**Código en R para simular un Ruido Blanco**

# Simulamos 100 números aleatorios que sean ruido blanco con media cero y

# varianza uno rnorm(n, mean = 0, sd = 1)

g = rnorm(100, 0, 1)

g

par(mfrow = c(2, 1)) ts.plot(g, main = "Nii(0,1)") acf(g, main = "Autocorrelación Simple", ylim = c(-1, 1), col = "black")

## Código en R para simular un camino aleatorio.

w = rnorm(100) # 100 números que suguen N(0,1)

x = w

for (t in 2:100) x[t] <- x[t - 1] + w[t] # genera un camino Aleatorio pues

# la primera diferencia es: x[t]-x[t-1]=w[t] (azar)

par(mfrow = c(2, 2))

ts.plot(x, main = "Camino aleatorio X\_t")

acf(x, main = "Autocorrelación Simple de Xt", ylim = c(-0.5, 0.5), col = "black")

d = diff(x)

ts.plot(d, main = "Primera diferencia de X\_t")

acf(d, main = "Autocorrelación Simple de d", ylim = c(-0.5, 0.5), ci.col = "black")

## Código en R para simular un proceso auroregresivo de orden 1 AR(1)

# Simulación de un proceso AR(1) con phi=0.4 order A specification of the

# non-seasonal part of the ARIMA model: the three components (p, d, q) are

# the AR order, the degree of differencing, and the MA order.

layout(matrix(c(1, 1, 2, 3), 2, 2, byrow = TRUE))

# ?arima.sim

AR = arima.sim(n = 100, model = list(order = c(1, 0, 0), ar = 0.4))

plot(AR, main = (expression(AR(1) ~ ~~~phi == +0.4)))

acf(AR, main = "Autocorrelación Simple", ylim = c(-0.5, 0.5), col = "black")

pacf(AR, main = "Autocorrelación Parcial", ylim = c(-0.5, 0.5), col = "black")

## Código en R para simular un proceso auroregresivo de orden 2 AR(2)

# Simulación de un proceso AR(1) con phi\_1=0.8897 y phi\_2=-0.4858

layout(matrix(c(1, 1, 2, 3), 2, 2, byrow = TRUE))

AR = arima.sim(n = 100, model = list(order = c(2, 0, 0), ar = c(0.8897, -0.4858)))

plot(AR, main = (expression(AR(2) ~ ~~~phi == c(0.8897, -0.4858))))

acf(AR, main = "Autocorrelación Simple", ylim = c(-0.5, 0.5), col = "black")

pacf(AR, main = "Autocorrelación Parcial", ylim = c(-0.5, 0.5), col = "black")

## Proceso de Poisson Homogeneo en R

# simulacion proceso poisson homogeneo

poisson\_h <- function(lambda,Tmax){

u <- runif(1)

t <- c(-log(u)/lambda)

T <- c(t)

i <- 1

while( t < Tmax){

u <- runif(1)

t <- t -log(u)/lambda

T <- c(T,t)

i <- i+1

}

T[i] <- Tmax

return(T)

}