a joint time series object

In [15]:
```
# Create a joint time series object that combines GDP growth (dgdp)
# and changes in hours worked (dhours), keeping only observations
# where both series are available
data = ts.intersect(dgdp, dhours)

# Display the first few observations of the combined dataset
head(data)

# Display the last few observations of the combined dataset
tail(data)
```

A Time Series: 6 × 2

	dgdp	dhours
1961 Q2	0.025005468	0.4
1961 Q3	0.024042802	0.3
1961 Q4	0.013513087	-0.6
1962 Q1	0.026262130	0.5
1962 Q2	0.007915743	0.1
1962 Q3	0.012708511	0.2

A Time Series: 6 × 2

	dgdp	dhours
2022 Q2	0.010229149	-0.1
2022 Q3	0.005138201	0.2
2022 Q4	-0.001370987	-0.1
2023 Q1	0.013454326	0.4
2023 Q2	0.001425423	0.2
2023 Q3	-0.002258095	0.1

ADL Step 2:

Find best-fitting ADL model

In [20]:
```
##############################################
## Exhaustive ADL Subset Search with AIC/BIC
##############################################

## ---- 0) Libraries ---------------------------------------------------------
## 'gets' provides the arx() function for regression with optional diagnostics.
library(gets)

## ---- 1) Create lagged variables -----------------------------------------
## Assume 'data' is a bivariate ts object:
##   - column 1: y (e.g., GDP growth)
##   - column 2: x (e.g., ΔHours worked)
## We build up to 4 lags for each series.
```

```r
y0 = data[, 1]                # y_t (dependent variable at time t)

y1 = lag(data[, 1], -1)       # y_{t-1}
y2 = lag(data[, 1], -2)       # y_{t-2}
y3 = lag(data[, 1], -3)       # y_{t-3}
y4 = lag(data[, 1], -4)       # y_{t-4}

x1 = lag(data[, 2], -1)       # x_{t-1}
x2 = lag(data[, 2], -2)       # x_{t-2}
x3 = lag(data[, 2], -3)       # x_{t-3}
x4 = lag(data[, 2], -4)       # x_{t-4}

## ---- 2) Align sample (common time span) -----------------------------------
## ts.intersect keeps only the time points where all included series are observed.
## This ensures comparable samples across all model specifications.
tempdata = ts.intersect(
  y0,
  y1, y2, y3, y4,
  x1, x2, x3, x4
)

## ---- 3) Define dependent and regressor matrices ---------------------------
## depvar: y_t aligned to the intersection sample.
## indvar: matrix of candidate regressors (lags of y and x).
depvar = tempdata[, 1]
indvar = tempdata[, 2:ncol(tempdata)]
## Give column names to aid interpretation
colnames(indvar) = c("y1","y2","y3","y4","x1","x2","x3","x4")

## ---- 4) Initialize containers for results --------------------------------
## We'll accumulate rows (one per model) in a list, then rbind at the end.
res_list = list()

iter = 1
model = arx(depvar, mc = TRUE)
res_list[[iter]] = data.frame(
      regressors = paste(""),
      AIC        = AIC(model),
      BIC        = BIC(model)
    )

## ---- 5) Define the model space -------------------------------------------
## We will try ALL non-empty subsets of the candidate regressors:
## sizes i = 1, 2, ..., kmax.
## WARNING: This is 2^kmax - 1 models (grows quickly with kmax).
kmax = ncol(indvar)   # total number of available regressors
k = 1:kmax            # index set for regressors

## ---- 6) Exhaustive subset loop -------------------------------------------
## Outer loop over subset size i
for (i in 1:kmax) {

  ## All combinations (columns) of size i from the set {1, ..., kmax}.
  ## combn returns a matrix with i rows and choose(kmax, i) columns.
  ksets = combn(k, i)

  ## Inner loop over each specific combination of regressors of size i
  for (j in 1:ncol(ksets)) {
    iter = iter + 1

    ## Indices of regressors to include in this model
    regressors = ksets[, j]

    ## Fit the ADL variant via arx():
    ## - depvar is the dependent variable (y_t)
    ## - mxreg supplies *exogenous* regressors (here: lags of y and x treated as "xreg")
    ## - mc = TRUE includes an intercept
    model = try(suppressWarnings(arx(depvar, mxreg = indvar[, regressors], mc = TRUE)), silent = TRUE)

    # If estimation fails
    # skip this specification and continue to the next pattern.
    if (inherits(model, "try-error")) next

    ## ---- 7) Store information criteria and regressor labels ----------------
    res_list[[iter]] = data.frame(
      regressors = paste(colnames(indvar)[regressors], collapse = " + "),
      AIC        = AIC(model),
      BIC        = BIC(model)
    )
  }
}

## ---- 8) Bind results into a single data frame -----------------------------
results = do.call(rbind, res_list)

## Quick look at the first and last few rows of the results table
head(results)
tail(results)

## ---- 9) Identify the best models by AIC and BIC ---------------------------
best_by_AIC = results[which.min(results$AIC), ]
best_by_BIC = results[which.min(results$BIC), ]
```

```
## Display the best specs under each criterion
best_by_AIC
best_by_BIC
```

A data.frame: 6 × 3

	regressors	AIC	BIC
	<chr>	<dbl>	<dbl>
1		-1452.622	-1449.116
2	y1	-1450.646	-1443.636
3	y2	-1450.769	-1443.758
4	y3	-1450.720	-1443.710
5	y4	-1452.254	-1445.243
6	x1	-1456.509	-1449.498

A data.frame: 6 × 3

	regressors	AIC	BIC
	<chr>	<dbl>	<dbl>
251	y1 + y2 + y3 + y4 + x2 + x3 + x4	-1445.081	-1417.039
252	y1 + y2 + y3 + x1 + x2 + x3 + x4	-1448.783	-1420.740
253	y1 + y2 + y4 + x1 + x2 + x3 + x4	-1449.866	-1421.823
254	y1 + y3 + y4 + x1 + x2 + x3 + x4	-1449.675	-1421.633
255	y2 + y3 + y4 + x1 + x2 + x3 + x4	-1449.852	-1421.810
256	y1 + y2 + y3 + y4 + x1 + x2 + x3 + x4	-1447.871	-1416.323

A data.frame: 1 × 3

	regressors	AIC	BIC
	<chr>	<dbl>	<dbl>
33	x1 + x3	-1456.901	-1446.385

A data.frame: 1 × 3

	regressors	AIC	BIC
	<chr>	<dbl>	<dbl>
6	x1	-1456.509	-1449.498

Interpretation of ADL Model Selection Loop

In this loop exercise, we first defined a **candidate set of regressors** (lags of GDP growth and lags of hours worked) and then **systematically examined every non-empty subset** of those candidates. For each subset, we estimated an ADL-style regression with a constant (no contemporaneous regressors to avoid endogeneity) and **recorded AIC and BIC**.

We then **ranked all models** by each criterion to evaluate the trade-off between fit and parsimony, and identified the **best specification under AIC and under BIC**.

In our results, **AIC** favored a richer hours-lag structure (lags 1 and 3), while **BIC** preferred a more parsimonious specification (lag 1 only).

The key takeaway is that we **explicitly searched** the space of plausible ADL regressors, **documented the search limits** (which lags were allowed), and **justified the final model choice** using transparent, reproducible criteria.

ADL Step 3:

FEstimate the ADL model

In [17]:
```
library(dynlm)
library(stargazer)
model.aic = dynlm(dgdp ~ L(dhours, c(1,3)), data)  # dgdp on dhours lags 1 & 3
model.bic = dynlm(dgdp ~ L(dhours, 1), data)       # dgdp on dhours lag 1

# How does the syntax work in general?
# m_ardl13 = dynlm(dgdp ~ L(dgdp, 1) + L(dhours, 1:3), data)          # ARDL(1,3)
# m_ardl31 = dynlm(dgdp ~ L(dgdp, 1:3) + L(dhours, 1), data)          # ARDL(3,1)
# m_ardl22 = dynlm(dgdp ~ L(dgdp, 1:2) + L(dhours, 1:2), data)        # ARDL(2,2)

stargazer(model.aic, model.bic, type = "text")     # print both models
```

```
===============================================================
                                Dependent variable:
                    -------------------------------------------
                                       dgdp
                         (1)                   (2)
---------------------------------------------------------------
L(dhours, c(1, 3))1        0.006**
                          (0.002)

L(dhours, c(1, 3))3        0.004
                          (0.002)

L(dhours, 1)                                  0.005**
                                             (0.002)

Constant                  0.008***            0.008***
                          (0.001)             (0.001)

---------------------------------------------------------------
Observations                247                  249
R2                         0.031                0.022
Adjusted R2                0.023                0.018
Residual Std. Error   0.012 (df = 244)      0.013 (df = 247)
F Statistic         3.893** (df = 2; 244) 5.617** (df = 1; 247)
===============================================================
Note:                               *p<0.1; **p<0.05; ***p<0.01
```

Interpretation of an ADL Model Regression Output

Because the left-hand side is $\Delta \log(\mathrm{GDP}_t)$, the coefficients measure effects on the **quarterly GDP growth rate** (in **percentage points** when multiplied by 100).

- **Intercept ≈ 0.008**
  → Average quarterly GDP growth is about **0.8%**.

Model (1)

- $L(\Delta \text{hours}, 1) = 0.006^{**}$
  → A **+1 hour** increase in weekly hours **from** $t-2$ **to** $t-1$ is **associated with +0.6 pp** higher GDP growth in quarter t.
- $L(\Delta \text{hours}, 3) = 0.004$
  → **Not statistically significant** — do not interpret as an effect.

Model (2)

- $L(\Delta \text{hours}, 1) = 0.005^{**}$
  → A **+1 hour** increase **from** $t-2$ **to** $t-1$ is **associated with +0.5 pp** higher GDP growth in quarter t.

%%shell

jupyter nbconvert --to html ///content/lecture_4.ipynb