Week 6 - Selecting and Estimating an ARMA Model with Deterministic Trend and Seasonality

Jonathan Thong

2023-03-31

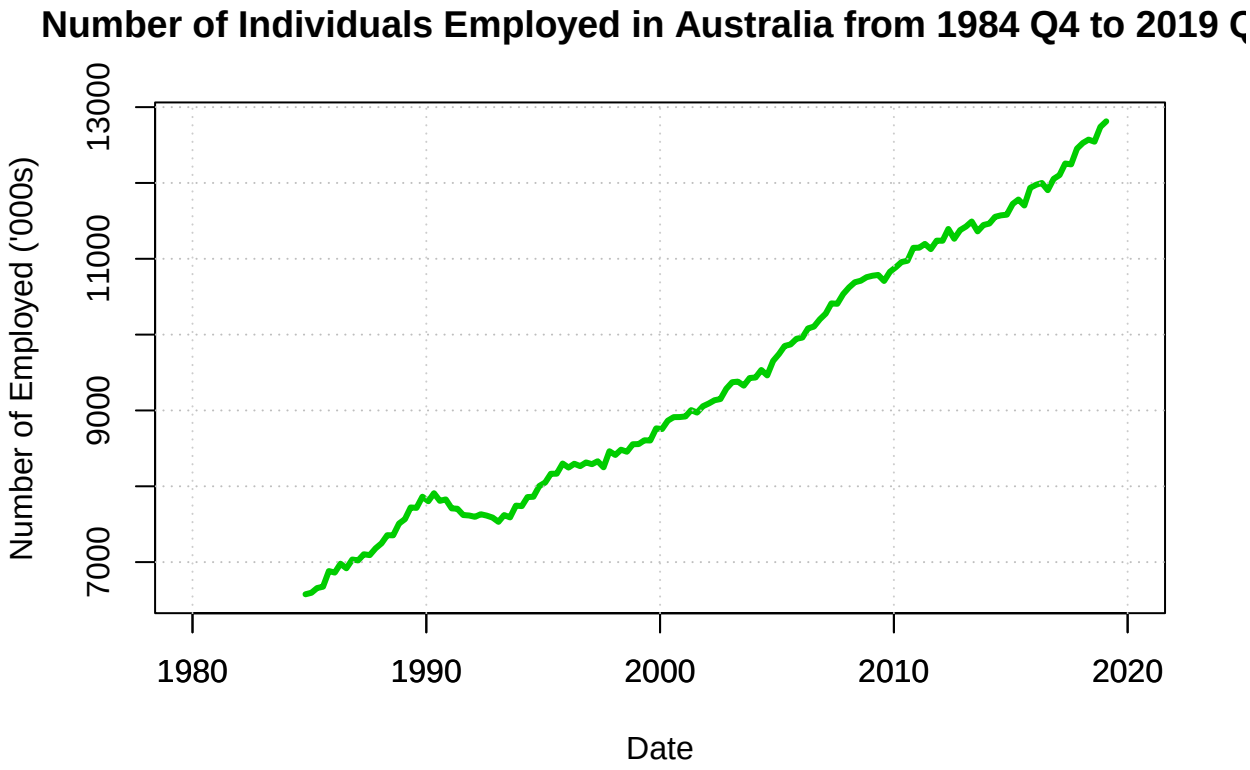
Describing the Data and Estimating the Deterministic Components

Let's start by importing and plotting the data. We can see that the time series possesses a clear upward trend. There are also some shorter term fluctuations which could potentially be modeled as seasonal fluctuations:

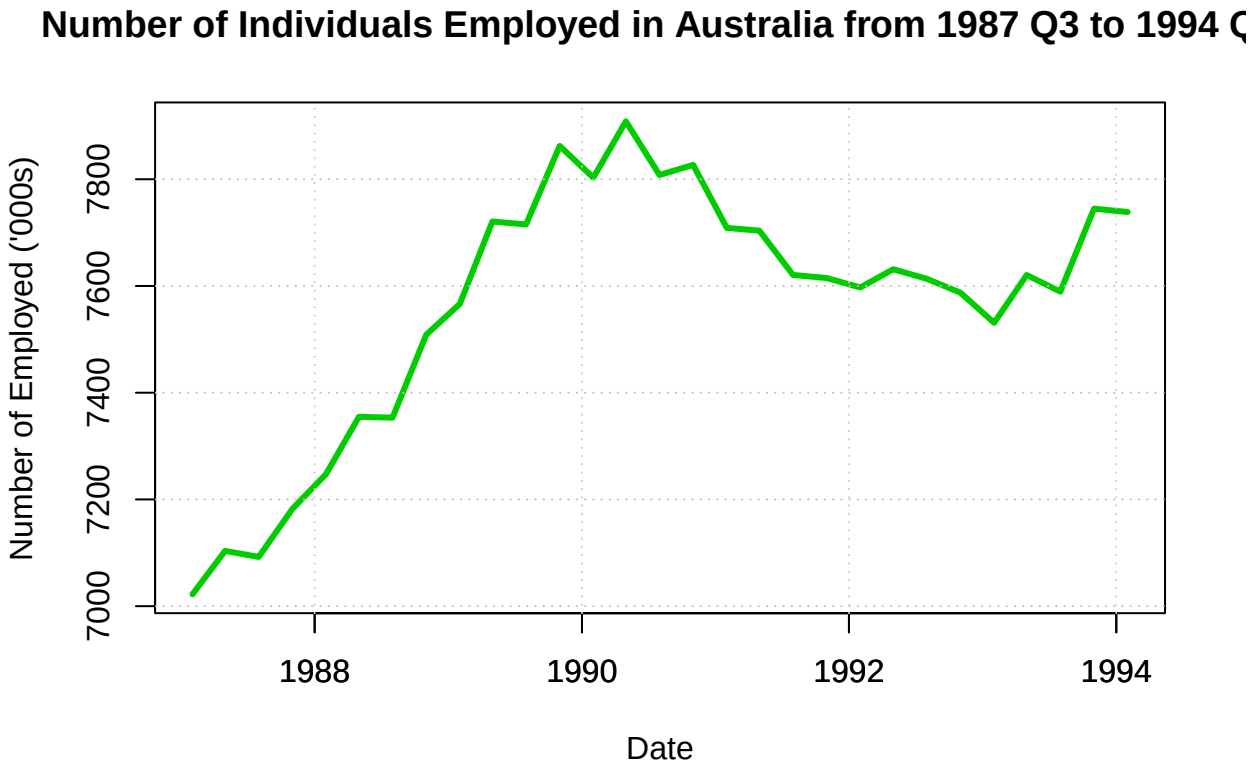
```
rm(list = ls())
data <- read.csv("ausemp.csv")

date <- seq(as.Date("1984/11/1"), as.Date("2019/2/1"), by ="quarter")

plot(date,data$ausemp,
main = "Number of Individuals Employed in Australia from 1984 Q4 to 2019 Q1",
xlab = "Date",
ylab = "Number of Employed ('000s)",
xlim = as.Date(c("1980/1/1","2020/1/1")),
col = "green3",
type ="l",
lwd = 3)
grid(nx = NA, ny = NULL,
     col = "grey", lwd = 1)
abline(v=axis.Date(1, date), col="grey80", lty = "dotted")
```




```
plot(date[10:38],data$ausemp[10:38],
main = "Number of Individuals Employed in Australia from 1987 Q3 to 1994 Q1",
xlab = "Date",
ylab = "Number of Employed ('000s)",
col = "green3",
type ="l",
lwd = 3)
grid(nx = NA, ny = NULL,
     col = "grey", lwd = 1)
abline(v=axis.Date(1, date[10:38]), col="grey80", lty = "dotted")
```



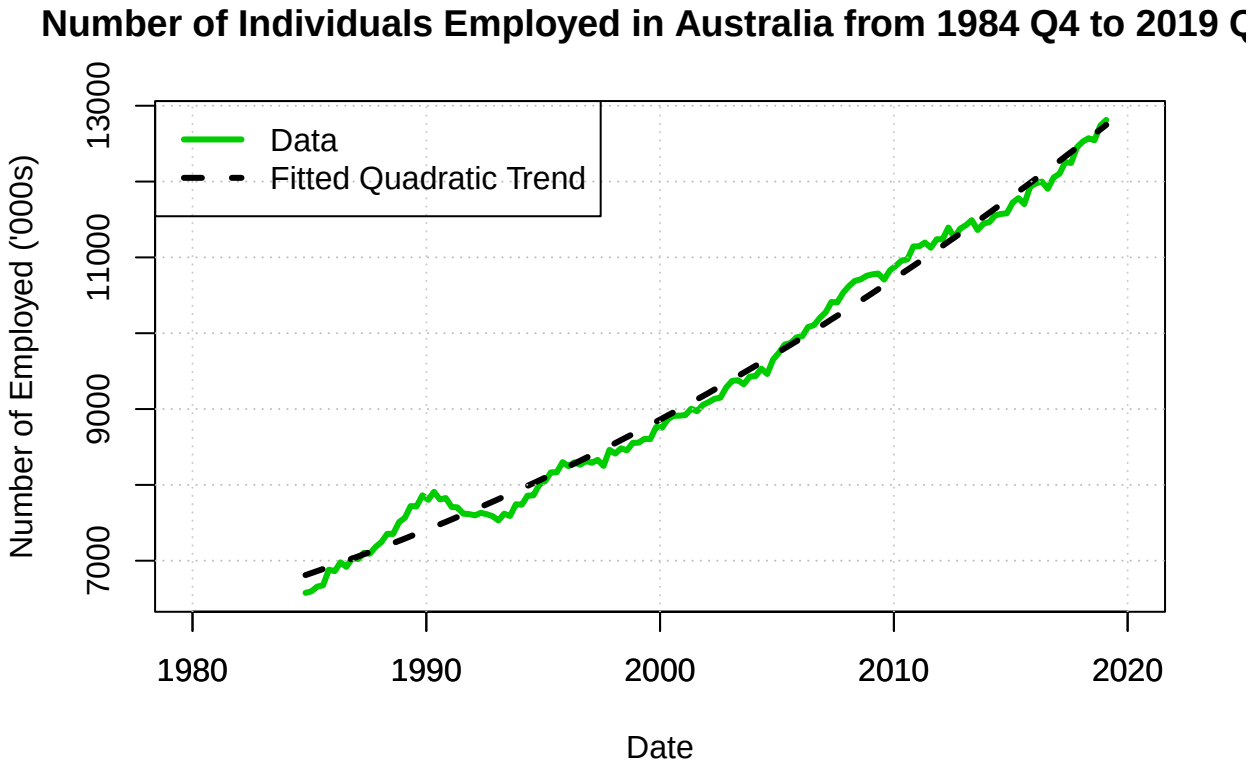
Let's first de-trend the data using a deterministic trend model. Looking at the plot above, there appears to be a bit of curvature to the upward trend, so a good starting point would be a quadratic or exponential trend specification:

```
T = length(data$ausemp)
time <- seq(1,T)
timesq <- time^2

trendmod <- lm(formula = data$ausemp ~ time + timesq)
summary(trendmod)
```

```
##
## Call:
## lm(formula = data$ausemp ~ time + timesq)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -291.78 -121.34  -44.00   90.81  477.59
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.784e+03  4.309e+01  157.45   <2e-16 ***
## time        2.597e+01  1.431e+00   18.15   <2e-16 ***
## timesq      1.249e-01  9.974e-03   12.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 166.3 on 135 degrees of freedom
## Multiple R-squared:  0.9911, Adjusted R-squared:  0.991
## F-statistic:  7515 on 2 and 135 DF,  p-value: < 2.2e-16
```

```
plot(date,data$ausemp,
main = "Number of Individuals Employed in Australia from 1984 Q4 to 2019 Q1",
xlab = "Date",
ylab = "Number of Employed ('000s)",
xlim = as.Date(c("1980/1/1","2020/1/1")),
col = "green3",
type ="l",
lwd = 3)
lines(date, trendmod$fitted.values, lty = "dashed", lwd = 3)
grid(nx = NA, ny = NULL,
     col = "grey", lwd = 1)
abline(v=axis.Date(1, date), col="grey80", lty = "dotted")
legend(x = "topleft",
       legend = c("Data", "Fitted Quadratic Trend"),
       lty = c("solid", "dashed"),
       lwd = 3.0,
       col = c("green3","black"))
```

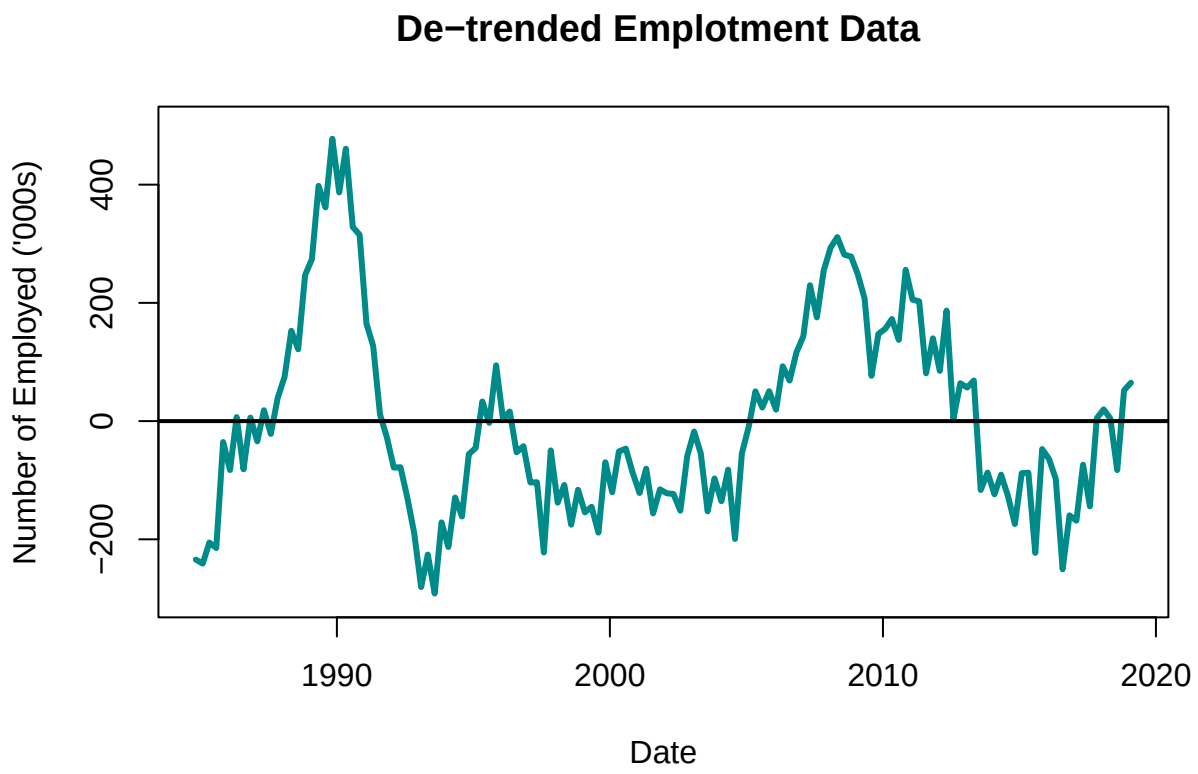


Once we have estimated the quadratic trend model, we can obtain the de-trended data as the residuals of the trend model:

```
data$detrended <- trendmod$residuals

plot(date,data$detrended,
main = "De-trended Emplotment Data ",
xlab = "Date",
ylab = "Number of Employed ('000s)",
ylim = c(-300,500),
col = "cyan4",
type ="l",
```

```
lwd = 3)
abline(0,0, lwd = 2)
```



We can then proceed to test whether there exists any significant deterministic seasonality using the same technique that we applied in Assignment 1. By excluding one of seasonal dummy variables and including a constant in our regression, the seasonal average corresponding to the excluded seasonal dummy becomes the baseline average so that the regression coefficients multiplying the remaining dummy variables now have the interpretation of deviations from the baseline. Looking at our estimation results we can see that none of these coefficients are statistically different from zero. Moreover, they are jointly equal to zero according to the reported overall F-statistic. Thus we can conclude that the seasonal means are not statistically different and we do not have to de-season our data and we can proceed to estimating the cyclical component by way of an ARMA specification.

```
s = 4

Q4 <- rep(c(1,0,0,0), ceiling(T/s)) # We start with Q4 because the first observation corresponds to the
Q1 <- rep(c(0,1,0,0), ceiling(T/s))
Q2 <- rep(c(0,0,1,0), ceiling(T/s))
Q3 <- rep(c(0,0,0,1), ceiling(T/s))

Q4 <- Q4[1:T]
Q1 <- Q1[1:T]
Q2 <- Q2[1:T]
Q3 <- Q3[1:T]

seasmod <- lm(formula = data$detrended ~ Q2 + Q3 + Q4)

summary(seasmod)
```

```
##
```

```
## Call:
## lm(formula = data$detrended ~ Q2 + Q3 + Q4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -276.53 -116.75  -37.43   97.51  458.47
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -4.223     27.789  -0.152    0.879
## Q2            33.708     39.587   0.851    0.396
## Q3           -40.592     39.587  -1.025    0.307
## Q4            23.338     39.299   0.594    0.554
##
## Residual standard error: 164.4 on 134 degrees of freedom
## Multiple R-squared:  0.0298, Adjusted R-squared:  0.008084
## F-statistic: 1.372 on 3 and 134 DF,  p-value: 0.254
```

Specifying and Estimating the Cyclical Component Using ARMA

Since we are working with real world data, we don't know what the true data generating process is. We will need to infer it statistically. As a starting point, let's perform our portmanteau tests on the de-trended data to verify that it is not white noise:

```
m = ceiling(sqrt(T))

Box.test(data$detrended, lag = m, type = "Box-Pierce")
```

```
##
##  Box-Pierce test
##
## data:  data$detrended
## X-squared = 462.6, df = 12, p-value < 2.2e-16
```

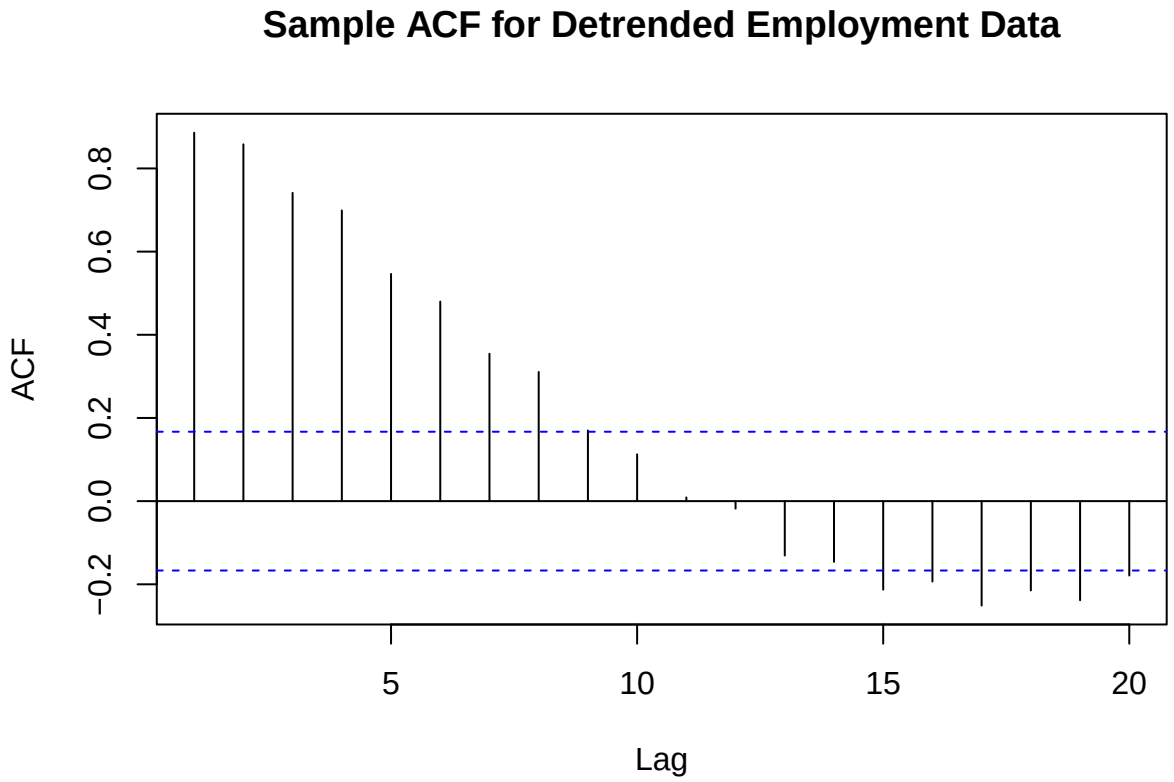
```
Box.test(data$detrended, lag = m, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  data$detrended
## X-squared = 480.61, df = 12, p-value < 2.2e-16
```

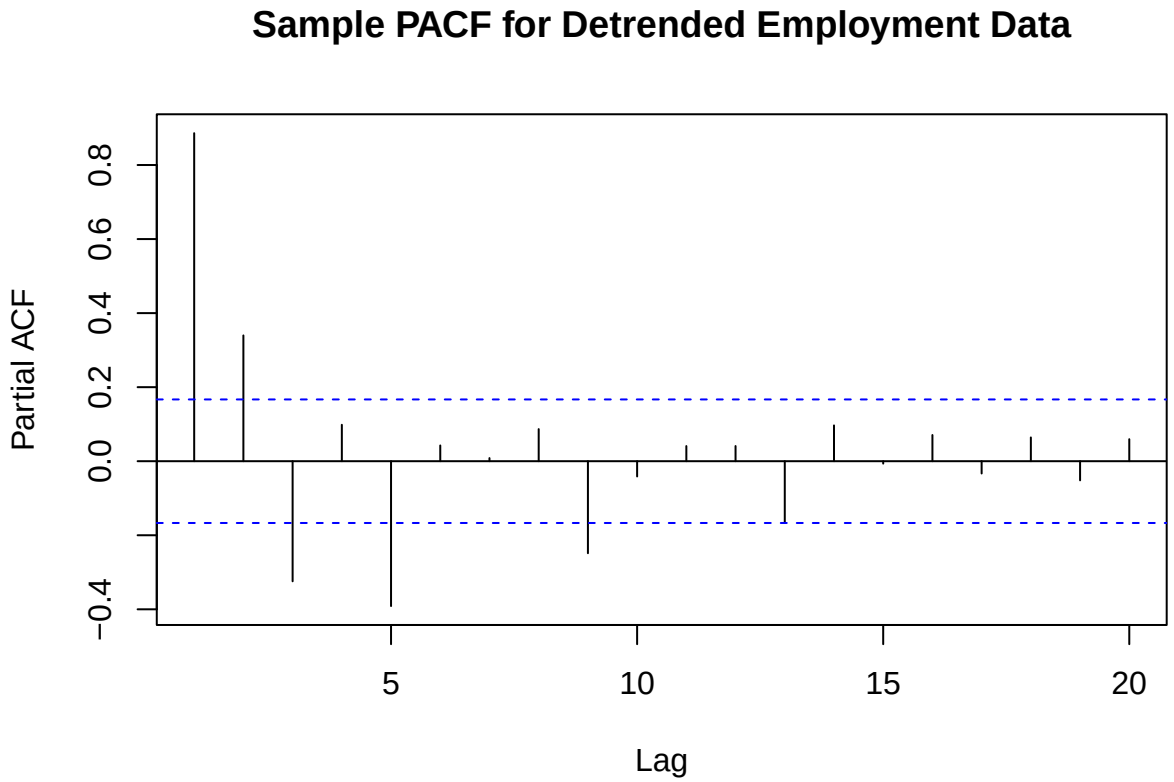
Then, having verified that dependence exists, we can proceed to have a look at the sample autocorrelations and partial autocorrelations.

```
acf.detrended <- acf(data$detrended, plot = FALSE)
pacf.detrended <- pacf(data$detrended, plot = FALSE)

plot(acf.detrended[1:20], main = "Sample ACF for Detrended Employment Data")
```




```
plot(pacf.detrended[1:20], main = "Sample PACF for Detrended Employment Data")
```



We can see from the above plots that the sample autocorrelations are decaying gradually and the partial autocorrelations appear to cut off at the fifth lag. However, there does appear to be a significant partial autocorrelation at the ninth lag too! Since we don't know the ARMA order of the true process, we will have

to try a range of candidate specifications and choose the one that fits the data the best. We will use the AIC and BIC as the basis for our model selection. Since we have to search across a number of different ARMA specifications, it is useful for us to write a loop! We will search from AR(1)-AR(9) and from MA(0) to MA(4) and we want to store the values of the information criteria in a data frame. To create this data frame, we will utilise the following code:

```
library(forecast)
```

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

```
pstart = 1
pend = 9

qstart = 0
qend = 4

prefix.ar <- "ar"
suffix.ar <- seq(from = pstart, to = pend)

prefix.ma <- "ma"
suffix.ma <- seq(from = qstart, to = qend)

ar.label <- paste(prefix.ar, suffix.ar, sep = "" )
ma.label <- paste(prefix.ma, suffix.ma, sep = "")

akaike <- data.frame(row.names = ar.label, matrix(0,pend,qend+1))
colnames(akaike) <- ma.label

bayes <- data.frame(row.names = ar.label, matrix(0,pend,qend+1))
colnames(bayes) <- ma.label
```

Now that we have specified the data frames into which we will store the values of the AIC and BIC of our candidate models, we will now write a loop to estimate them. Note that in most applications, there will be a set of ARMA specifications for which the estimation will not converge. We will exclude these models from consideration by setting their AIC and BIC values to *NA*:

```
for(i in pstart:pend){

  for(j in (qstart+1):(qend+1)){

    model <- arima(data$detrended,
                   order = c(i,0,j-1),
                   include.mean = FALSE,
                   method = "ML")

    if (model$code == 0){
    akaike[i,j] <- AIC(model)
    bayes[i,j]<- BIC(model)}
    else {akaike[i,j] <- NA
    bayes[i,j]<- NA}
  }
}
```

```
## Warning in log(s2): NaNs produced
```

```
## Warning in log(s2): NaNs produced

## Warning in log(s2): NaNs produced

## Warning in arima(data$detrended, order = c(i, 0, j - 1), include.mean = FALSE, :
## possible convergence problem: optim gave code = 1

## Warning in log(s2): NaNs produced

## Warning in arima(data$detrended, order = c(i, 0, j - 1), include.mean = FALSE, :
## possible convergence problem: optim gave code = 1

## Warning in arima(data$detrended, order = c(i, 0, j - 1), include.mean = FALSE, :
## possible convergence problem: optim gave code = 1

## Warning in arima(data$detrended, order = c(i, 0, j - 1), include.mean = FALSE, :
## possible convergence problem: optim gave code = 1

## Warning in arima(data$detrended, order = c(i, 0, j - 1), include.mean = FALSE, :
## possible convergence problem: optim gave code = 1

## Warning in log(s2): NaNs produced

## Warning in arima(data$detrended, order = c(i, 0, j - 1), include.mean = FALSE, :
## possible convergence problem: optim gave code = 1

## Warning in arima(data$detrended, order = c(i, 0, j - 1), include.mean = FALSE, :
## possible convergence problem: optim gave code = 1

## Warning in arima(data$detrended, order = c(i, 0, j - 1), include.mean = FALSE, :
## possible convergence problem: optim gave code = 1

## Warning in arima(data$detrended, order = c(i, 0, j - 1), include.mean = FALSE, :
## possible convergence problem: optim gave code = 1
```

Then, to find the model that minimises the AIC and BIC, we simple need to locate the coordinates of the minimum value of these data frames:

```
akaike.row <- rownames(akaike)[which(akaike == min(akaike, na.rm = TRUE),arr.ind = TRUE)[ , 1]]
akaike.col <- colnames(akaike)[which(akaike == min(akaike, na.rm = TRUE),arr.ind = TRUE)[ , 2]]

akaike.choice <- c(akaike.row, akaike.col)

bayes.row <- rownames(bayes)[which(bayes == min(bayes, na.rm = TRUE),arr.ind = TRUE)[ , 1]]
bayes.col <- colnames(akaike)[which(bayes == min(bayes, na.rm = TRUE),arr.ind = TRUE)[ , 2]]

bayes.choice <- c(bayes.row, bayes.col)
```

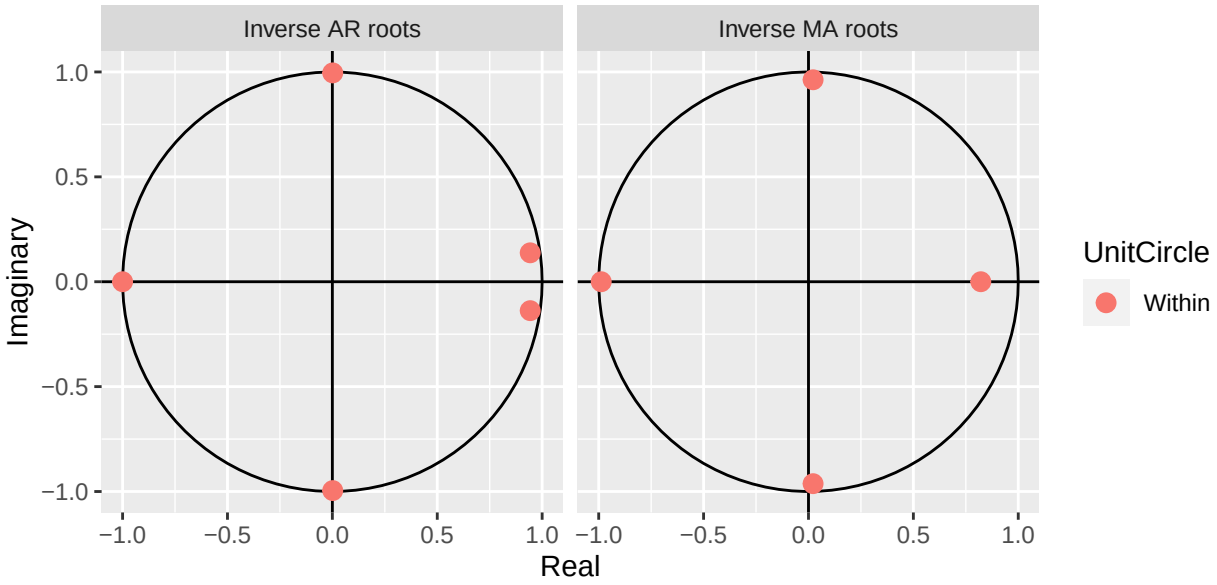
Thus, both the AIC and BIC agree that an ARMA(5,4) is the model that best fits the data:

```
bestmod <- arima(data$detrended,
                 order = c(5,0,4),
                 include.mean = FALSE,
                 method = "ML")
summary(bestmod)
```

```
##
```

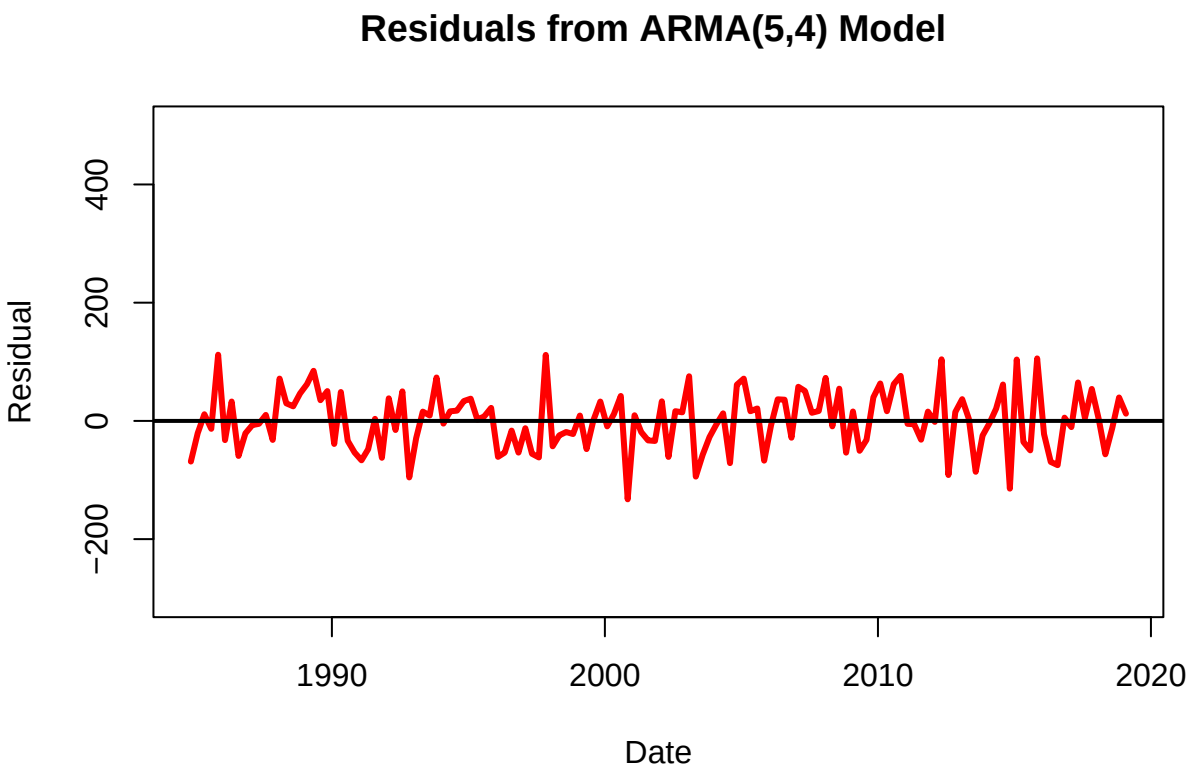
```
## Call:
## arima(x = data$detrended, order = c(5, 0, 4), include.mean = FALSE, method = "ML")
##
## Coefficients:
##          ar1      ar2      ar3     ar4      ar5     ma1     ma2     ma3
##       0.8898  -0.0166  -0.0334  0.9721  -0.9004  0.1225  0.1083  0.1900
## s.e.  0.0758   0.0219   0.0252  0.0246   0.0690  0.1467  0.1203  0.1528
##           ma4
##       -0.7525
## s.e.   0.1276
##
## sigma^2 estimated as 2388:  log likelihood = -737,  aic = 1494
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.6042093 48.87115 39.27138 -39.74189 104.4956 0.6378159
##                    ACF1
## Training set -0.0755377
```

```
autoplot(bestmod)
```



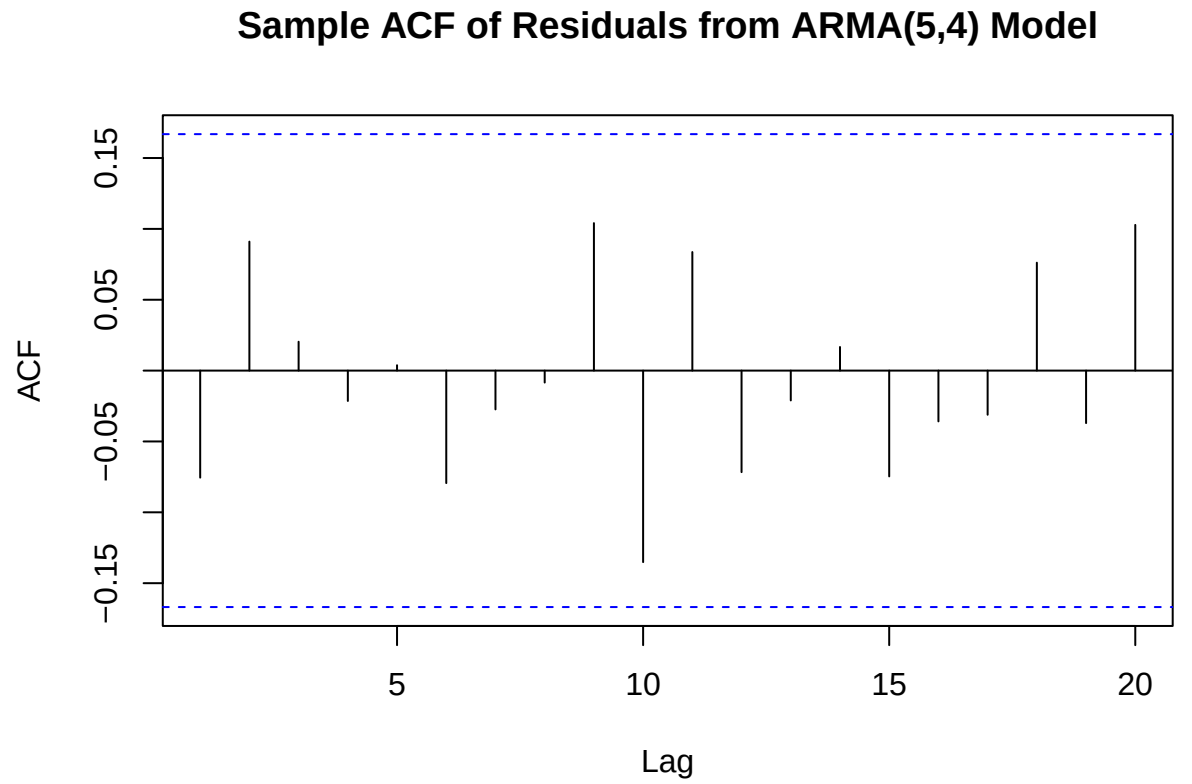
To see whether this specification has accounted for all the dependence in the data, let's have a look at the residuals:

```
plot(date,bestmod$residuals,
main = "Residuals from ARMA(5,4) Model ",
xlab = "Date",
ylab = "Residual",
ylim = c(-300,500),
col = "red",
type ="l",
lwd = 3)
abline(0,0, lwd = 2)
```

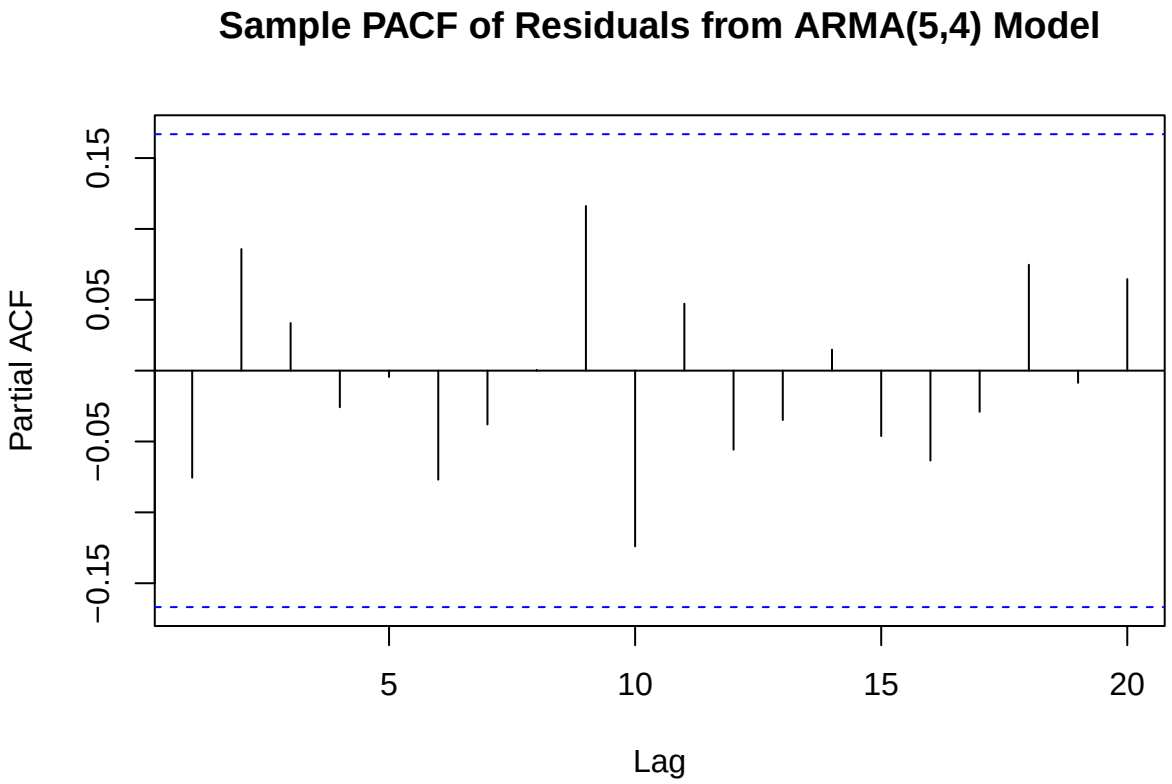



```
acf.resid <- acf(bestmod$residuals, plot = FALSE)
pacf.resid <- pacf(bestmod$residuals, plot = FALSE)

plot(acf.resid[1:20], main = "Sample ACF of Residuals from ARMA(5,4) Model")
```



```r
plot(pacf.resid[1:20], main = "Sample PACF of Residuals from ARMA(5,4) Model")
```



Applying our portmanteau tests, we can verify that the residuals are indeed white noise!

```r
Box.test(bestmod$residuals, type = "Box-Pierce", l = m)
```

```
##
##  Box-Pierce test
##
## data:  bestmod$residuals
## X-squared = 8.7255, df = 12, p-value = 0.7262
```

```r
Box.test(bestmod$residuals, type = "Ljung-Box", l = m)
```

```
##
##  Box-Ljung test
##
## data:  bestmod$residuals
## X-squared = 9.3834, df = 12, p-value = 0.6699
```