R-Week8-9

Jonathan Thong

2023-05-13

Simulating An Integrated Series

Let's begin by simulating \$ T = 1000\$ observations from a white noise series,

$$\epsilon_t \sim N(0,1)$$

```r
rm(list = ls())

T = 1000
e = rnorm(T)
```

Having obtained our white noise series, let's now proceed to construct an ARMA(2,2). Recall that an ARMA(2,2) is described by the equation,

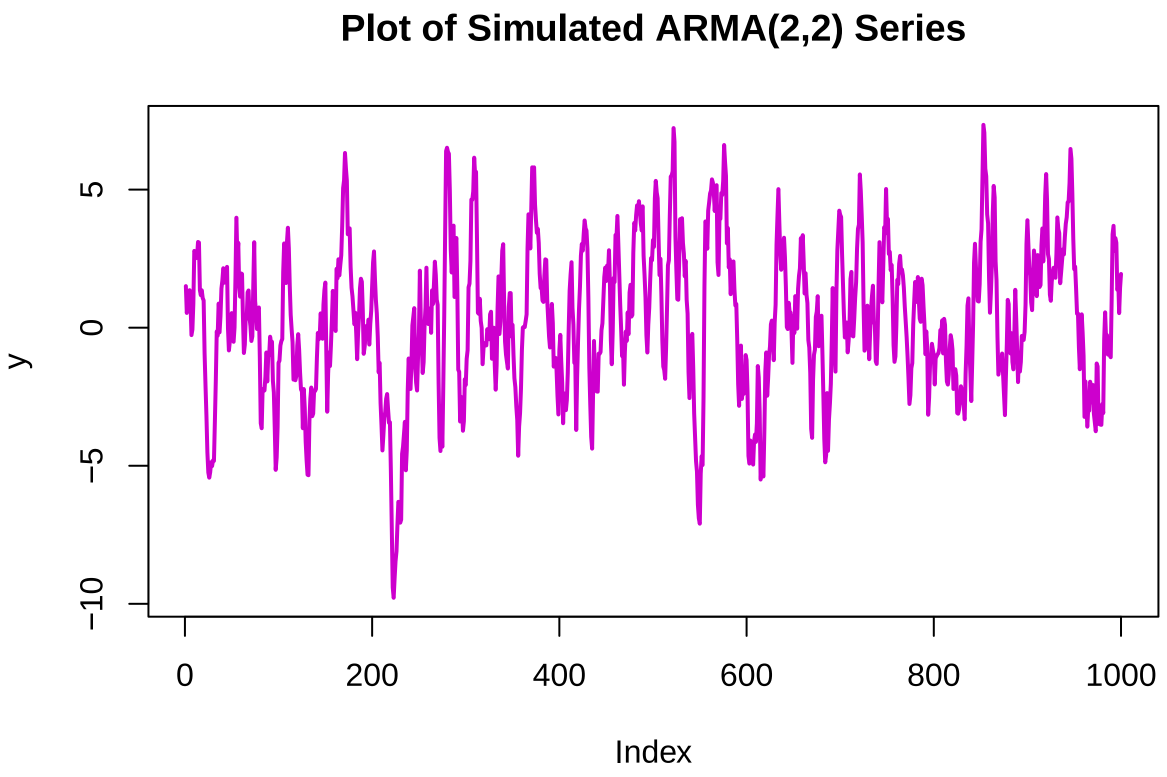
$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

Using the first two observations of the white noise series as initial values, and setting $\phi_1 = 0.5$, $\phi_2 = 0.3$, $\theta_1 = 0.6$ and $\theta_2 = 0.2$, we can proceed to construct the ARMA(2,2) using a loop.

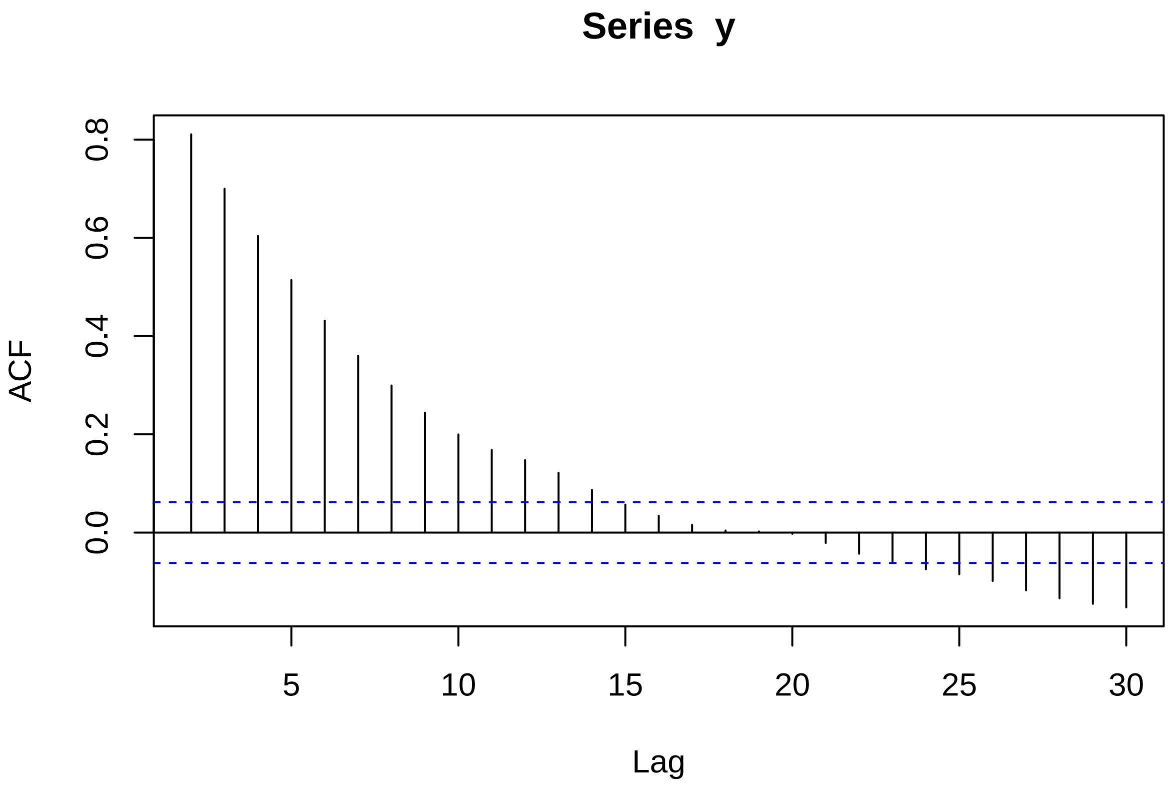
```r
y <- rep(NA,T)
y[1:2] <- e[1:2]

phi1 = 0.5
phi2 = 0.3
theta1 = 0.6
theta2 = 0.2

for (i in 3:T){
     y[i] = phi1*y[i-1] + phi2*y[i-2] + e[i] + theta1*e[i-1] + theta2*e[i-2]
}
```

We can then have a look at the plot of the series as well as the sample autocorrelation and partial autocorrelations. From the sample PACF we can clearly discern the AR(2) component from the first two significant partial autocorrelations. The MA(2) components are less discernible as the cutoff that we would see in the sample autocorrelations is obscured by the autoregressive structure.

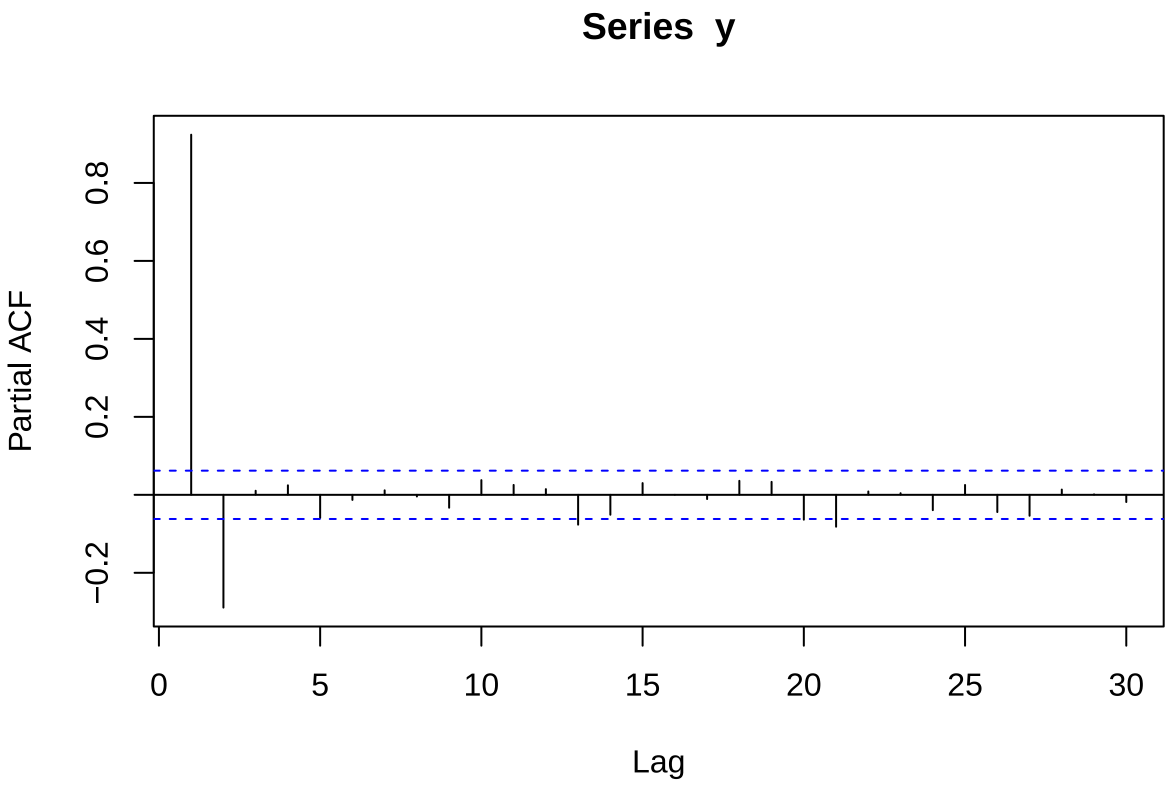
```r
plot(y,
     main = "Plot of Simulated ARMA(2,2) Series",
     type = "l",
     col = "magenta3",
     lwd = 2.0)
```



```
acf.y <- acf(y, plot = FALSE)
pacf.y <- pacf(y, plot = FALSE)

plot(acf.y[2:30])
```



```
plot(pacf.y)
```



Given the values of phi_1 and phi_2 we know that the series that we have just computed is covariance stationary. To generate a series that is integrated of order 1, we simply have to define:

$$Q_t = \sum_{j=1}^{t} Y_j$$

We know that Q_t is an I(1) series since the first difference of the series is stationary

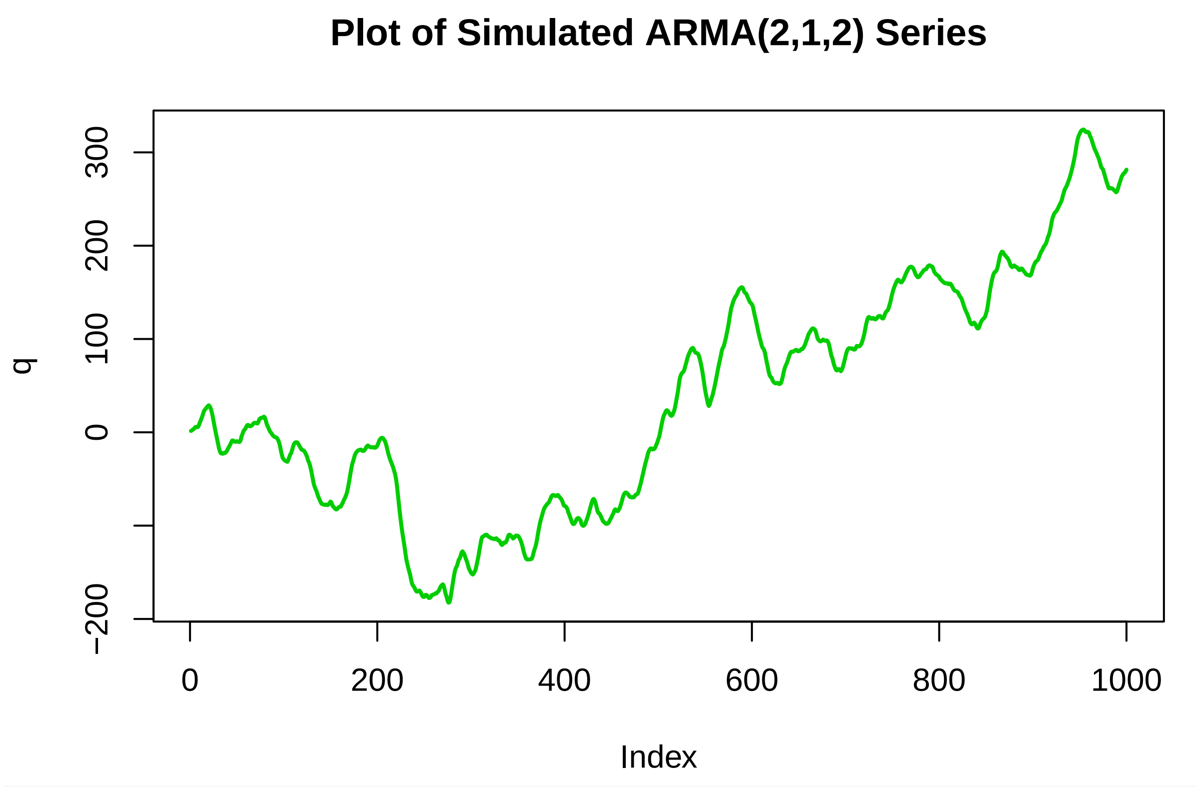
$$Q_t - Q_{t-1} = \sum_{j=1}^{t} Y_j - \sum_{j=1}^{t-1} Y_j = Y_t$$

Therefore Q_t is an ARIMA(2,1,2). We can use the **cumsum()** function to compute this:

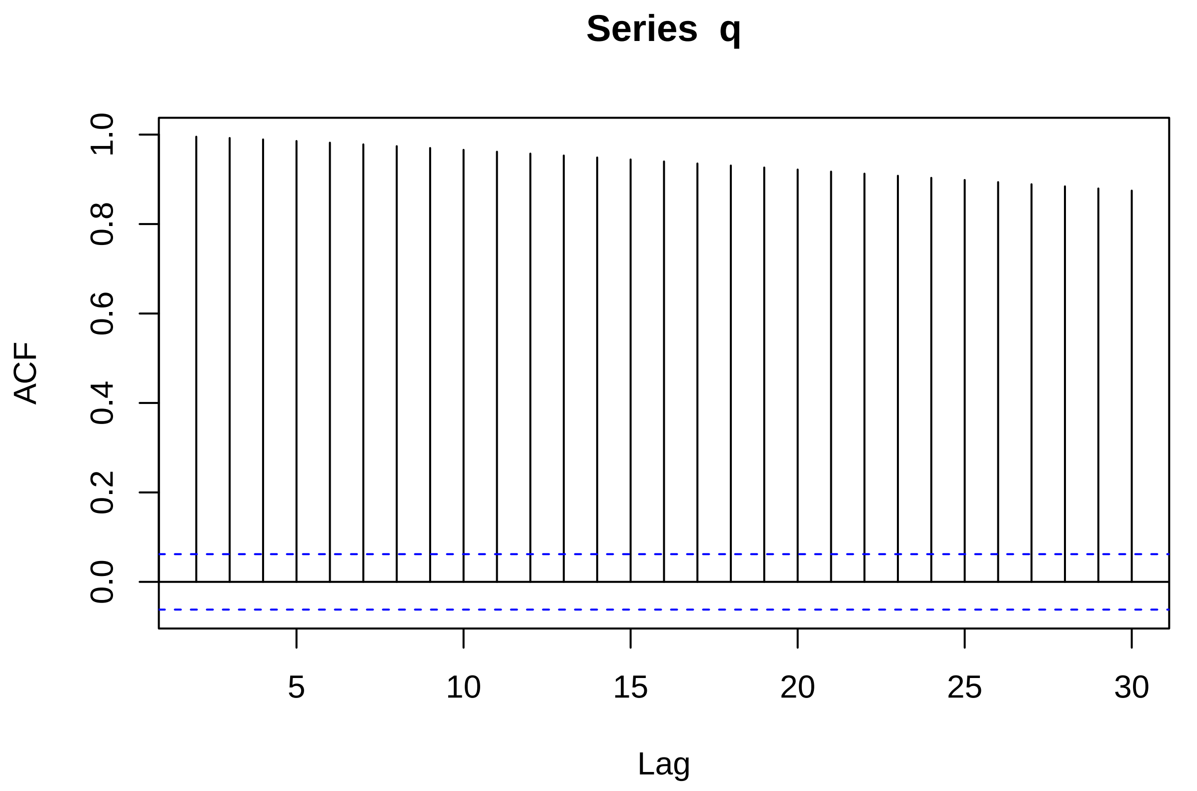
```
q = cumsum(y)
```

Looking at the plot, sample ACF and PACF of this new series, we can see that the ARIMA(2,1,2) is clearly non-stationary as the sample autocorrelations are decaying slowly (i.e., NOT at a geomtric rate), the first order partial autocorrelation is equal to 1.

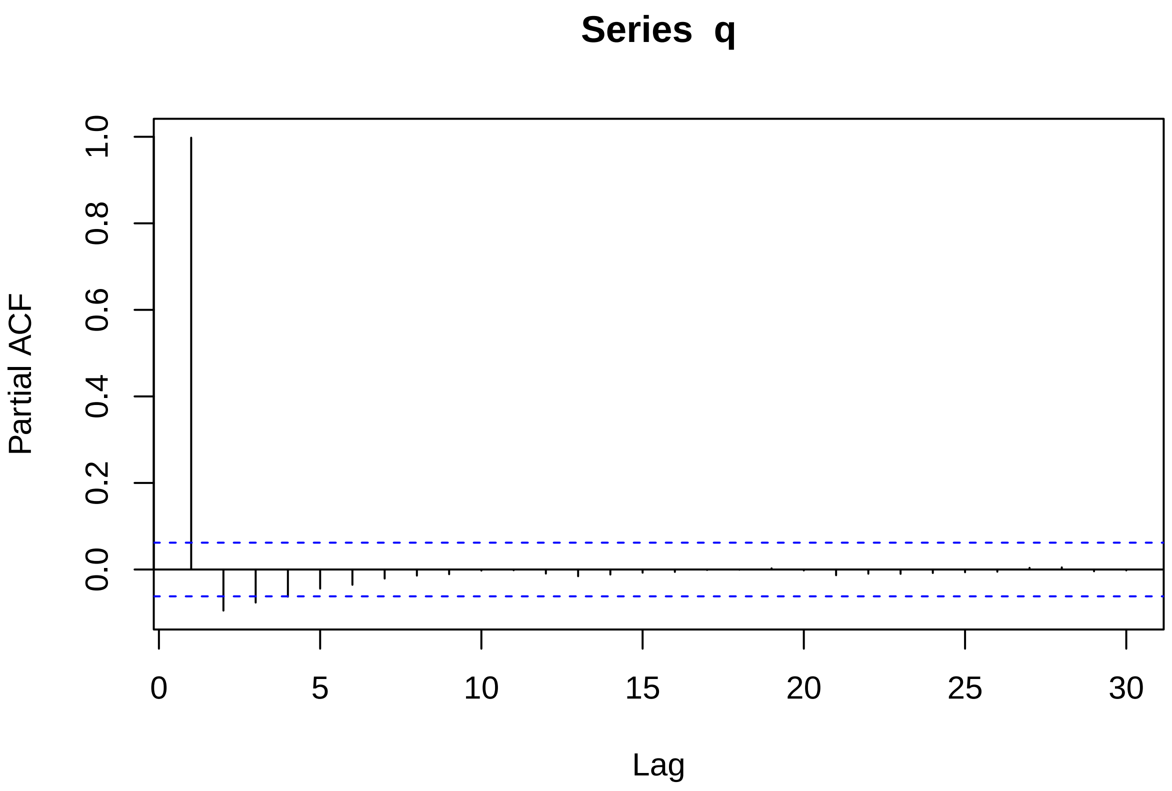
```
plot(q,
     main = "Plot of Simulated ARMA(2,1,2) Series",
     type = "l",
     col = "green3",
     lwd = 2.0)
```



```r
acf.q <- acf(q, plot = FALSE)
pacf.q <- pacf(q, plot = FALSE)

plot(acf.q[2:30])
```



```
plot(pacf.q)
```



We can then proceed to use our I(1) series Q_t to compute a series that is integrated of order 2 by defining,

$$Z_t = \sum_{k=1}^{t} Q_k$$

We know that Z_t is an I(2) series since the first difference of the series is I(1),

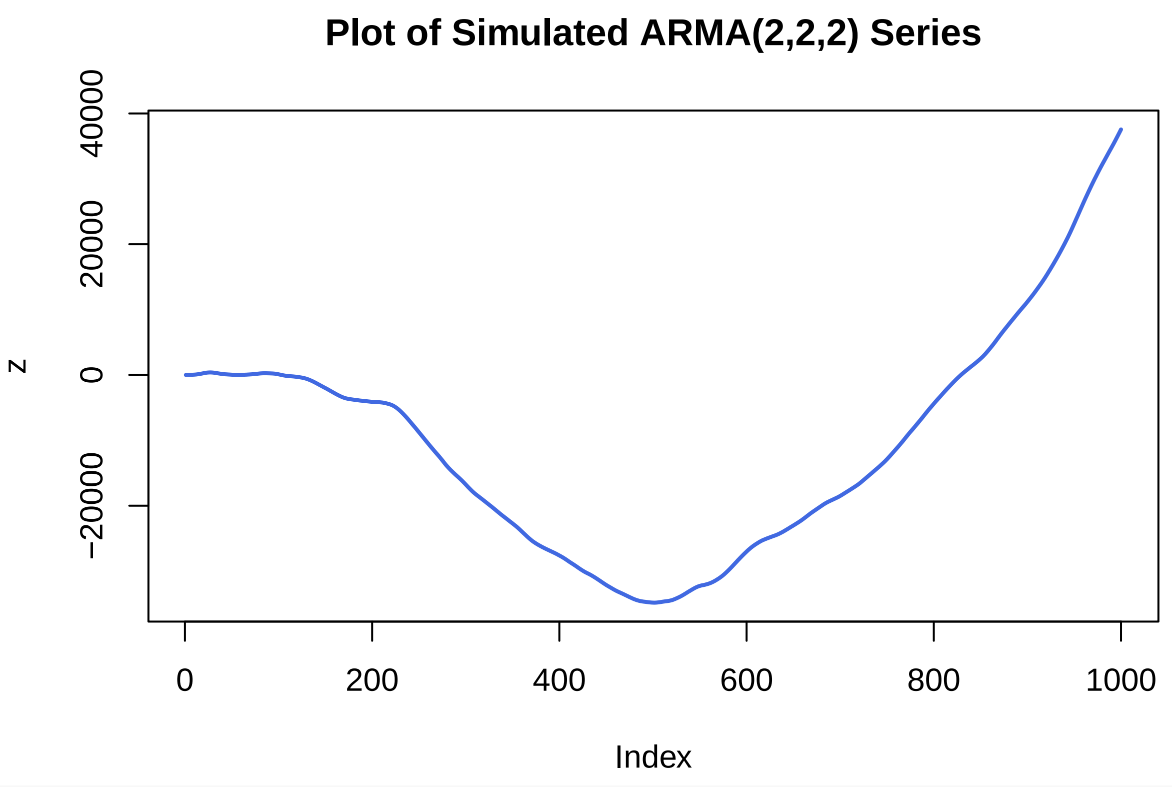
$$Z_t - Z_{t-1} = \sum_{k=1}^{t} Q_k - \sum_{k=1}^{t-1} Q_k = Q_t$$

Therefore Z_t is an ARIMA(2,2,2). We can again use the **cumsum()** function to compute this:

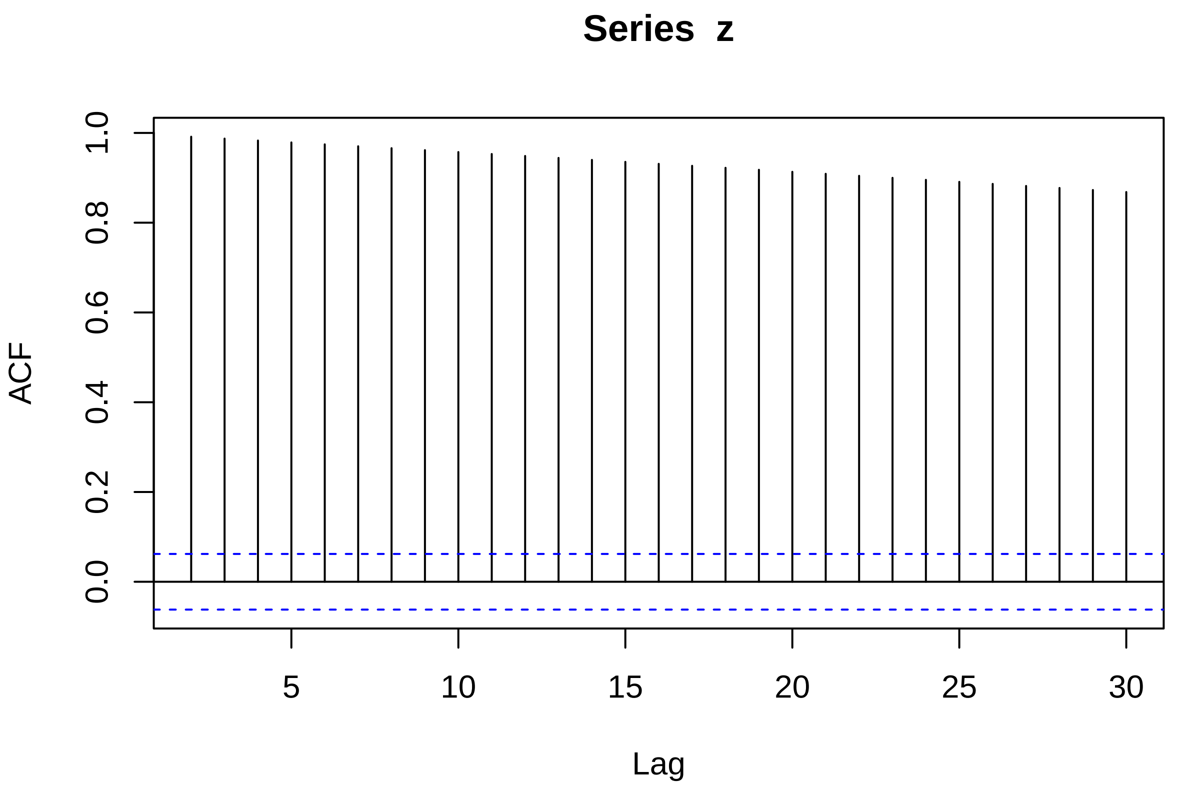
```
z = cumsum(q)
```

Looking at the plot, sample ACF and PACF of this new series, we can again see that the ARIMA(2,2,2) is clearly non-stationary as the sample autocorrelations are decaying slowly (i.e., NOT at a geometric rate), the first order partial autocorrelation is equal to 1. We can also see that the data is trending upwards.

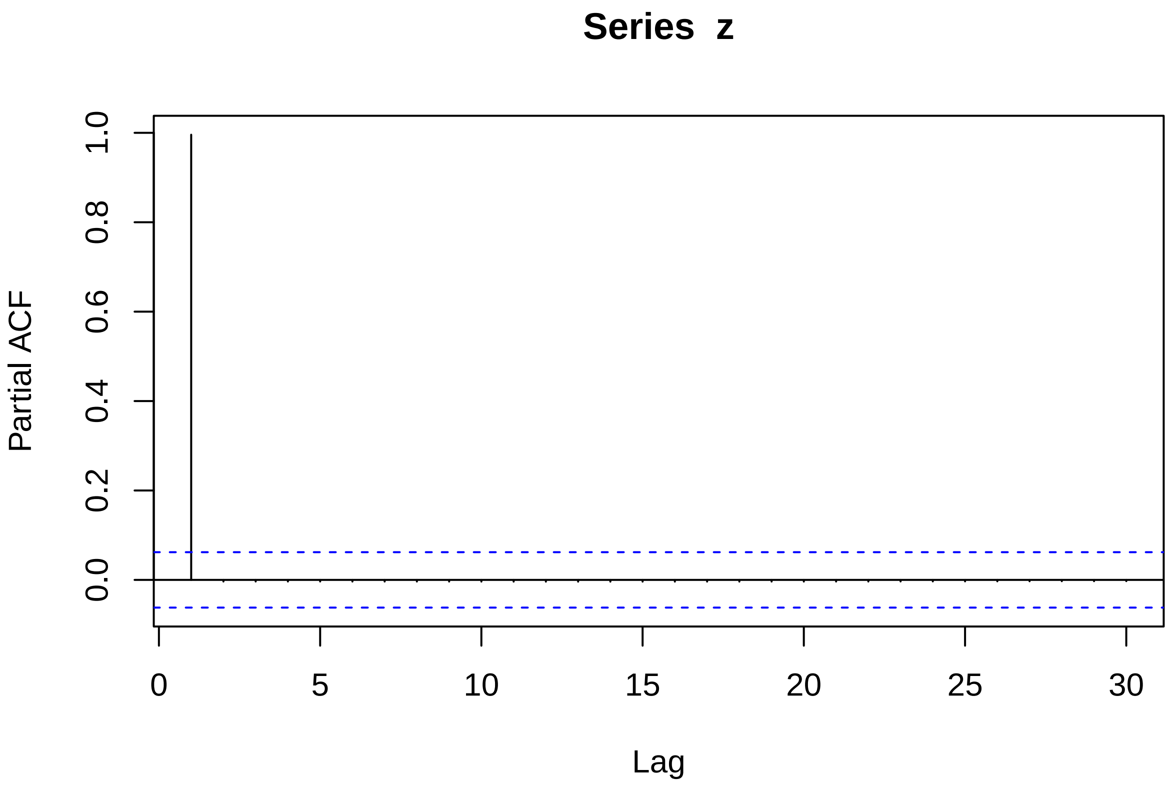
```
plot(z,
     main = "Plot of Simulated ARMA(2,2,2) Series",
     type = "l",
     col = "royalblue",
     lwd = 2.0)
```



```
acf.z <- acf(z, plot = FALSE)
pacf.z <- pacf(z, plot = FALSE)

plot(acf.z[2:30])
```



```r
plot(pacf.z)
```



Performing an Augmented Dickey Fuller Test

Let's now use our simulated I(2) series to demonstrate the Ng & Perron (1995) ADF testing procedure that we discussed in class and the tutorial. First we set the value of the max lag length for lagged differences in the ADF testing equation:

$$k_{max} = 12\left(\frac{T}{100}\right)^{\frac{1}{4}}$$

```r
library(urca)
kmax = ceiling(12*(T/100)^(1/4))
```

Then, we perform the ADF test iteratively, stating at $k = k_{max}$ and checking the absolute value of the t-statistic associated with the coefficient estimate on the last lagged difference. If the t-statistic is greater than 1.6 in absolute value, then set $k = k_{max}$ and use the testing equation to perform the test. Otherwise, reduce the lag length by one and repeat the process. To simplify the process, we can again use a loop with an if statement which stops the loop once the above condition is met. Also note below that we are estimating an ADF test in which there is a drift included in the testing equation. This is a reasonable component to include given the plot of Z_t.

```r
for (i in 1:kmax-1){

  k = kmax+1-i

  df.z = ur.df(z, type = "drift", lags = k)

  tstat = df.z@testreg$coefficients[k+2,3]
  if(abs(tstat) >= 1.6){break}
}

k
```

```
## [1] 22
```

Let's have a look at the test results implied by the above loop. The output of the ADF test performed by the **urca** package can be a little tricky to read as it will report multiple test statistics depending on the form of the testing equation used. For instance, when we include a drift term in the testing equation, it has the form:

$$\Delta Y_t = \mu + \rho Y_{t-1} + \sum_{j=2}^{k} \beta_j \Delta Y_{t-j+1} + \epsilon_t$$

Looking at the output below we can see that there are two test statistics reported. The first test statistic corresponds to the standard Dickey Fuller hypothesis test:

$$H_0 : \rho = 0 H_A : \rho \neq 0$$

And the associated critical value is labeled in the output below as *tau2*. As we can see, the test statistic does not exceed the critical value in absolute value at all levels of significance (i.e., 1%, 5% and 10%). Therefore we fail reject the null hypothesis and conclude that there exists a unit root in Z_t. An unsurprising result!

The second test statistic corresponds to the joint hypothesis test

$$H_0 : \rho = \mu = 0 H_A : \rho \text{ and/or } \mu \neq 0$$

And the associated critical value is labeled in the output below as *phi1*. If this second test statistic does not exceed this critical value in absolute value, then we can conclude that both ρ and μ are jointly equal to zero and thus we did not need to specify the ADF equation with a drift term.

On the other hand, if the second test statistic exceeds this critical value, we can conclude that either ρ and/or μ is statistically different from zero. If we have failed to reject the null hypothesis in the first test (i.e., concluded that $\rho = 0$), then this means that $\mu \neq 0$ and we were correct in including a drift term.

For practical purposes however, we need only focus on the first test.

```
summary(df.z)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.13666 -0.64977 -0.01885  0.66558  2.95263
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.419e-02  4.508e-02   0.315  0.75292    
## z.lag.1      -1.621e-06  2.303e-06  -0.704  0.48157    
## z.diff.lag1   2.190e+00  3.227e-02  67.865  < 2e-16 ***
## z.diff.lag2  -1.467e+00  7.784e-02 -18.850  < 2e-16 ***
## z.diff.lag3   2.283e-01  9.124e-02   2.502  0.01251 *  
## z.diff.lag4   1.365e-01  9.163e-02   1.490  0.13653    
## z.diff.lag5  -1.253e-01  9.191e-02  -1.364  0.17298    
## z.diff.lag6   1.718e-02  9.207e-02   0.187  0.85199    
## z.diff.lag7   9.315e-03  9.207e-02   0.101  0.91943    
## z.diff.lag8   5.821e-02  9.201e-02   0.633  0.52713    
## z.diff.lag9  -1.070e-01  9.190e-02  -1.165  0.24444    
## z.diff.lag10  8.144e-02  9.192e-02   0.886  0.37586    
## z.diff.lag11 -4.383e-02  9.192e-02  -0.477  0.63357    
## z.diff.lag12  1.081e-01  9.193e-02   1.176  0.24001    
## z.diff.lag13 -8.063e-02  9.198e-02  -0.877  0.38091    
## z.diff.lag14 -1.063e-01  9.192e-02  -1.157  0.24758    
## z.diff.lag15  1.261e-01  9.190e-02   1.372  0.17031    
## z.diff.lag16  4.572e-03  9.202e-02   0.050  0.96038    
## z.diff.lag17 -7.253e-02  9.209e-02  -0.788  0.43112    
## z.diff.lag18  2.206e-02  9.207e-02   0.240  0.81071    
## z.diff.lag19  1.057e-01  9.193e-02   1.150  0.25052    
## z.diff.lag20 -4.990e-02  9.187e-02  -0.543  0.58717    
## z.diff.lag21 -1.194e-01  7.852e-02  -1.521  0.12854    
## z.diff.lag22  8.476e-02  3.260e-02   2.600  0.00947 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.9942 on 953 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999 
## F-statistic: 6.987e+05 on 23 and 953 DF,  p-value: < 2.2e-16
## 
## 
## Value of test-statistic is: -0.7041 0.8201 
## 
## Critical values for test statistics: 
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

Having identified a unit root in Z_t, the next step is to compute its first difference and apply the ADF test to see whether this first difference is stationary.

```r
x = diff(z)

for (i in 1:kmax-1){

  k = kmax+1-i

  df.x = ur.df(x, type = "drift", lags = k)

  tstat = df.x@testreg$coefficients[k+2,3]

  if(abs(tstat) >= 1.6){break}
}
```

```
k
```

```
## [1] 21
```

Looking at the results below, we can again see that the first test statistic is less than the critical value in absolute value for all significance levels. Thus we can conclude that there exists a unit root in the first difference of Z_t.

```
summary(df.x)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.10993 -0.64683 -0.01214  0.66179  2.99584
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.0356004  0.0332694   1.070   0.2849
## z.lag.1      -0.0000340  0.0002608  -0.130   0.8963
## z.diff.lag1   1.1907204  0.0322564  36.914  < 2e-16 ***
## z.diff.lag2  -0.2771496  0.0502566  -5.515  4.5e-08 ***
## z.diff.lag3  -0.0486999  0.0510461  -0.954   0.3403
## z.diff.lag4   0.0879418  0.0511243   1.720   0.0857 .
## z.diff.lag5  -0.0376505  0.0512705  -0.734   0.4629
## z.diff.lag6  -0.0203444  0.0512909  -0.397   0.6917
## z.diff.lag7  -0.0109332  0.0513052  -0.213   0.8313
## z.diff.lag8   0.0470889  0.0511958   0.920   0.3579
## z.diff.lag9  -0.0599486  0.0511830  -1.171   0.2418
## z.diff.lag10  0.0217182  0.0511877   0.424   0.6715
## z.diff.lag11 -0.0221736  0.0511843  -0.433   0.6650
## z.diff.lag12  0.0858615  0.0512185   1.676   0.0940 .
## z.diff.lag13  0.0051471  0.0512570   0.100   0.9200
## z.diff.lag14 -0.1012075  0.0511890  -1.977   0.0483 *
## z.diff.lag15  0.0249261  0.0512852   0.486   0.6271
## z.diff.lag16  0.0295308  0.0513168   0.575   0.5651
## z.diff.lag17 -0.0431428  0.0513331  -0.840   0.4009
## z.diff.lag18 -0.0208550  0.0512745  -0.407   0.6843
## z.diff.lag19  0.0848530  0.0512786   1.655   0.0983 .
## z.diff.lag20  0.0343668  0.0506638   0.678   0.4977
## z.diff.lag21 -0.0833700  0.0325332  -2.563   0.0105 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9939 on 954 degrees of freedom
## Multiple R-squared:  0.8711, Adjusted R-squared:  0.8682
## F-statistic: 293.1 on 22 and 954 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -0.1303 0.5835
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

We now have to difference the data again and apply the ADF test to see whether this second difference is stationary.

```
w = diff(x)

for (i in 1:kmax-1){

  k = kmax+1-i

  df.w = ur.df(w, type = "drift", lags = k)

  tstat = df.w@testreg$coefficients[k+2,3]

  if(abs(tstat) >= 1.6){break}
}

k
```

```
## [1] 20
```

Looking at the results below, we now see that the first test statistic is greater than the critical value in absolute value for all significance levels. Thus we reject the null hypothesis and conclude that the second difference is stationary. Since we have had to difference Z_t twice in order to achieve stationarity, we can conclude that Z_t is integrated of order 2 (i.e., I(2)). This should be an unsurprising result as we have constructed Z_t to have this property!

```
summary(df.w)
```

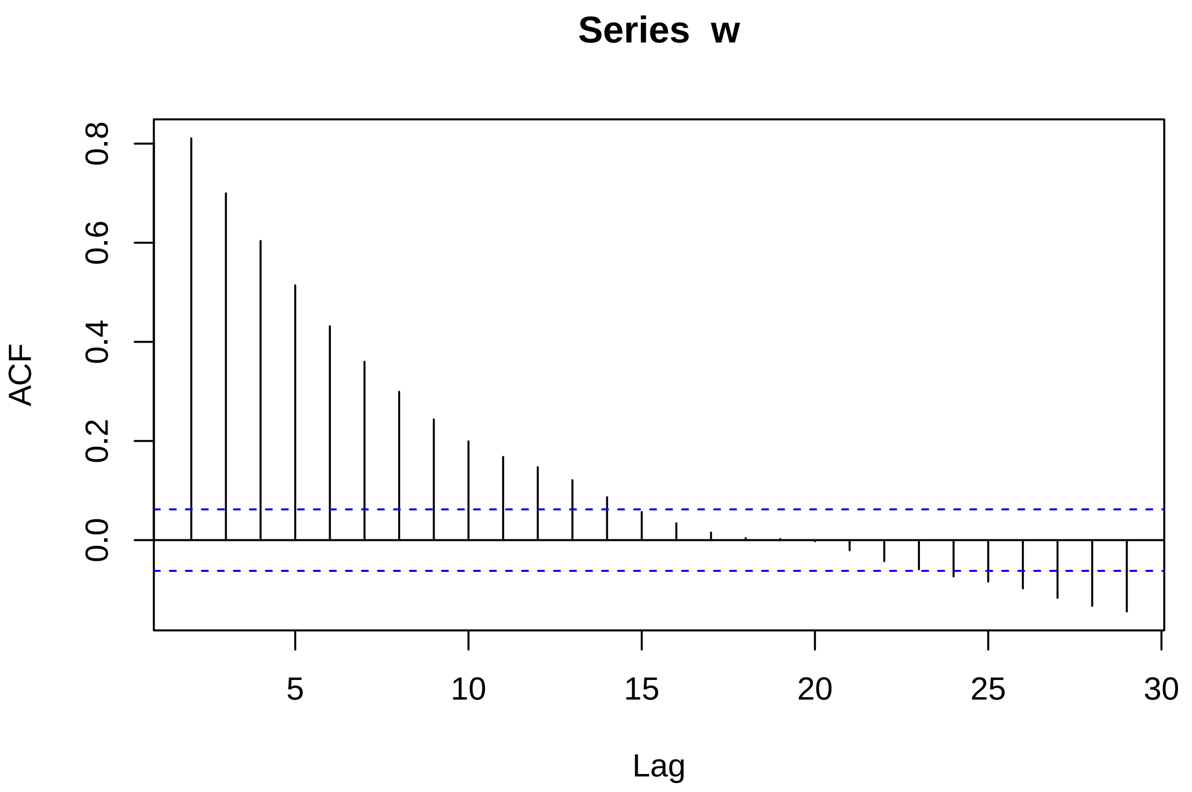
```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression drift
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.11239 -0.64909 -0.01422  0.66233  2.99484
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.034491   0.032145   1.073   0.2836
## z.lag.1      -0.113995   0.018428  -6.186 9.14e-10 ***
## z.diff.lag1   0.304728   0.034061   8.946  < 2e-16 ***
## z.diff.lag2   0.027565   0.035414   0.778   0.4365
## z.diff.lag3  -0.021193   0.035085  -0.604   0.5460
## z.diff.lag4   0.066726   0.034860   1.914   0.0559 .
## z.diff.lag5   0.029038   0.034862   0.833   0.4051
## z.diff.lag6   0.008660   0.034676   0.250   0.8028
## z.diff.lag7  -0.002302   0.034407  -0.067   0.9467
## z.diff.lag8   0.044767   0.034346   1.303   0.1927
## z.diff.lag9  -0.015194   0.034336  -0.443   0.6582
## z.diff.lag10  0.006506   0.034137   0.191   0.8489
## z.diff.lag11 -0.015684   0.034009  -0.461   0.6448
## z.diff.lag12  0.070137   0.033727   2.080   0.0378 *
## z.diff.lag13  0.075265   0.033712   2.233   0.0258 *
## z.diff.lag14 -0.025973   0.033558  -0.774   0.4392
## z.diff.lag15 -0.001082   0.033399  -0.032   0.9742
## z.diff.lag16  0.028437   0.033315   0.854   0.3936
## z.diff.lag17 -0.014728   0.033299  -0.442   0.6584
## z.diff.lag18 -0.035616   0.032992  -1.080   0.2806
## z.diff.lag19  0.049238   0.033011   1.492   0.1361
## z.diff.lag20  0.083651   0.032445   2.578   0.0101 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9934 on 955 degrees of freedom
## Multiple R-squared:  0.1469, Adjusted R-squared:  0.1281
## F-statistic: 7.828 on 21 and 955 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -6.186 19.1445
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

Estimating and Forecasting an Integrated Time Series

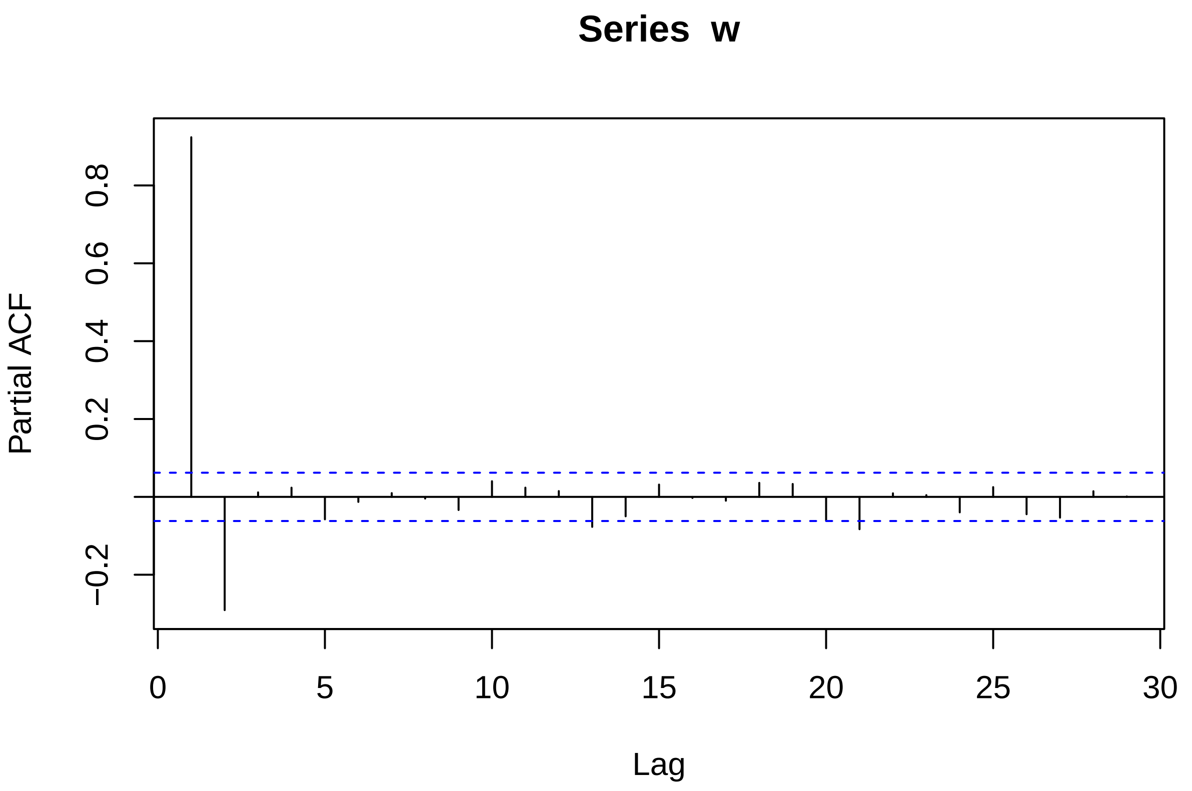
Having determined the order of integration for Z_t, we can proceed to specify and estimate our model in the same way as before.

Looking at the sample PACF of our second differenced data, we can see that we should not be looking at models with very high AR or MA orders.

```r
acf.w <- acf(w, plot = FALSE)
pacf.w <- pacf(w, plot = FALSE)

plot(acf.w[2:30])
```



```
plot(pacf.w)
```



So we will search from an ARIMA(1,2,0) to an ARIMA(4,2,4). We will use a slightly modified version of the loop that I wrote for searching across a set of candidate ARMA specifications. The modification allows the loop to continue when there is an error in the estimation:

```r
library(forecast)
```

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

```r
pstart = 1
pend = 4

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(NA,(pend-pstart+1),qend+1))
colnames(akaike) <- ma.label

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

for(i in pstart:pend){

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

    model <- try(Arima(z,
                    order = c(i,2,j-1),
                    include.mean = FALSE,
                    include.drift = FALSE,
                    method = "CSS-ML"))
    if(!inherits(model, "try-error") && model$code == 0){
       akaike[i,j] <- AIC(model)
       bayes[i,j]<- BIC(model)}
  }
}
```

Then, we will want to identify the model that yields the lowest AIC or BIC

```r
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)
print(akaike.choice)
```

```
## [1] "ar3" "ma4"
```

```r
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)
```

```
print(bayes.choice)
```

```
## [1] "ar2" "ma0"
```

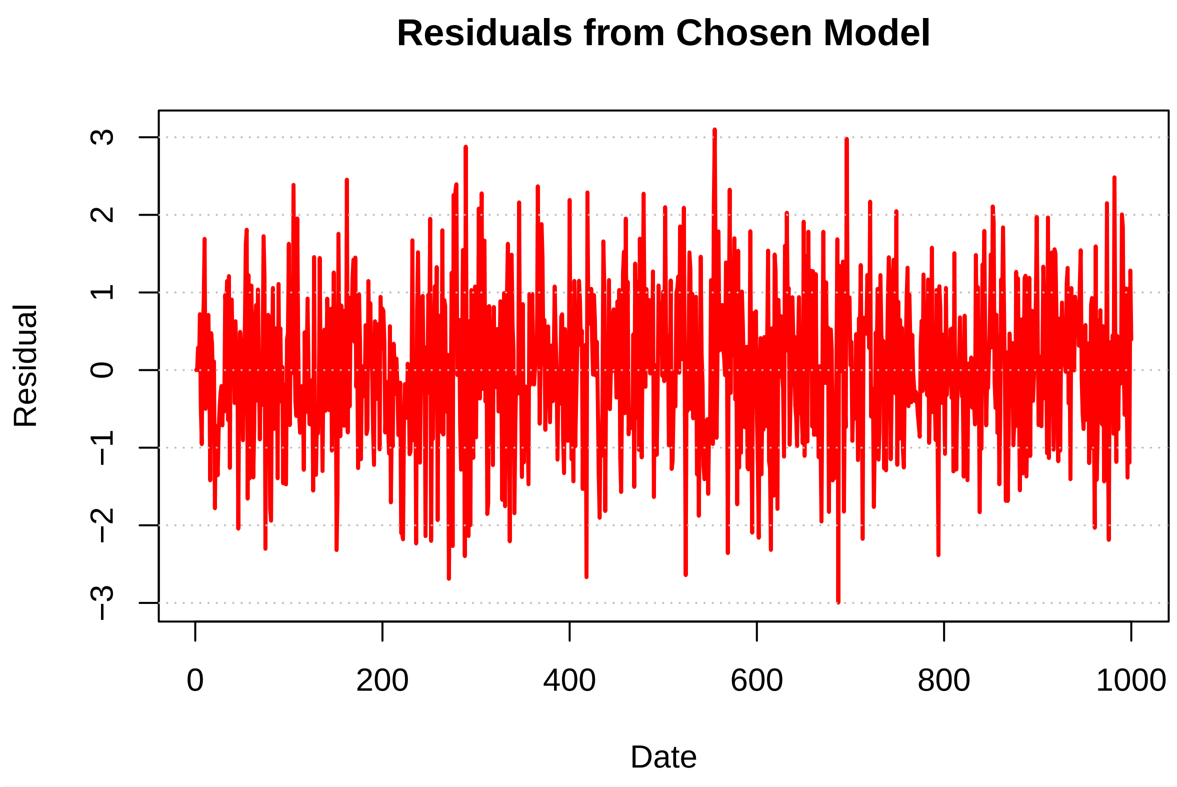
We can compare this to the one chosen by the **auto.arima()** function:

```
bestmod <- auto.arima(z, d = 2, max.p = 4, max.q = 4)
bestmod
```

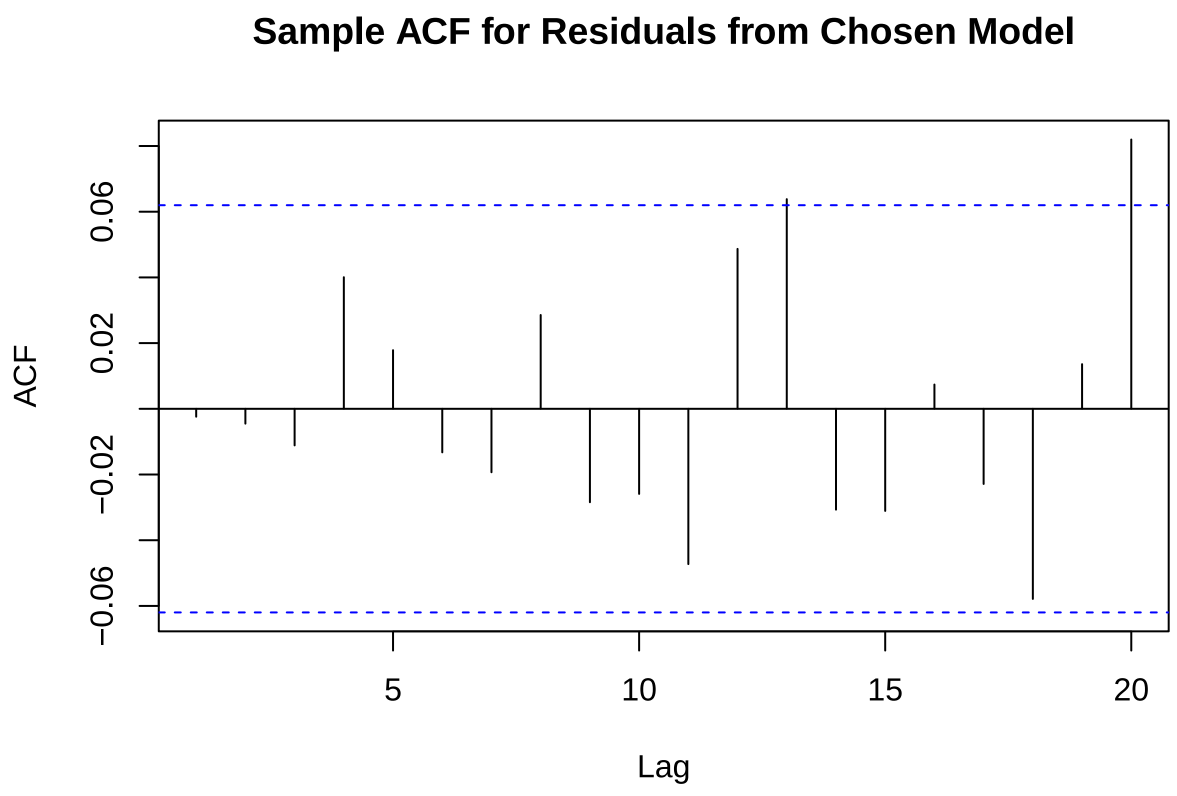
```
## Series: z
## ARIMA(1,2,2)
##
## Coefficients:
##          ar1     ma1     ma2
##       0.8613  0.3401  0.1115
## s.e.  0.0194  0.0369  0.0358
##
## sigma^2 = 0.9903:  log likelihood = -1410.8
## AIC=2829.6   AICc=2829.64   BIC=2849.22
```

Now let's have a look at the residuals from our chosen model:

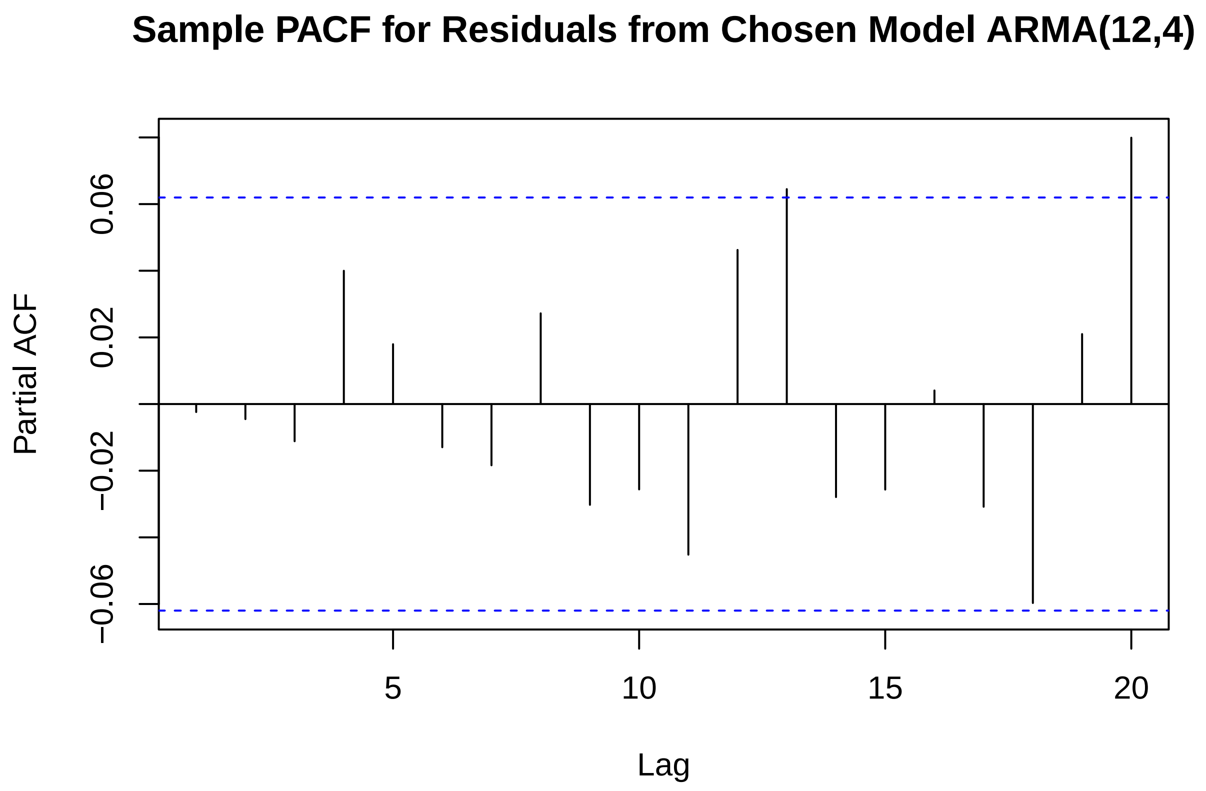
```
res = resid(bestmod)

plot(res,
     main = "Residuals from Chosen Model",
     xlab = "Date",
     ylab = "Residual",
     type = "l",
     col = "red",
     lwd = 2.0)
grid(nx = NA, ny = NULL,
     col = "grey", lwd = 1)
```



```r
acf.res <- acf(res, plot = FALSE)

pacf.res <- pacf(res, plot = FALSE)

plot(acf.res[1:20], main = "Sample ACF for Residuals from Chosen Model")
```



```r
plot(pacf.res[1:20], main = "Sample PACF for Residuals from Chosen Model ARMA(12,4)")
```



```r
m <- ceiling(sqrt(T))

Box.test(res, lag = m, type = "Box-Pierce")
```

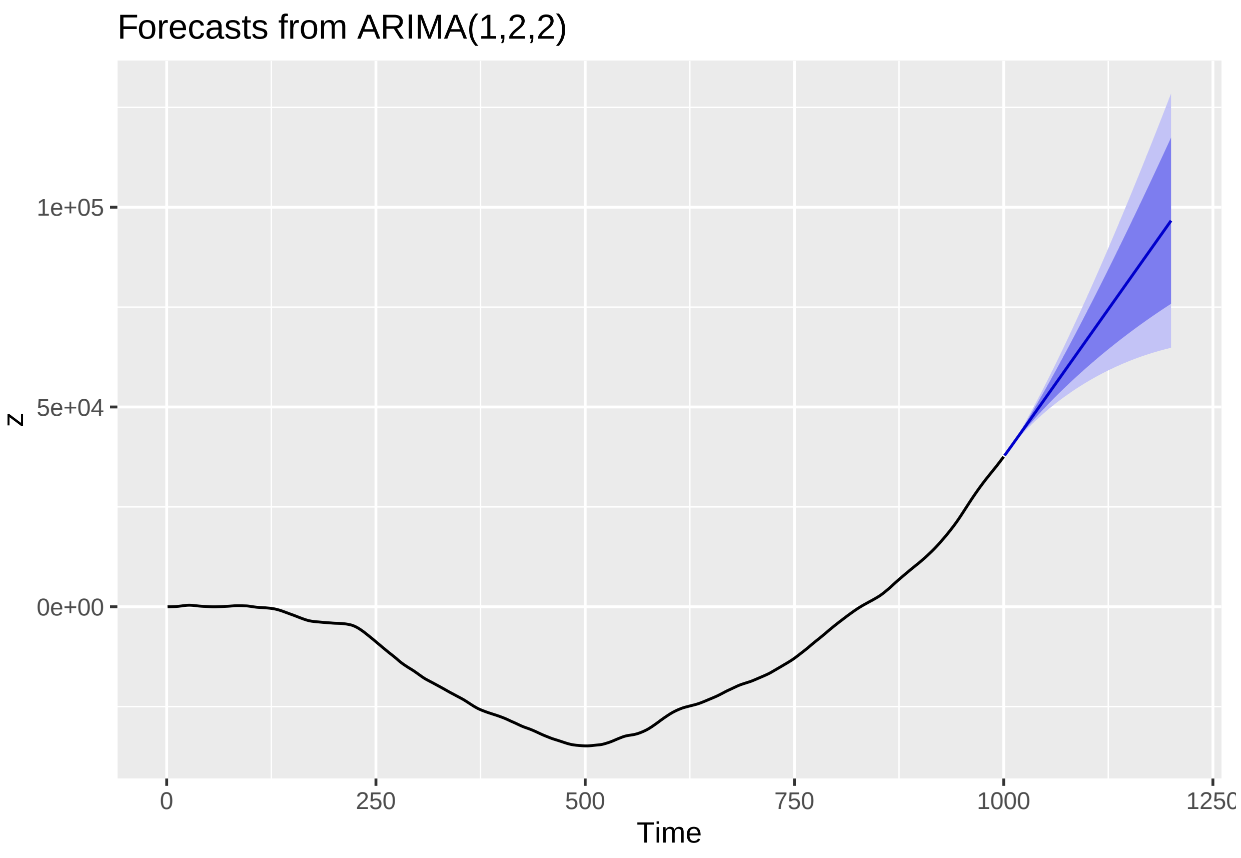
```
##
##  Box-Pierce test
##
## data:  res
## X-squared = 39.13, df = 32, p-value = 0.1802
```

```r
Box.test(res, lag = m, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  res
## X-squared = 39.949, df = 32, p-value = 0.1578
```

Finally, we can proceed to use the **forecast()** function to generate point and interval forecasts from our chosen model:

```r
h = 200

bestmod.for <- forecast(bestmod, h)

autoplot(bestmod.for)
```


Forecasts from ARIMA(1,2,2)
z
1e+05
5e+04
0e+00
0
250
500
750
1000
1250
Time