# Quiz2

Dante Velasquez

4/26/2020

## Load Libraries

```
library(TSA)
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
library(forecast)
```

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

```
## Registered S3 methods overwritten by 'forecast':
##   method       from
##   fitted.Arima TSA
##   plot.Arima   TSA
```

```
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
library(tidyverse)
```

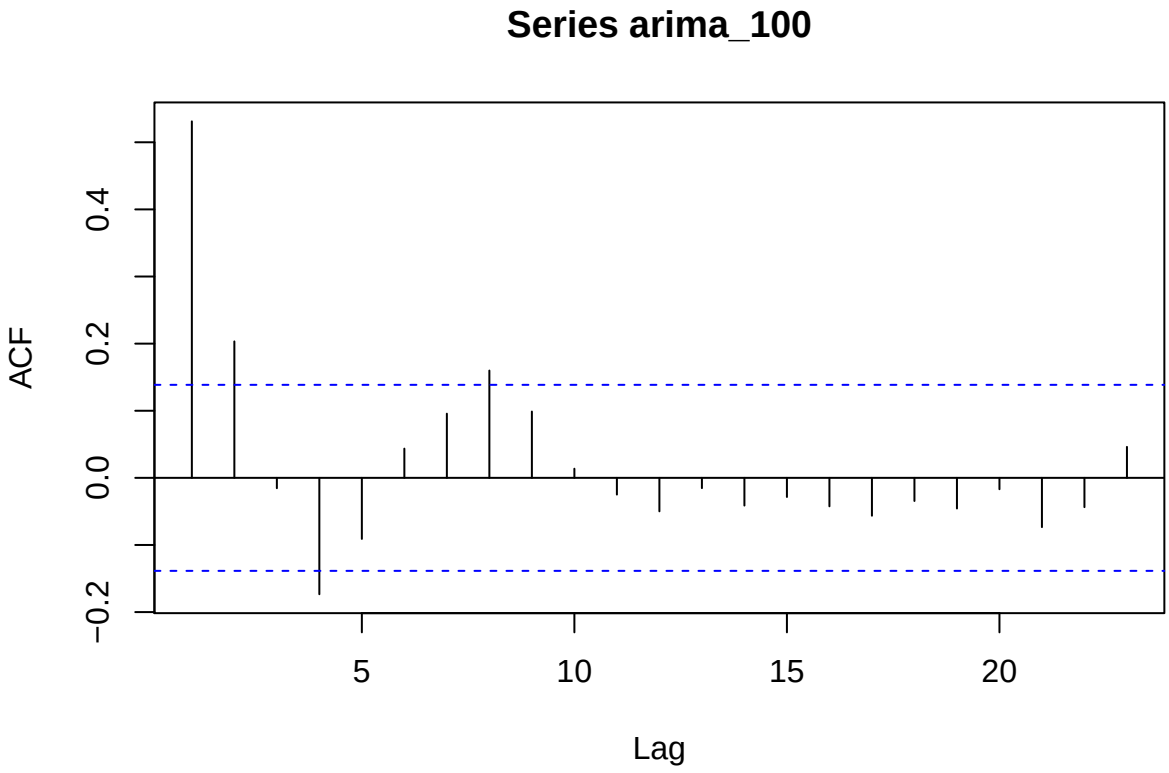
```
## -- Attaching packages ---------------------------------------------------------------------- tid
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
```

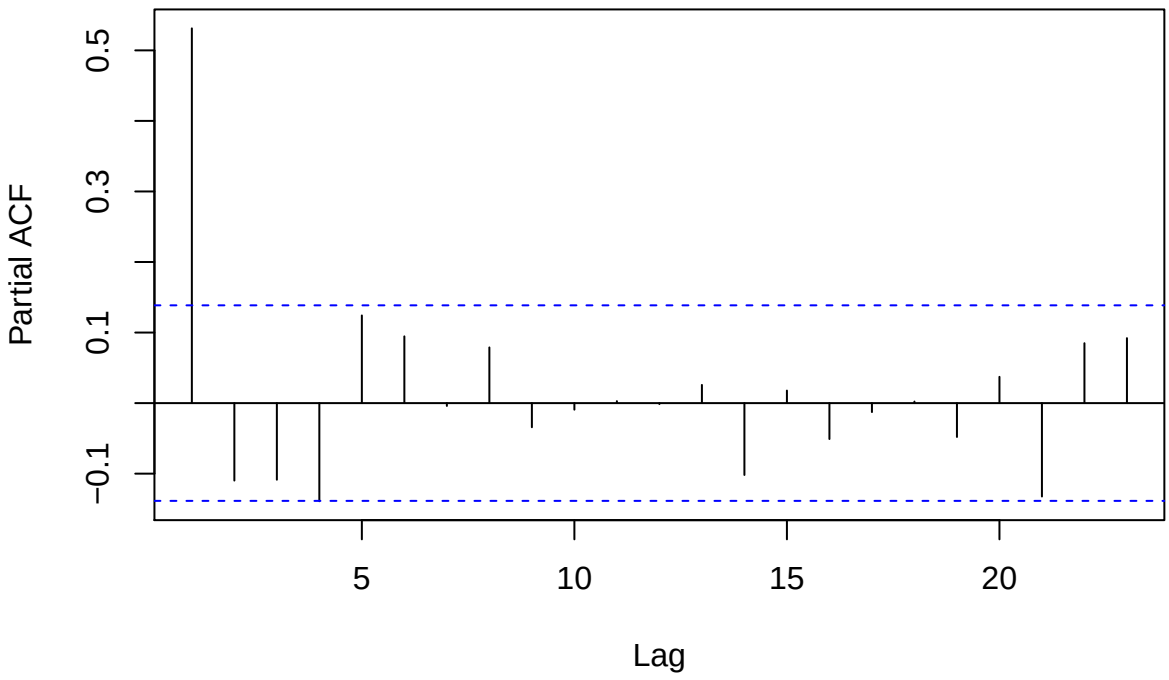
```
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
## -- Conflicts ------------------------------------------------------------------------------ tidyverse
## x dplyr::filter() masks stats::filter()
## x dplyr::first()  masks xts::first()
## x dplyr::lag()    masks stats::lag()
## x dplyr::last()   masks xts::last()
## x readr::spec()   masks TSA::spec()
```

## Problem 2

```r
set.seed(1224)

phi <- c(0.6, -0.3)
theta <- c(-0.7, -0.1)
stde <- 10
mu <- 100
N <- 200

white_noise <- rnorm(n = N, mean = mu, sd = stde)

arima_100 <- arima.sim(n = N, model = list(order = c(1, 0, 0), ar = phi[1]), sd = stde) + mu
arima_010 <- arima.sim(n = N, model = list(order = c(0, 1, 0)), sd = stde) + mu
arima_001 <- arima.sim(n = N, model = list(order = c(0, 0, 1), ma = theta[1]), sd = stde) + mu

arima_200 <- arima.sim(n = N, model = list(order = c(2, 0, 0), ar = phi), sd = stde) + mu
arima_002 <- arima.sim(n = N, model = list(order = c(0, 0, 2), ma = theta), sd = stde) + mu
arima_212 <- arima.sim(n = N, model = list(order = c(2, 1, 2), ar = phi, ma = theta), sd = stde) + mu

arima_211 <- arima.sim(n = N, model = list(order = c(2, 1, 1), ar = phi, ma = theta[1]), sd = stde) + mu
arima_112 <- arima.sim(n = N, model = list(order = c(1, 1, 2), ar = phi[1], ma = theta), sd = stde) + mu
arima_131 <- arima.sim(n = N, model = list(order = c(1, 3, 1), ar = phi[1], ma = theta[1]), sd = stde) +

# Arima(1, 0, 0)
acf(arima_100)
```



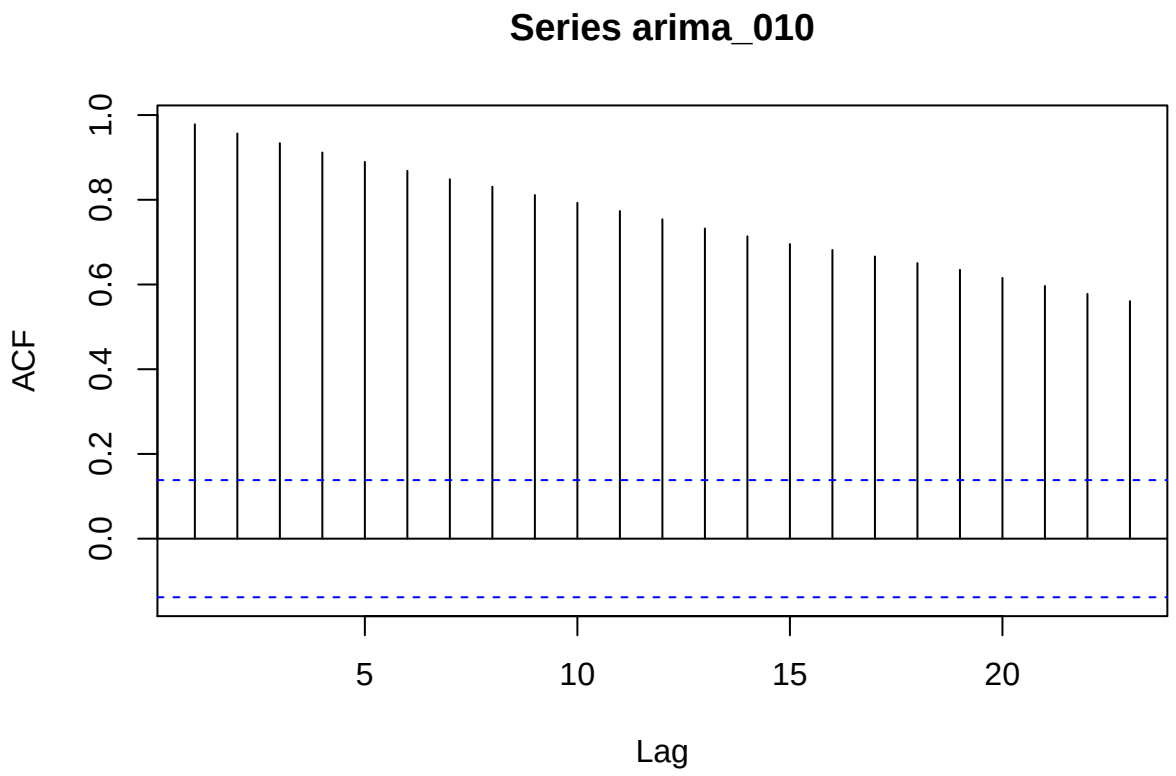

```
pacf(arima_100)
```



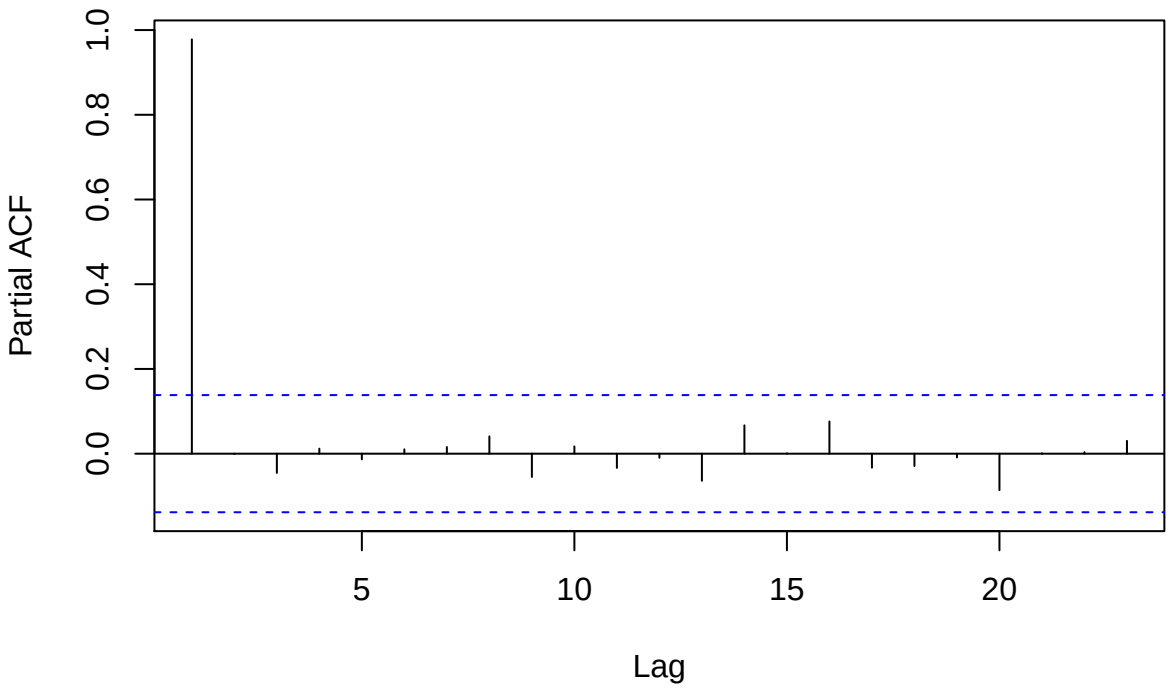

```
eacf(arima_100)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x o x o o o x o o o  o  o  o
## 1 x x o x o o o o o o o  o  o  o
## 2 x o x o o o o o o o o  o  o  o
## 3 x o x o o o o o o o o  o  o  o
## 4 x x x o o x o o o o o  o  o  o
## 5 x x x x o o o o o o o  o  o  o
## 6 o x x x x o o o o o o  o  o  o
## 7 o x x x x o o o o o o  o  o  o
```

```
# Arima(0, 1, 0)
acf(arima_010)
```




```
pacf(arima_010)
```



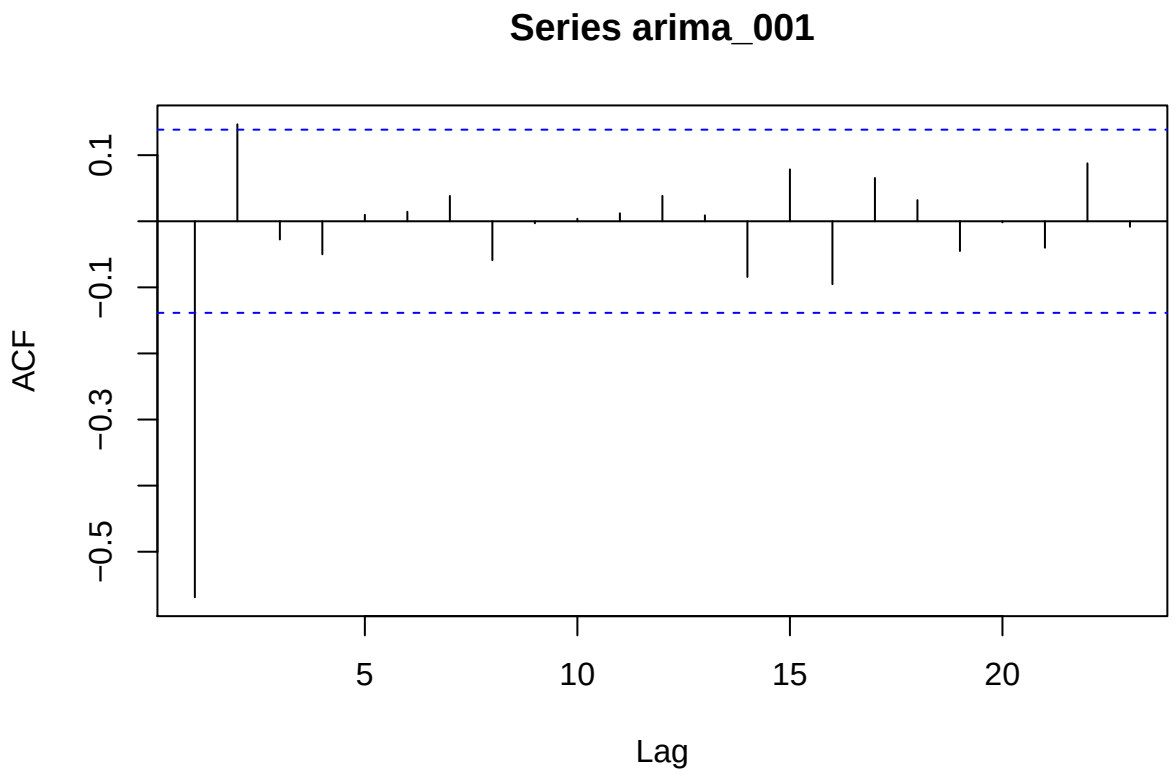

```
print('arima(0, 1, 0)')
```

```
## [1] "arima(0, 1, 0)"
```

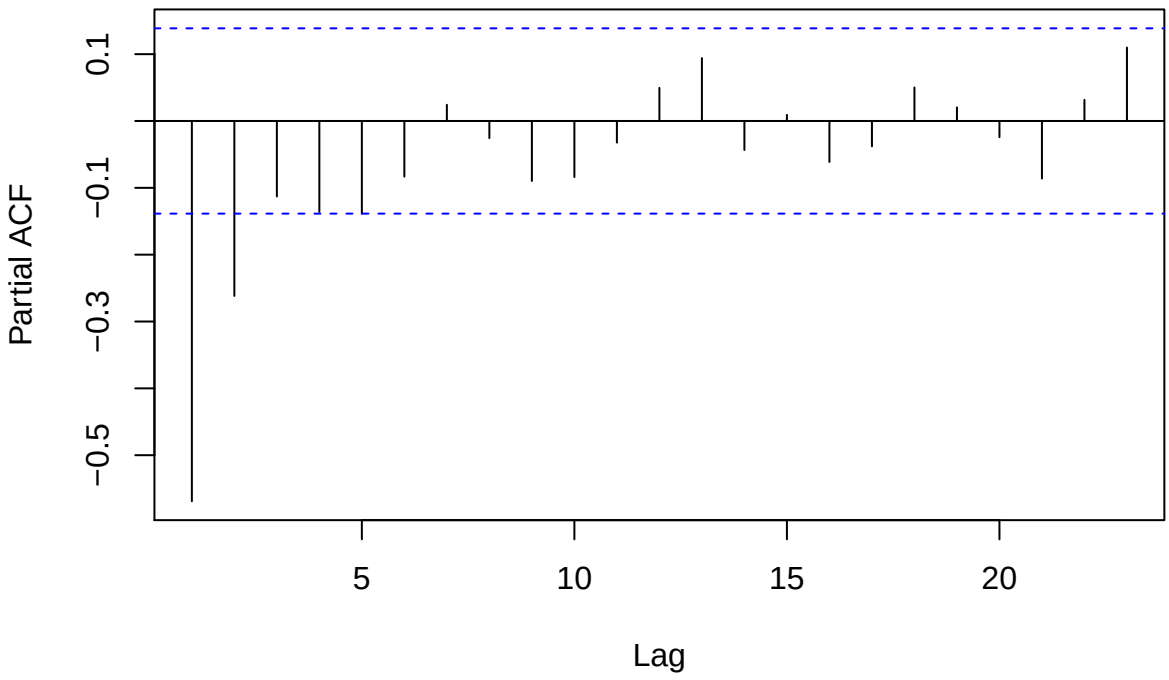
```
eacf(arima_010)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x x x x  x  x  x
## 1 o o o o o o o o o o o  o  o  o
## 2 x o o o o o o o o o o  o  o  o
## 3 x x o o o o o o o o o  o  o  o
## 4 x x x o o o o o o o o  o  o  o
## 5 o x x x o o o o o o o  o  o  o
## 6 x x o o o o o o o o o  o  o  o
## 7 o x o x x x o o o o o  o  o  o
```

```
# Arima(0, 0, 1)
acf(arima_001)
```




```
pacf(arima_001)
```



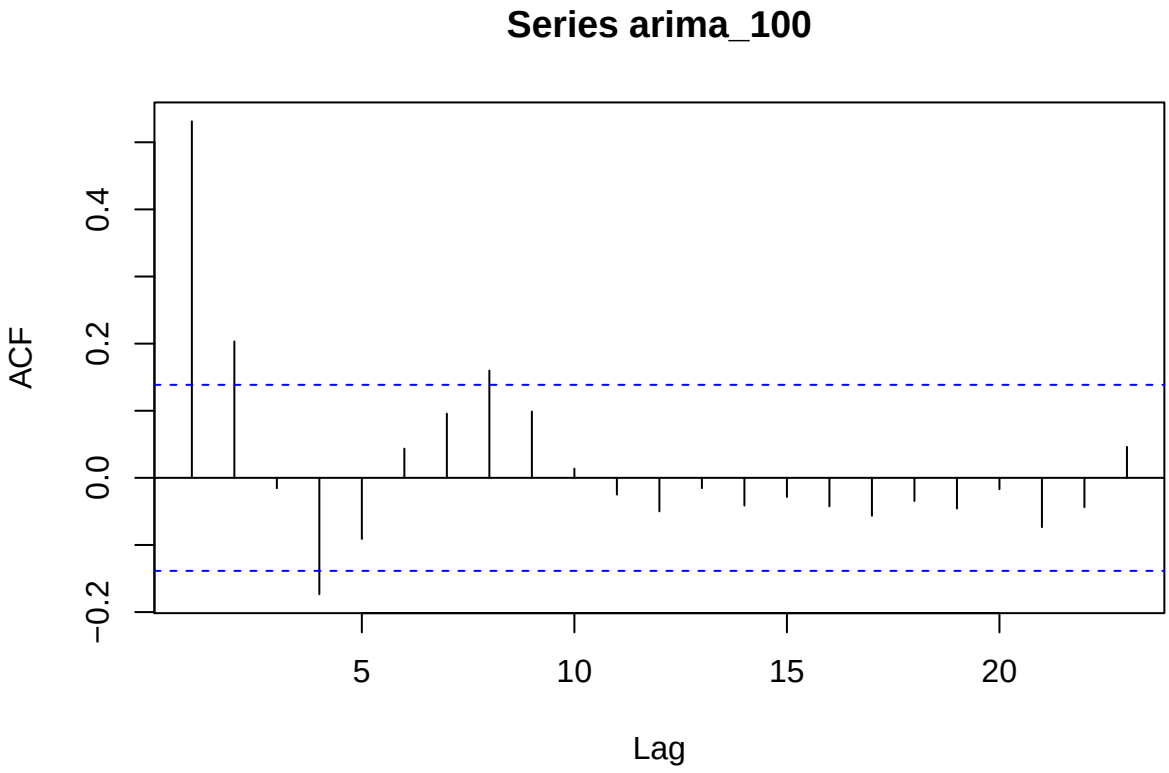

```
print('arima(0, 0, 1)')
```

```
## [1] "arima(0, 0, 1)"
```

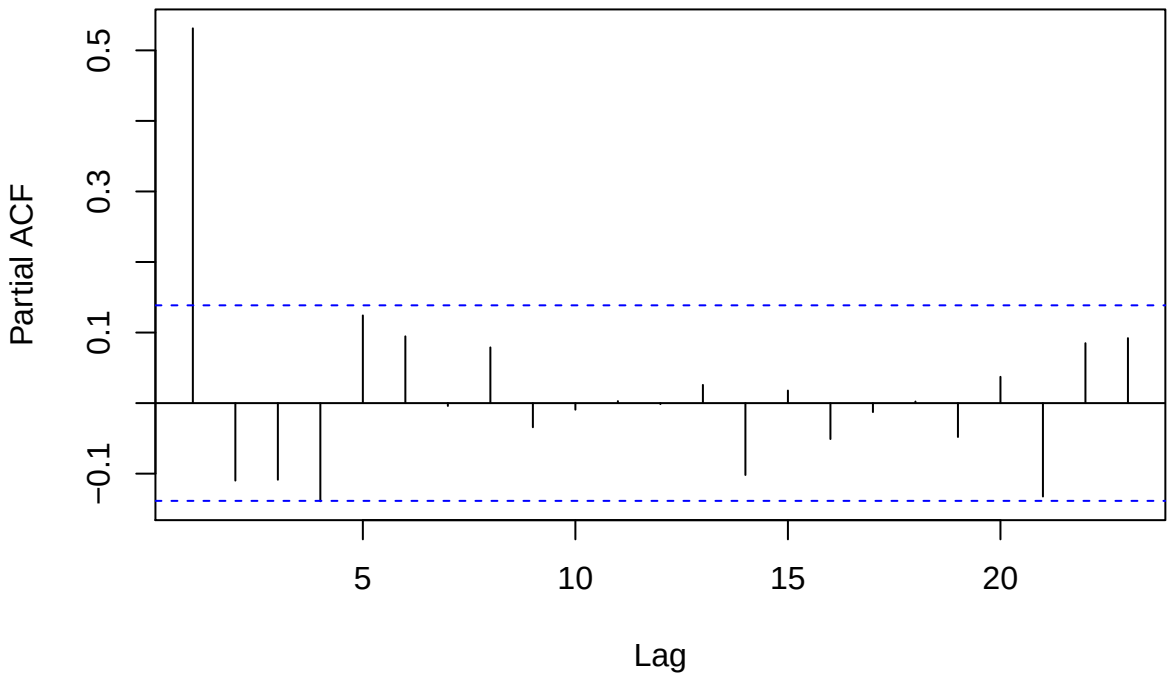
```
eacf(arima_001)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x o o o o o o o o o  o  o  o
## 1 x o o o o o o o o o o  o  o  o
## 2 x o x o o o o o o o o  o  o  o
## 3 x o x o o o o o o o o  o  o  o
## 4 x o o x x o o o o o o  o  o  o
## 5 x x o x o o o o o o o  o  o  o
## 6 x o x x o x x o o o o  o  o  o
## 7 x o x x x x o o o o o  o  o  o
```

```
# Arima(2, 0, 0)
acf(arima_100)
```




```
pacf(arima_100)
```



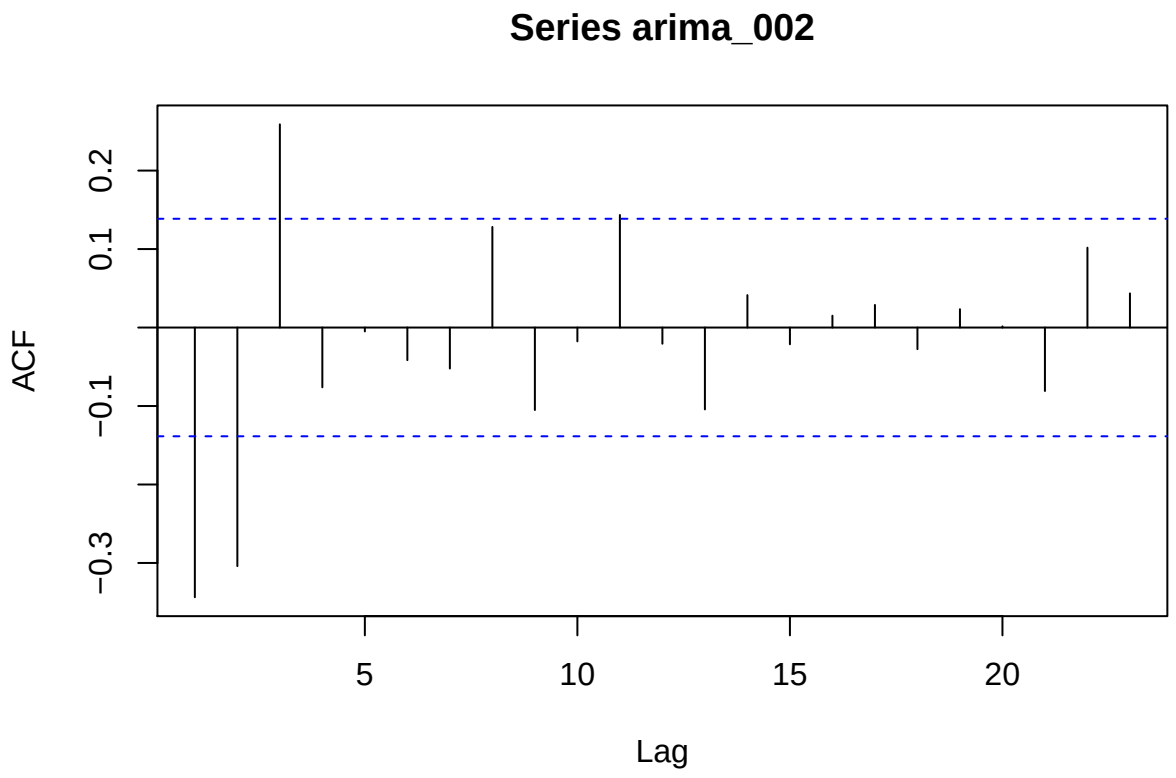

```
print('arima(1, 0, 0)')
```

```
## [1] "arima(1, 0, 0)"
```

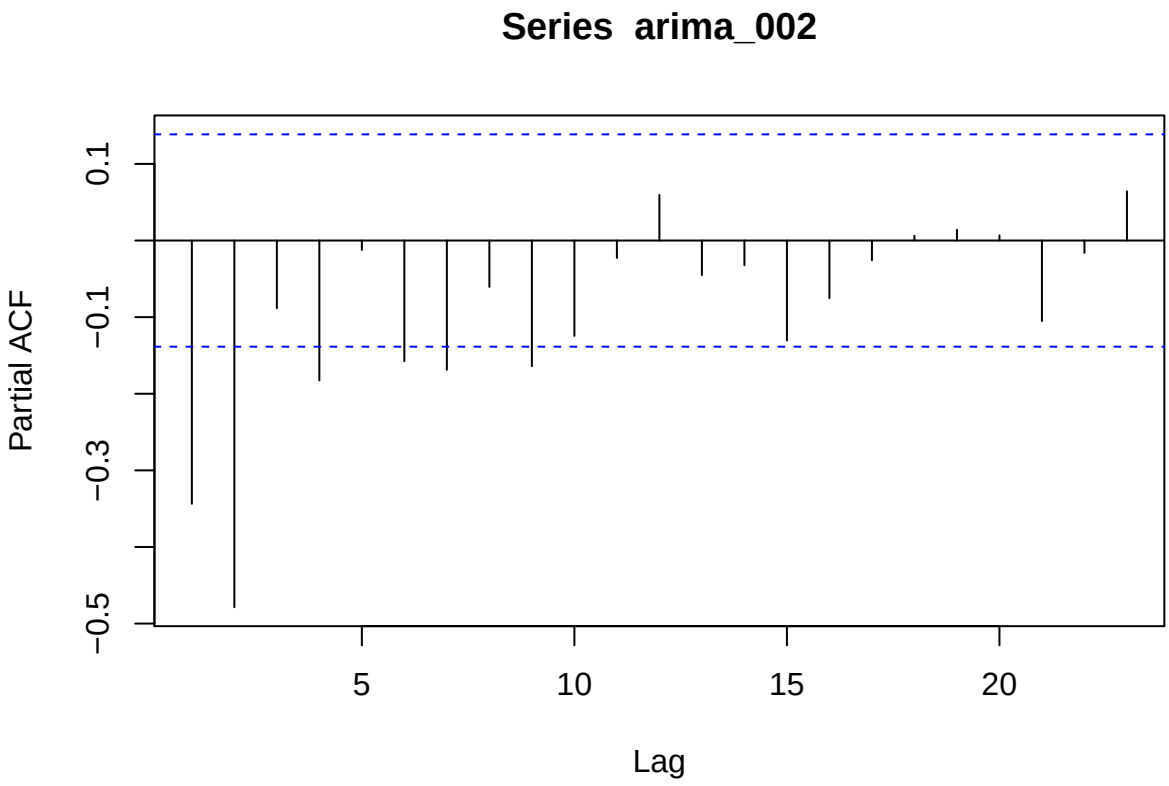
```
eacf(arima_100)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x o x o o o x o o o  o  o  o
## 1 x x o x o o o o o o o  o  o  o
## 2 x o x o o o o o o o o  o  o  o
## 3 x o x o o o o o o o o  o  o  o
## 4 x x x o o x o o o o o  o  o  o
## 5 x x x x o o o o o o o  o  o  o
## 6 o x x x x o o o o o o  o  o  o
## 7 o x x x x o o o o o o  o  o  o
```

```
# Arima(0, 0, 2)
acf(arima_002)
```




```
pacf(arima_002)
```



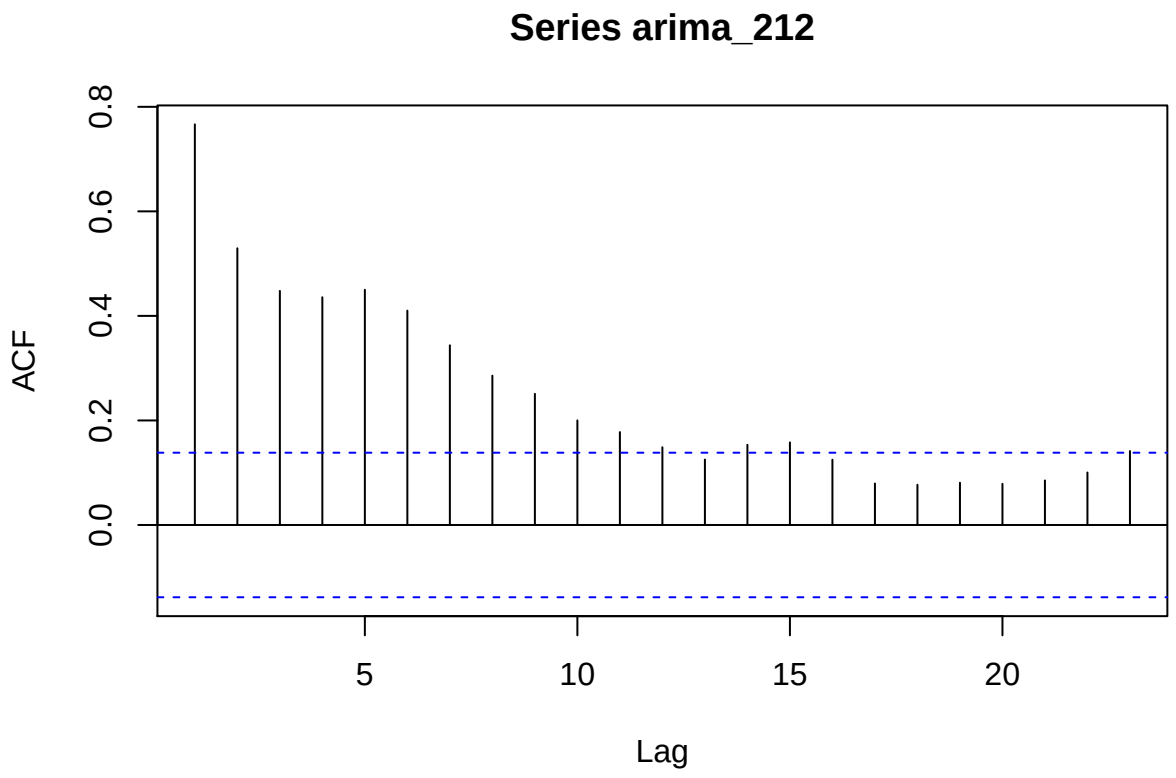

```
print('arima(0, 0, 2)')
```

```
## [1] "arima(0, 0, 2)"
```

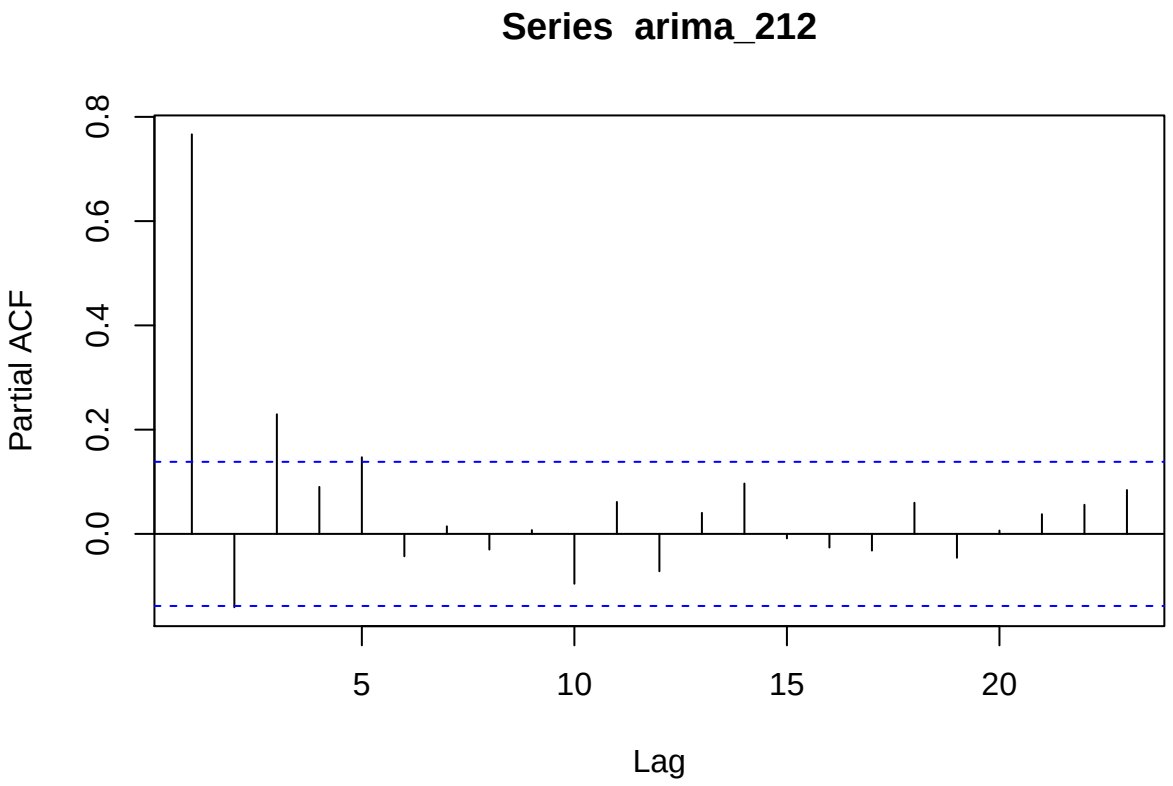
```
eacf(arima_002)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x o o o o o o o o  o  o  o
## 1 x x x o o o o o o o x  o  o  o
## 2 x x o o o o o o o o o  o  o  o
## 3 x x o o o o o o o o o  o  o  o
## 4 o x o o x x o o o o o  o  o  o
## 5 o x x o x o o o o o o  o  o  o
## 6 x x x x o o x o o o o  o  o  o
## 7 x x x x x o x o o o o  o  o  o
```

```
# Arima(2, 1, 2)
acf(arima_212)
```




```
pacf(arima_212)
```



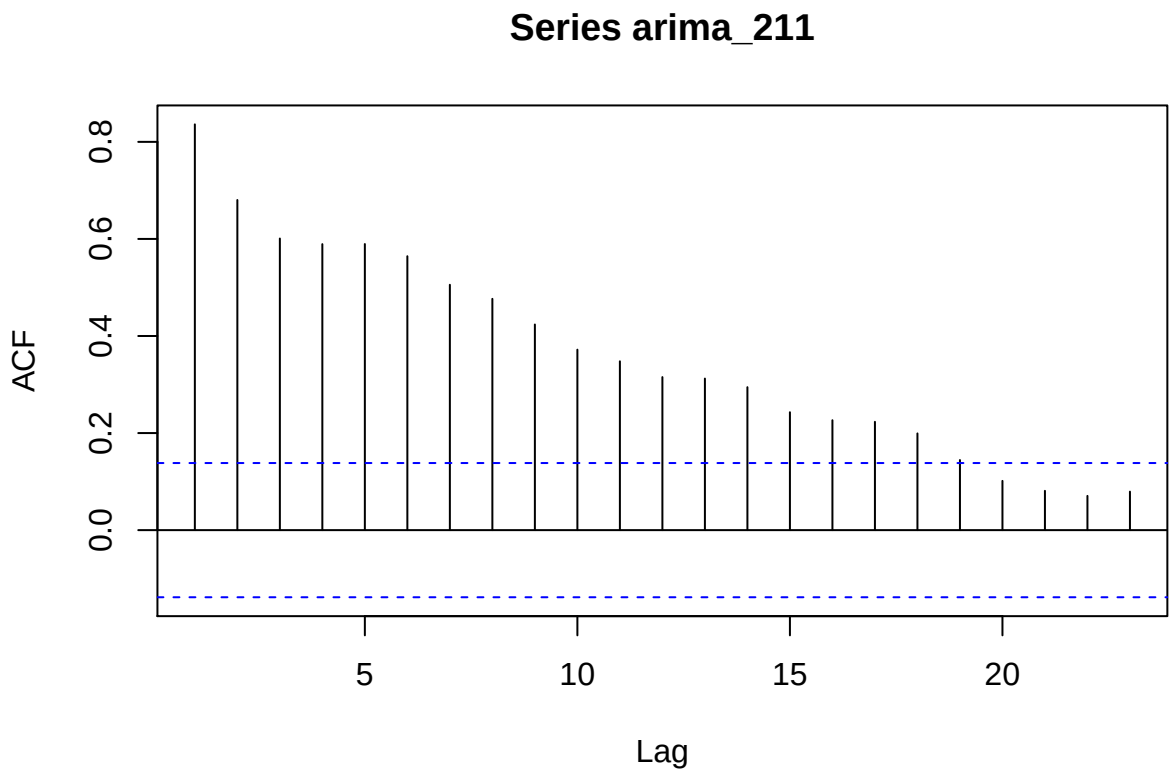

```
print('arima(2, 1, 2)')
```

```
## [1] "arima(2, 1, 2)"
```

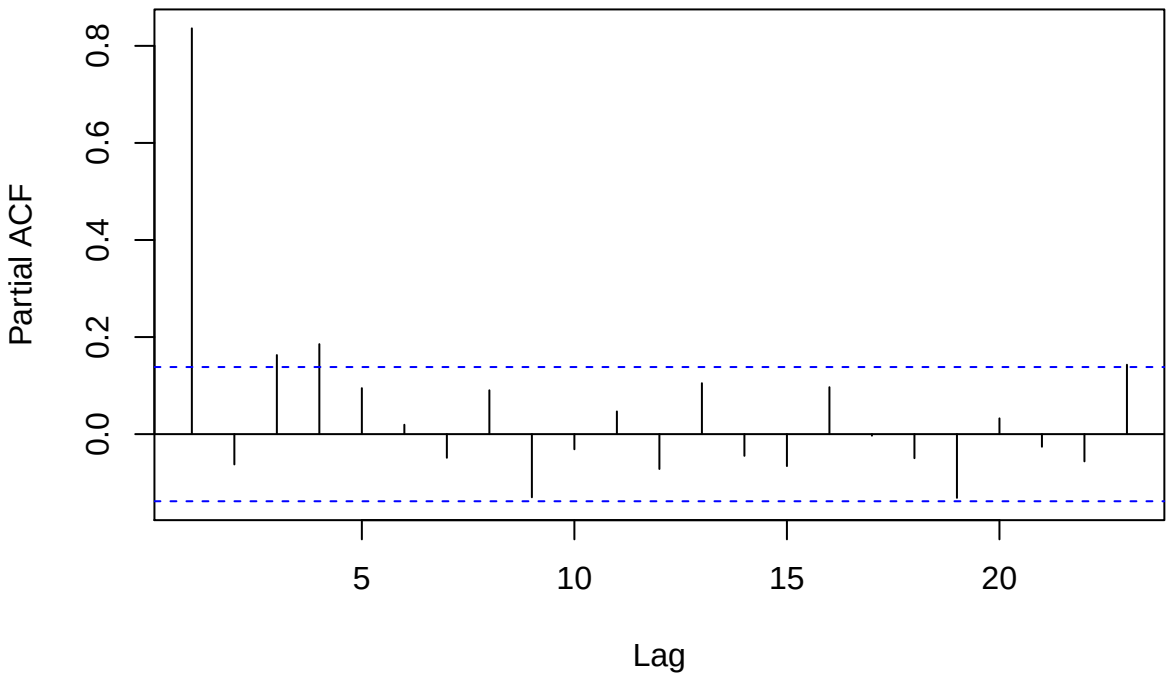
```
eacf(arima_212)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x x x x  x  o  x
## 1 x x x o o o o o o o o  o  o  o
## 2 x x o o o o o o o o o  o  o  o
## 3 x x o x o o o o o o o  o  o  o
## 4 x x x o o o o o o o o  o  o  o
## 5 x x o o o o o o o o o  o  o  o
## 6 x o o o o o o o o o o  o  o  o
## 7 x x o o x o o o o o o  o  o  o
```

```
# Arima(2, 1, 1)
acf(arima_211)
```




```
pacf(arima_211)
```



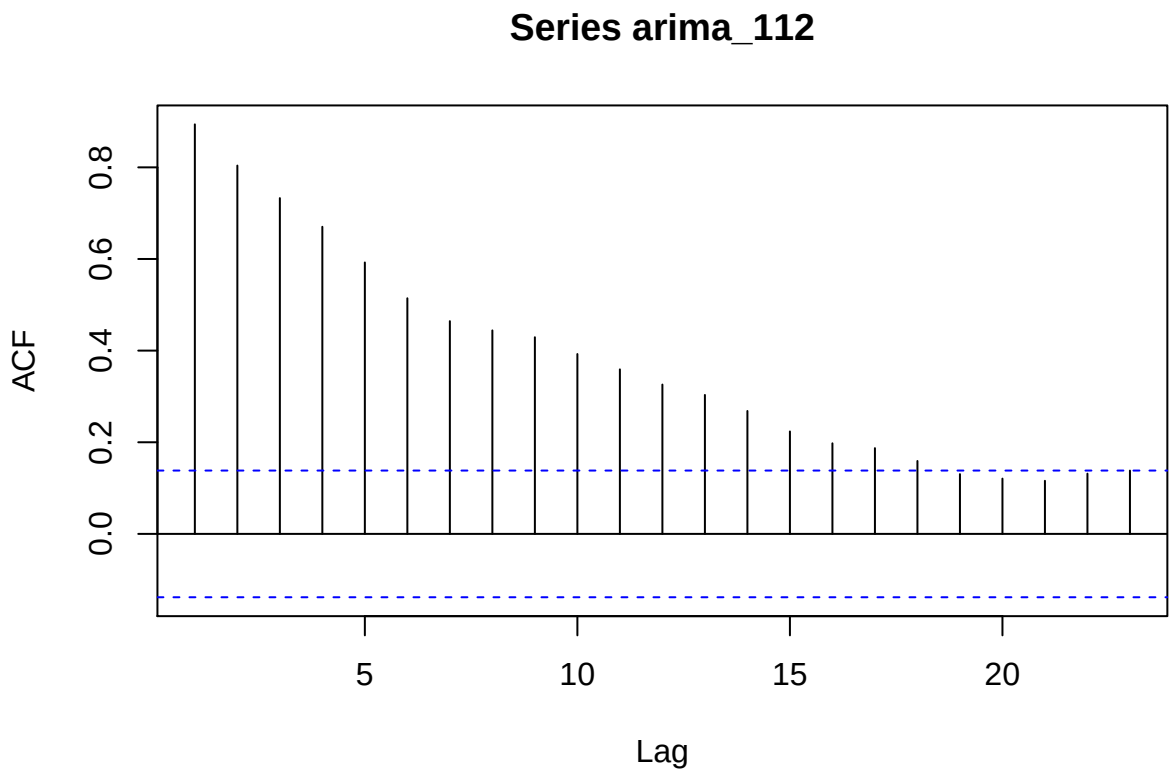

```
print('arima(2, 1, 1)')
```

```
## [1] "arima(2, 1, 1)"
```

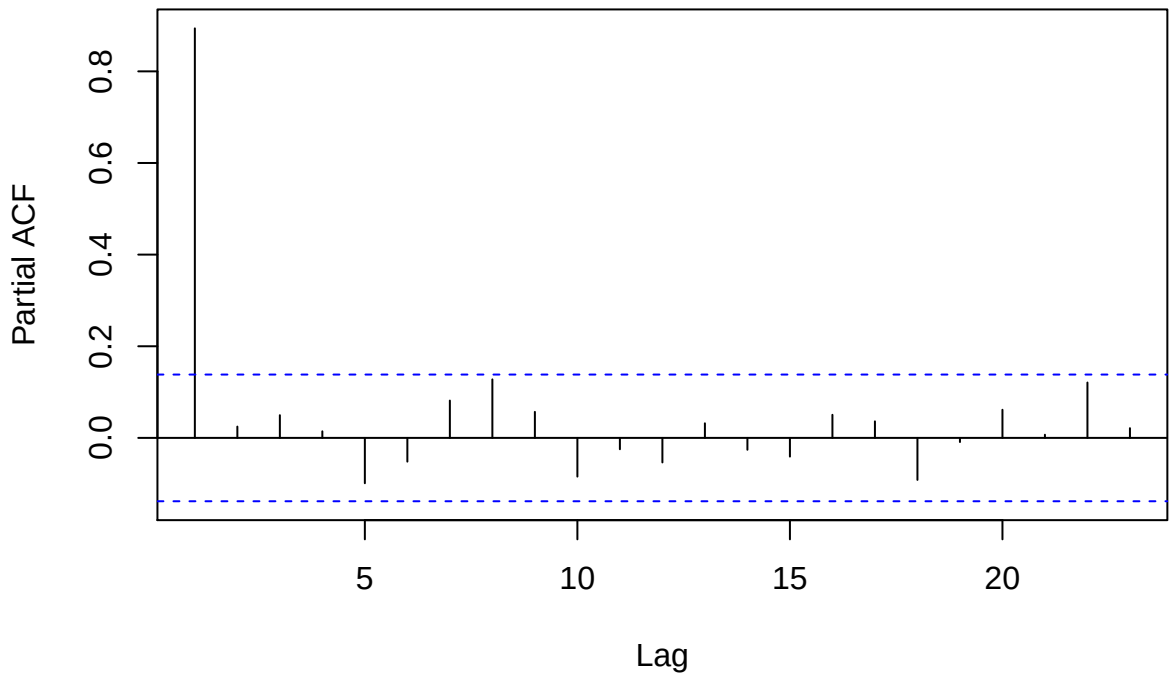
```
eacf(arima_211)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x x x x  x  x  x
## 1 o x x o o o o o o o o  o  o  o
## 2 x o x o o o o o o o o  o  o  o
## 3 x x o o o o o o o o o  o  o  o
## 4 x x o o o o o o o o o  o  o  o
## 5 x x o o o o o o o o o  o  o  o
## 6 x o o o o o o o o o o  o  o  o
## 7 o x x o x x o o o o o  o  o  o
```

```
# Arima(1, 1, 2)
acf(arima_112)
```




```
pacf(arima_112)
```



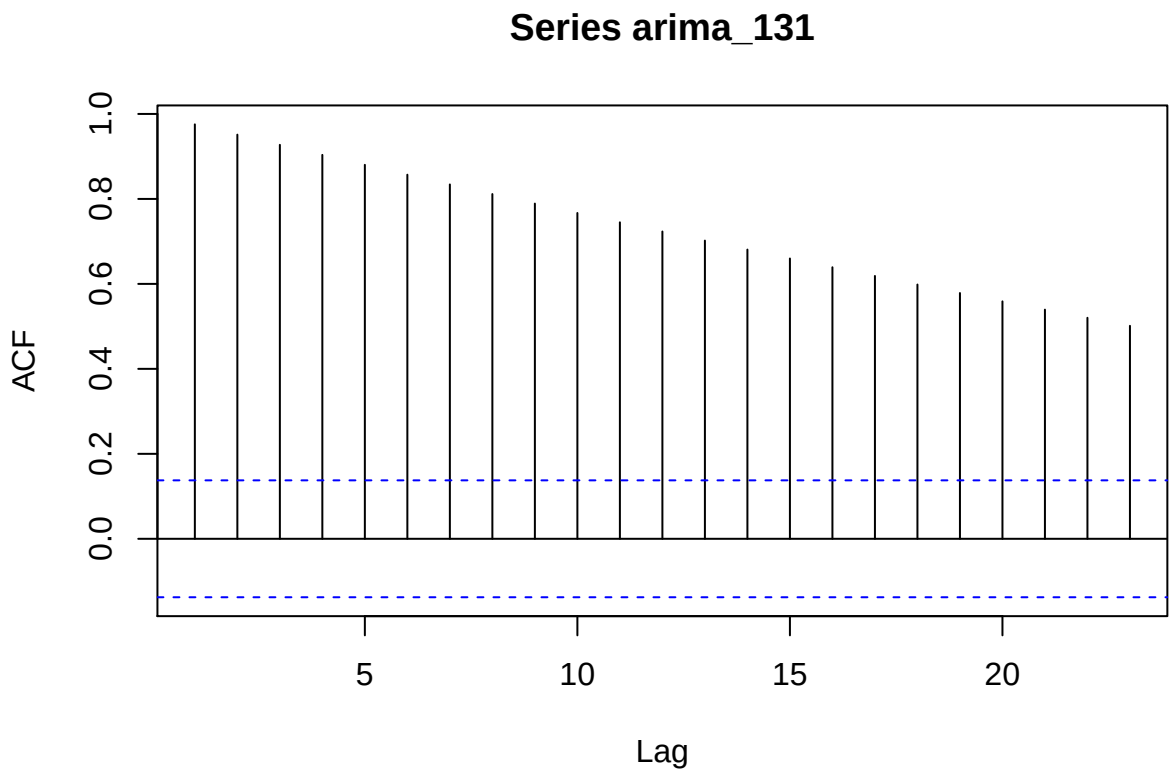

```
print('arima(1, 1, 2)')
```

```
## [1] "arima(1, 1, 2)"
```

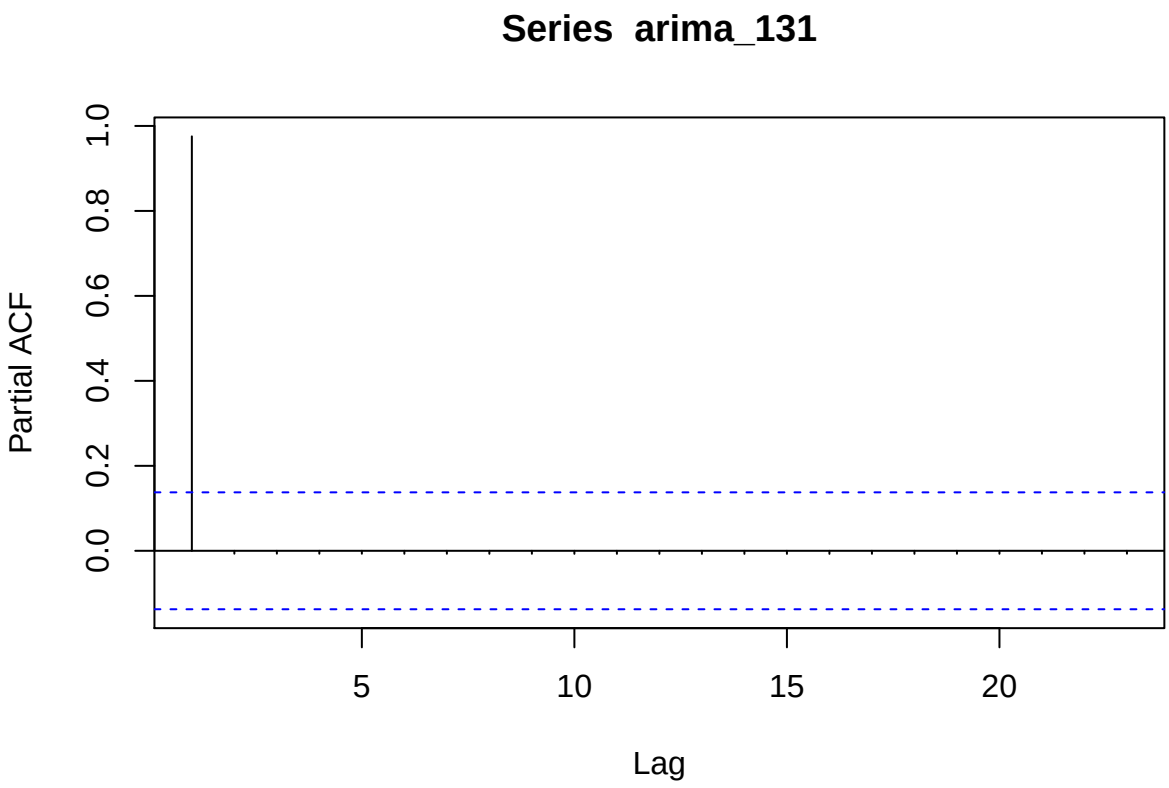
```
eacf(arima_112)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x x x x  x  x  x
## 1 o o o o o o o o o o o  o  o  o
## 2 x o o o o o o o o o o  o  o  o
## 3 x x o o o o o o o o o  o  o  o
## 4 o x x o o o o o o o o  o  o  o
## 5 x x x o o o o o o o o  o  o  o
## 6 x x o o o o o o o o o  o  o  o
## 7 x x o o o o o o o o o  o  o  o
```

```
# Arima(1, 3, 1)
acf(arima_131)
```




```
pacf(arima_131)
```




```
#eacf(arima_131)
```

### Problem 3

```
getSymbols("FB", src = "yahoo")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```
## [1] "FB"
```

```
getSymbols("AAPL", src = "yahoo")
```

```
## [1] "AAPL"
```

```
getSymbols("INTC", src = "yahoo")
```

```
## [1] "INTC"
```

```
getSymbols("IBM", src = "yahoo")
```

```
## [1] "IBM"
```

```
getSymbols("NVDA", src = "yahoo")
```

```
## [1] "NVDA"
```

```
getSymbols("SIRI", src = "yahoo")
```

```
## [1] "SIRI"
```

```
getSymbols("SPOT", src = "yahoo")
```

```
## [1] "SPOT"
```

```
getSymbols("GOOGL", src = "yahoo")
```

```
## [1] "GOOGL"
```

```
getSymbols("AMZN", src = "yahoo")
```

```
## [1] "AMZN"
```

```
getSymbols("WFC", src = "yahoo")
```

```
## [1] "WFC"
```

```
stocks <- list(AAPL, AMZN, FB, GOOGL, IBM, INTC, NVDA, SIRI, SPOT, WFC)

N_points <- 200
window_size <- 99
closing <- lapply(stocks, function(s) {
  l <- dim(s)[1]
  return(s[(l - N_points + 1):l, 4])
})

percent_success <- lapply(1:10, function(i) {
  d <- closing[[i]]

  # Create Holt Winter for every 99 point window
  stock_closing_forecasts <- rollapply(d, window_size, function(w) {

    # Create Holt-Winter Model
    hw <- HoltWinters(w, beta = FALSE, gamma = FALSE)

    # Forecast
    forecasted_value <- forecast(hw, h = 1)$mean

    return(forecasted_value)
  })

  NonNAindex <- which(!is.na(stock_closing_forecasts))
  j <- min(NonNAindex) + 1
  n <- length(stock_closing_forecasts)

  # Keep only NonNA values to assess accuracy
  stock_closing_forecasts <- stock_closing_forecasts[j:n]
  d <- d[j:n]
  m <- length(d)
```

```
  pred_direction <- sign(diff(stock_closing_forecasts))[-1]
  actual_direction <- sign(diff(d))[-1]


  n_success <- sum(pred_direction == actual_direction)
  return(n_success / m)
})

print(percent_success)
```

```
## [[1]]
## [1] 0.9306931
##
## [[2]]
## [1] 0.960396
##
## [[3]]
## [1] 0.950495
##
## [[4]]
## [1] 0.9207921
##
## [[5]]
## [1] 0.980198
##
## [[6]]
## [1] 0.950495
##
## [[7]]
## [1] 0.9207921
##
## [[8]]
## [1] 0.960396
##
## [[9]]
## [1] 0.970297
##
## [[10]]
## [1] 0.980198
```