Time-Series Data Analysis

ARIMA Models

jsn-jin @ UCLA

Contents

Some Stochastic Processes

Stationarity

A stochastic process $\{Y_t\}_{t=1}^{\infty}$ is **strictly stationary** if, for any set of subscripts $t+1, t+2, \ldots, t+r$ with any given finite integer r, the joint distribution of

$$(Y_{t+1}, Y_{t+2}, \ldots, Y_{t+r})$$

is the same as the joint distribution of

$$(Y_{t+k+1}, Y_{t+k+2} \ldots, Y_{t+k+r})$$

with any time shift k.

A stochastic process $\{Y_t\}_{t=1}^{\infty}$ is **covariance stationary** if

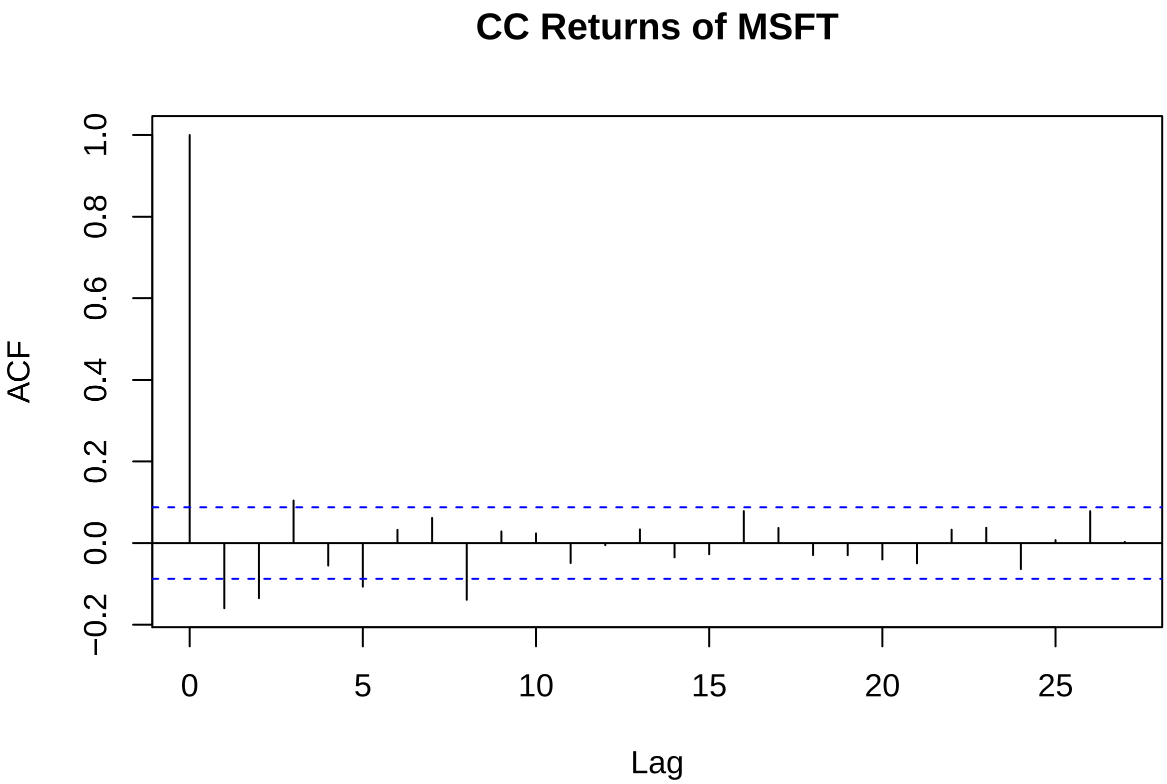
- $E[|Y_t|^2] \leq \infty$
- $E[Y_t] = \mu$ and $Var(Y_t) = \sigma^2$ for all t (time-invariant mean and variance)
- $Cov(Y_t, Y_{t+j}) = \gamma_j$ depends on j and not on t (time-invariant covariance)

ACF in R

The sample autocorrelation function (ACF) can be reported by the function `acf()`

```r
df <- read.csv(file = "MSFT.csv", head = TRUE, sep = ",", stringsAsFactors = FALSE)
df$Date <- as.Date(df$Date, "%Y-%m-%d")

N <- length(df$Adj.Close)
cc.ret <- log(df$Adj.Close[2:N]) - log(df$Adj.Close[1:(N-1)])

df.ret <- data.frame(df$Date[2:N], cc.ret)
```

```r
colnames(df.ret)[1] <- "Date" # rename the first column

acf(df.ret$cc.ret, main = "CC Returns of MSFT", plot = TRUE)
```

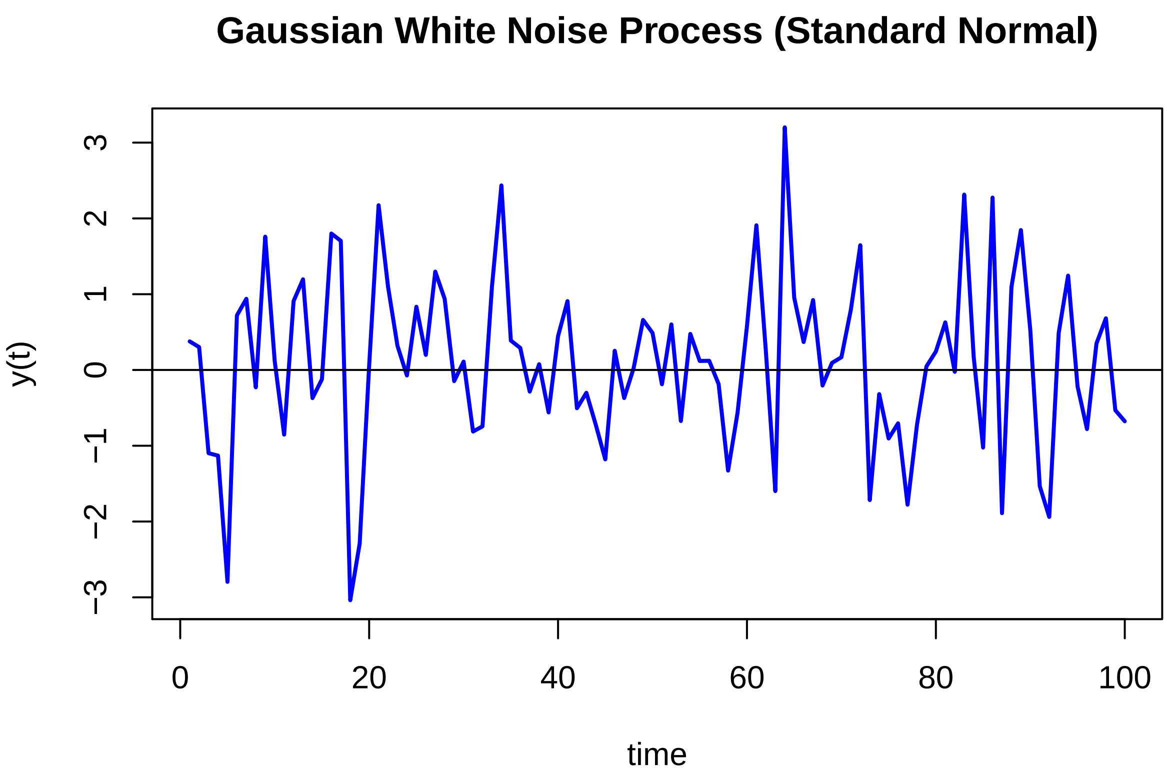


White Noise Processes

The code below are from An Introduction to Computational Finance and Financial Econometrics, by Eric Zivot. All errors are mine.

Let $\varepsilon_t \overset{\text{i.i.d.}}{\sim} N(0, \sigma^2)$. $\{\varepsilon_t\}$ is called a Gaussian (Nornal) White Noise process, or, $\varepsilon_t \sim GWN(0, \sigma^2)$. Here is an example.

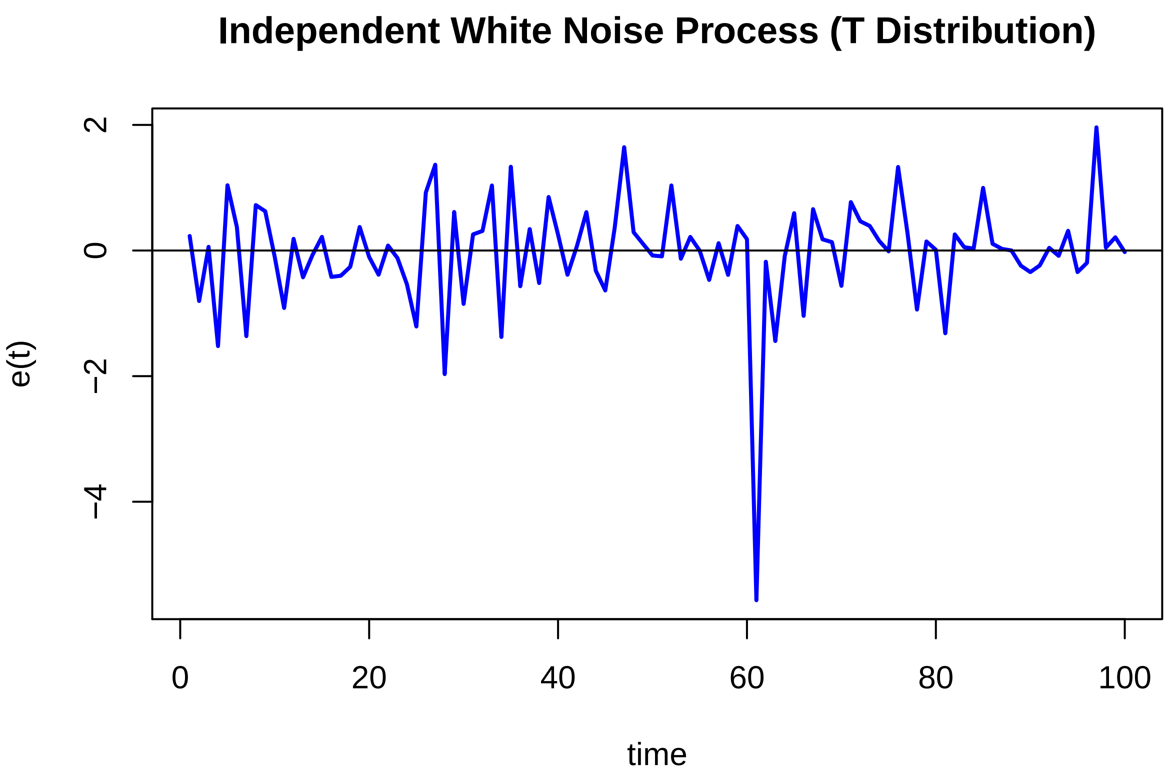
```r
set.seed(2020)

N <- 100

e1 = rnorm(N, 0, 1)

ts.plot(e1,
        main = "Gaussian White Noise Process (Standard Normal)",
        xlab = "time",
        ylab = "y(t)",
        col = "blue",
        lwd = 2)

abline(h = 0.00)
```



The function `ts.plot()` creates a time series line plot with a dummy time index.

We can relax assumption of normal distribution, and consider the independent white noise (IWN): $\varepsilon_t \overset{i.i.d.}{\sim} (0, \sigma^2)$.

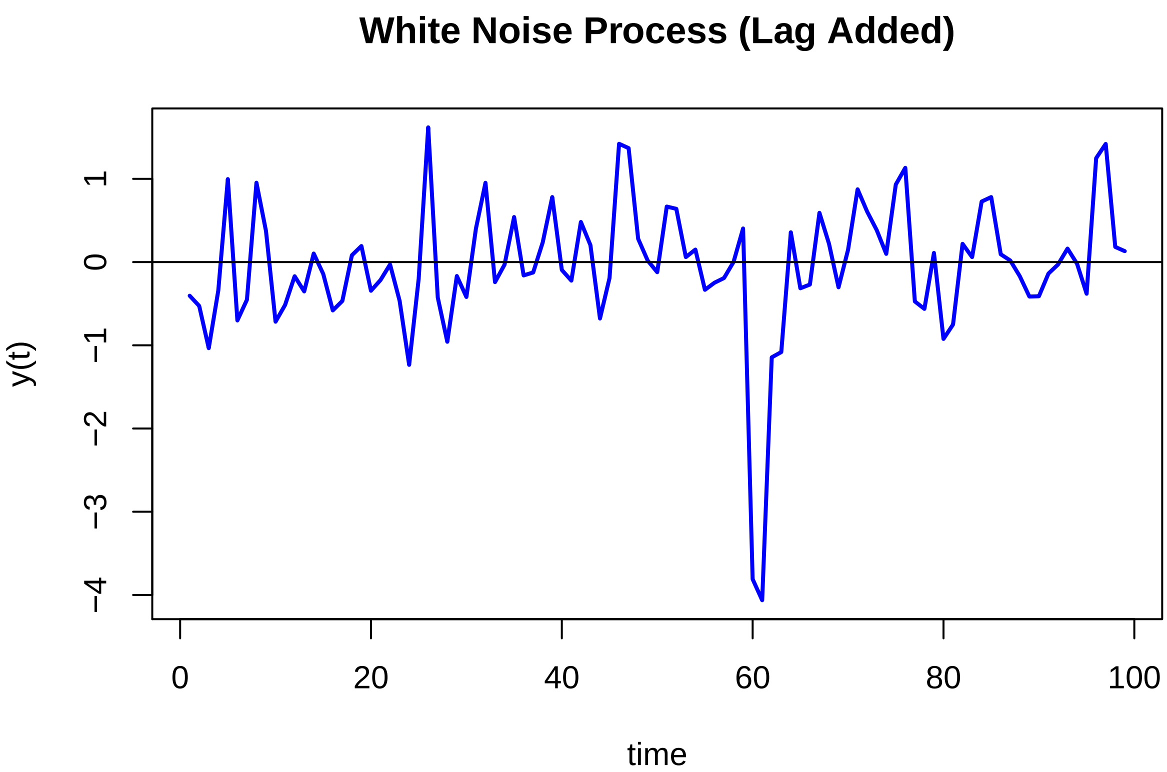
```
set.seed(2020)

N <- 100
v <- 3

e2 = rt(N, df = 3) / sqrt(v/(v-2)) # follows a T distribution

ts.plot(e2,
        main = "Independent White Noise Process (T Distribution)",
        xlab = "time",
        ylab = "e(t)",
        col = "blue",
        lwd = 2)

abline(h = 0.00)
```



This simulated IWN process has more extreme observations.

We can further drop the i.i.d. assumption, and consider the (weak) white noises: $\varepsilon_t \sim WN(0, \sigma^2)$.

```
set.seed(2020)

e3 <- sqrt(0.5) * e2[2:N] + sqrt(0.5) * e2[1:N-1]

ts.plot(e3,
        main = "White Noise Process (Lag Added)",
        xlab = "time",
        ylab = "y(t)",
        col = "blue",
        lwd = 2)

abline(h = 0.00)
```



ARIMA Models - MA(2) and AR(2)

Moving Average Models (MA)

Consider an MA(2) process:

$$Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2},$$

where $\epsilon_t \sim GWN(0, \sigma^2)$, and θ_1, θ_2 are finite. The moments of the MA(2) are

$$E[Y_t] = \mu$$

$$Var(Y_t) = \sigma^2(1 + \theta_1^2 + \theta_2^2)$$

$$Cov(Y_t, Y_{t+j}) = \begin{cases} (\theta_1 + \theta_1\theta_2)\sigma^2 & \text{if } j = 1, \\ \theta_2\sigma^2 & \text{if } j = 2, \\ 0 & \text{otherwise.} \end{cases}$$

We can verify that the MA(2) is covariance stationary.

Simulation

Simulate an MA(2) process with

$$\mu = 1,\ \theta_1 = 0.5,\ \theta_2 = 0.25,\ \sigma^2 = 1$$

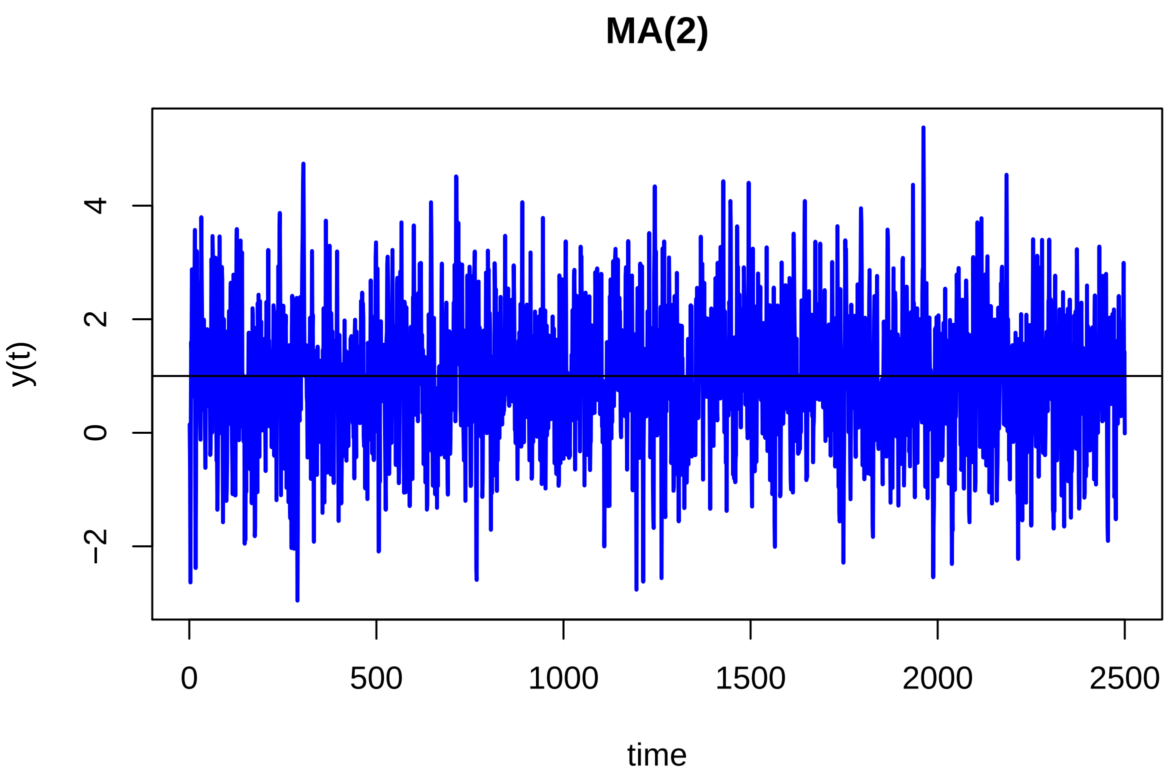
```r
n.obs <- 2500 + 2 # add 2 to compensate MA(2)

mu <- 1
sigma.e <- 1

theta1 <- 0.5
theta2 <- 0.25

set.seed(2020)

e <- rnorm(n.obs, sd = sigma.e) # generate the white noise
em1 <- c(0, e[1:(n.obs - 1)]) # white noise 1 period before
em2 <- c(0, 0, e[1:(n.obs - 2)]) # white noise 2 periods before

# generate our data
y <- mu + e + theta1 * em1 + theta2 * em2

head(y, n = 5)
```

```
## [1]  1.3769721  1.4900344  0.1469940 -0.6040304 -2.6362431
```

```r
y <- y[-(1:2)] # drop the first two observations (because they don't have noises in pervious periods)

head(y, n = 5)
```

```
## [1]  0.14699405 -0.60403040 -2.63624306  0.03970486  1.60027419
```

```r
ts.plot(y,
        main = "MA(2)",
        xlab = "time",
        ylab = "y(t)",
        col = "blue",
        lwd = 2)

abline(h = mu)
```



Or, we can use the (build-in) function `arima.sim()`

```
set.seed(2020)

error.model <- function(n){rnorm(n, sd = sigma.e)}

n.obs <- 2500

# simulate from an ARIMA (Autoregressive Intergrated Moving Average) model
# arima.sim(model, n, rand.gen = a function to generate noise)

y <- arima.sim(model = list(ma = c(theta1, theta2)), n = n.obs, rand.gen = error.model) + mu

head(y, 5)
```

```
## [1]  0.14699405 -0.60403040 -2.63624306  0.03970486  1.60027419
```

```
help(arima.sim)
```

As we see, this method is equivalent to last method.

ACFs

Next, we check the autocorrelations.

```
ARMAacf(ar = 0, ma = c(theta1, theta2), lag.max = 10)
```

```
##         0         1         2         3         4         5         6         7 
## 1.0000000 0.4761905 0.1904762 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 
##         8         9        10 
## 0.0000000 0.0000000 0.0000000
```

```r
sprintf("The lag 1 autocorrelation is %f", (theta1 + theta1 * theta2)/(1 + theta1^2 + theta2^2))
```

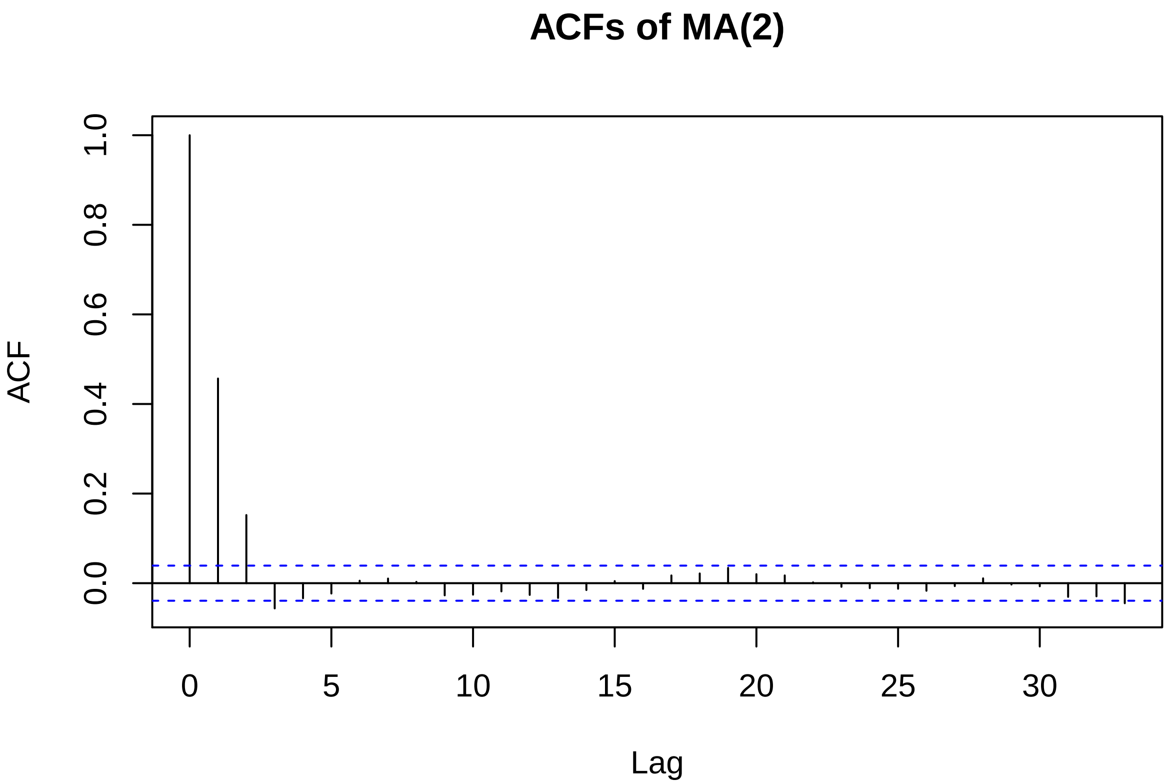
```
## [1] "The lag 1 autocorrelation is 0.476190"
```

```r
sprintf("The lag 2 autocorrelation is %f", theta2/(1 + theta1^2 + theta2^2))
```

```
## [1] "The lag 2 autocorrelation is 0.190476"
```

The sample ACFs are reported as follows.

```r
acf(y, main = "ACFs of MA(2)")
```




```r
cat("The sample ACFs are:")
```

```
## The sample ACFs are:
```

```r
acf(y, plot = FALSE)[1:10,]
```

```
##
## Autocorrelations of series 'y', by lag
##
##      1      2      3      4      5      6      7      8      9     10
##  0.457  0.152 -0.056 -0.034 -0.023  0.006  0.010  0.003 -0.027 -0.026
```

Estimation

We can fit an MA(2) model to the simulated data to get estimates $(\hat{\mu}, \hat{\theta}_1, \hat{\theta}_2, \hat{\sigma}^2)$.

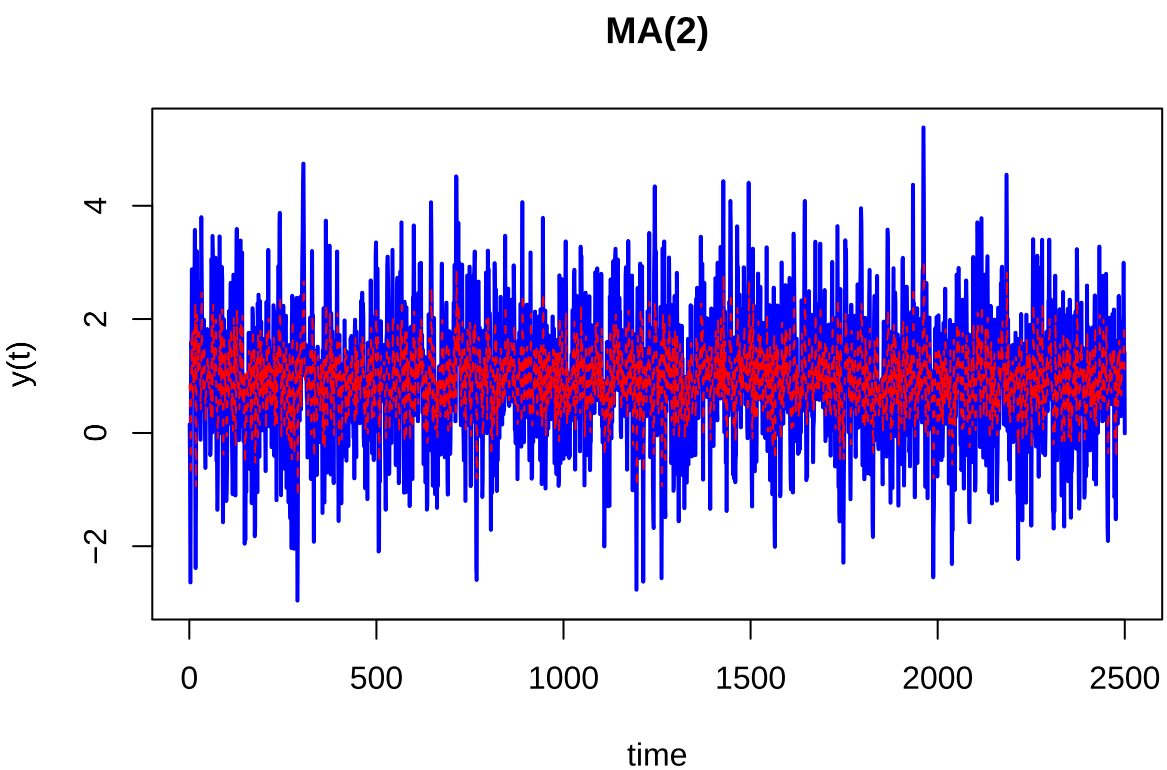
```r
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
# Arima function fits ARIMA model to univariate time series
# Arima(y, order = c(0,0,0), include.mean = TRUE)
MA <- Arima(y, order = c(0, 0, 2), include.mean = TRUE) # order = c(AR order, the degree of differencin
MA
```

```
## Series: y
## ARIMA(0,0,2) with non-zero mean
##
## Coefficients:
##          ma1     ma2    mean
##       0.5009  0.2515  0.9510
## s.e.  0.0196  0.0197  0.0356
##
## sigma^2 estimated as 1.033:  log likelihood=-3586.81
## AIC=7181.63   AICc=7181.64   BIC=7204.92
```

```r
# this Arima function is just like lm.
# but, instead of fit an linear model, Arima function fits a ARIMA model on the data
# however, becuase different ARIMA models can be very different, we have to tell ARIMA what is the orde

residuals <- residuals(MA)
MA_fitted <- y - residuals

ts.plot(y,
        main = "MA(2)",
        xlab = "time",
        ylab = "y(t)",
        col = "blue",
        lwd = 2)

points(MA_fitted, type = "l", col = 2, lty = 2)
```



Autoregressive models (AR)

Consider an AR(2) process:

$$Y_t - \mu = \phi_1(Y_{t-1} - \mu) + \phi_2(Y_{t-2} - \mu) + \epsilon_t,$$

(Note: the trend is taken out in this AR model)

where $|\phi_1 + \phi_2| < 1$. The moments of the AR(2) are

$$\begin{aligned} E[Y_t] &= \mu, \\ \gamma_0 = Var(Y_t) &= \frac{\sigma^2}{1 - \phi_1\rho_1 - \phi_2\rho_2}, \\ \rho_1 = Corr(Y_t, Y_{t-1}) &= \frac{\phi_1}{1 - \phi_2}, \\ \rho_2 = Corr(Y_t, Y_{t-2}) &= \frac{\phi_1^2 + (1 - \phi_2)\phi_2}{1 - \phi_2}, \\ \rho_\tau = Corr(Y_t, Y_{t-\tau}) &= \phi_1\rho_{\tau-1} + \phi_2\rho_{\tau-2}, \tau = 3, 4, \ldots \end{aligned}$$

We can verify that the AR(2) is covariance stationary.

Simulation

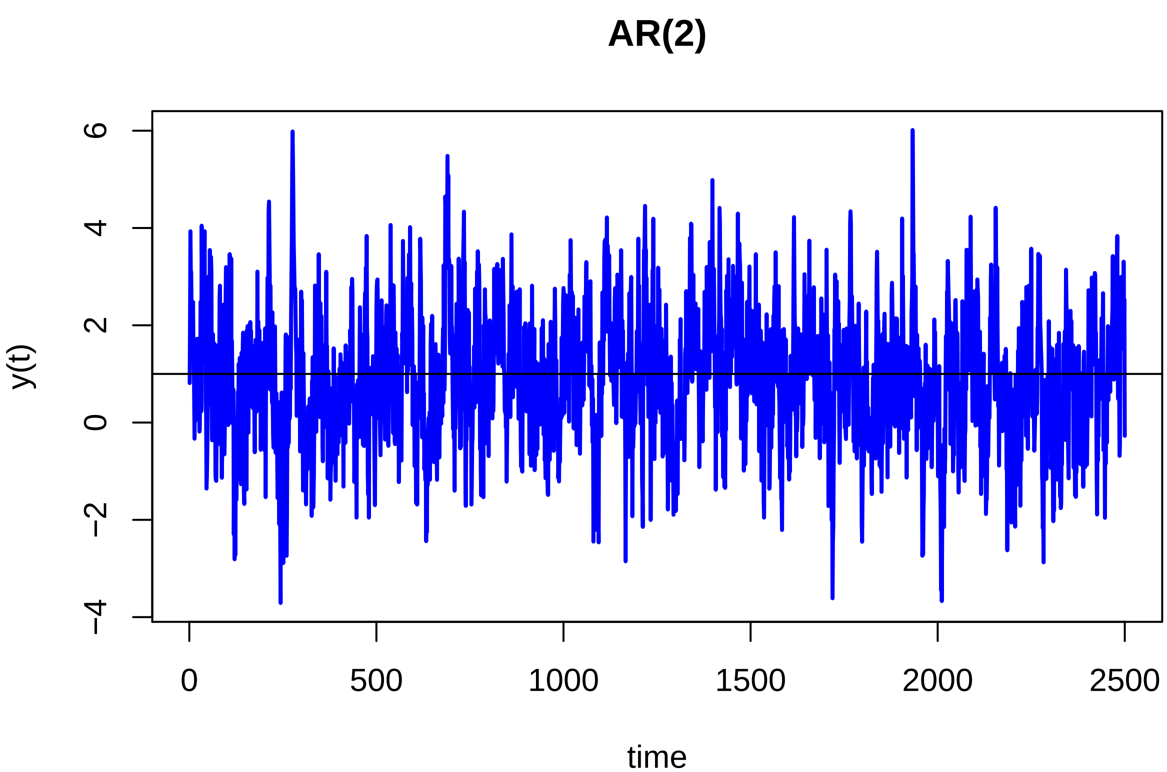
we can use the function `arima.sim()` to generate the simulated data.

```
n.obs <- 2500

mu <- 1
sigma.e <- 1
```

```
phi1 <- 0.5
phi2 <- 0.25

set.seed(2020)

error.model <- function(n){rnorm(n, sd = sigma.e)}

n.obs <-  2500

y <- arima.sim(model = list(order = c(2, 0, 0), ar = c(phi1, phi2)), n = n.obs, rand.gen = error.model)

head(y, n = 5)
```

```
## Time Series:
## Start = 1
## End = 5
## Frequency = 1
## [1] 0.8118779 2.0892871 3.9329867 3.1269336 3.0873412
```

```
ts.plot(y,
        main = "AR(2)",
        xlab = "time",
        ylab = "y(t)",
        col = "blue",
        lwd = 2)

abline(h = mu)
```



ACFs

Next, we check the autocorrelations.

$$
\begin{aligned}
\rho_1 &= Corr(Y_t, Y_{t-1}) = \frac{\phi_1}{1-\phi_2}, \\
\rho_2 &= Corr(Y_t, Y_{t-2}) = \frac{\phi_1^2 + (1-\phi_2)\phi_2}{1-\phi_2},
\end{aligned}
$$

```
ARMAacf(ar = c(phi1, phi2), ma = 0, lag.max = 10)
```

```
##         0         1         2         3         4         5         6         7
## 1.0000000 0.6666667 0.5833333 0.4583333 0.3750000 0.3020833 0.2447917 0.1979167
##         8         9        10
## 0.1601562 0.1295573 0.1048177
```

```
sprintf("The lag 1 autocorrelation is %f", phi1 / (1 - phi2))
```

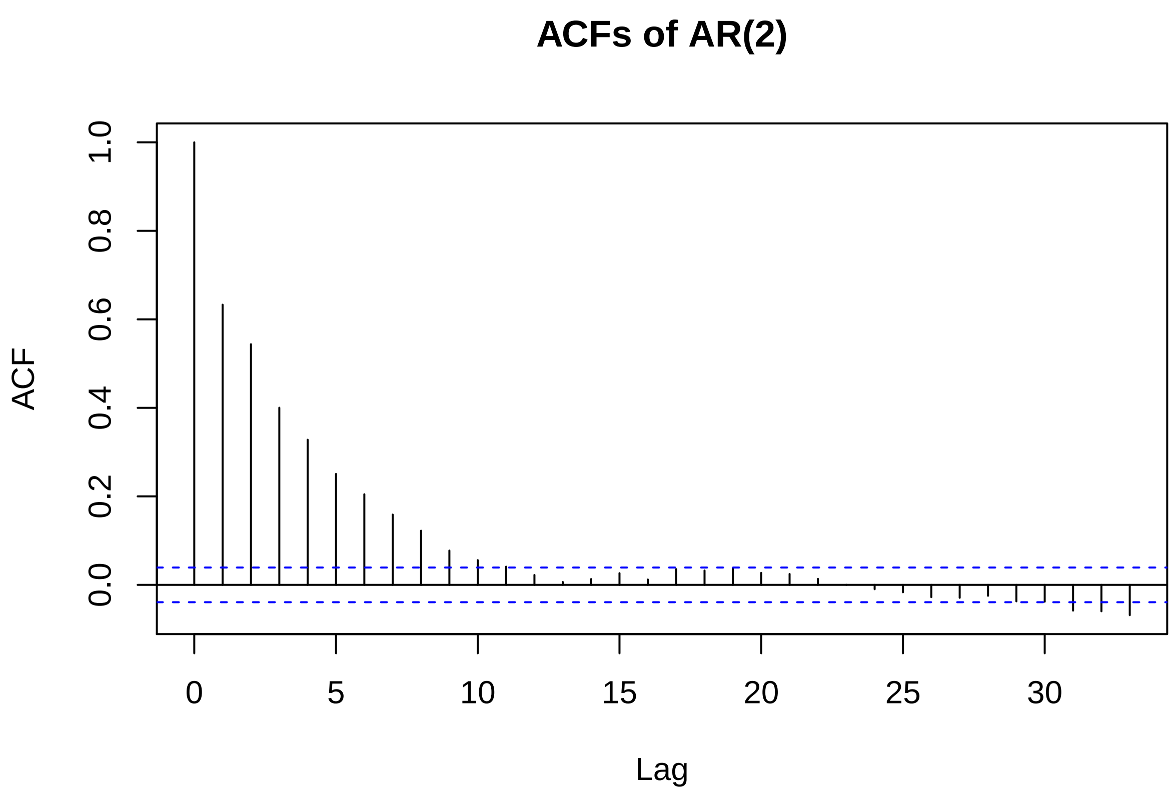
```
## [1] "The lag 1 autocorrelation is 0.666667"
```

```
sprintf("The lag 2 autocorrelation is %f", (phi1^2 + (1 - phi2) * phi2) / (1 - phi2))
```

```
## [1] "The lag 2 autocorrelation is 0.583333"
```

The sample ACFs are reported as follows.

```
acf(y, main = "ACFs of AR(2)")
```




```
cat("The sample ACFs are:")
```

```
## The sample ACFs are:
```

```r
acf(y, plot = FALSE)[1:10,]
```

```
##
## Autocorrelations of series 'y', by lag
##
##     1     2     3     4     5     6     7     8     9    10
## 0.633 0.544 0.400 0.328 0.251 0.205 0.159 0.122 0.078 0.056
```

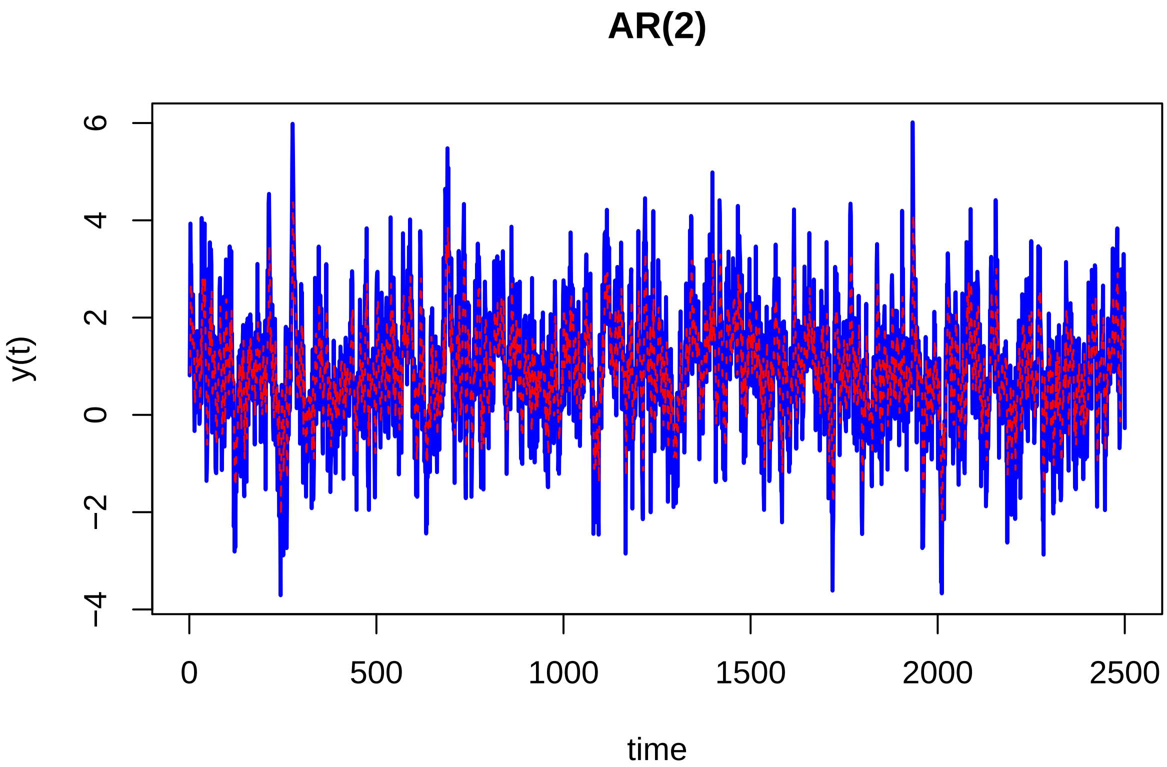
Estimation

We can fit an AR(2) model to the simulated data to get estimates $(\hat{\mu}, \hat{\phi}_1, \hat{\phi}_2, \hat{\sigma}^2)$.

```r
library(forecast)

AR <- Arima(y, order = c(2, 0, 0), include.mean = TRUE)
AR
```

```
## Series: y
## ARIMA(2,0,0) with non-zero mean
##
## Coefficients:
##          ar1     ar2    mean
##       0.4819  0.2384  0.8925
## s.e.  0.0194  0.0194  0.0722
##
## sigma^2 estimated as 1.025:  log likelihood=-3576.81
## AIC=7161.61   AICc=7161.63   BIC=7184.91
```

```r
residuals <- residuals(AR)
MA_fitted <- y - residuals

ts.plot(y,
        main = "AR(2)",
        xlab = "time",
        ylab = "y(t)",
        col = "blue",
        lwd = 2)

points(MA_fitted, type = "l", col = 2, lty = 2)
```


AR(2)
y(t)
time
6
4
2
0
−2
−4
0
500
1000
1500
2000
2500