Econ 144: Homework #3

Problem 1.

Consider the following MA(2) process:

$$y_t = 0.7 - 2\varepsilon_{t-1} + 1.35\varepsilon_{t-2} + \varepsilon_t$$

1a. Obtain the theoretical autocorrelation function up to lag 10.

The theoretical autocorrelations are:

$$\rho_1 = \frac{\theta_1\theta_2 + \theta_1}{1 + \theta_1^2 + \theta_2^2} = ((-2) * 1.35 - 2)/(1 + (-2)^2) + 1.35^2) = -0.688$$

$$\rho_2 = \frac{\theta_2}{1 + \theta_1^2 + \theta_2^2} = 1.35/(1 + (-2)^2) + 1.35^2) = 0.197$$

$$\rho_3 = \rho_4 = ... = \rho_{10} = 0.$$

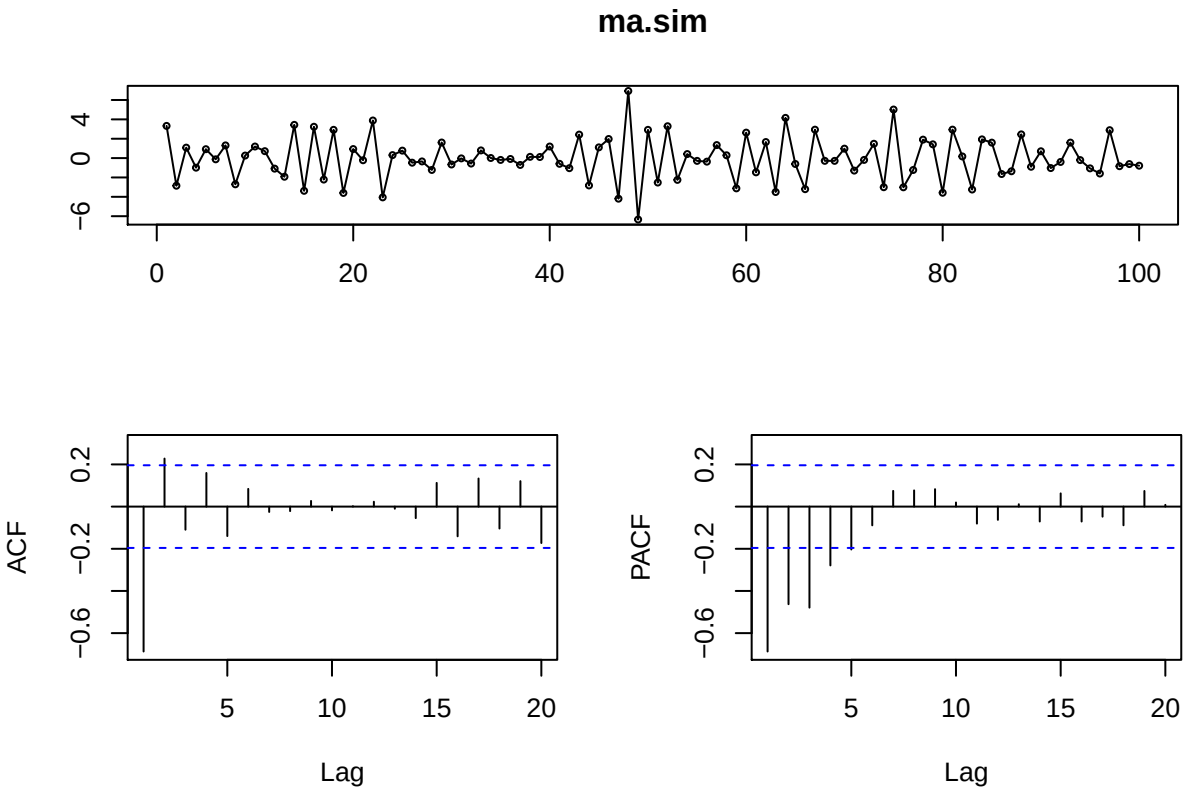
1b. Simulate the process for t = 1, 2, . . . , 100, and compute the sample autocorrelation function up to lag 10.

```
ma.sim <- arima.sim(list(order = c(0, 0, 2), ma = c(-2, 1.35)),
    n = 100)
ma.sim
```

```
## Time Series:
## Start = 1
## End = 100
## Frequency = 1
##   [1]  3.32342540 -2.85173546  1.07849130 -0.98870669  0.92097291
##   [6] -0.12566431  1.30954255 -2.70665504  0.26265981  1.19537221
##  [11]  0.71702436 -1.08910967 -1.92531747  3.43044709 -3.37974592
##  [16]  3.23506776 -2.21803653  2.91876887 -3.60010693  0.93422217
##  [21] -0.22358205  3.88300985 -4.05554604  0.30964671  0.77129855
##  [26] -0.47488459 -0.34404222 -1.22399075  1.60903640 -0.66185022
##  [31] -0.03029519 -0.56572567  0.79401090  0.01016819 -0.18982322
##  [36] -0.09066561 -0.70808950  0.12969928  0.11238815  1.19485797
##  [41] -0.59144151 -1.04177673  2.42584533 -2.82480741  1.10595193
##  [46]  1.97080175 -4.19260209  6.92889668 -6.34163411  2.91677259
##  [51] -2.52453423  3.29209206 -2.25005510  0.41601294 -0.28664795
##  [56] -0.36714245  1.34339052  0.28951778 -3.12594091  2.63004105
##  [61] -1.47035609  1.65793013 -3.50190589  4.15295428 -0.60721886
```

```
##  [66] -3.21036305  2.93658470 -0.28365090 -0.28571941  0.97371742
##  [71] -1.29081049 -0.18812411  1.47873023 -3.00808072  5.01433346
##  [76] -3.01562212 -1.22864172  1.90594938  1.42144113 -3.57899439
##  [81]  2.93889437  0.17182013 -3.24685595  1.93278334  1.59678017
##  [86] -1.64300550 -1.36136030  2.44778585 -0.89564401  0.70470607
##  [91] -1.02866715 -0.40039652  1.59904347 -0.20739963 -1.05919658
##  [96] -1.59363813  2.88449521 -0.83738966 -0.61765412 -0.78494030
```

```
tsdisplay(ma.sim)
```



There will be some fluctuation in terms of what we observe, based upon the seed we are using to generate these random simulations. Additionally, we are using a pretty small sample size ($n = 100$) in the simulation. If we were to use a larger sample size (i.e., $n = 1000$), we would probably observe values in the ACF/PACF's that are closer to that of the theoretical values.

Problem 2.

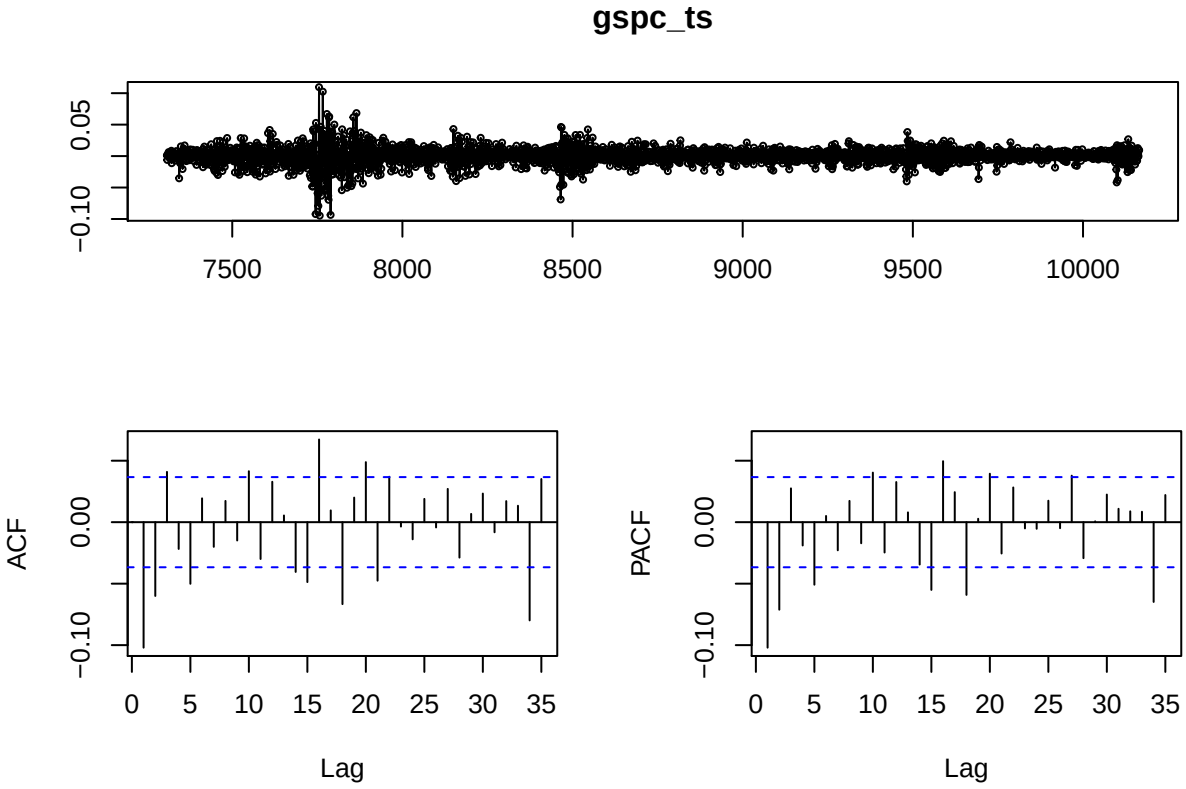
We begin by downloading the returns for the S&P 500 using the `getReturns()` function from the stockPortfolio library and computing the ACF/PACF's.

```
## '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.

##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).

## [1] "GSPC"
```



The model does not look like it will take on the form of a simple AR or MA process. There appear to be some interacting factors, as there are statistically significant spikes in both the ACF and the PACF at lags up to 5.

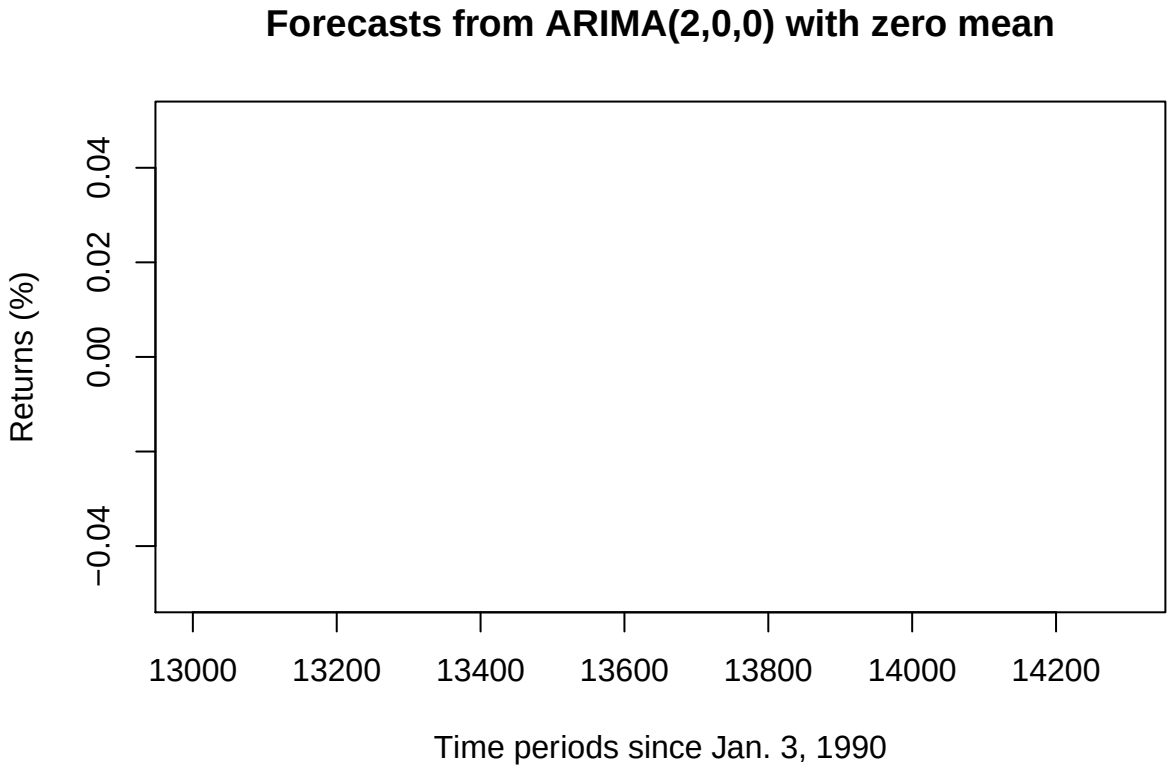
We use `auto.arima` to estimate the process as well. This gives us an ARIMA(2,0,5) process.

```
model <- auto.arima(gspc_ts)
model
```

```
## Series: gspc_ts
## ARIMA(2,0,0) with zero mean
##
## Coefficients:
##           ar1      ar2
##       -0.1089  -0.0707
```

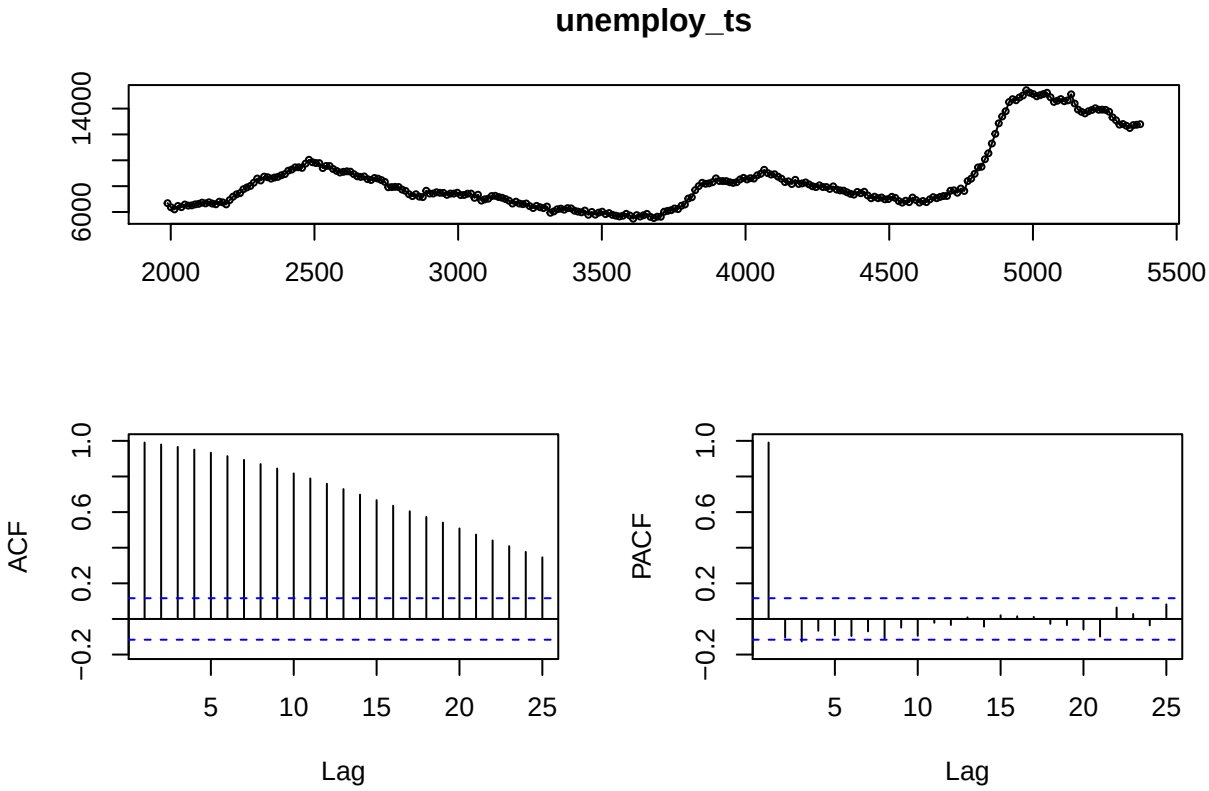
```
## s.e.   0.0187   0.0187
##
## sigma^2 estimated as 0.0001569:  log likelihood=8462.79
## AIC=-16919.58   AICc=-16919.57   BIC=-16901.71
```

We forecast 100 steps ahead. We find that the forecasted values are mostly flat, which indicates that the proposed ARIMA model, despite the relative complexity of the orders, fails to capture much of the dynamics within the series. This is consistent with financial theory, in that stock returns (especially for indices) should be mostly random due to market efficiency.



Problem 3.

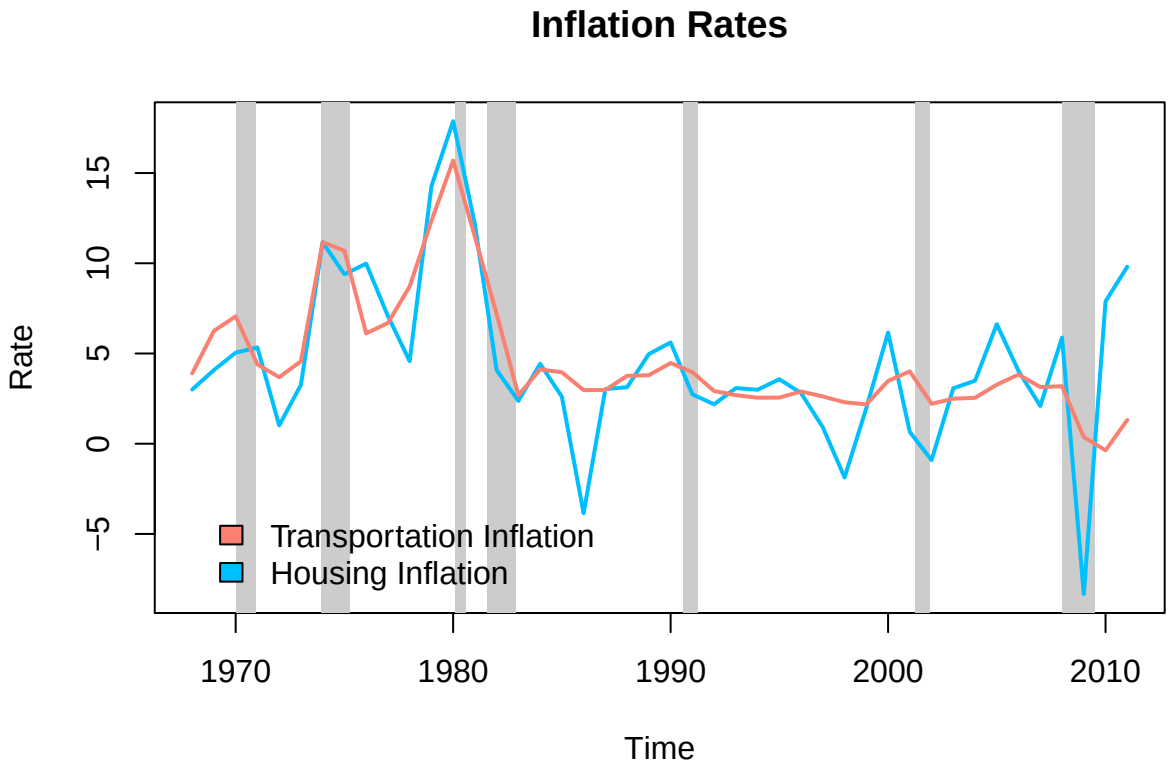
We begin by taking the unemployment rate and plotting it, along with the autocorrelation functions.
.



It appears that we may have an issue with stationarity, as we see spikes in the PACF that are close to 1. This would mean that an autoregressive process may not be a very suitable model to explain the dependence within the series. More robust measures could be taken to examine whether or not there does exist such a unit root problem. Regardless, we should consider differencing the data.

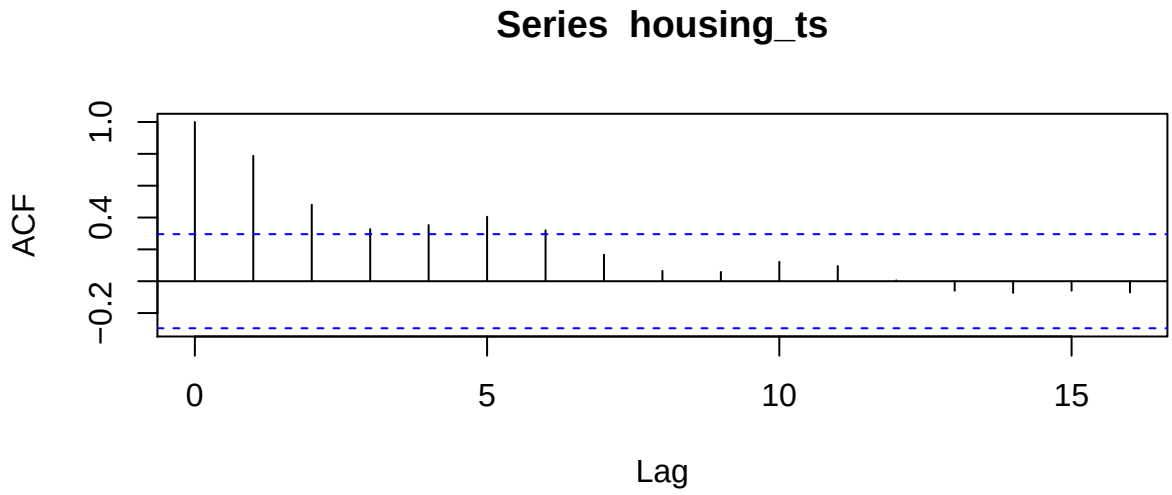
Problem 4.

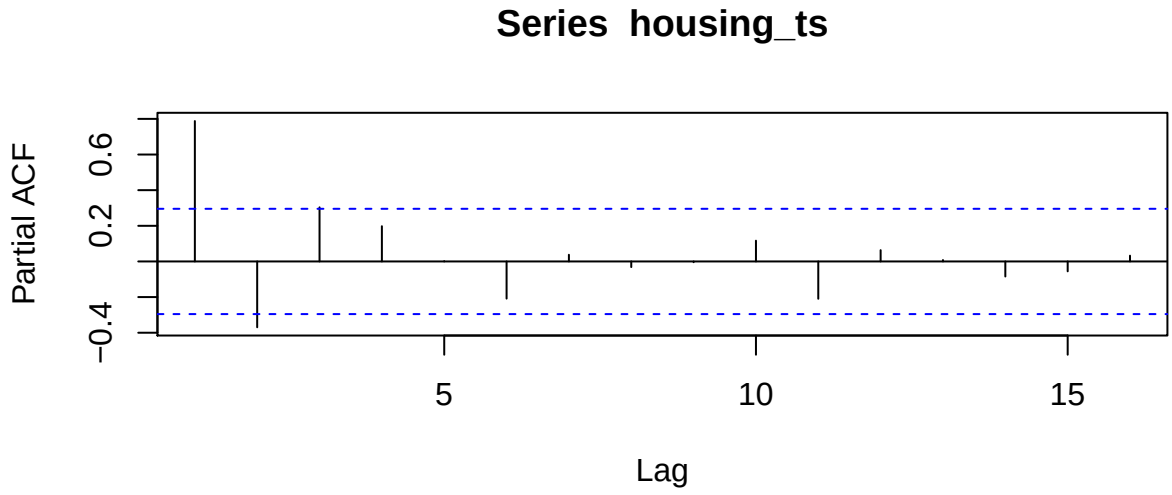
We take the housing and transportation inflation components from the Consumer Price Index from the BLS website. We plot both together.



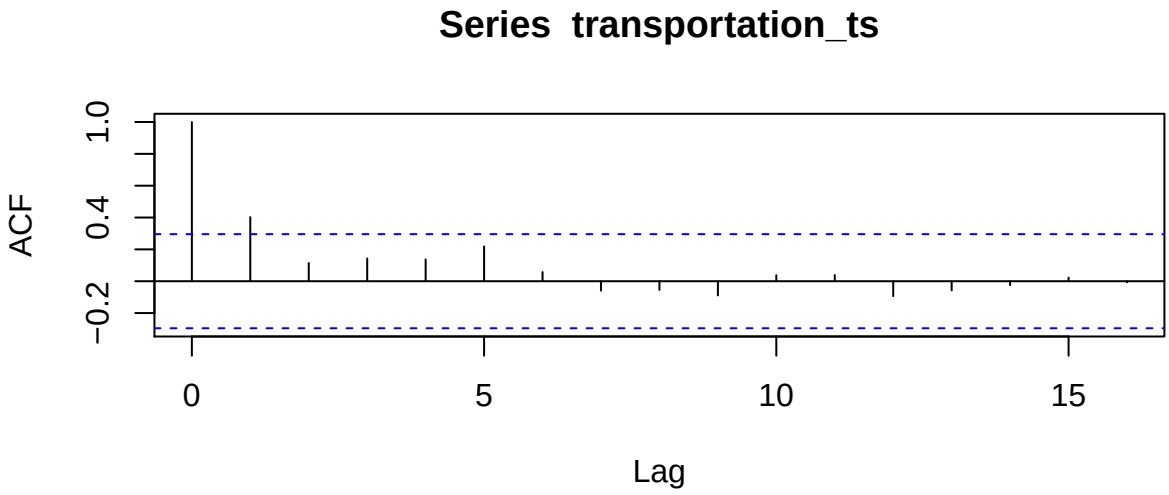
To examine whether the inflation rate for these components can be modeled as an AR(2) process, we compute the autocorrelation functions for both series:

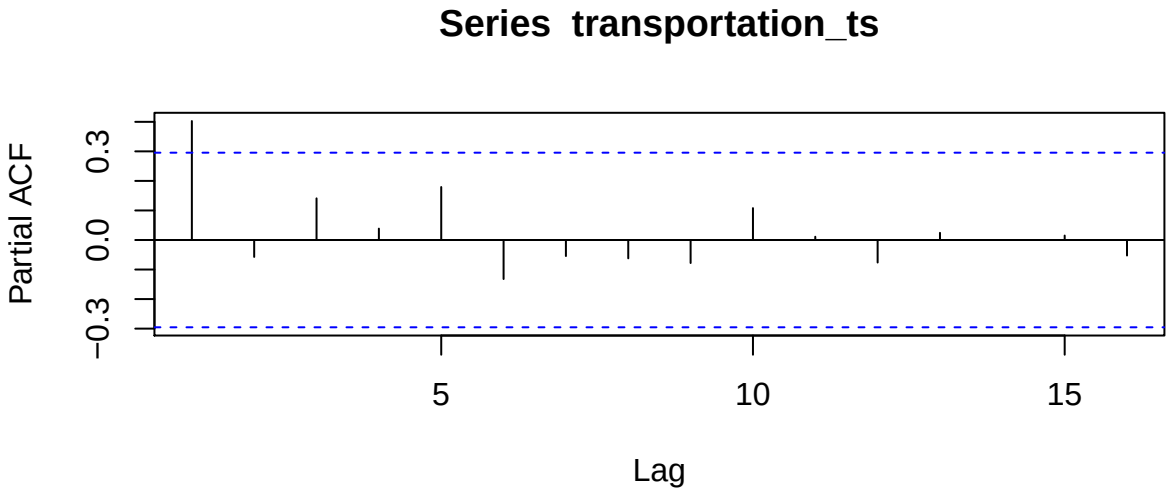
```
par(mfrow = c(2, 1))
acf(housing_ts)
pacf(housing_ts)
```






```
par(mfrow = c(2, 1))
acf(transportation_ts)
pacf(transportation_ts)
```





It appears from the ACF/PACF's that the housing series could be modeled as an AR(2) due to the large spikes in the PACF at lags 1 and 2, and the gradual decay in the ACF. A look at `auto.arima` shows that the estimated model from `auto.arima` is an ARMA model of order (2,2) with an integrated component of order 1.

```
auto.arima(housing_ts)  #2,1,2
```

```
## Series: housing_ts
## ARIMA(2,1,2)
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       0.6976  -0.9291  -0.4953  0.6634
## s.e.  0.1066   0.0824   0.1919  0.1816
##
## sigma^2 estimated as 3.115:  log likelihood=-84.06
## AIC=178.12   AICc=179.74   BIC=186.93
```

Meanwhile, from the ACF/PACF's of the transportation series, it would appear that fitting an

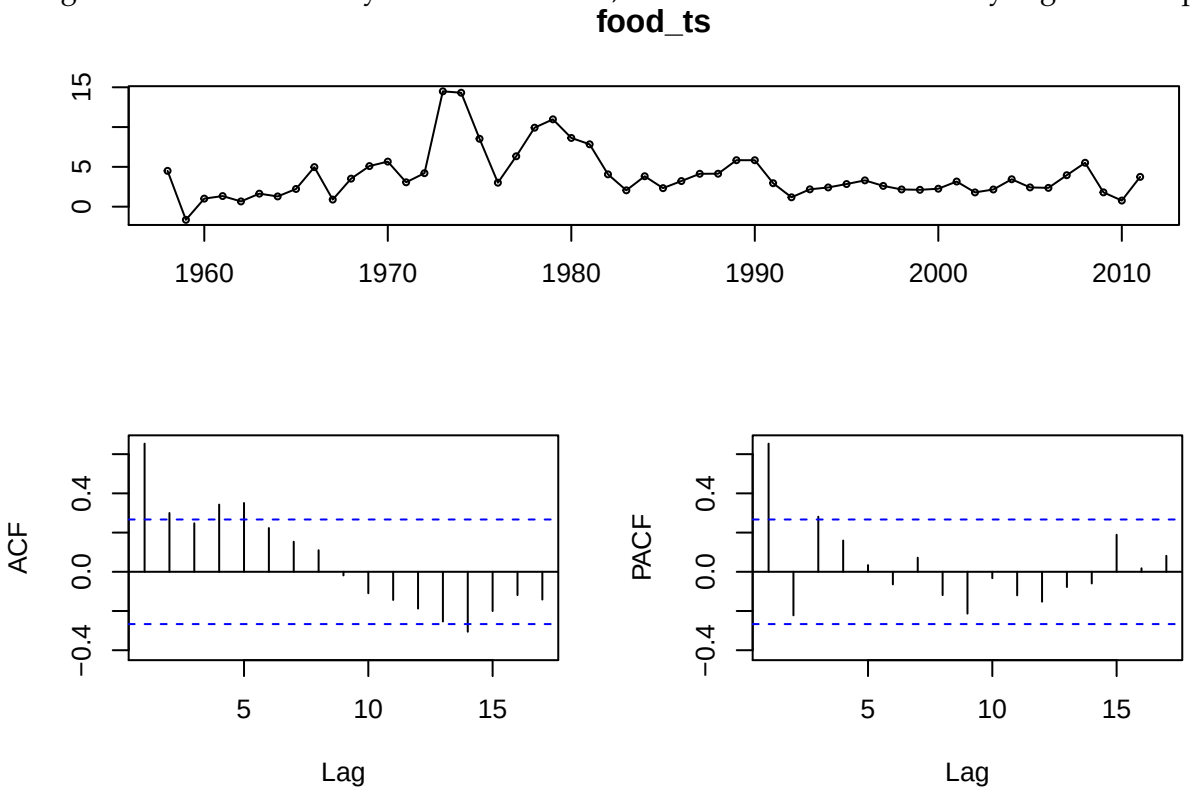
AR(2) would not be somewhat unfounded. We look at `auto.arima`'s estimated model and find that it estimates an moving average model of order 1.

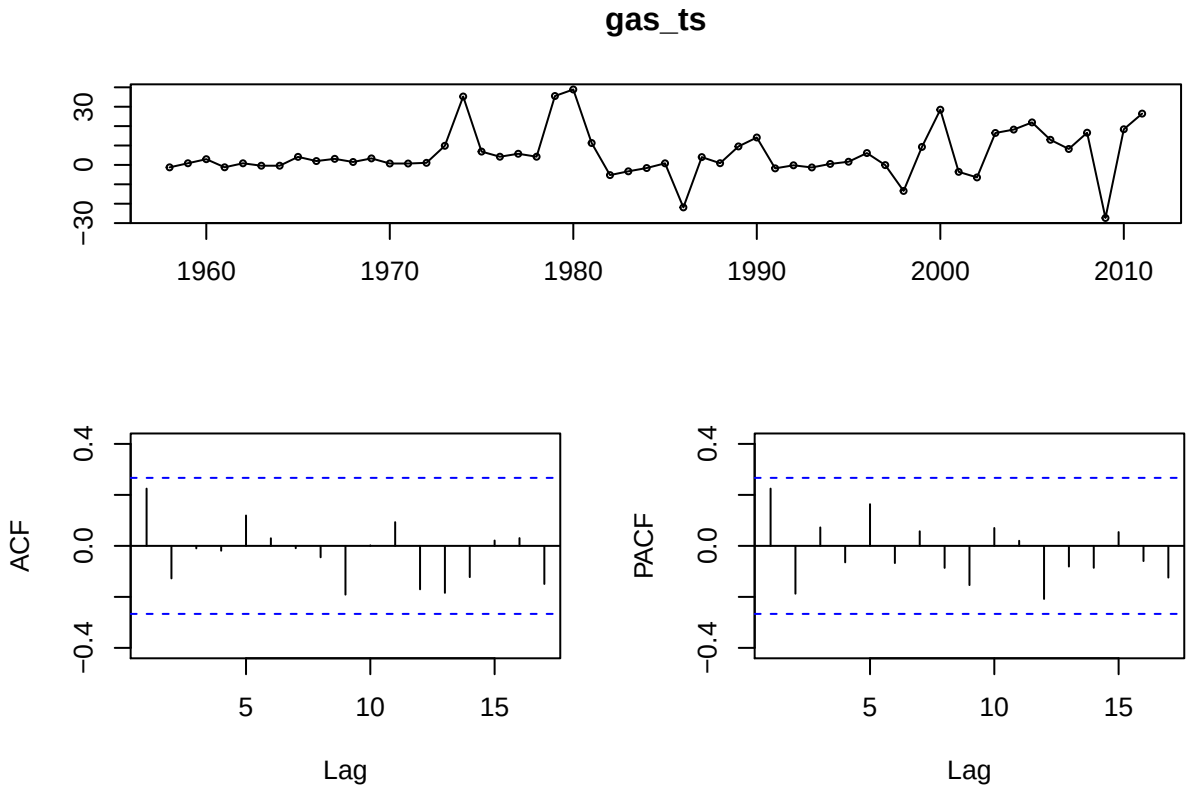
```
auto.arima(transportation_ts)  #0, 0, 1
```

```
## Series: transportation_ts
## ARIMA(0,0,1) with non-zero mean
##
## Coefficients:
##          ma1    mean
##       0.4336  4.3965
## s.e.  0.1442  0.8751
##
## sigma^2 estimated as 17.4:  log likelihood=-124.36
## AIC=254.72   AICc=255.32   BIC=260.07
```

Problem 5.

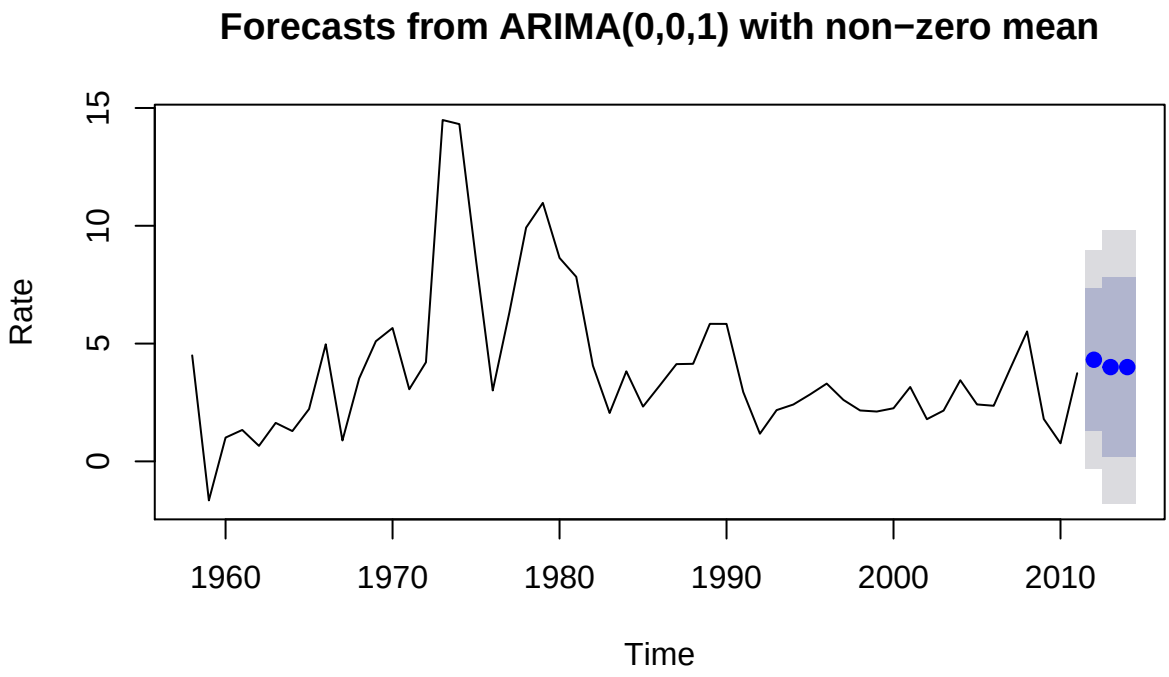
We now look at the food and gas inflation series. Looking at the corresponding ACF/PACF's of both series, it appears that the food series shows signs of an MA(1) process, while the gas series looks mostly like white noise, as there are no statistically significant spikes.





We fit an MA(1) model to the food series and provide the 3-day ahead forecast:

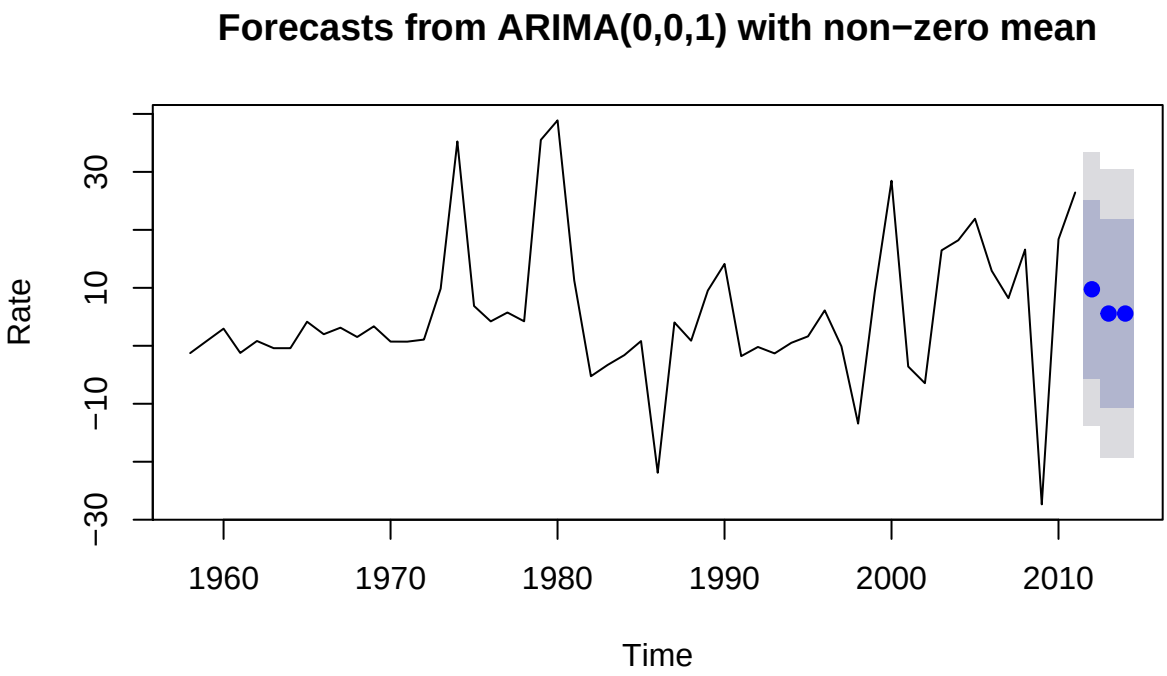
```
# Models: Food
model1 <- Arima(food_ts, c(0, 0, 1))
plot(forecast(model1, h = 3), ylab = "Rate", xlab = "Time")
```



The gas series is technically just white noise, so it would be incorrect to fit an MA(1) to the series.

However, just to see what the 3-day ahead forecasts will be, we fit an MA(1) anyway:

```
# Gas
model2 <- Arima(gas_ts, c(0, 0, 1))
plot(forecast(model2, h = 3), ylab = "Rate", xlab = "Time")
```



For the values of the actual point forecasts and intervals, you may output them from the forecast() function (or refer to the slides for the appropriate equations). Because we are using an MA(1) process, we expect any forecasts beyond an $h = 1$ horizon to converge back to μ.

R-Code:

```
rm(list = ls(all = TRUE))

# Load Libraries
library(lattice)
library(foreign)
library(MASS)
library(car)
require(stats)
require(stats4)
library(KernSmooth)
library(fastICA)
library(cluster)
library(leaps)
library(mgcv)
library(rpart)
library(pan)
```

```
library(mgcv)
library(DAAG)
library("TTR")
library(tis)
require("datasets")
require(graphics)
library("forecast")
require(astsa)
library(xtable)
library(stats)
library(quantmod)
library(stockPortfolio)
library(tseries)

# Problem 1 (6.4) b. Simulate the process for t = 1,2, ...
# 100, and compute the sample autocorrelation function up to
# lag 10.
ma.sim <- arima.sim(list(order = c(0, 0, 2), ma = c(-2, 1.35)),
    n = 100)
ma.sim
tsdisplay(ma.sim)

# Problem 2 (6.10)
gspc <- getReturns("^GSPC", freq = "day", start = "1990-01-03")
gspc_ts <- ts(rev(gspc$R), start = as.Date("1990-01-03"))
par(mfrow = c(2, 1))
acf(gspc_ts)
pacf(gspc_ts)
model <- auto.arima(gspc_ts)  #ARIMA(2,0,5)

dates <- rownames(gspc$R)
dates <- rev(dates)
dates <- as.Date(dates)
forecasted_dates <- seq(dates[length(dates)] + 1, by = 1, length = 10)
quartz()
plot(forecast(model, h = 100), xlim = c(13000, 14300), ylim = c(-0.05,
    0.05), ylab = "Returns (%)", xlab = "Time periods since Jan. 3, 1990")

# Problem 3 (7.2)
unemploy <- read.csv("~/Desktop/unemploy.csv", header = TRUE,
    stringsAsFactors = FALSE)
# Calculate the autocorrelation functions
unemploy_ts <- ts(unemploy[, 2], start = 1989, freq = 1/12)
time <- seq(1989, by = 1/12, length = length(unemploy_ts))

# Plot of the data
plot(time, unemploy_ts, type = "l", ylab = "Unemployment (in thousands)",
```

```
    xlab = "Time", main = "Unemployment over time")
nberShade()
lines(time, unemploy_ts)

# ACF/PACF
par(mfrow = c(2, 1))
acf(unemploy_ts)
pacf(unemploy_ts)

# Reason the values of the autocorrelation coefficients for
# different displacements of time.  Stationarity issue


# Problem 4 (7.6) Download components of the CPI from the BLS
# website
cpi <- read.csv("~/Desktop/cpi.csv", header = TRUE, stringsAsFactor = FALSE)

# We look at housing and transportation inflation
housing_ts <- ts(na.omit(cpi[, 3]), start = 1968)
transportation_ts <- ts(na.omit(cpi[, 5]), start = 1968)
time <- seq(1968, by = 1, length = length(housing_ts))
plot(time, transportation_ts, type = "l", main = "Inflation Rates",
    ylab = "Rate", xlab = "Time")
nberShade()
lines(time, transportation_ts, col = "deepskyblue", lwd = 2)
lines(time, housing_ts, col = "salmon", lwd = 2)
legend(1968, -3, c("Transportation Inflation", "Housing Inflation"),
    fill = c("salmon", "deepskyblue"), cex = 0.97, bg = "white",
    bty = "n")

# Analyze whether the inflation rate for these components can
# be modeled as an AR(2) process
par(mfrow = c(2, 1))
acf(housing_ts)
pacf(housing_ts)
# Yes

par(mfrow = c(2, 1))
acf(transportation_ts)
pacf(transportation_ts)
# No

auto.arima(housing_ts)  #2,1,2
auto.arima(transportation_ts)  #0, 0, 1

# Problem 5 (7.7)
cpi2 <- read.csv("~/Desktop/cpi_food.csv", header = TRUE, stringsAsFactors = FALSE)
```

```r
# Food and gas inflation
food_ts <- ts(na.omit(cpi2[, 3]), start = 1958)
gas_ts <- ts(na.omit(cpi2[, 5]), start = 1958)

# ACF/PACF:
tsdisplay(food_ts)
# MA(1)?

tsdisplay(gas_ts)
# Gas looks like white noise

# Models: Food
model1 <- Arima(food_ts, c(0, 0, 1))
plot(forecast(model1, h = 3), ylab = "Rate", xlab = "Time")

# Gas
model2 <- Arima(gas_ts, c(0, 0, 1))
plot(forecast(model2, h = 3), ylab = "Rate", xlab = "Time")
```