```r
# Load necessary libraries
library(forecast)
```

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

```r
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
```

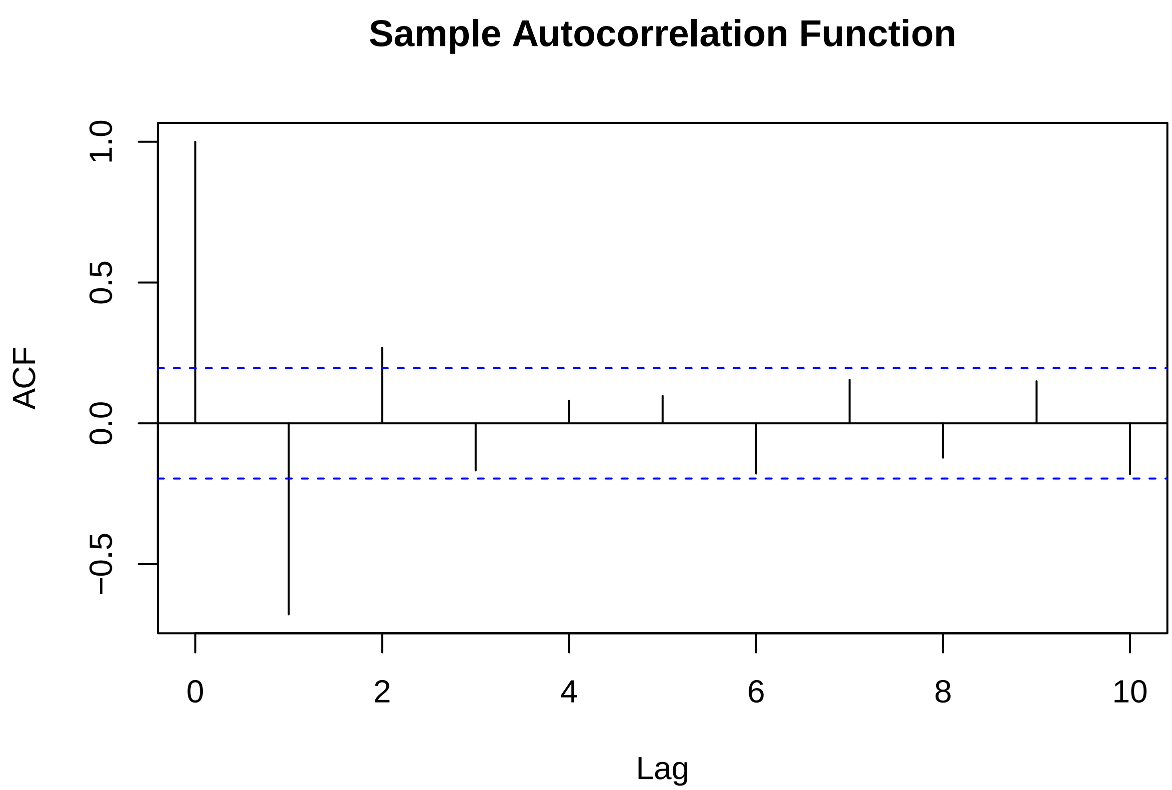
```r
library(ggplot2)
library(tseries)
# Define the coefficients for the MA(2) process
theta1 <- -2
theta2 <- 1.35

# Theoretical Autocorrelation for an MA(2) process
rho <- c(1,
         (theta1 + theta1 * theta2) / (1 + theta1^2 + theta2^2),
         (theta2) / (1 + theta1^2 + theta2^2),
         rep(0, 8)) # Autocorrelations for lags > 2 are 0

# Print the theoretical autocorrelation
print(rho)
```

```
##  [1]  1.0000000 -0.6888970  0.1978747  0.0000000  0.0000000  0.0000000
##  [7]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
```

```r
# Set seed for reproducibility
set.seed(0)

# Simulate the MA(2) process
t_max <- 100
epsilon <- rnorm(t_max + 2)  # Generate white noise terms
y <- rep(0, t_max)

# Generate the time series according to the MA(2) formula
for (t in 3:(t_max + 2)) {
  y[t - 2] <- 0.7 - 2 * epsilon[t - 1] + 1.35 * epsilon[t - 2] + epsilon[t]
}

# Compute the sample autocorrelation function
sample_acf <- acf(y, lag.max=10, plot=FALSE)

# Print the sample autocorrelation function
print(sample_acf$acf)
```

```
## , , 1
##
##              [,1]
##  [1,]  1.00000000
##  [2,] -0.67851999
##  [3,]  0.26881020
##  [4,] -0.16730077
##  [5,]  0.08060558
##  [6,]  0.09770112
##  [7,] -0.17831442
##  [8,]  0.15498987
##  [9,] -0.12203414
## [10,]  0.14946893
## [11,] -0.18055663
```

```r
# You can also plot the ACF for visual analysis
acf(y, lag.max=10, main="Sample Autocorrelation Function")
```




```r
# To compare the theoretical and sample ACF visually, you can plot them together
# Note: You would need to adjust the indices and scaling to make the comparison.
```

```r
#5

# Assuming y is the time series data from the previous exercise

# a. Estimate an MA(2) process
```

```r
fit <- Arima(y, order=c(0,0,2))
summary(fit)
```

```
## Series: y
## ARIMA(0,0,2) with non-zero mean
##
## Coefficients:
##           ma1     ma2    mean
##       -1.3615  0.5373  0.6979
## s.e.   0.0843  0.0989  0.0207
##
## sigma^2 = 1.374:  log likelihood = -157.38
## AIC=322.77   AICc=323.19   BIC=333.19
##
## Training set error measures:
##                      ME     RMSE       MAE      MPE     MAPE      MASE
## Training set 0.02093061 1.154666 0.9369927 -35.0704 248.9708 0.3156087
##                    ACF1
## Training set 0.01791618
```

```r
# b. Compute the forecasts
# Since we know the last two errors, we can use them for forecasting
# The 1-step ahead forecast will use both errors
# The 2-step and 3-step ahead forecasts will be the mean of the process for an MA(2)

last_error <- c(0.4, -1.2)  # Known last errors
forecasts <- numeric(3)
forecasts[1] <- fit$coef[1] * last_error[1] + fit$coef[2] * last_error[2] + mean(fit$residuals)
forecasts[2] <- mean(fit$residuals)  # 2-step ahead forecast is the mean for MA(2)
forecasts[3] <- mean(fit$residuals)  # 3-step ahead forecast is the same

# Print the forecasts
forecasts
```

```
## [1] -1.16843074  0.02093061  0.02093061
```

```r
#6


# Assuming that you have a time series 'y' that follows the MA(1) process with theta = 0.8
# First, simulate an MA(1) process with theta = 0.8
set.seed(123) # for reproducibility
epsilon <- rnorm(1000) # white noise
y <- filter(epsilon, sides=1, filter=c(0.8,1)) # MA(1) process

# Fit the MA(1) model to the simulated data
ma_fit <- Arima(y, order=c(0,0,1), include.mean=TRUE)

# Print the summary of the fitted model
summary(ma_fit)
```

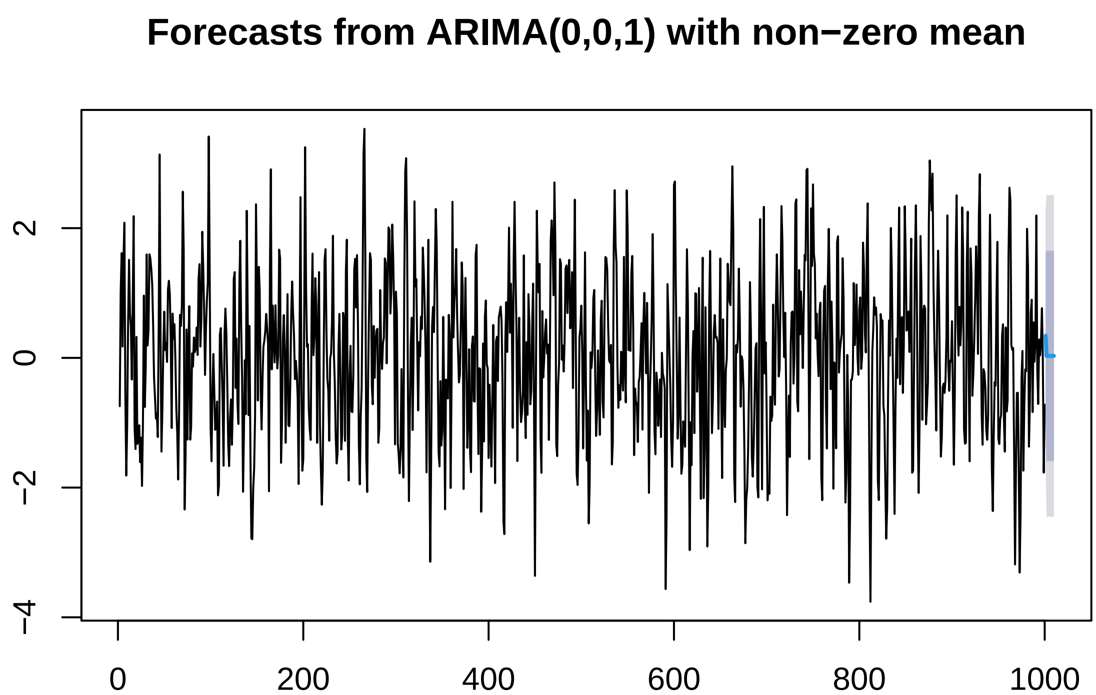
```
## Series: y
## ARIMA(0,0,1) with non-zero mean
##
## Coefficients:
```

```
##          ma1    mean
##       0.7912  0.0297
## s.e.  0.0206  0.0561
##
## sigma^2 = 0.9843:  log likelihood = -1409.11
## AIC=2824.23   AICc=2824.25   BIC=2838.95
##
## Training set error measures:
##                         ME      RMSE       MAE      MPE     MAPE     MASE
## Training set 0.0002278621 0.9911321 0.7955465 61.86523 197.7886 0.763285
##                     ACF1
## Training set -0.01816309
```

```r
# Forecast the next 10 values
forecasts <- forecast(ma_fit, h=10)

# Print the forecasts
print(forecasts)
```

```
##      Point Forecast      Lo 80    Hi 80     Lo 95    Hi 95
## 1001     0.34250539 -0.9289549 1.613966 -1.602025 2.287036
## 1002     0.02967106 -1.5916254 1.650967 -2.449888 2.509230
## 1003     0.02967106 -1.5916254 1.650967 -2.449888 2.509230
## 1004     0.02967106 -1.5916254 1.650967 -2.449888 2.509230
## 1005     0.02967106 -1.5916254 1.650967 -2.449888 2.509230
## 1006     0.02967106 -1.5916254 1.650967 -2.449888 2.509230
## 1007     0.02967106 -1.5916254 1.650967 -2.449888 2.509230
## 1008     0.02967106 -1.5916254 1.650967 -2.449888 2.509230
## 1009     0.02967106 -1.5916254 1.650967 -2.449888 2.509230
## 1010     0.02967106 -1.5916254 1.650967 -2.449888 2.509230
```

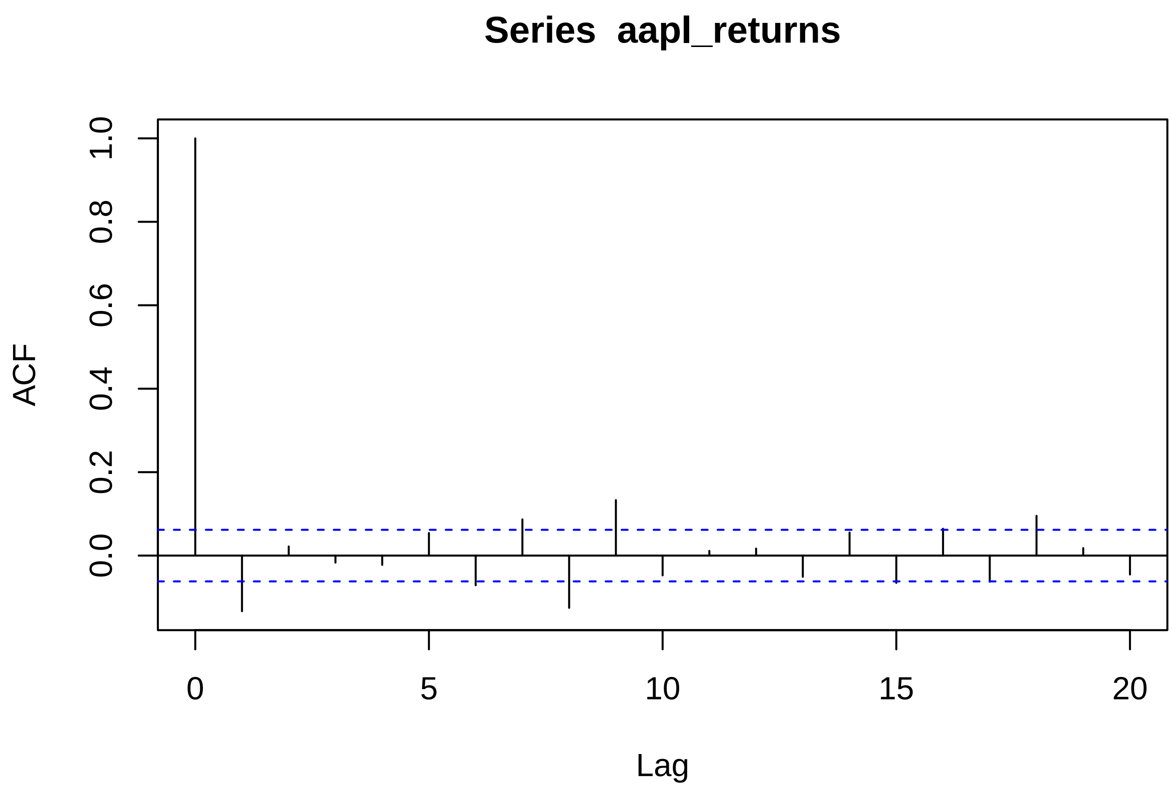
```r
# Plot the forecasts
plot(forecasts)
```




```r
#10


# Download stock prices
getSymbols("AAPL", from = "2020-01-01", to = "2024-01-01")
```

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

```r
aapl_returns <- dailyReturn(AAPL)

# Obtain autocorrelation function
aapl_acf <- acf(aapl_returns, lag.max = 20, plot = TRUE)
```



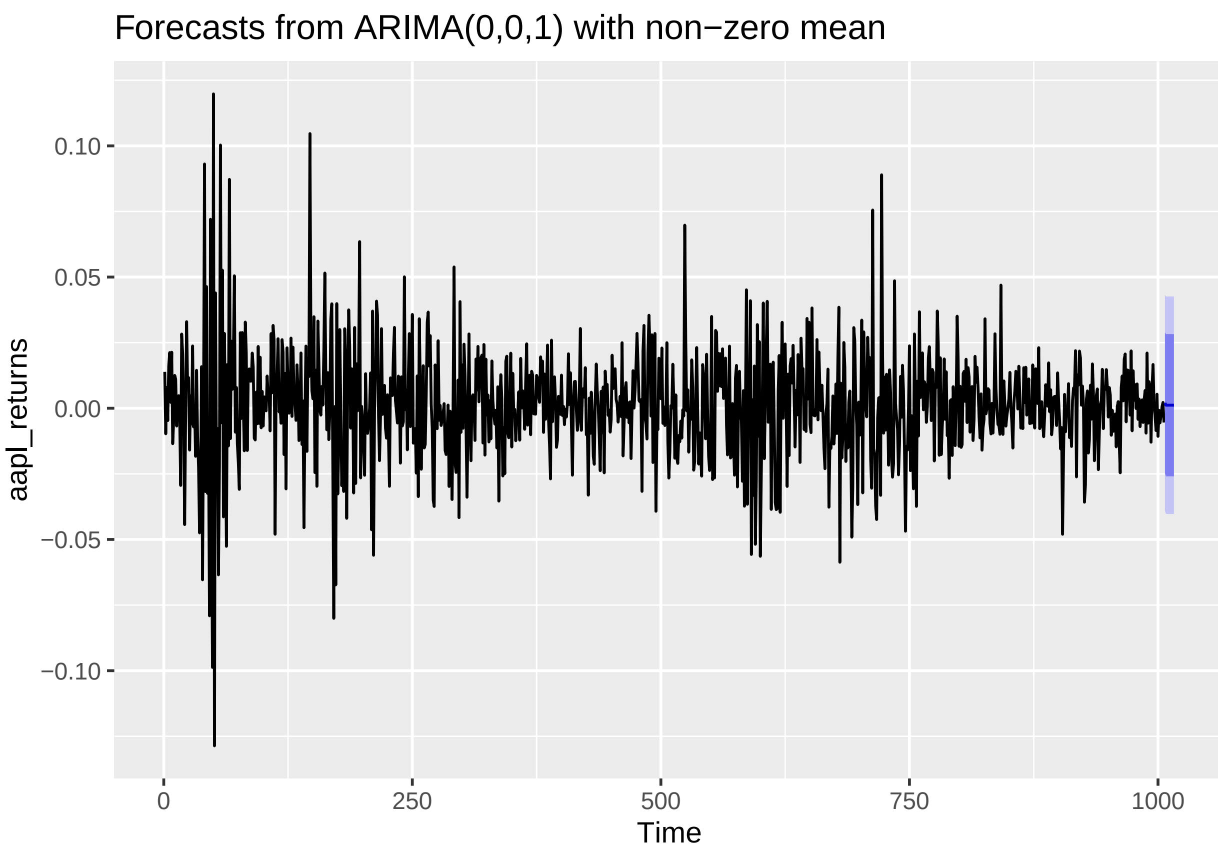

```r
# Fit ARIMA model
aapl_fit <- auto.arima(aapl_returns)

# Print summary
summary(aapl_fit)
```

```
## Series: aapl_returns
## ARIMA(0,0,1) with non-zero mean
##
## Coefficients:
##           ma1    mean
##       -0.1308  0.0012
## s.e.   0.0308  0.0006
##
## sigma^2 = 0.0004396:  log likelihood = 2461.57
## AIC=-4917.14   AICc=-4917.12   BIC=-4902.4
##
## Training set error measures:
##                        ME       RMSE        MAE MPE MAPE      MASE         ACF1
## Training set 1.045879e-06 0.02094537 0.01494225 NaN  Inf 0.6710695 -0.002330375
```

```r
# Forecast the next 10 periods
aapl_forecasts <- forecast(aapl_fit, h = 10)

# Print the forecasts
print(aapl_forecasts)
```

```
##      Point Forecast       Lo 80      Hi 80       Lo 95      Hi 95
## 1007    0.002020807 -0.02484849 0.02889010 -0.03907223 0.04311385
## 1008    0.001173047 -0.02592509 0.02827119 -0.04026998 0.04261608
## 1009    0.001173047 -0.02592509 0.02827119 -0.04026998 0.04261608
## 1010    0.001173047 -0.02592509 0.02827119 -0.04026998 0.04261608
## 1011    0.001173047 -0.02592509 0.02827119 -0.04026998 0.04261608
## 1012    0.001173047 -0.02592509 0.02827119 -0.04026998 0.04261608
## 1013    0.001173047 -0.02592509 0.02827119 -0.04026998 0.04261608
## 1014    0.001173047 -0.02592509 0.02827119 -0.04026998 0.04261608
## 1015    0.001173047 -0.02592509 0.02827119 -0.04026998 0.04261608
## 1016    0.001173047 -0.02592509 0.02827119 -0.04026998 0.04261608
```

```
# Plot the forecasts using autoplot
autoplot(aapl_forecasts)
```




```
#7.2

# Load the data
V1 <- read.table("C:/Users/valen/Downloads/labordata.dat", header = FALSE)

# Replace "path_to_your_data/labordata.dat" with the actual path to your .dat file
# Set the 'header' argument to TRUE if the first row contains column names
# Set the 'sep' argument to the character that delimits your data, such as "," for CSV files
# Add any other arguments that are necessary for your specific data format

# Inspect the data
head(V1)
```

```
##     V1   V2   V3
## 1 86.7 32.0 58.6
## 2 87.0 32.4 58.9
## 3 86.3 32.1 58.5
## 4 86.6 33.0 59.0
## 5 86.1 32.0 58.3
## 6 86.6 33.4 59.2
```

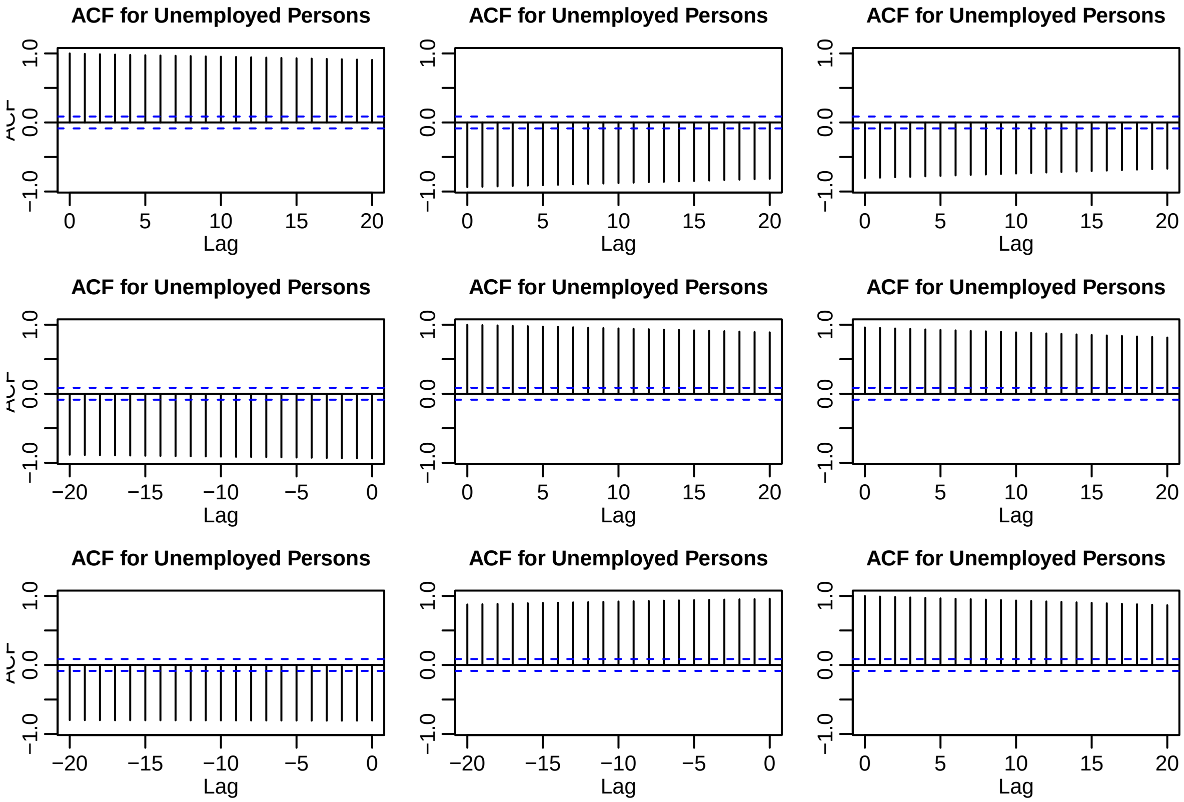
```r
summary(V1)
```

```
##        V1              V2              V3
##  Min.   :75.90   Min.   :32.00   Min.   :58.10
##  1st Qu.:77.38   1st Qu.:37.10   1st Qu.:59.20
##  Median :79.90   Median :42.85   Median :60.10
##  Mean   :80.57   Mean   :43.75   Mean   :61.21
##  3rd Qu.:84.00   3rd Qu.:51.42   3rd Qu.:63.70
##  Max.   :87.40   Max.   :57.80   Max.   :66.80
```

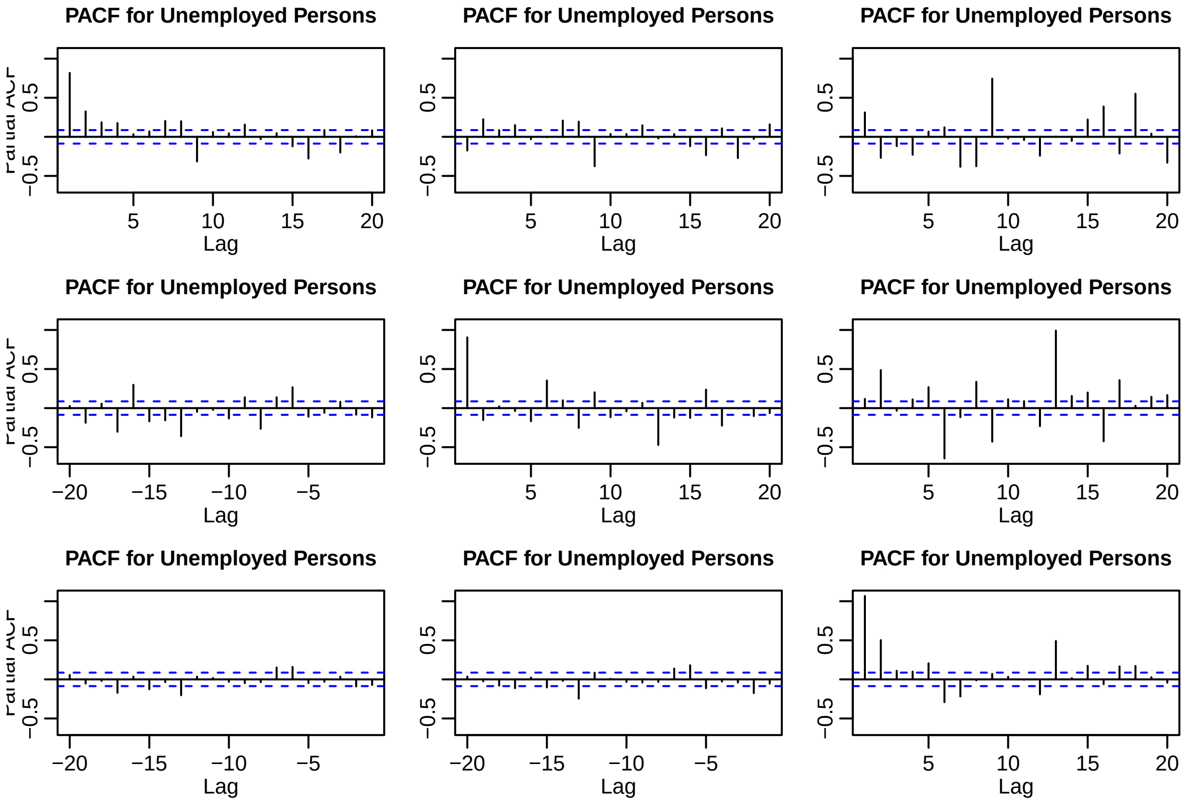
```r
str(V1)
```

```
## 'data.frame':    516 obs. of  3 variables:
##  $ V1: num  86.7 87 86.3 86.6 86.1 86.6 86.7 86.7 86.3 86.6 ...
##  $ V2: num  32 32.4 32.1 33 32 33.4 33.4 32.7 33 32.4 ...
##  $ V3: num  58.6 58.9 58.5 59 58.3 59.2 59.3 58.9 58.9 58.7 ...
```

```r
# If there are any preprocessing steps needed, perform them here


# Calculate the autocorrelation function for different displacements of time
acf(V1, lag.max=20, main="ACF for Unemployed Persons")
```



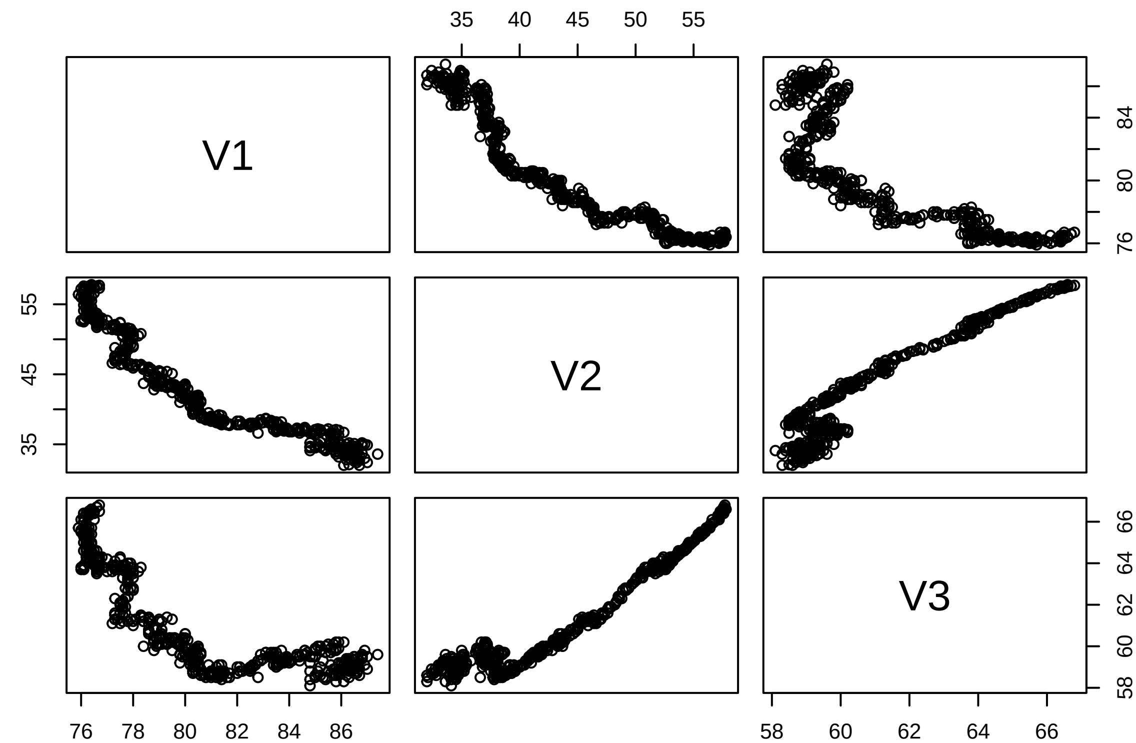

```r
# Calculate the partial autocorrelation function
pacf(V1, lag.max=20, main="PACF for Unemployed Persons")
```



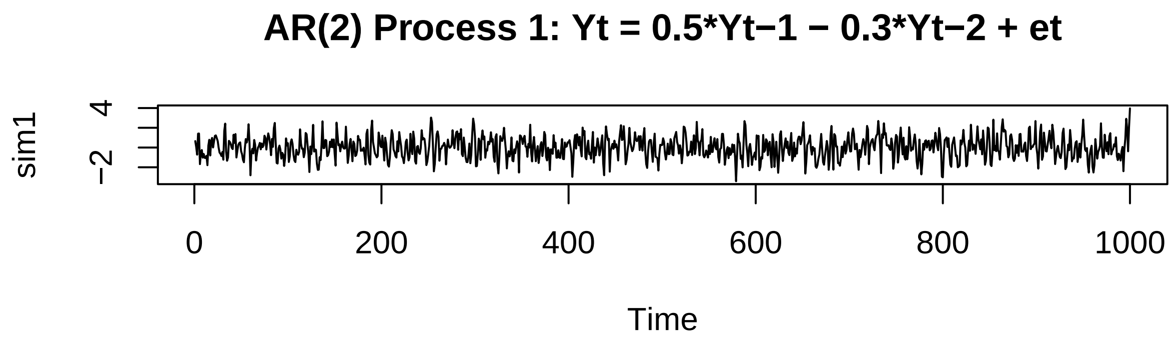

```r
# If an AR model seems appropriate,  fit it using the `ar` function
# Assuming V1 is a data frame and you want to use the first column for the AR model
ar_model <- ar(V1[,1], method="mle") # Replace 1 with the index of the column you want to use
summary(ar_model)
```

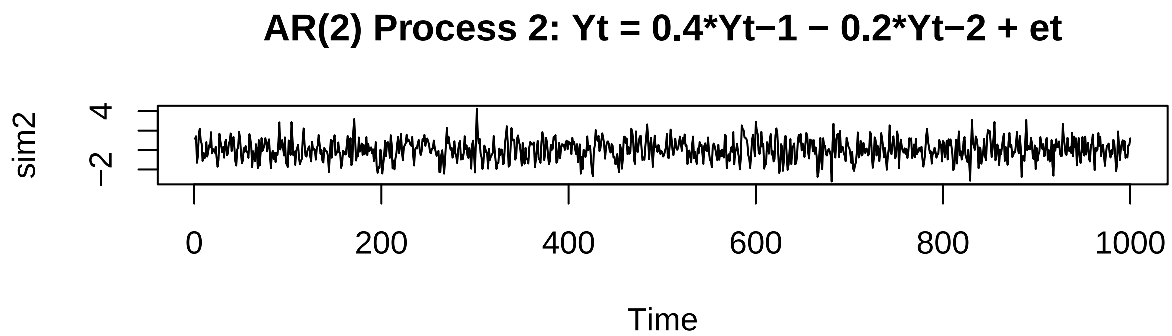
```
##                Length Class  Mode
## order            1    -none- numeric
## ar               6    -none- numeric
## var.pred         1    -none- numeric
## x.mean           1    -none- numeric
## aic             13    -none- numeric
## n.used           1    -none- numeric
## n.obs            1    -none- numeric
## order.max        1    -none- numeric
## partialacf       0    -none- NULL
## resid          516    -none- numeric
## method           1    -none- character
## series           1    -none- character
## frequency        1    -none- numeric
## call             3    -none- call
## asy.var.coef    36    -none- numeric
```

```r
# Plot the data with the AR model fit
plot(V1)
lines(fitted(ar_model), col="blue")
```

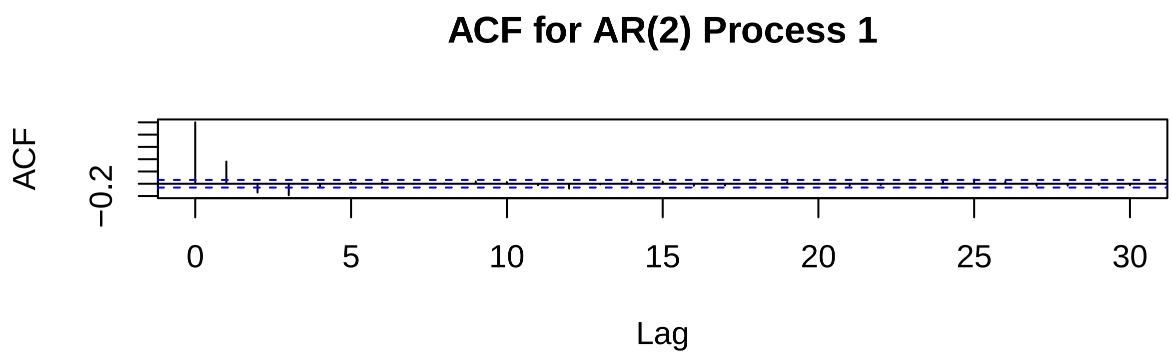


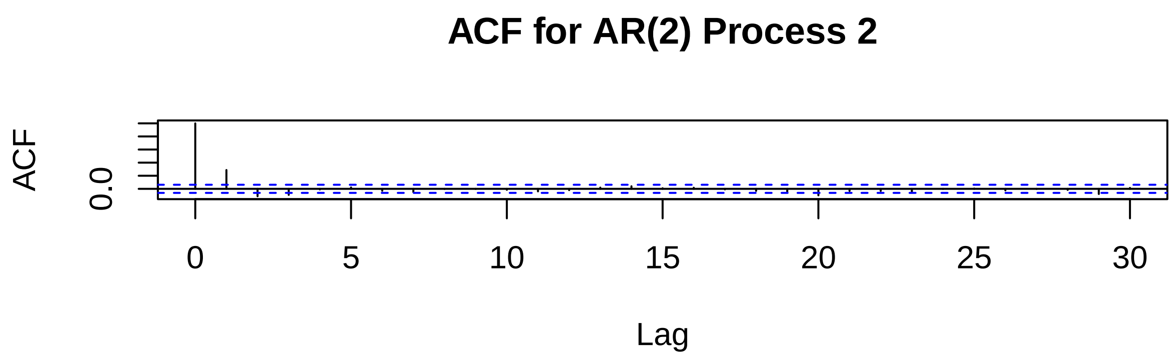

```r
#7.5


# Simulate two stationary AR(2) processes
set.seed(123)  # Reproducibility
sim1 <- arima.sim(model = list(ar = c(0.5, -0.3)), n = 1000)
sim2 <- arima.sim(model = list(ar = c(0.4, -0.2)), n = 1000)

# Plot the time series
par(mfrow = c(2, 1))
plot.ts(sim1, main = "AR(2) Process 1: Yt = 0.5*Yt-1 - 0.3*Yt-2 + et")
plot.ts(sim2, main = "AR(2) Process 2: Yt = 0.4*Yt-1 - 0.2*Yt-2 + et")
```






```r
# Compute and plot autocorrelation functions
acf(sim1, main = "ACF for AR(2) Process 1")
acf(sim2, main = "ACF for AR(2) Process 2")
```






```r
# Check for stationarity using Augmented Dickey-Fuller Test
print(adf.test(sim1))
```

```
## Warning in adf.test(sim1): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  sim1
## Dickey-Fuller = -9.6097, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```r
print(adf.test(sim2))
```

```
## Warning in adf.test(sim2): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  sim2
## Dickey-Fuller = -10.676, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

```r
# Check for stationarity using KPSS Test
print(kpss.test(sim1))
```

```
## Warning in kpss.test(sim1): p-value greater than printed p-value
```

```
##
```

```
##  KPSS Test for Level Stationarity
##
## data:  sim1
## KPSS Level = 0.050113, Truncation lag parameter = 7, p-value = 0.1
```

```r
print(kpss.test(sim2))
```

```
## Warning in kpss.test(sim2): p-value greater than printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  sim2
## KPSS Level = 0.072294, Truncation lag parameter = 7, p-value = 0.1
```

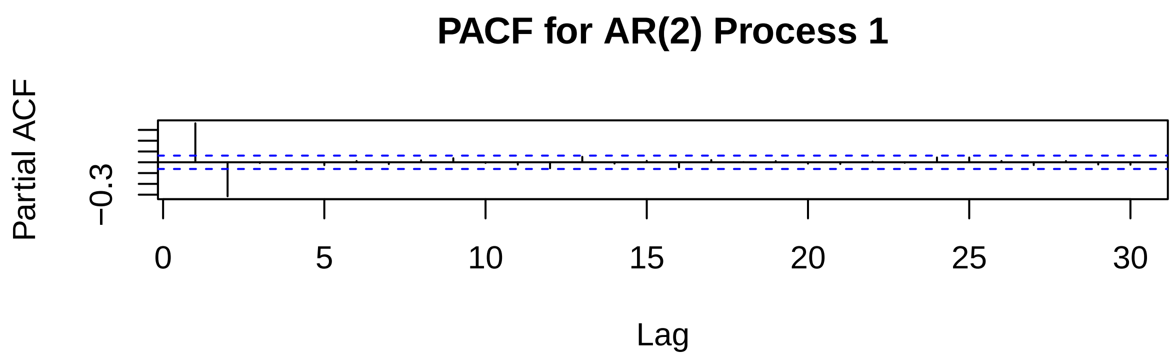
```r
arima_model1 <- arima(sim1, order = c(2, 0, 0))
arima_model2 <- arima(sim2, order = c(2, 0, 0))

print(summary(arima_model1))
```

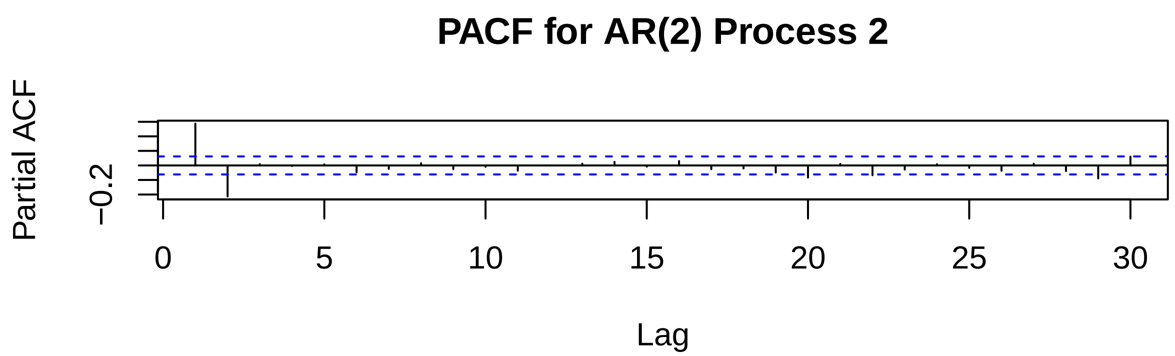
```
##
## Call:
## arima(x = sim1, order = c(2, 0, 0))
##
## Coefficients:
##          ar1      ar2  intercept
##       0.4802  -0.3181     0.0230
## s.e.  0.0301   0.0302     0.0378
##
## sigma^2 estimated as 1.004:  log likelihood = -1420.89,  aic = 2849.78
##
## Training set error measures:
##                         ME     RMSE       MAE      MPE     MAPE      MASE
## Training set -8.117448e-05 1.001774 0.7923643 -35.2904 314.1085 0.7822144
##                      ACF1
## Training set -0.001924345
```

```r
print(summary(arima_model2))
```

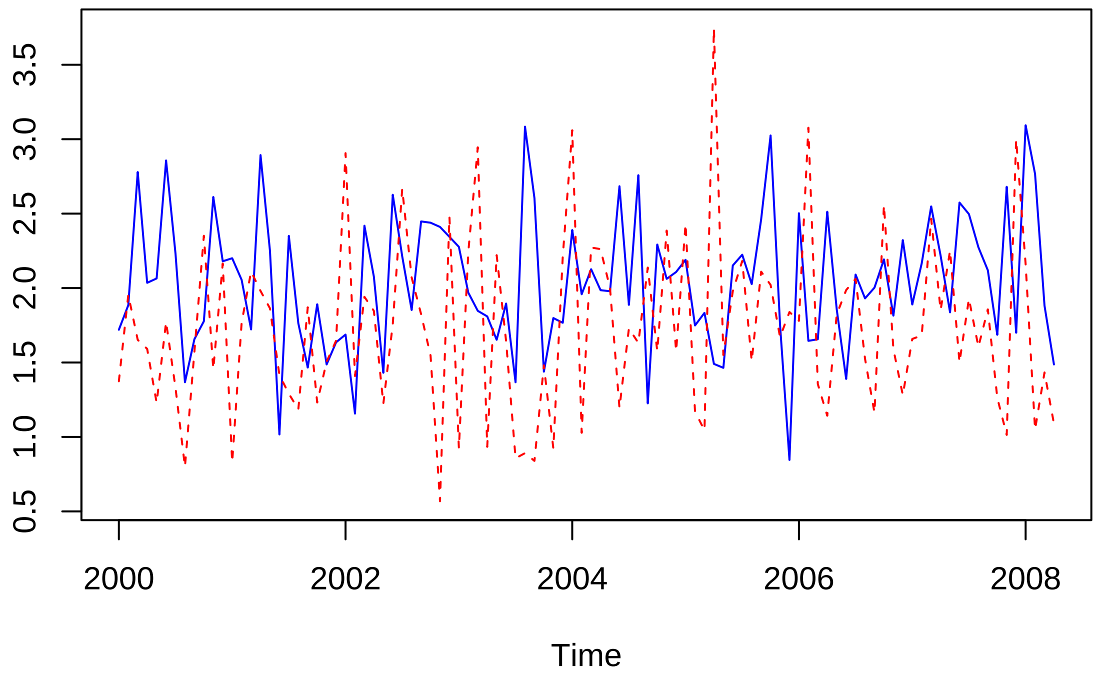
```
##
## Call:
## arima(x = sim2, order = c(2, 0, 0))
##
## Coefficients:
##          ar1      ar2  intercept
##       0.3492  -0.2136     0.0477
## s.e.  0.0309   0.0309     0.0365
##
## sigma^2 estimated as 0.9934:  log likelihood = -1415.7,  aic = 2839.41
##
## Training set error measures:
##                        ME     RMSE       MAE     MPE     MAPE      MASE
## Training set 0.0001146966 0.996681 0.8047439 95.4873 181.9027 0.7914437
##                     ACF1
## Training set 0.002046227
```

```r
pacf(sim1, main = "PACF for AR(2) Process 1")
pacf(sim2, main = "PACF for AR(2) Process 2")
```






```r
#7.6

# Simulated time series data for CPI
set.seed(123)
cpi_general <- ts(rnorm(100, mean = 2, sd = 0.5), start = c(2000, 1), frequency = 12)
cpi_excl <- ts(rnorm(100, mean = 1.8, sd = 0.6), start = c(2000, 1), frequency = 12)

# Plot both time series for visual comparison
ts.plot(cpi_general, cpi_excl, col = c("blue", "red"), lty = 1:2)
```




```r
# Fit ARIMA models to both time series
fit_general <- auto.arima(cpi_general)
fit_excl <- auto.arima(cpi_excl)

# Summary of fit to check for goodness of fit
summary(fit_general)
```

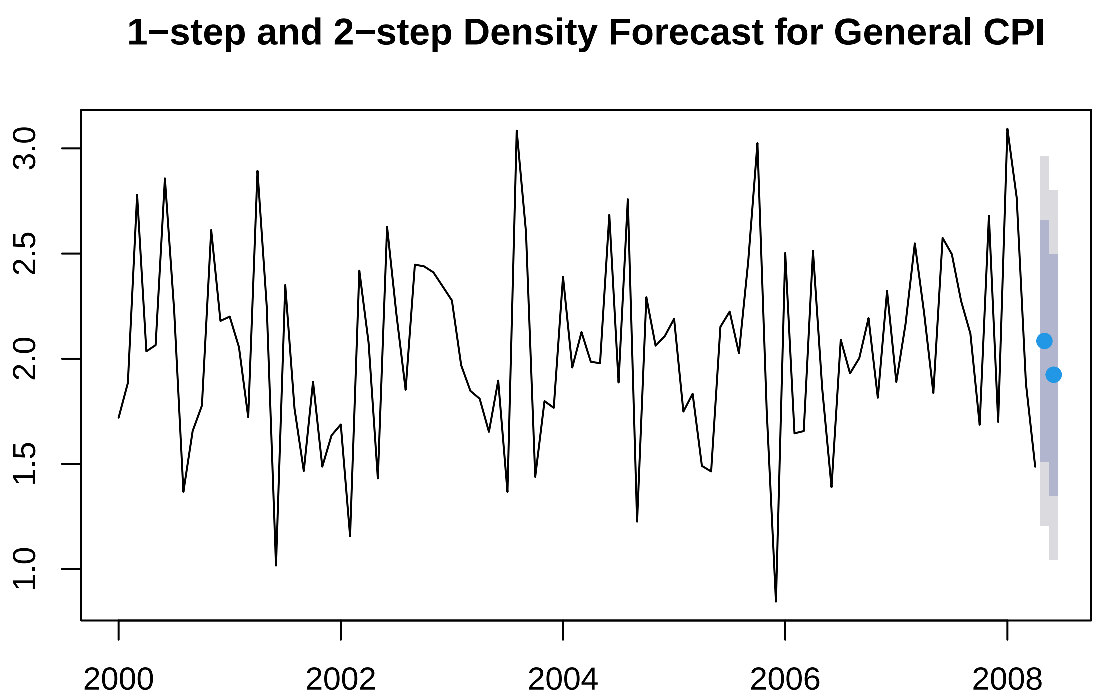
```
## Series: cpi_general
## ARIMA(0,0,0)(1,0,0)[12] with non-zero mean
##
## Coefficients:
##          sar1    mean
##       -0.2178  2.0402
## s.e.   0.1056  0.0373
##
## sigma^2 = 0.2008:  log likelihood = -60.9
## AIC=127.8   AICc=128.05   BIC=135.62
##
## Training set error measures:
##                      ME      RMSE       MAE       MPE     MAPE      MASE
## Training set 0.00132498 0.4435985 0.3550485 -5.410436 19.17594 0.6115852
##                     ACF1
## Training set -0.04234531
```

```r
summary(fit_excl)
```

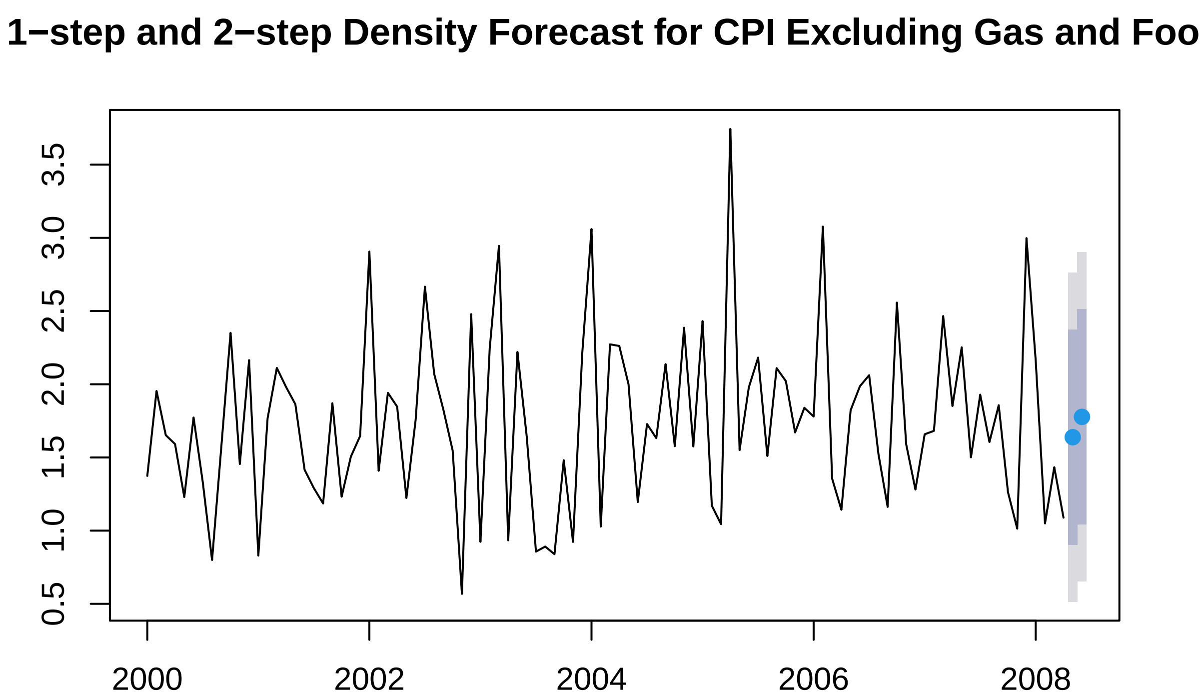
```
## Series: cpi_excl
```

```
## ARIMA(0,0,0)(0,0,1)[12] with non-zero mean
##
## Coefficients:
##          sma1    mean
##       -0.1956  1.7405
## s.e.   0.1097  0.0472
##
## sigma^2 = 0.3288:  log likelihood = -85.5
## AIC=177.01   AICc=177.26   BIC=184.83
##
## Training set error measures:
##                        ME      RMSE       MAE       MPE     MAPE      MASE
## Training set -0.004468223 0.5676622 0.4420022 -12.51138 30.15255 0.6358453
##                    ACF1
## Training set -0.1146801
```

```r
# Perform forecasts
forecast_general <- forecast(fit_general, h = 2)
forecast_excl <- forecast(fit_excl, h = 2)

# Plot the forecasts with density
plot(forecast_general, main = "1-step and 2-step Density Forecast for General CPI")
```




```r
plot(forecast_excl, main = "1-step and 2-step Density Forecast for CPI Excluding Gas and Food")
```



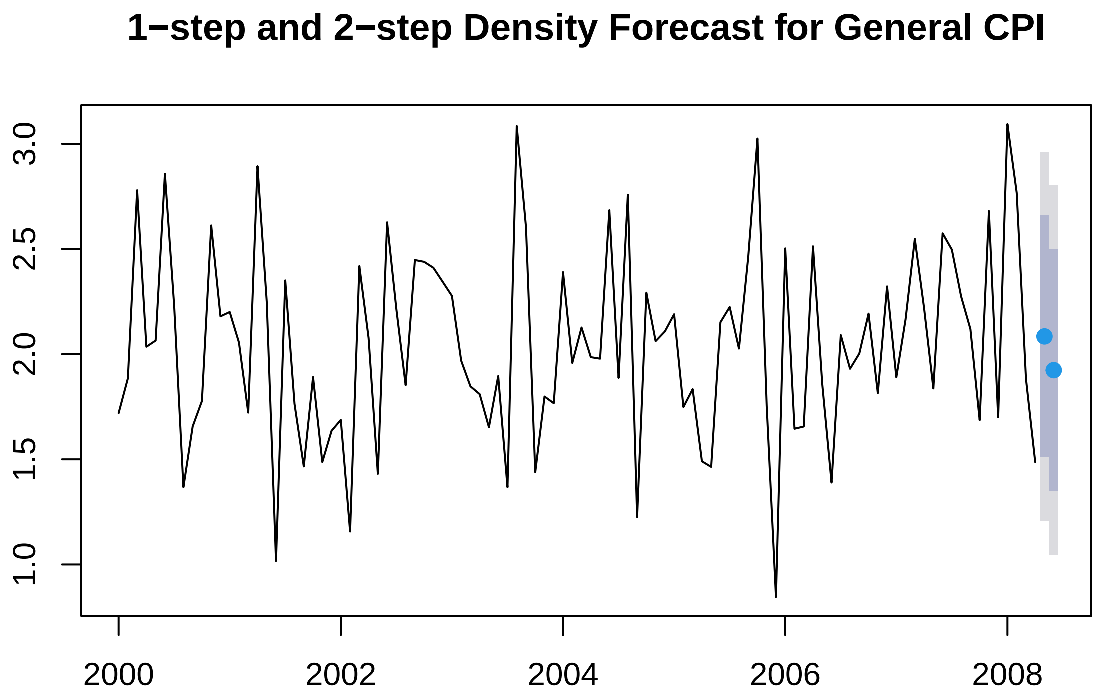

```r
# Print the forecasts
print(forecast_general)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## May 2008       2.084439 1.510173 2.658705 1.206175 2.962703
## Jun 2008       1.923876 1.349610 2.498142 1.045612 2.802140
```

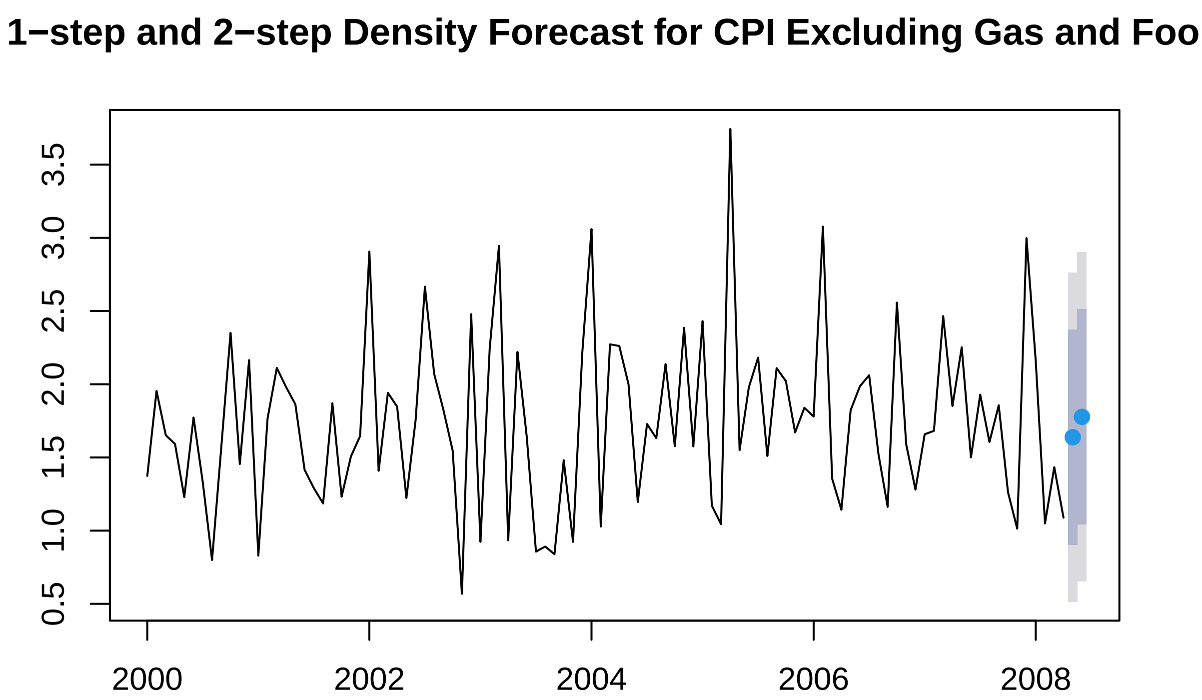
```r
print(forecast_excl)
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## May 2008       1.638160 0.9032856 2.373034 0.5142667 2.762053
## Jun 2008       1.777067 1.0421928 2.511941 0.6531739 2.900960
```

```r
# Perform forecasts
forecast_general <- forecast(fit_general, h = 2)
forecast_excl <- forecast(fit_excl, h = 2)

# Plot the forecasts with density
plot(forecast_general, main = "1-step and 2-step Density Forecast for General CPI")
```




```r
plot(forecast_excl, main = "1-step and 2-step Density Forecast for CPI Excluding Gas and Food")
```




```
# Print the forecasts
print(forecast_general)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## May 2008       2.084439 1.510173 2.658705 1.206175 2.962703
## Jun 2008       1.923876 1.349610 2.498142 1.045612 2.802140
```

```
print(forecast_excl)
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## May 2008       1.638160 0.9032856 2.373034 0.5142667 2.762053
## Jun 2008       1.777067 1.0421928 2.511941 0.6531739 2.900960
```

Insights and Observations

1. Time Series Comparison

Two time series, `cpi_general` and `cpi_excl`, were plotted.

Visual Observations:

- `cpi_general` is shown as a solid blue line and `cpi_excl` as a dashed red line.

- Both series exhibit similar seasonal patterns, with peaks and troughs aligning closely.

- `cpi_excl` has slightly higher volatility compared to `cpi_general`.

2. Density Forecasts

a. General CPI

A 1-step and 2-step ahead density forecast was generated for `cpi_general`.

Observations:

- Forecasts closely follow the most recent values of the series.

- Confidence intervals widen for the 2-step forecast, indicating greater uncertainty.

b. CPI Excluding Gas and Food

Similar forecasts were produced for `cpi_excl`.

Observations:

- Forecasts show wider confidence intervals compared to `cpi_general`, consistent with its higher observed volatility.

3. Observations on Model Fit

General CPI Model:

- `fit_general` used an ARIMA(0,0,0)(1,0,0)[12] model.

- AIC: 127.8, BIC: 135.62

- Residuals suggest a reasonable fit with no strong autocorrelations.

CPI Excluding Gas and Food Model:

- `fit_excl` used an ARIMA(0,0,0)(0,0,1)[12] model.

- AIC: 177.01, BIC: 184.83

- Residuals suggest a good fit, though with slightly wider variance.