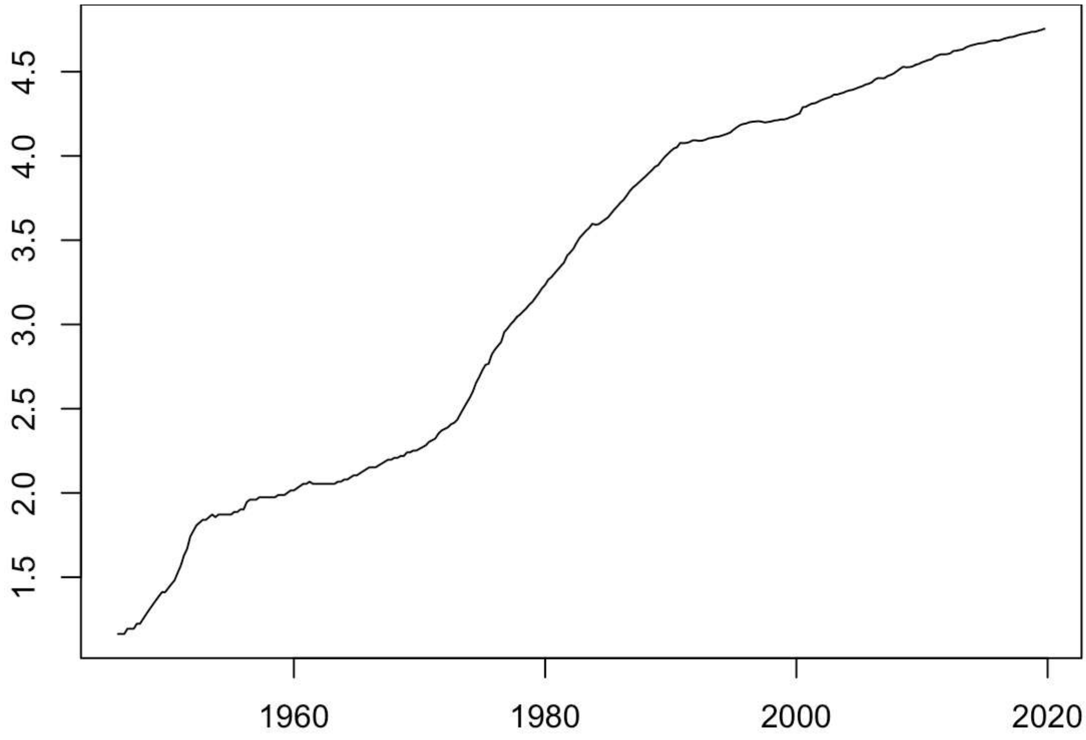
Tutorial 6 Answers

1. Reviewing the unit root test on log CPI:

```
dt <- read.csv("CPI.csv")
Y <- ts(log(dt$CPI), start=c(1922,2), end=c(2025,2), frequency=4)
Y <- window(Y, start=c(1946,1), end=c(2019,4))
```

```
plot(Y)
```

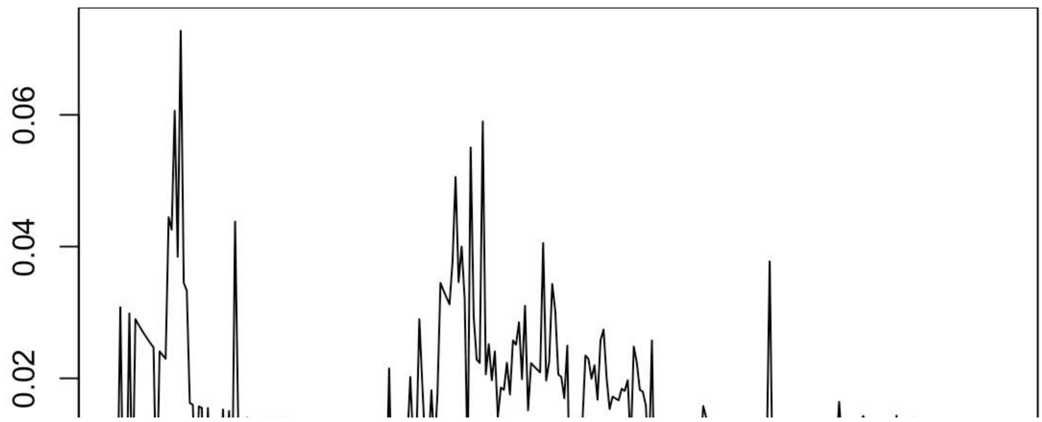


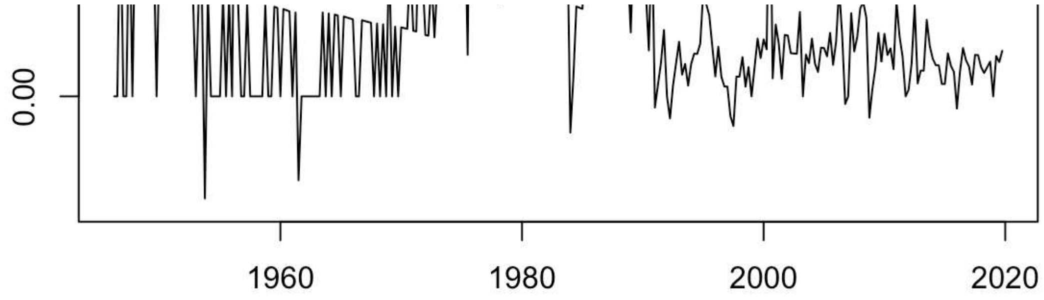

```
library(urca)
adf <- ur.df(Y, type="trend", selectlags="AIC")
p_adf <- punitroot(adf@teststat[1], trend="ct", statistic="t")
```

```
Since p_adf = 0.997 > 0.05, H0:unit root is not rejected.
```

a. Differencing the log of CPI gives inflation:

```
DY <- diff(Y)
plot(DY)
```





There is no evidence of a linear trend in this time series. There are some extended periods (early 1950's and 1970-80's) where inflation was higher, but these are not indicative of an overall upwards or downwards trend.

b. Carrying out the unit root test on inflation:

```
adf <- ur.df(DY, type="drift", selectlags="AIC")
p_adf <- punitroot(adf@teststat[1], trend="c", statistic="t")
```

```
Since p_adf = 0.0000 < 0.05, H0:unit root is rejected.
```

c. Code for model search:

```
library(forecast)
pmax <- 4
qmax <- 4
LBp <- matrix(nrow=pmax+1, ncol=qmax+1)
rownames(LBp) <- paste0("p=", 0:pmax)
colnames(LBp) <- paste0("q=", 0:qmax)
AICc <- LBp
for (p in 0:pmax){
  for (q in 0:qmax){
    eq <- Arima(DY, order=c(p,0,q))
    capture.output(LBp[p+1,q+1] <- checkresiduals(eq,
      plot=FALSE)$p.value)
    AICc[p+1,q+1] <- eq$aicc
  }
}
```

Residual autocorrelation significant:

```
print(1*(LBp<0.05))
```

```
    q=0 q=1 q=2 q=3 q=4
p=0   1   1   1   1   1
p=1   1   1   0   0   0
p=2   1   1   0   0   0
p=3   1   1   0   0   0
p=4   1   1   0   0   0
```

Model to minimise AICc:

```
print(AICc==min(AICc))
```

```
     q=0   q=1   q=2   q=3   q=4
p=0 FALSE FALSE FALSE FALSE FALSE
p=1 FALSE FALSE FALSE  TRUE FALSE
p=2 FALSE FALSE FALSE FALSE FALSE
p=3 FALSE FALSE FALSE FALSE FALSE
p=4 FALSE FALSE FALSE FALSE FALSE
```

The chosen model is ARMA(1,3) for ΔY_t, or ARIMA(1,1,3) for Y_t.

d. Firstly here is the one-step-ahead forecast for inflation, which is `DY`, i.e. $\Delta \log \mathrm{CPI}_t$.

```
eq <- Arima(DY, order=c(1,0,3))
ForecastInflation_2020q1 <- forecast(eq, h=1)$mean
print(ForecastInflation_2020q1)
```

```
           Qtr1
2020 0.00710347
```

Now let's calculate the forecast for the CPI. A simple way is to apply the inflation forecast for 2020q1 to the observed value of the CPI for 2019q4:

```
# Create a ts of CPI (i.e. not log CPI)
CPI <- ts(dt$CPI, start=c(1922,2), end=c(2025,2), frequency=4)

# CPI in 2019q4, last date of the estimation sample
CPI_2019q4 <- CPI[which(time(CPI)==2019.75)]

# Forecast CPI for 2020q1 is CPI in 2019q4 increased by
# forecast inflation for 2020q1
ForecastCPI_2020q1 <- CPI_2019q4*(1+ForecastInflation_2020q1)
print(ForecastCPI_2020q1)
```

```
         Qtr1
2020 117.0254
```

A different way which will give a very similar answer is to add the forecast for ΔY_{n+1} to the actual value for Y_n to produce a forecast for Y_{n+1}. Noting that $Y_{n+1} = \log \mathrm{CPI}_{n+1}$, we can take the *exponential* of the forecast for Y_{n+1} to be the forecast for CPI_{n+1}.

```
n <- length(Y)
ForecastY_2020q1 <- Y[n]+ForecastInflation_2020q1
print(ForecastY_2020q1)
```

```
         Qtr1
2020 4.762416
```

```
ForecastCPI_2020q1 <- exp(ForecastY_2020q1)
print(ForecastCPI_2020q1)
```

```
         Qtr1
2020 117.0284
```

Note this differs in the third decimal place to the previous approach. Neither is literally "correct": both are approximations that, for practical purposes, are essentially equivalent.

Another approach that directly models `Y` is to specify the model

```
eq1 <- Arima(Y, order=c(1,1,3), xreg=(1:n))
```

This model is

$$\begin{aligned} Y_t &= \beta_1 t + Z_t \\ \Delta Z_t &= \phi_1 \Delta Z_{t-1} + U_t + \theta_1 U_{t-1} + \theta_2 U_{t-2} + \theta_3 U_{t-3} \end{aligned}$$

The linear trend in Y_t is captured by the `xreg=(1:n)` option in the estimation, which produces a linear trend variable of the form `(1,2,...,n)`. Previously we have used `time(Y)` as the trend variable, and we could do that here as well, but `(1:n)` is used here to allow direct comparison of coefficient estimates between `eq` and `eq1`. The model corresponding to `eq` above can be expressed

$$\begin{aligned} \Delta Y_t &= \beta_1 + \Delta Z_t \\ \Delta Z_t &= \phi_1 \Delta Z_{t-1} + U_t + \theta_1 U_{t-1} + \theta_2 U_{t-2} + \theta_3 U_{t-3} \end{aligned}$$

i.e. it is essentially the same other than in `eq` we use the first difference series `DY` directly with an ARIMA(1,0,3) specification, while in `eq1` we model `Y` and specify the required difference with the $d = 1$ component of the ARIMA(1,1,3) specification. These are two versions of the same model and give equivalent coefficient estimates:

```
print(eq)
```

```
Series: DY
ARIMA(1,0,3) with non-zero mean

Coefficients:
         ar1      ma1     ma2     ma3    mean
      0.8316  -0.6377  0.2380  0.1276  0.0120
s.e.  0.0501   0.0701  0.0629  0.0679  0.0022

sigma^2 = 8.244e-05:  log likelihood = 970.4
AIC=-1928.81   AICc=-1928.52   BIC=-1906.69
```

```
print(eq1)
```

```
Series: Y
Regression with ARIMA(1,1,3) errors

Coefficients:
         ar1      ma1     ma2     ma3    xreg
      0.8316  -0.6377  0.2380  0.1276  0.0120
s.e.  0.0501   0.0701  0.0629  0.0679  0.0022
```

```
sigma^2 = 8.245e-05:  log likelihood = 970.4
AIC=-1928.81   AICc=-1928.52   BIC=-1906.69
```

The forecast for `Y` from `eq1` is

```
ForecastY_2020q1 <- forecast(eq1, xreg=(n+1))$mean
print(ForecastY_2020q1)
```

```
         Qtr1
2020 4.762416
```

which is the same as the second forecast above.

Obvious question : should we use `eq` or `eq1`? Since they are equivalent representations of the same model it doesn't matter. In a case such as this where inflation is likely the quantity of most interest it may be convenient to use the `eq` form in terms of `DY` and then deduce CPI forecasts from there. If the only interest were in the level variable `Y` then the `eq1` form in terms of `Y` may be more convenient. Ultimately either is fine.

Note the step where we first calculate the forecast for $Y_{n+1} = \log \text{CPI}_{n+1}$ and then take the exponential to convert this into a forecast for CPI_{n+1}. This is very common practice and superficially seems watertight, but it turns out the resulting forecast is not a conditional expectation. In particular we have the forecast for Y_{n+1} being

$$E(Y_{n+1}|\mathscr{Y}_n) = E(\log \text{CPI}_{n+1}|\mathscr{Y}_n)$$

Taking the exponential:

$$\begin{aligned}\exp\left(E(Y_{n+1}|\mathscr{Y}_n)\right) &= \exp E(\log \text{CPI}_{n+1}|\mathscr{Y}_n)\\ &\neq E(\exp \log \text{CPI}_{n+1}|\mathscr{Y}_n)\\ &= E(\text{CPI}_{n+1}|\mathscr{Y}_n)\end{aligned}$$

The $\neq$ step occurs because the nonlinear $\exp(.)$ operation does not pass through the linear expectation operation. If we have (an estimate of) the conditional expectation of $\log \text{CPI}_{n+1}$ we do not strictly speaking obtain the conditional expectation of CPI_{n+1} by taking the exponential. To find $E(\text{CPI}_{n+1}|\mathscr{Y}_n)$ from $E(\log \text{CPI}_{n+1}|\mathscr{Y}_n)$ would require a more complicated calculation that would involve assumptions and/or estimates of the entire distribution of the CPI variable, not just the conditional mean. Luckily, in practice simply taking the exponential tends to work well in providing a good approximation. In fact research suggests it tends to work better in practice than the theoretically correct formula. So we will continue to take exponentials whenever required.

2. Consider a general representation

$$Y_t = \beta_0 + \beta_1 t + Z_t$$

where the specification of Z_t is not important for this argument. The first difference of Y_t is $\Delta Y_t = Y_t - Y_{t-1}$. The general representation for Y_{t-1} is

$$Y_{t-1} = \beta_0 + \beta_1(t-1) + Z_{t-1},$$

and subtracting this from Y_t gives

$$\begin{aligned} Y_t - Y_{t-1} &= \beta_1[t-(t-1)] + Z_t - Z_{t-1} \\ \Delta Y_t &= \beta_1 + \Delta Z_t \end{aligned}$$

So if Y_t has a linear trend then ΔY_t will not. The implication is also that the intercept included in a model for ΔY_t corresponds to the slope of the linear trend in Y_t. In this form β_1 is called the "drift", which explains the use of the term in the R command when including an intercept in a Dickey-Fuller test.