

# Workshop 3: MA and AR Simulations

STAT 464/864 - Fall 2024  
Discrete Time Series Analysis  
Skye P. Griffith, Queen's University

## Setup

As usual, we need to load the `itsmr` package from the textbook. We're also going to load `knitr`, which should be installed by default. (Else, you can install it by running the code `install.packages("knitr")` in the CONSOLE.)

```
library(itsmr)
library(knitr)
set.seed(420) # we're also setting a *randomness* seed.
```

## Data

We'll be making our own data, today, according to the following models:

$$\begin{aligned}\text{MA}(2): \quad X_t &= Z_t + \theta_1 Z_{t-1} + \theta_2 Z_{t-2} \\ Z_t &\sim wn(\sigma_Z^2) \\ \{\theta_1, \theta_2, \sigma_Z^2\} &:= \{0.25, 0.75, 4\}\end{aligned}$$

---

$$\begin{aligned}\text{AR}(2): \quad Y_t &= \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + W_t \\ W_t &\sim wn(\sigma_W^2) \\ \{\phi_1, \phi_2, \sigma_W^2\} &:= \{0.25, 0.75, 4\}\end{aligned}$$

For integer  $t \in \{1, \dots, N = 500\}$

## Creating an MA(2) using ITSMR

We want to simulate the series

$$X_t = Z_t + (0.25)Z_{t-1} + (0.75)Z_{t-2}$$

### Simulation

The easy way to do this is by using the functions `sim` and `specify`, from ITSMR.

```
# --- Specify Parameters
# theta <- c(0.25, 0.75)
# s      <- 2
# N      <- 500

# # --- Simulate!
# model <- specify(, # give it your MA parameters
#                  sigma2 = ) # noise variance
#
# x      <- sim(n = ,      # number of simulations
#              a = ) # specify ARMA model
```

### Plotting

Let's plot  $x_t$  and compare it to plain white noise. We'll also plot the corresponding ACFs. Since this is simulated data, there are no units or scientific details to specify.

```
par(mfrow = c(2,1), mar = c(4,4,4,1))

# # --- Create noise series z(t)
# z <- -----

# # --- Time plots
# _____, main = ""
# mtext("z(t): simulated wn(0,4)") # subtitle

# _____, main = "x(t): simulated MA(2)"
# mtext( paste("theta =", list(theta))) # puts theta vals in the subtitle
```

```

par(mfrow = c(2,1), mar = c(4,4,4,1))

# # --- ACF plots
# _____, main = ""
# mtext("ACF of z(t): white noise wn(0,4)")
#
# _____, main = "ACF of x(t): simulated MA(2)"
# mtext(paste("theta =", list(theta) ))

```

**Time plots:** Notice the MA(2) series looks *smoother* than the white noise series. This makes sense, as each observation is a linear combination of previous values.

**ACF plots** According to the white noise hypothesis test, from class, we are allowed to enter the rejection region  $\alpha \times \max\{h\} = 0.05 \times 26 = 1.3 \rightarrow 1$  time before we have to reject the null.

So,  $z_t$  passes the white noise test, but  $x_t$  does not. This makes sense:  $x_t$  is highly correlated with the previous two timepoints, and that correlation should be proportional to the MA(2) coefficients.

## Creating an AR(2) from scratch

1. Create the white noise series  $\{w_t\}_{t=1}^{500}$ . This series is obviously *random*, but at each time ( $t$ ) where we create a new observation  $y_t = \text{blablabla} + w_t$ , we need to be drawing from the *same* observed series  $\{w_t\}$ .

We'll create way more than  $N$  observations, because we want to iteratively establish the shape of our model before formally storing the data. Let's choose  $M = 2500$ .

2. We have to get our AR(2) started, somehow. The first observation  $y_1$  is supposed to be made of past  $y_t$  values, but there is no  $y_0$ ! So let's start with white noise — not our  $w_t$  series, but some other white noise  $v_t$  with the same variance  $\sigma_W^2$ . We call this process “seeding” the series.

Again, this vector will be length  $M$ , not length  $N$ . We'll seed the first  $N$  entries of  $y_t$  with some noise unrelated to  $w_t$ .

3. Use a **for** loop to fill the remaining “un-seeded”  $y_t$  vector according to the desired model.
4. Throw out the first 2000 points as “burn-in” values. The remaining 500 observations form our final simulation  $y_t$ .

```

# # --- Specify Parameters
# phi <- c(0.25, 0.75)
# s    <- 2
# N    <- 500
# M    <- 5*N

# # --- 1. White noise w_t (length = M)!
# w <- -----

# # --- 2. Seed the series
# v <- rnorm(N, sd = s) # some other white noise
# y <- c(v, rep(0,M-N)) # seed 1 to N, fill rest with 0's, total = M points

# # --- 3. Simulate M-length AR(2)
# for(t in (N+1):M) {
#   # ----- # our model!
# }

# # --- 4. Throw away first (M-N) points as 'burn-in'
# y.full <- y # store the full series for later in the workshop :)
#
# w <- w[(M-N+1):M]
# y <- y[(M-N+1):M]

```

## Plotting

```

par(mar = c(4,4,3.5,1))

# _____, main = "y(t): simulated AR(2)"
# mtext(paste("phi =", list(phi) ))
#
# _____, main = "ACF of y(t): simulated AR(2)"
# mtext(paste("phi =", list(phi) ))

```

## Comparing the sample ACFs to the theoretical ACF

Recall from class, for an MA(2) process  $X_t$ ,

$$\gamma_X(0) = \sigma_Z^2(1 + \theta_1^2 + \theta_2^2)$$

$$\gamma_X(1) = \sigma_Z^2(\theta_1 + \theta_1\theta_2)$$

$$\gamma_X(2) = \sigma_Z^2\theta_2$$

Note  $\gamma_X$  is symmetric, and it's zero for  $|h| \geq 3$ .

Thus, we can capture the entire behaviour of  $\gamma_X$  (and  $\rho_X$ ) using only  $h = 0, 1, 2$ .

To get  $\rho_X(h)$ , we just standardize by dividing  $\gamma_X(h)/\gamma_X(0)$ .

```
# rho.y <- c( _____, # h=0
#           _____, # h=1
#           _____,) # h=2
#
# rho.y <- _____ # standardize by rho(0)
#
# acf(x, main = "ACF of x(t): simulated MA(2)")
# mtext(paste("theta =", list(theta) ))
#
# points(_____) # plot true rho over sample rho
```

## Linear Regression

Let's produce and plot the simple linear regression model:

$$y_i = \beta x_i + \varepsilon_i.$$

To avoid overwriting our time series, we'll call the predictor and response  $u, v$  instead of  $x, y$ .

```
# u <- rnorm(N) # predictor (just using noise, here)
# e <- rnorm(N) # noise vector
# b <- 2         # true regression coefficient
#
# v <- _____ # response
#
# # --- Create model
# model1 <- lm(_____) # -1 removes intercept
#
# # --- Plot line of best fit
# plot(u,v, main = "Simple Linear Regression")
# _____, col = "blue")
```

## Estimating the AR(2) coefficients $\phi_1$ and $\phi_2$

```
# model2 <-  
#   lm(_____  
#  
# phi.hat <- _____  
#  
# rownames(phi.hat) <- c("phi 1", "phi 2")  
#  
# kable(phi.hat)
```

The estimates  $\hat{\phi}_1$  and  $\hat{\phi}_2$  are quite good. We can see that by comparing the absolute difference  $|\hat{\phi} - \phi|$  to the standard error column, above. Or, we can look at the confidence intervals directly:

```
# # --- within 1.96 times the standard error?  
# result <- data.frame(  
#   estimate = phi.hat[,1],  
#   confint(model2),  
#   "within CI?" = ((phi.hat[,1] > confint(model2)[,1])  
#                   &(phi.hat[,1] < confint(model2)[,2]))  
# )  
# colnames(result)[2:3] <- colnames(confint(model2))  
#  
# kable(result)
```