AutoRegressive Integrated Moving Average Models

- White noise, autoregressive (AR), moving average (MA), and ARMA models
- Stationarity, detrending, differencing, and seasonality

Autoregressive (AR) models

- AR models are models in which the values of a variable in one period is related to the its values in previous periods. That means, values of y_t are affected by the values of y_{t-1} in the past.
- For example, the amount of money in your bank account in one month is related to the amount in your account in a previous month.
- AR(p) is an autoregressive model with p lags:

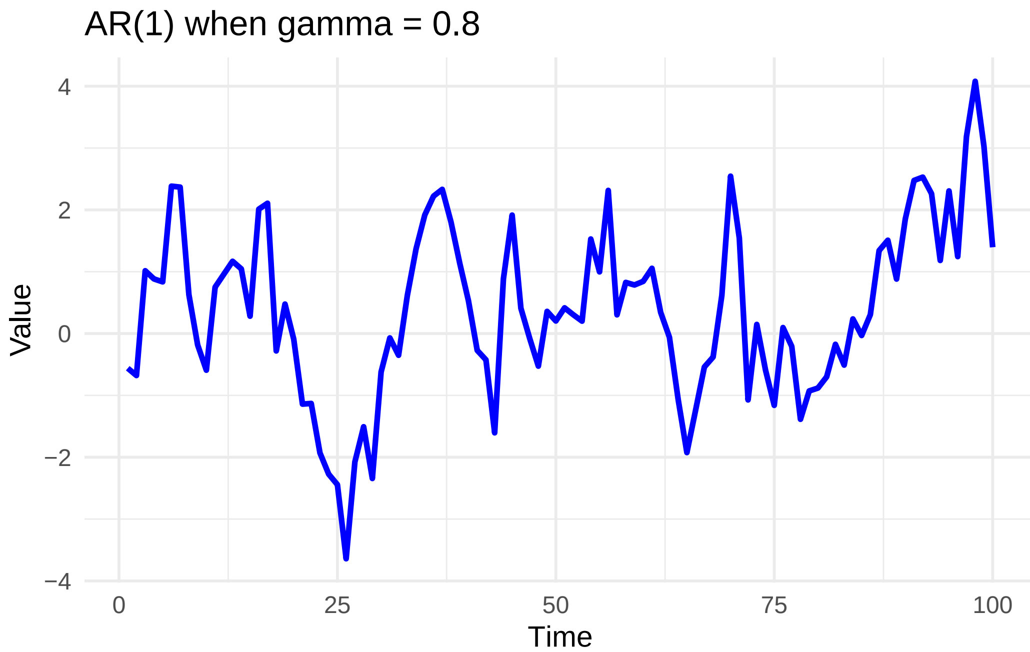
$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \epsilon_t$$

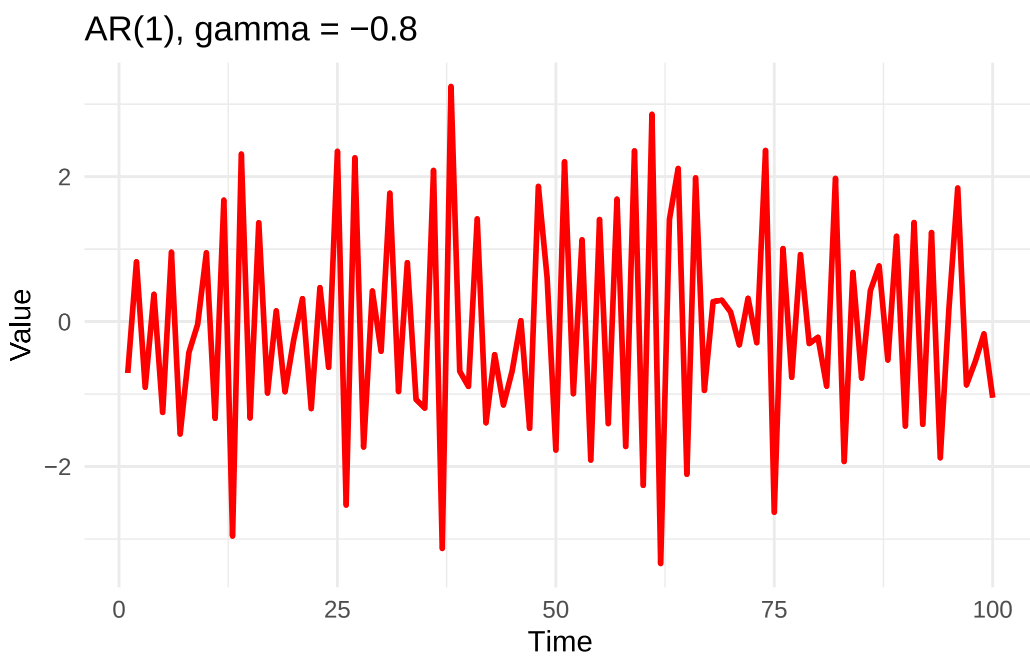
where μ is a constant and γ is the coefficient for the lagged variable in time $t = 1, 2, ..., p$.

- When $p = 1$, we will have AR(1) process and is expressed as:

$$y_t = \mu + \gamma_1 y_{t-1} + \epsilon_t$$

Example plots of AR process:





Moving average (MA) models

- Moving average (MA) models account for the possibility of a relationship between a variable and the residuals from previous periods.
- MA(q) is a moving average model with q-lags:

$$y_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$

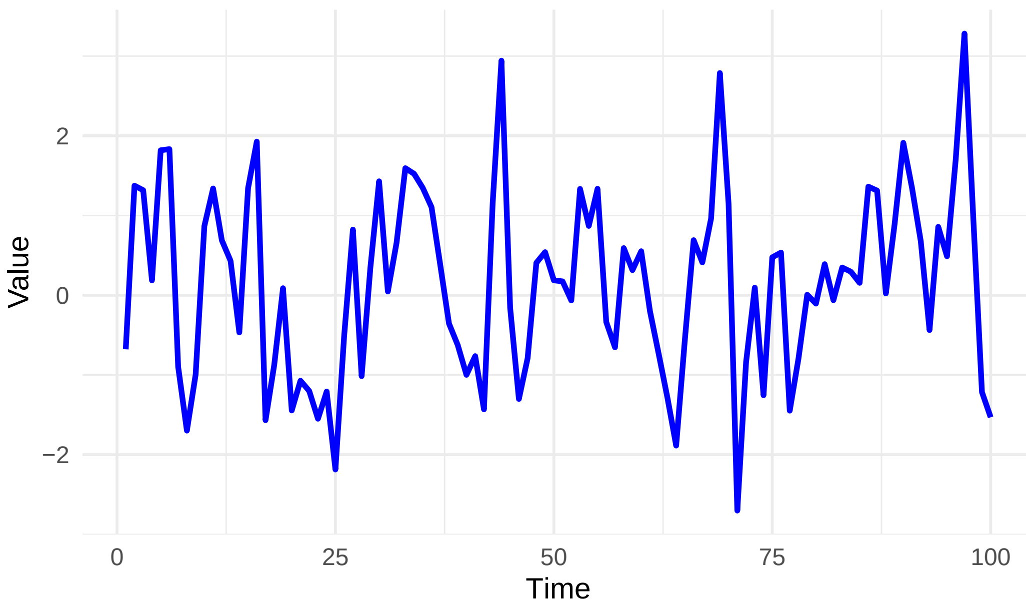
where θ_i is the coefficient for the lagged error term in time $t = 1, 2, ..., q$.

- MA(1) model is expressed as:

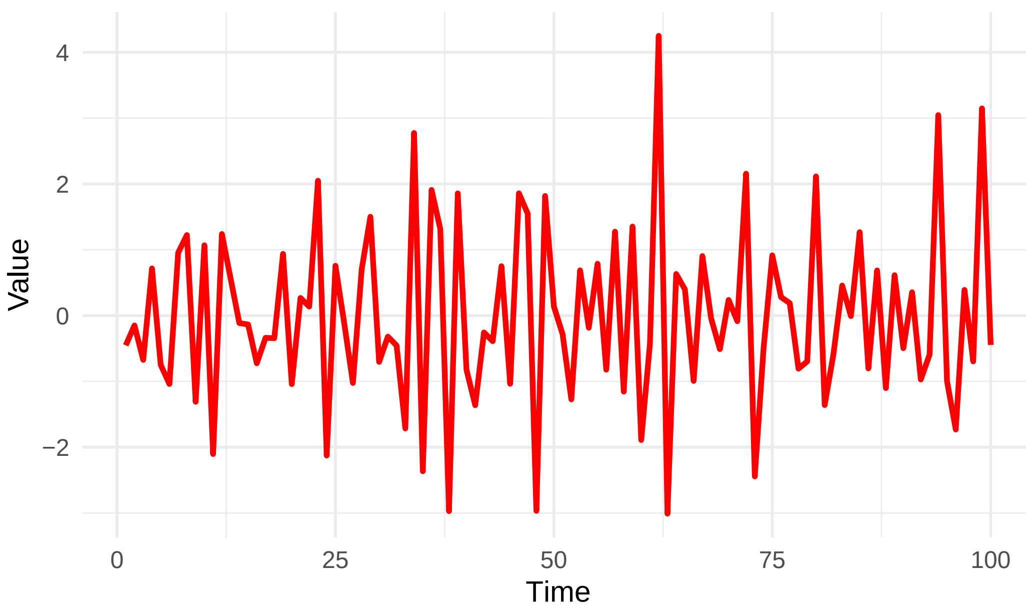
$$y_t = \mu + \epsilon_t + \theta \epsilon_{t-1}$$

- In an MA(1) model, today's value of y depends on today's shock and yesterday's shock. Those shocks (errors) are not observed in your data — you only observe y.

MA(1), theta = 0.8



MA(1), theta = −0.8



Difference between AR and MA

- AR model: uses past y → you observe it
- MA model: uses past errors → you do not observe them

Imagine you want to explain today's temperature using:

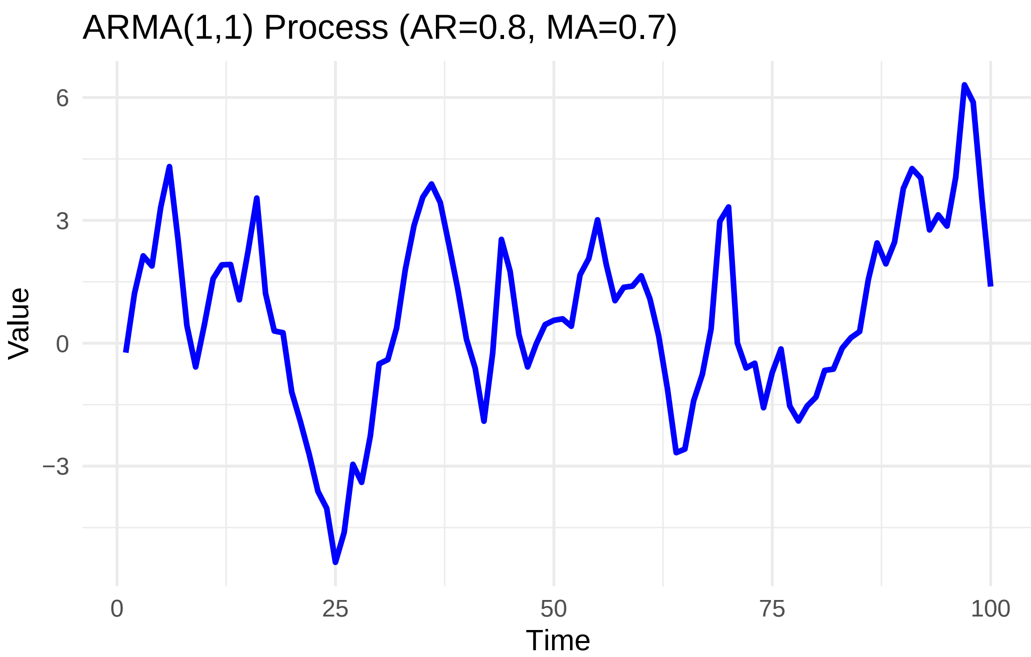
- yesterday's temperature → you have it
- yesterday's measurement mistake → you don't have it
- MA models depend on unobserved shocks, so they must be estimated by methods that infer those shocks (like arima()), not by lm().

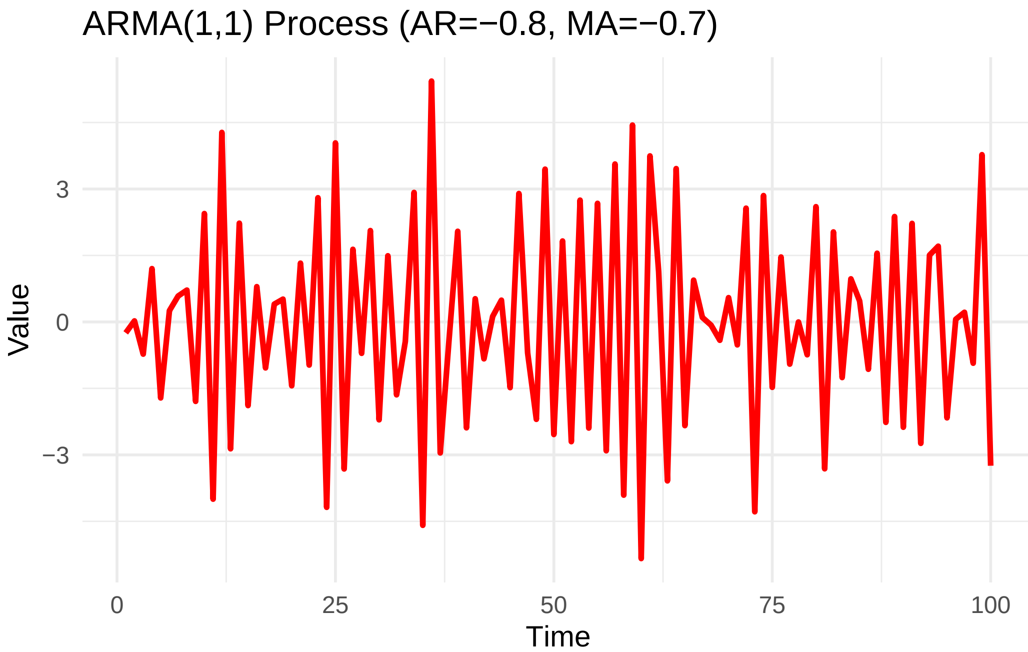
You must guess the mistakes while fitting the model.That's what arima() does.

Autoregressive moving average (ARMA) models

- Autoregressive moving average (ARMA) models combine both p autoregressive terms and q moving average terms, also called ARMA(p,q).

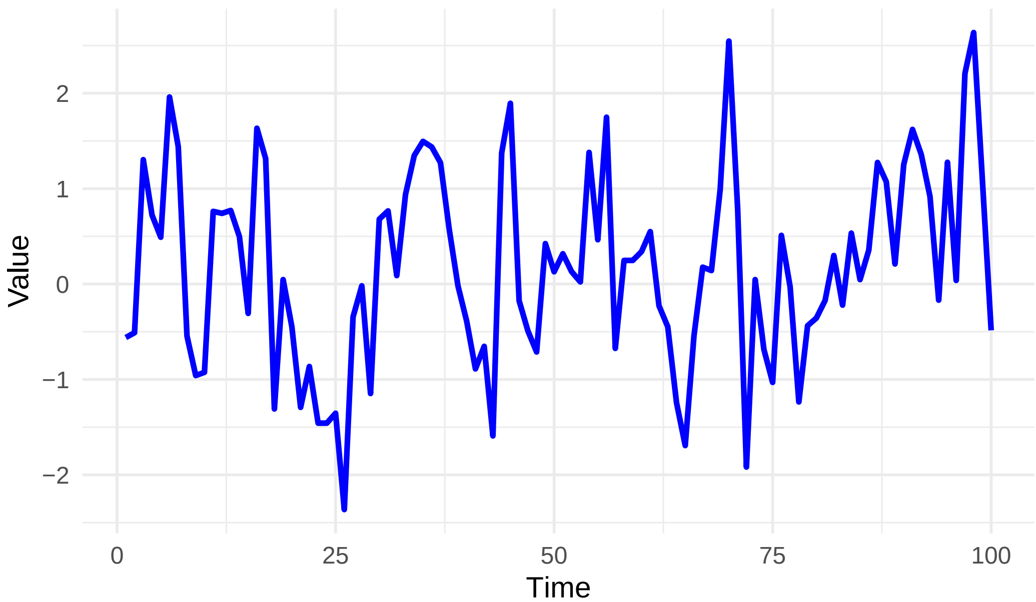
$$y_t = \mu + \sum_{i=1}^{p} \gamma_i y_{t-i} + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$$



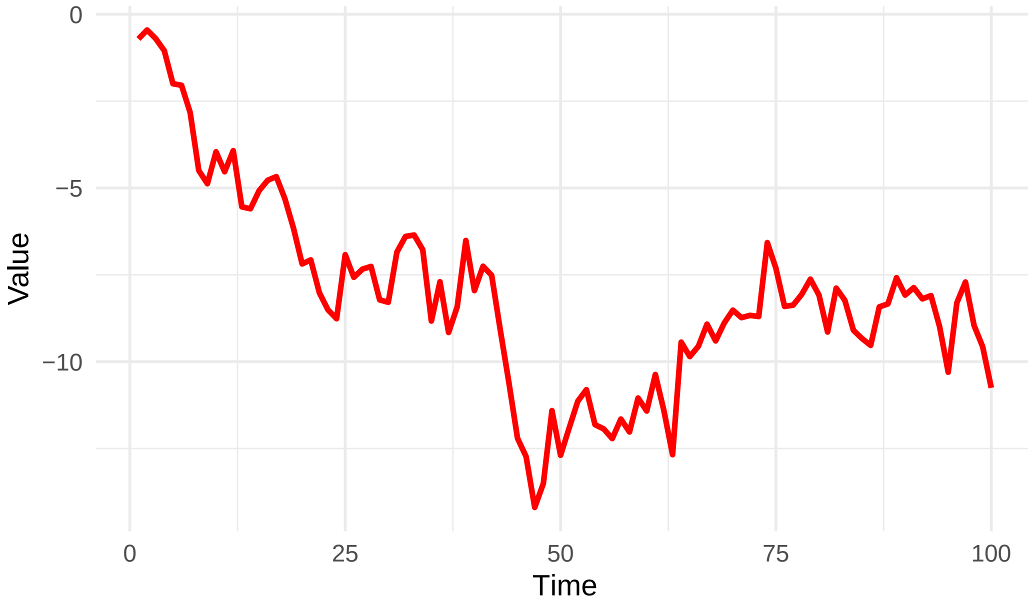

ARMA(1,1) Process (AR=−0.8, MA=−0.7)
3
0
−3
Value
0
25
50
75
100
Time

Fitting ARIMA Model

- Modeling an ARMA(p,q) process requires stationary time series/process.
- A stationary process has a mean and variance that do not change over time and the process does not have trends.
- Is the figure below stationary or nonstationary?



- What about the figure below, is it stationary or nonstationary?

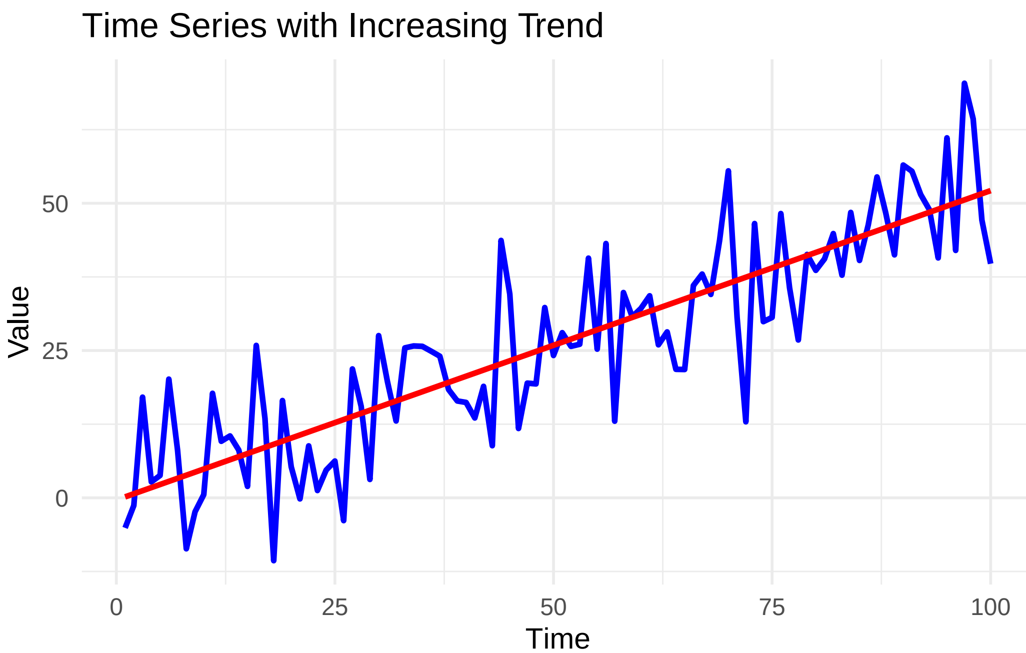


Ways of converting nonstationary series to stationary series

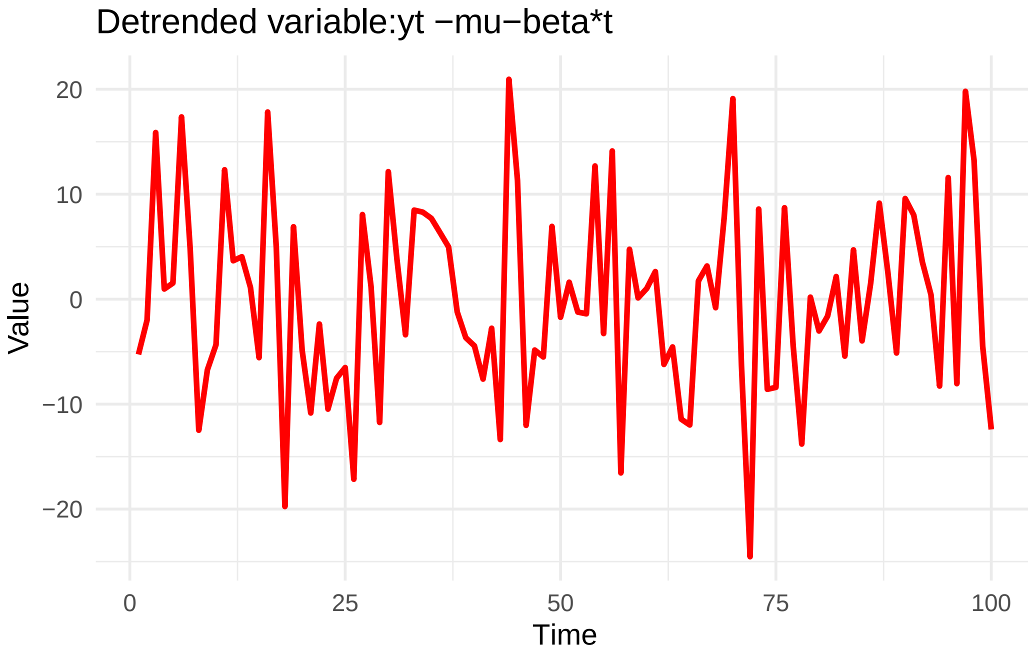
Detrending

- A variable can be detrended by regressing the variable on a time trend and obtaining the residuals.

$$y_t = \mu + \beta t + \epsilon_t$$

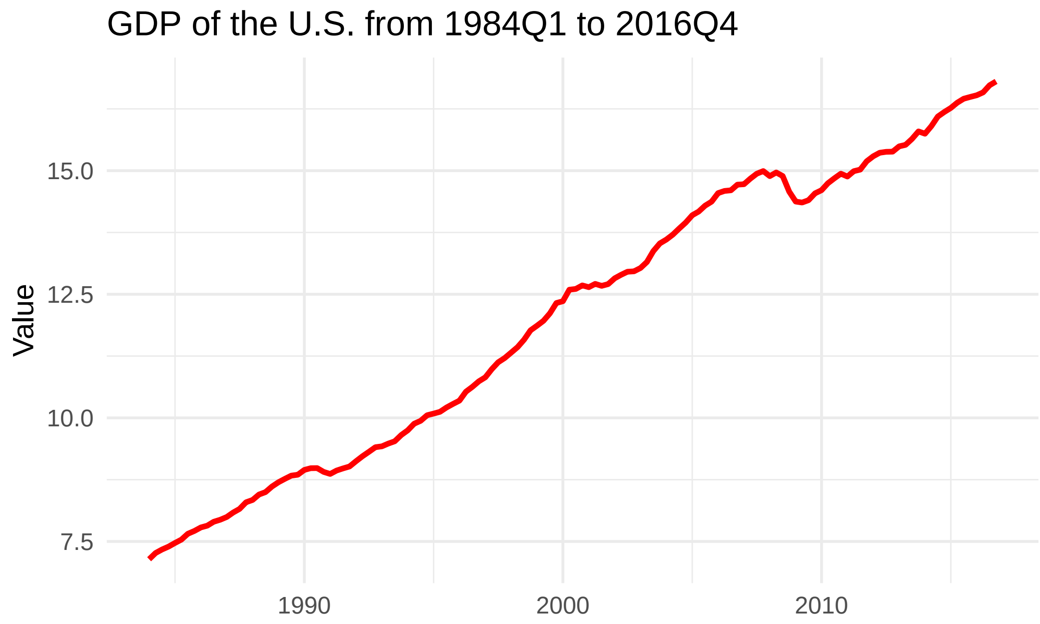


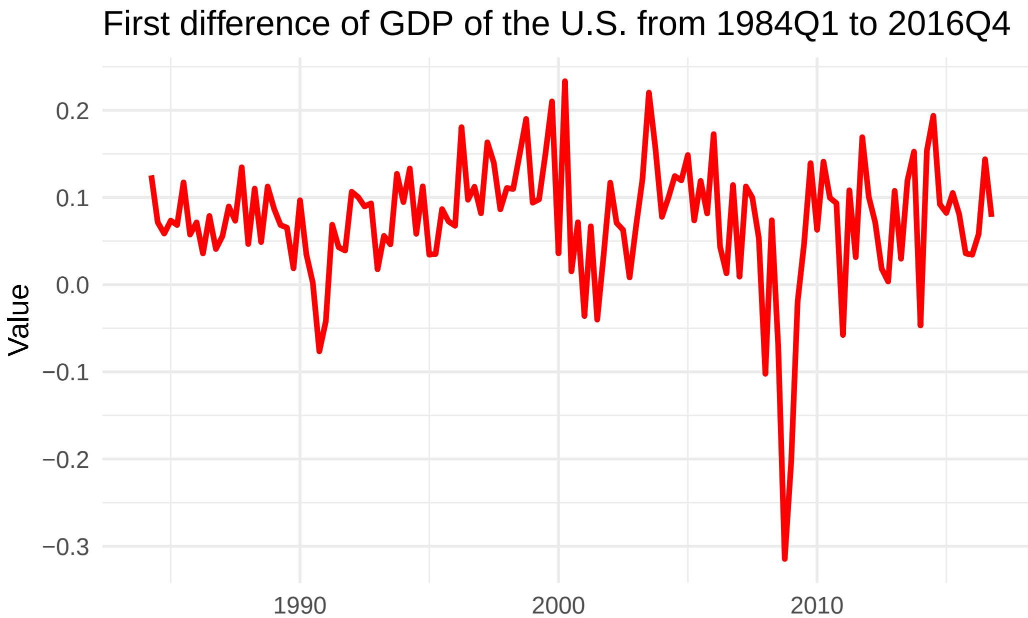
- If we remove/subtract the trend term from the above plot, the series looks:



Differencing

- When a variable y_t is not stationary, a common solution is to use differenced variable: $\Delta y_t = y_t - y_{t-1}$, for first order differences.
- **ARIMA (p,d,q)** denotes an ARMA model with **p autoregressive lags**, **q moving average lags**, and **difference in the order of d.**




First difference of GDP of the U.S. from 1984Q1 to 2016Q4
Value
0.2
0.1
0.0
−0.1
−0.2
−0.3
1990
2000
2010

Seasonality

- Seasonality is a particular type of autocorrelation pattern where patterns occur every "season," like monthly, quarterly, etc.
- For example, quarterly data may have the same pattern in the same quarter from one year to the next.
- Seasonality must also be corrected before a time series model can be fitted.

Estimation step

- Estimate ARMA models and examine the various coefficients.
- The goal is to select a stationary and parsimonious model that has significant coefficients and a good fit.

Diagnostic checking step

- If the model fits well, then the residuals from the model should resemble a while noise process.
- Check for normality looking at a histogram of the residuals or by using a quantile-quantile (Q-Q) plot.
- Check for independence by examining the ACF and PACF of the residuals, which should look like a white noise.
- The Ljung-Box-Pierce statistic performs a test of the magnitude of the autocorrelations of the correlations as a group.
- Examine goodness of fit using the Akaike Information Criteria (AIC) and Bayesian Information Criteria (BIC). Use most parsimonious model with lowest AIC and/or BIC.

Forecasting using ARIMA model

```
# Fit ARIMA model: two approach: using auto.arima() function from the

# forecast package and using the ARMA() function.
```

```
rm(list = ls())
library(tidyverse)
#browseURL("http://www.principlesofeconometrics.com/poe5/data/def/usmacro.def")
load(url("http://www.principlesofeconometrics.com/poe5/data/rdata/usmacro.rdata"))

head(usmacro)
```

```
    dateid01      g    inf   u
1 1948-01-01  2.267  2.119 3.7
2 1948-04-01  2.517  1.592 3.7
3 1948-07-01  2.418  1.684 3.8
4 1948-10-01  0.429 -0.918 3.8
5 1949-01-01 -1.888 -0.951 4.7
6 1949-04-01 -1.344 -0.109 5.9
```

```r
# Forecasting
library(forecast)

# Forecasting using AR(2) model using arima()

#?arima()

u <- ts(usmacro$u, frequency = 4, start = c(1948,1))
#u

fit_ar <- arima(u, order = c(2,0,0))  # AR(2)
summary(fit_ar)
```

```
Call:
arima(x = u, order = c(2, 0, 0))

Coefficients:
         ar1      ar2  intercept
      1.6129  -0.6612     5.7484
s.e.  0.0449   0.0451     0.3596

sigma^2 estimated as 0.08579:  log likelihood = -54.15,  aic = 116.3

Training set error measures:
                      ME      RMSE       MAE        MPE     MAPE      MASE
Training set 0.003681312 0.2929013 0.2142276 -0.2214522 3.833826 0.7917105
                   ACF1
Training set 0.06778812
```

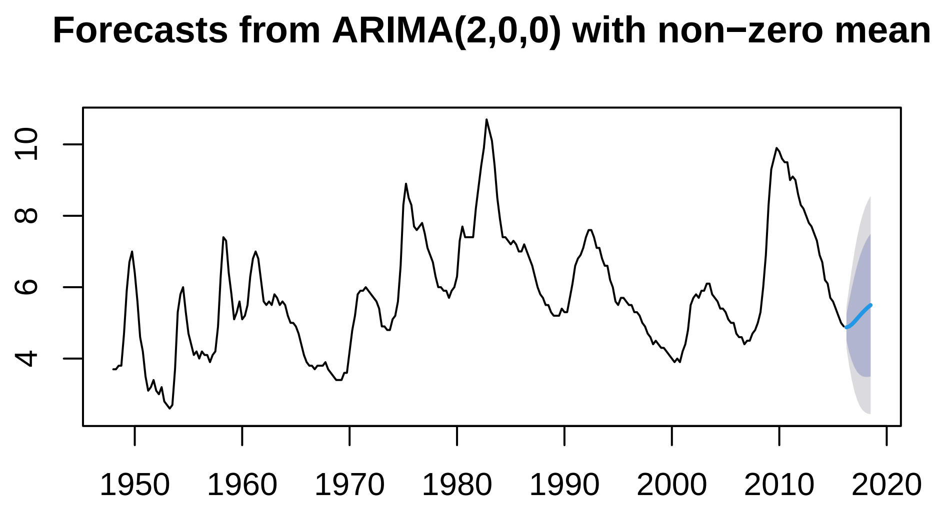
Use the model to forecast the unemployment rate in the next 10 quarters

```
# forcast next 10 time points
ar_forecast  <- forecast(fit_ar,h=10)
ar_forecast
```

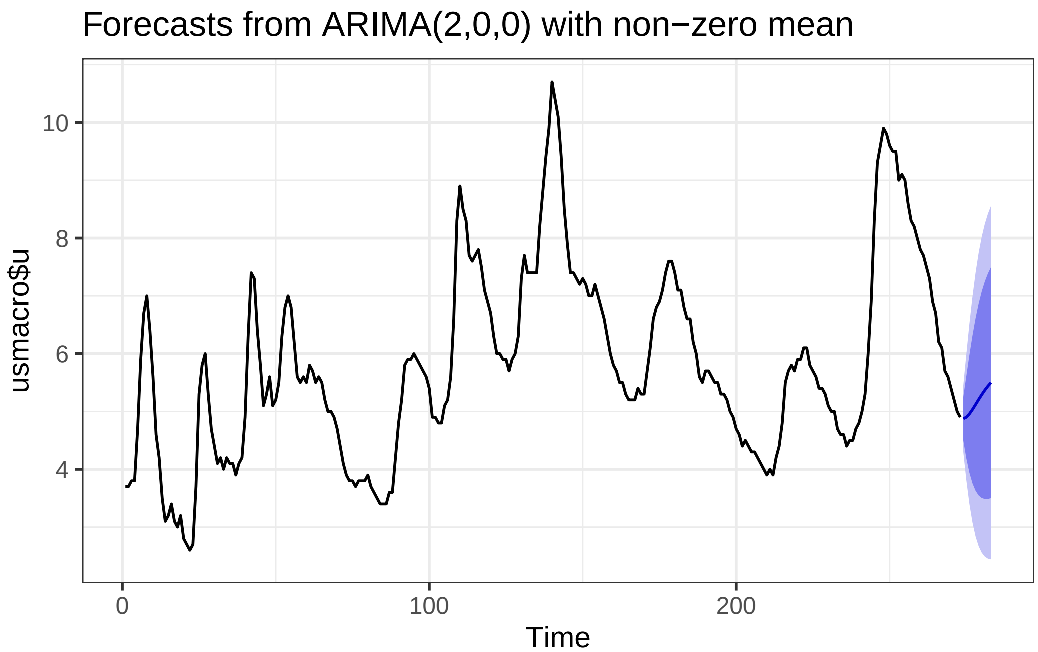
```
        Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2016 Q2       4.874878 4.499510 5.250246 4.300802 5.448954
2016 Q3       4.900482 4.188128 5.612836 3.811030 5.989934
2016 Q4       4.958390 3.939634 5.977146 3.400337 6.516443
2017 Q1       5.034859 3.755224 6.314495 3.077826 6.991892
2017 Q2       5.119907 3.627823 6.611990 2.837962 7.401851
2017 Q3       5.206516 3.547329 6.865702 2.669009 7.744022
2017 Q4       5.289971 3.503216 7.076727 2.557365 8.022578
2018 Q1       5.367309 3.485802 7.248816 2.489792 8.244826
2018 Q2       5.436863 3.486809 7.386918 2.454513 8.419214
2018 Q3       5.497910 3.499544 7.496276 2.441672 8.554148
```

- plot the data and forecast

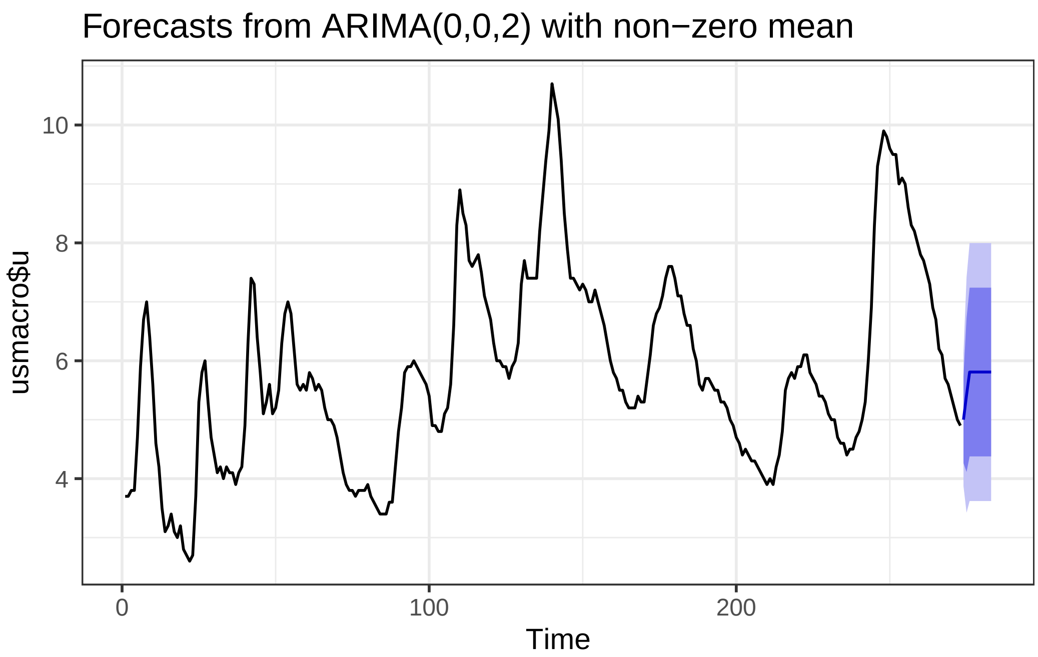
```r
#Plot
plot(ar_forecast)
```



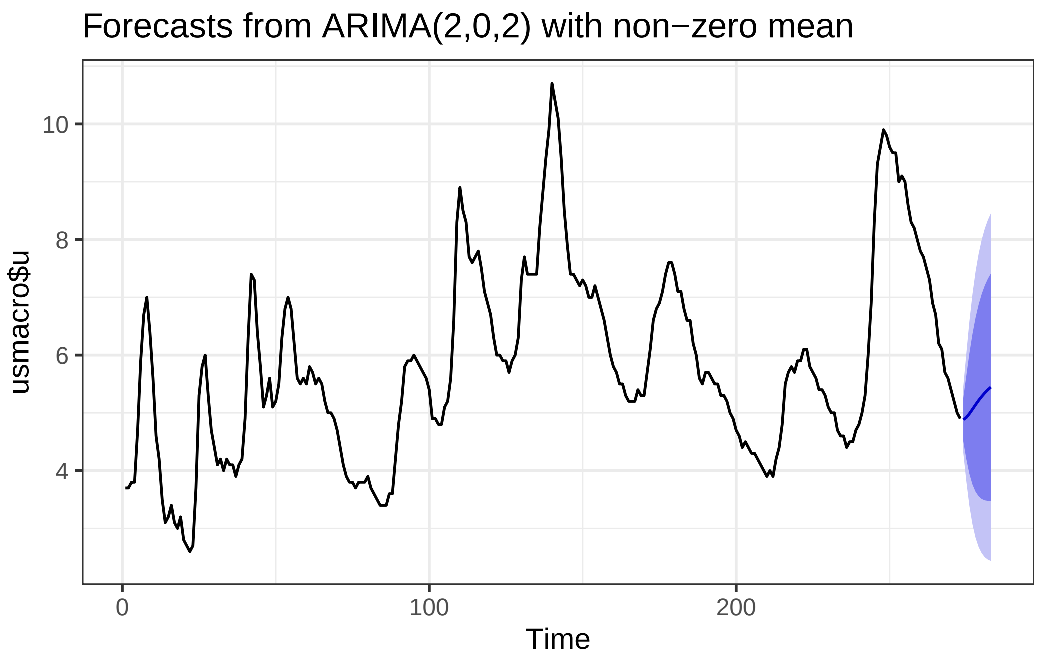
```
# Alternatively
arima(usmacro$u, order = c(2,0,0)) %>% forecast(h=10) %>% autoplot + theme_bw()
```



```r
# Forcasting using MA(2)
arima(usmacro$u, order = c(0,0,2))  %>% forecast(h=10) %>% autoplot + theme_bw()
```



```
# Forecasting using ARMA(2,0,2)
arima(usmacro$u, order = c(2,0,2))  %>% forecast(h=10) %>% autoplot + theme_bw()
```



- notice the difference of the following two lines of codes

```
arima(usmacro$u, order = c(2,0,0)) %>% forecast(h=10) + theme_bw()
```

```
NULL
```

```
# defining u as ts() object
u <- ts(usmacro$u, frequency = 4, start = c(1948,1))
arima(u, order = c(2,0,0)) %>% forecast(h=10)
```

```
        Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2016 Q2       4.874878 4.499510 5.250246 4.300802 5.448954
2016 Q3       4.900482 4.188128 5.612836 3.811030 5.989934
2016 Q4       4.958390 3.939634 5.977146 3.400337 6.516443
2017 Q1       5.034859 3.755224 6.314495 3.077826 6.991892
2017 Q2       5.119907 3.627823 6.611990 2.837962 7.401851
2017 Q3       5.206516 3.547329 6.865702 2.669009 7.744022
2017 Q4       5.289971 3.503216 7.076727 2.557365 8.022578
2018 Q1       5.367309 3.485802 7.248816 2.489792 8.244826
2018 Q2       5.436863 3.486809 7.386918 2.454513 8.419214
2018 Q3       5.497910 3.499544 7.496276 2.441672 8.554148
```

```
#arima(u, order = c(2,0,0)) %>% forecast(h=10) %>% autoplot
```

Forecasting using ARDL model

The ARDL(2,1) model:

$$U_t = \delta + \beta_1 U_{t-1} + \beta_2 U_{t-2} + \alpha_1 G_{t-1} + v_t$$

```r
# Example 9.7 Forecasting unemployment with an ARDL(2,1) model

usmacro.lag <- cbind( u = usmacro[,"u"],
                      g = usmacro[,"g"],
                      gLag1 = dplyr::lag(usmacro[,"g"],1))

head(usmacro.lag)
```

```
       u      g  gLag1
[1,] 3.7  2.267     NA
[2,] 3.7  2.517  2.267
[3,] 3.8  2.418  2.517
[4,] 3.8  0.429  2.418
[5,] 4.7 -1.888  0.429
[6,] 5.9 -1.344 -1.888
```

```
Series: usmacro.lag[, "u"]
Regression with ARIMA(2,0,0) errors

Coefficients:
         ar1      ar2  intercept     xreg
      1.6109  -0.6590     5.7602  -0.0057
s.e.  0.0454   0.0456     0.3622   0.0119

sigma^2 = 0.08732:  log likelihood = -54.34
AIC=118.68   AICc=118.9   BIC=136.71
```

```r
# Forecasting
fc2 <- forecast(fit2, h=3,xreg=cbind(xreg = c(usmacro[,"g"][273],0.869,1.069)))

fc2
```

```
    Point Forecast    Lo 80    Hi 80   Lo 95    Hi 95
274       4.876016 4.497326 5.254705 4.29686 5.455171
275       4.898494 4.180496 5.616492 3.80041 5.996577
276       4.955545 3.929425 5.981664 3.38623 6.524859
```

```r
# plot the forecast value and interval
autoplot(fc2) + ylab("Unemployment") +
  ggtitle("Forecast unemployment with future GDP growth")+
  theme_bw()
```