Continuous Assessment Homework 1

June 3, 2024

Manuel Andrés Hernández Alonso, niub20274855

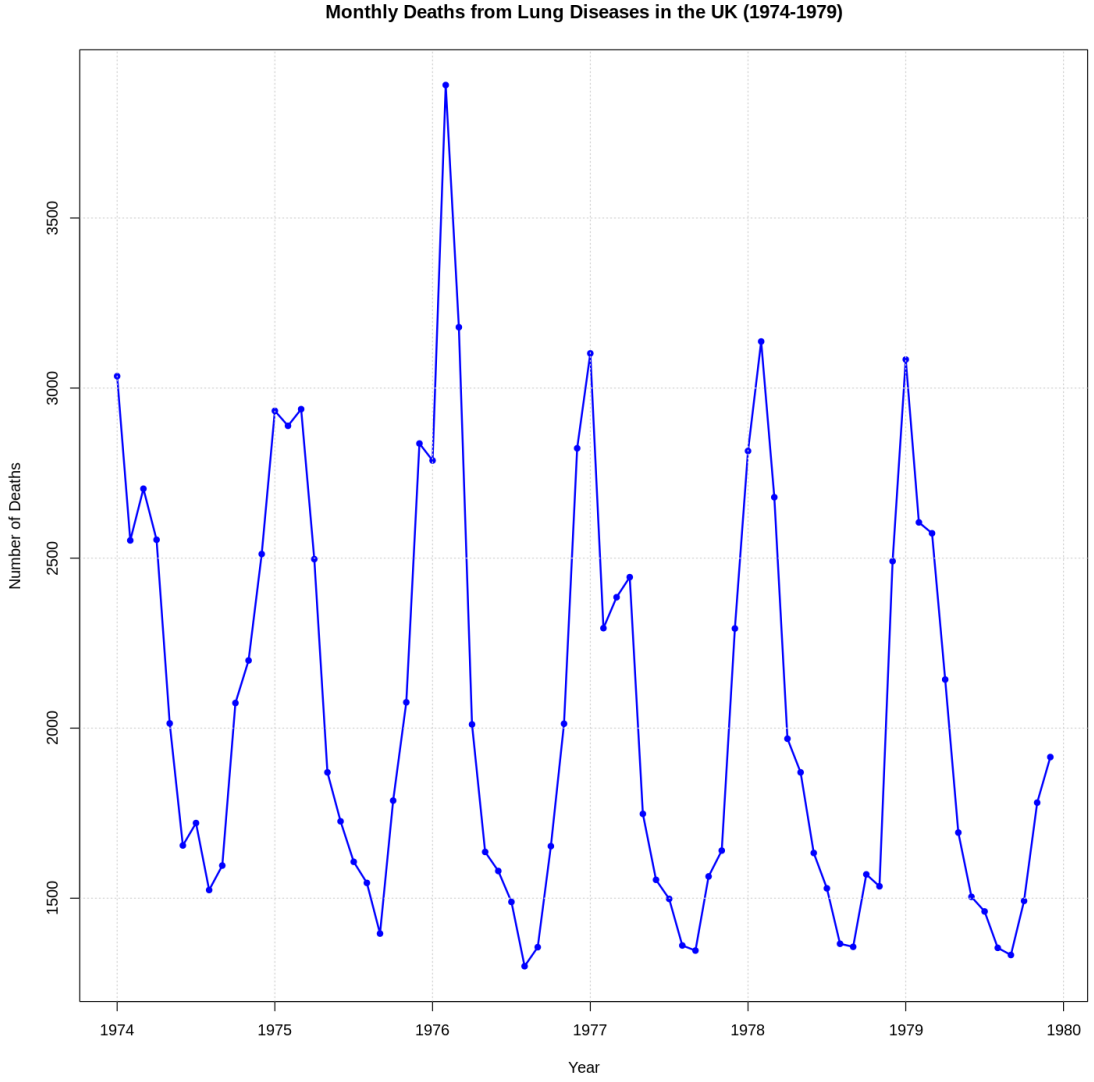
[287]:
```
library(repr)
options(repr.plot.width = 12, repr.plot.height = 12)
```

1 Exercise 1

Read the file ldeaths in the folder datasets of R. Make the graphical representation. Identify and estimate the trend, the seasonal component and the residual component. Are the residuals a sample of an IID noise?

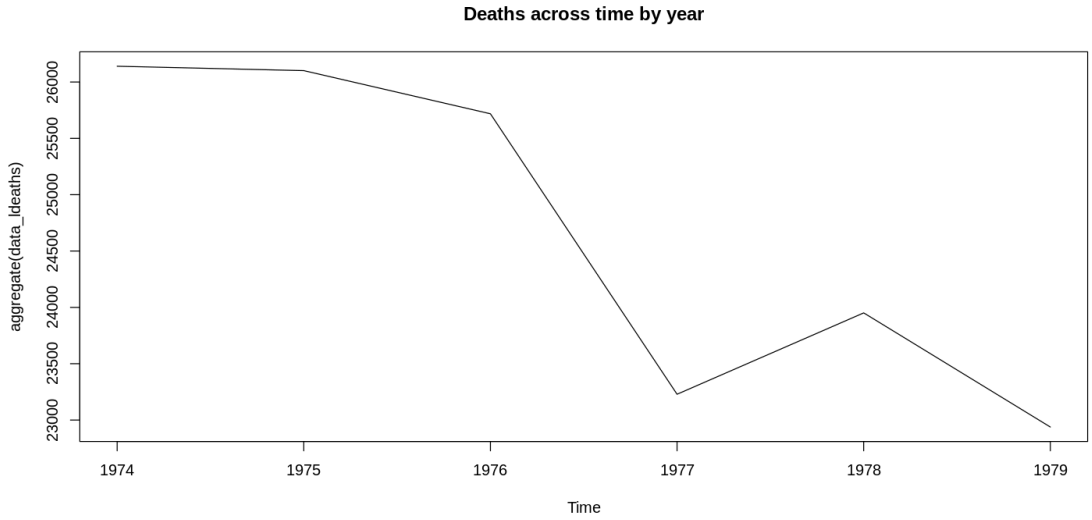
Firstly, we'll plot the data of the ldeaths dataset from the R datasets.

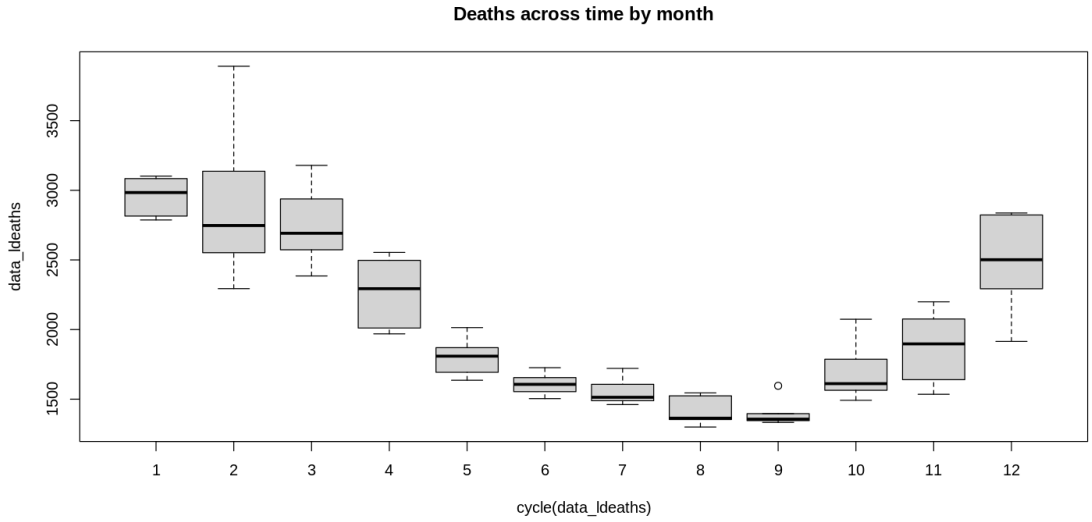
[395]:
```
data_ldeaths <- ldeaths

plot(data_ldeaths,
     main="Monthly Deaths from Lung Diseases in the UK (1974-1979)",
     ylab="Number of Deaths",
     xlab="Year",
     col="blue",
     lwd=2,
     type="o",
     pch=20)

grid()
```



[396]:
```
t <- 1:(6*12) # Generate the indexes of the timepoints
```

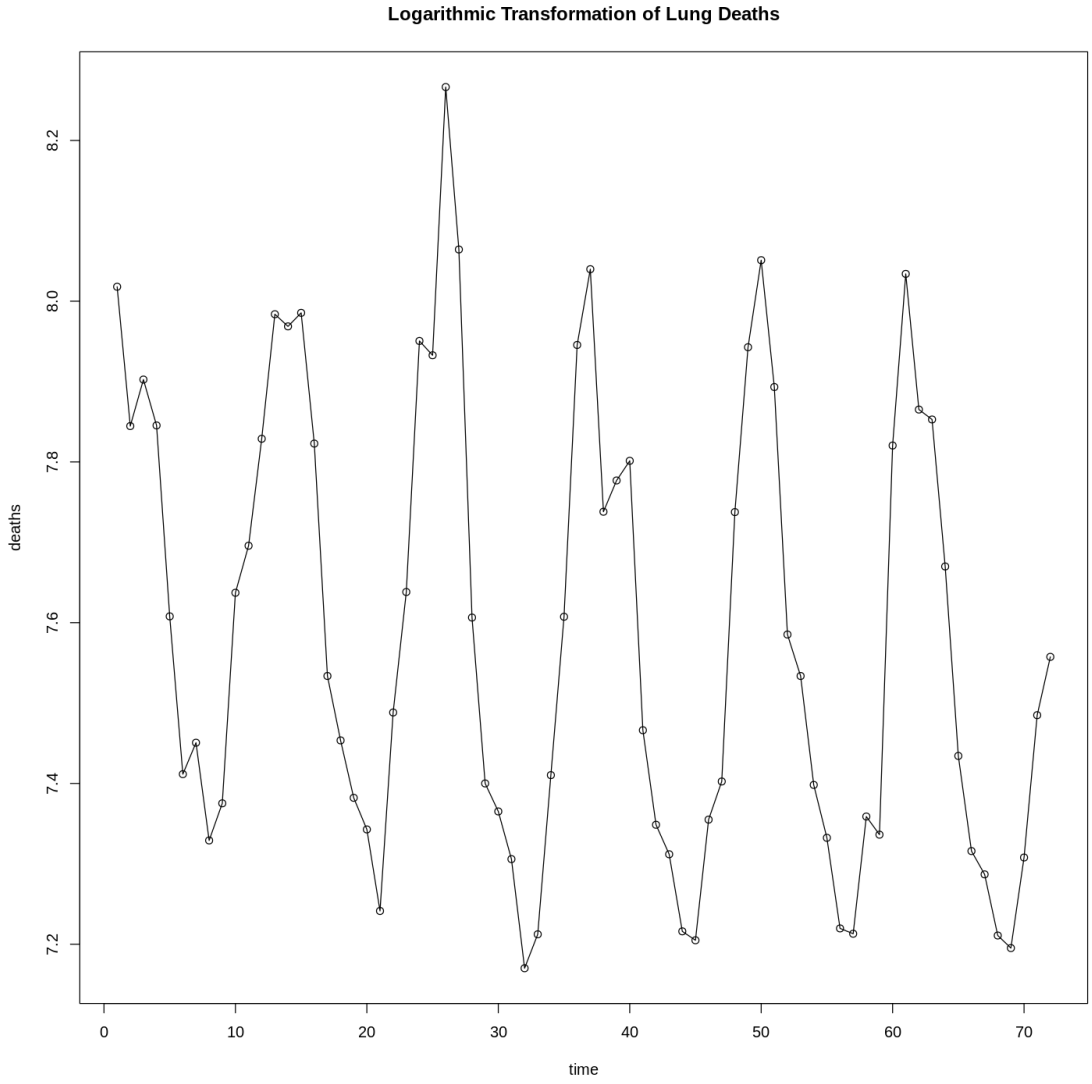
[397]:
```
layout (1:2)
plot(aggregate(data_ldeaths), main="Deaths across time by year")
boxplot(data_ldeaths~cycle(data_ldeaths), main="Deaths across time by month")
```





Here we can see the data across years, and the mean and variance of the month cycles.

[398]:
```
y<-log(data_ldeaths)
plot (t, y, xlab="time", ylab="deaths", type="o" , main="Logarithmic␣
 ↪Transformation of Lung Deaths")
```



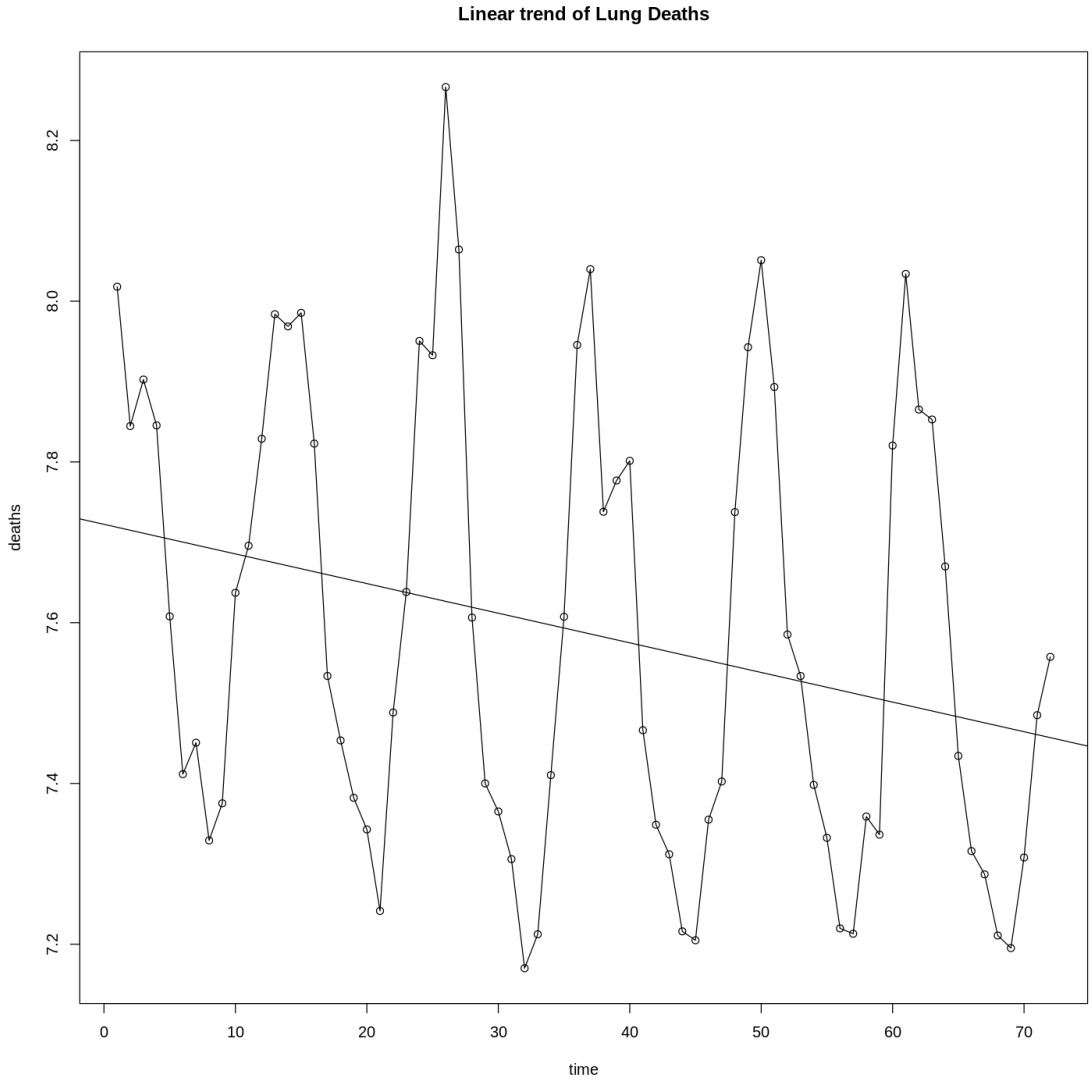
We now proceed to calculate the linear trend of the data, as we can see we get an intercept b of approximately 7.7223 and a t of -0.0036 in:

$$y = tX + b$$

[399]:
```
line<-lm(y~t)
summary(line)
plot(t,y,type="o",xlab="time",ylab="deaths", main="Linear trend of Lung Deaths");
 ↪abline(line)
```

```
Call:
lm(formula = y ~ t)
```

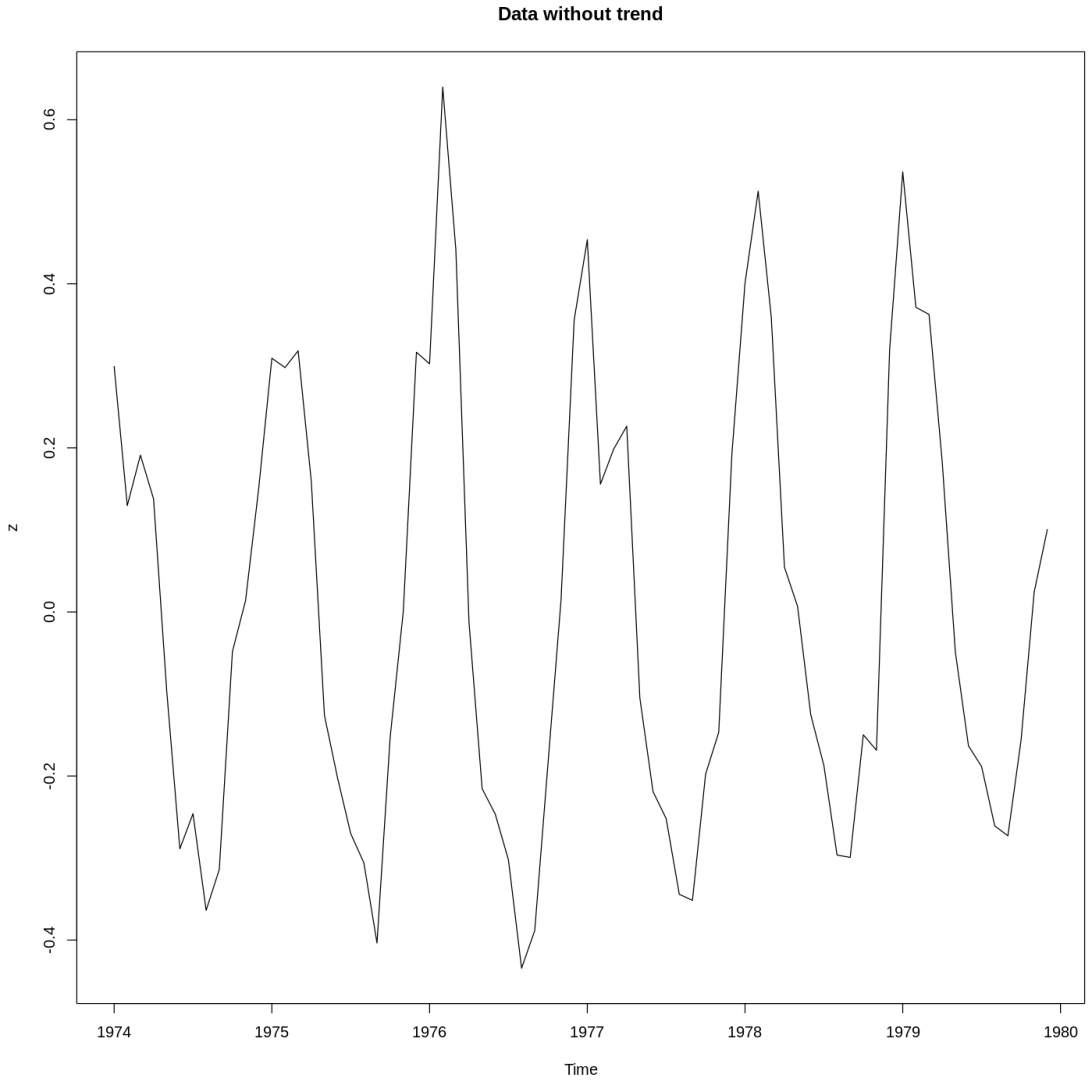
```
Residuals:
     Min       1Q   Median       3Q      Max
-0.43429 -0.22570 -0.04839  0.20538  0.63989

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.722376   0.065789 117.381   <2e-16 ***
t           -0.003686   0.001566  -2.353   0.0214 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2762 on 70 degrees of freedom
Multiple R-squared:  0.07332,        Adjusted R-squared:  0.06009
F-statistic: 5.539 on 1 and 70 DF,  p-value: 0.02141
```



Now we can remove the trend from the data.

[400]:
```
y<-log(data_ldeaths)
lm(y~t)
trend<- -0.003686*t+7.722376
z<-y-trend
plot.ts(z, main="Data without trend")
```

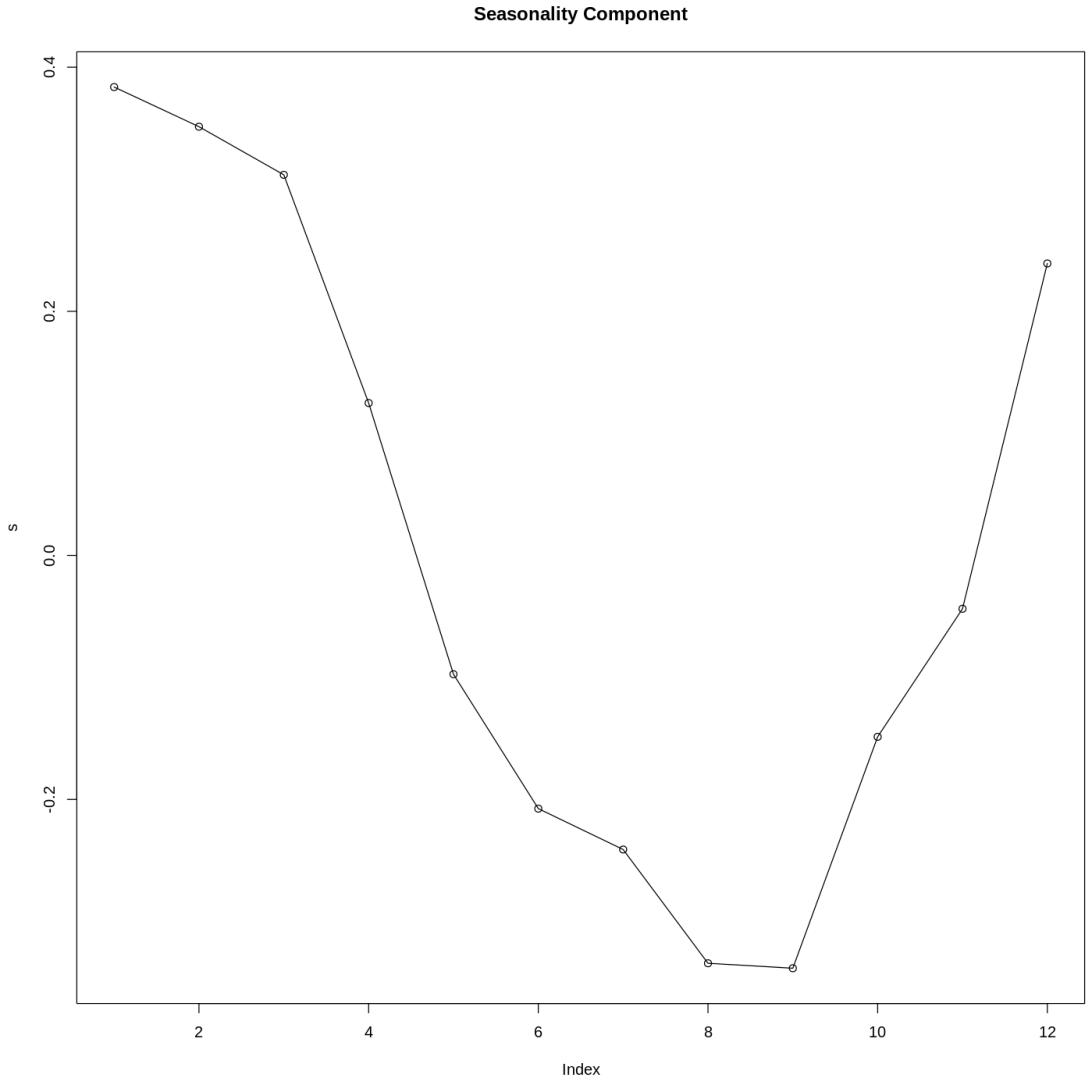
```
Call:
lm(formula = y ~ t)

Coefficients:
(Intercept)            t
   7.722376    -0.003686
```



We can now compute the seasonality component by segmenting the time series into a matrix with 12 columns that represent the months.

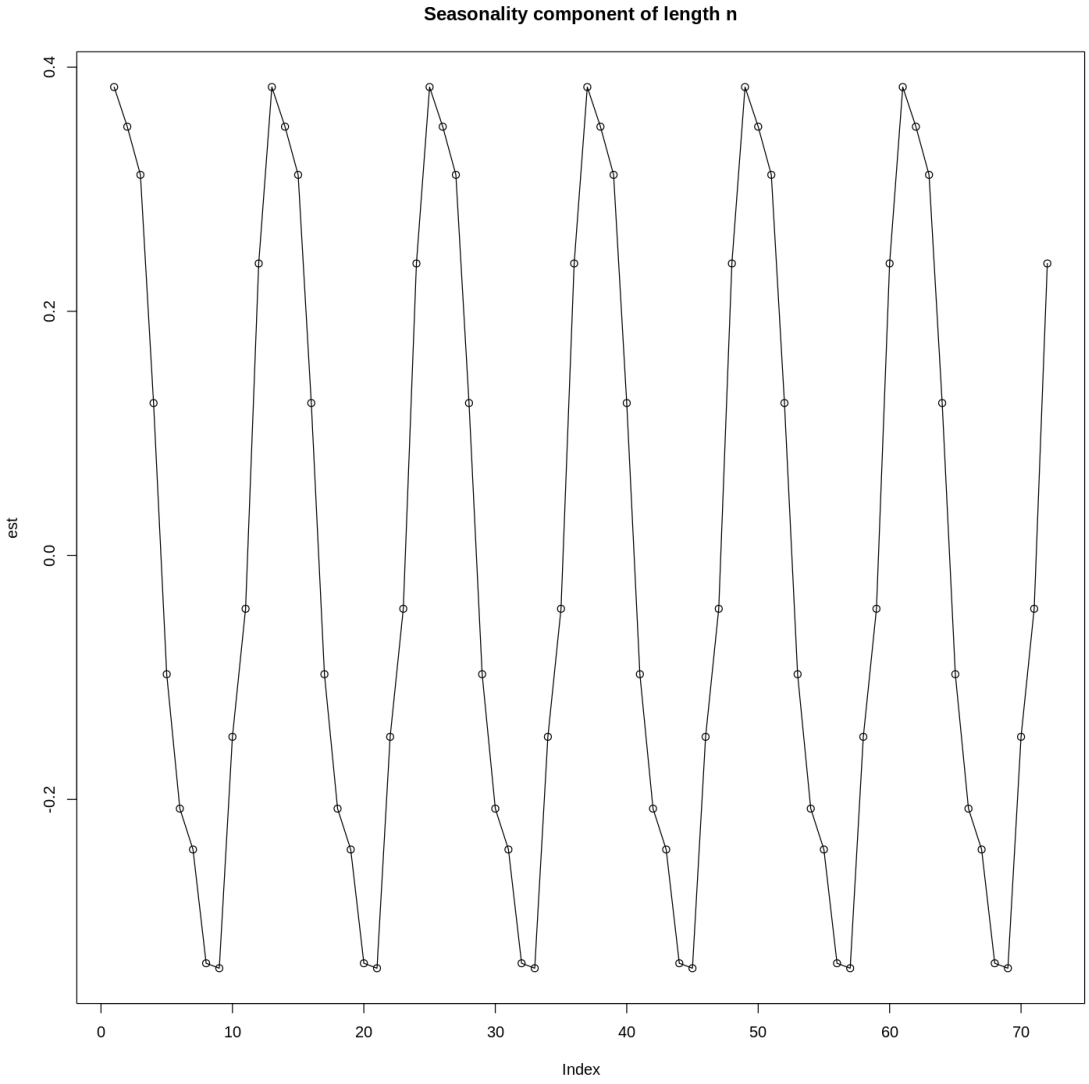
[401]:
```
z <- matrix(z, ncol=12, byrow=TRUE)
z <- t(z)
```

[402]:
```
e<-apply(z,1,mean)
ee<-mean(e)
s<-e-ee
plot (s,type="o", main="Seasonality Component")
```

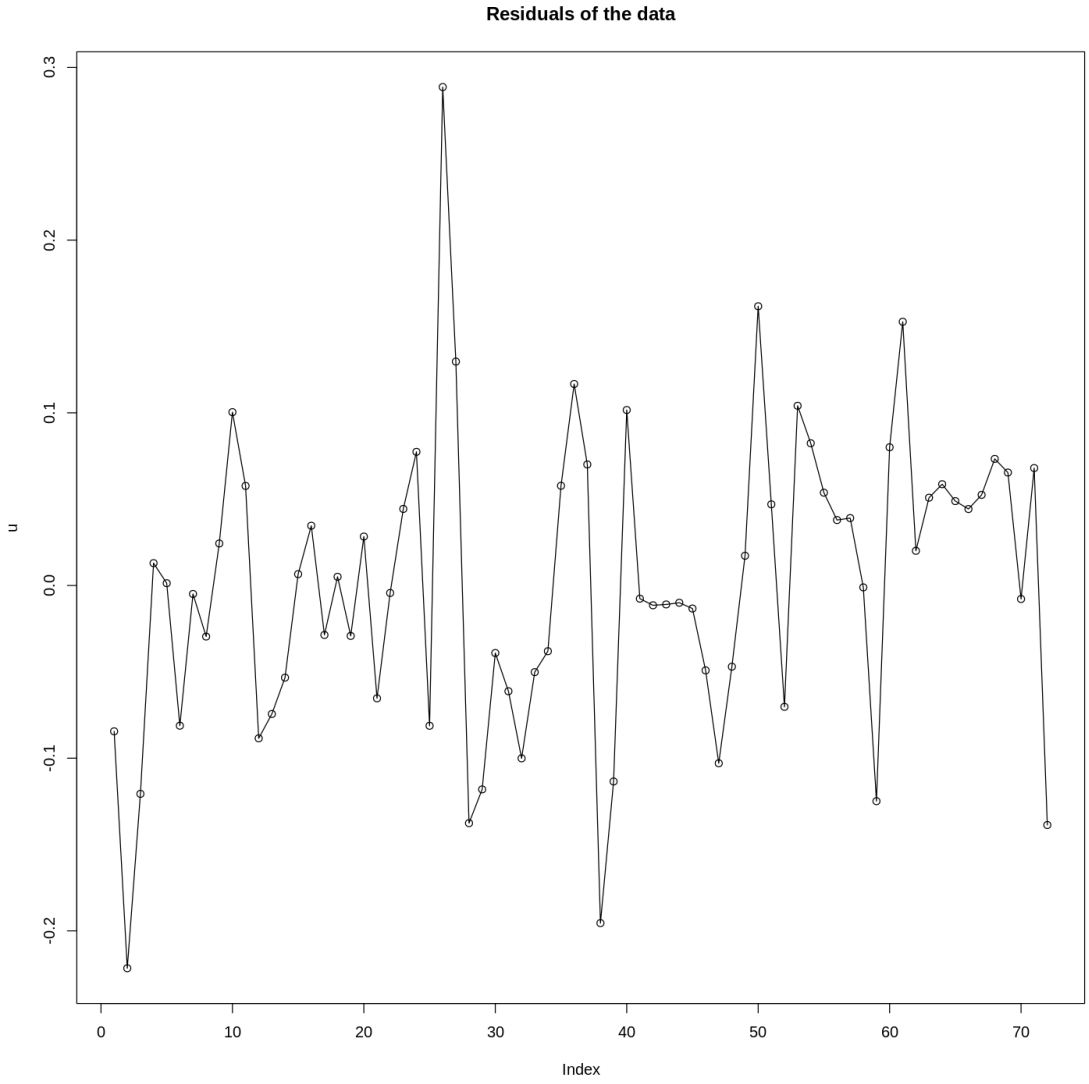


We can now repeat this component to match the length of the original series and substract it from the data.

[403]:
```
est<-array(s,12*6)
plot (est,type="o", main="Seasonality component of length n")
```

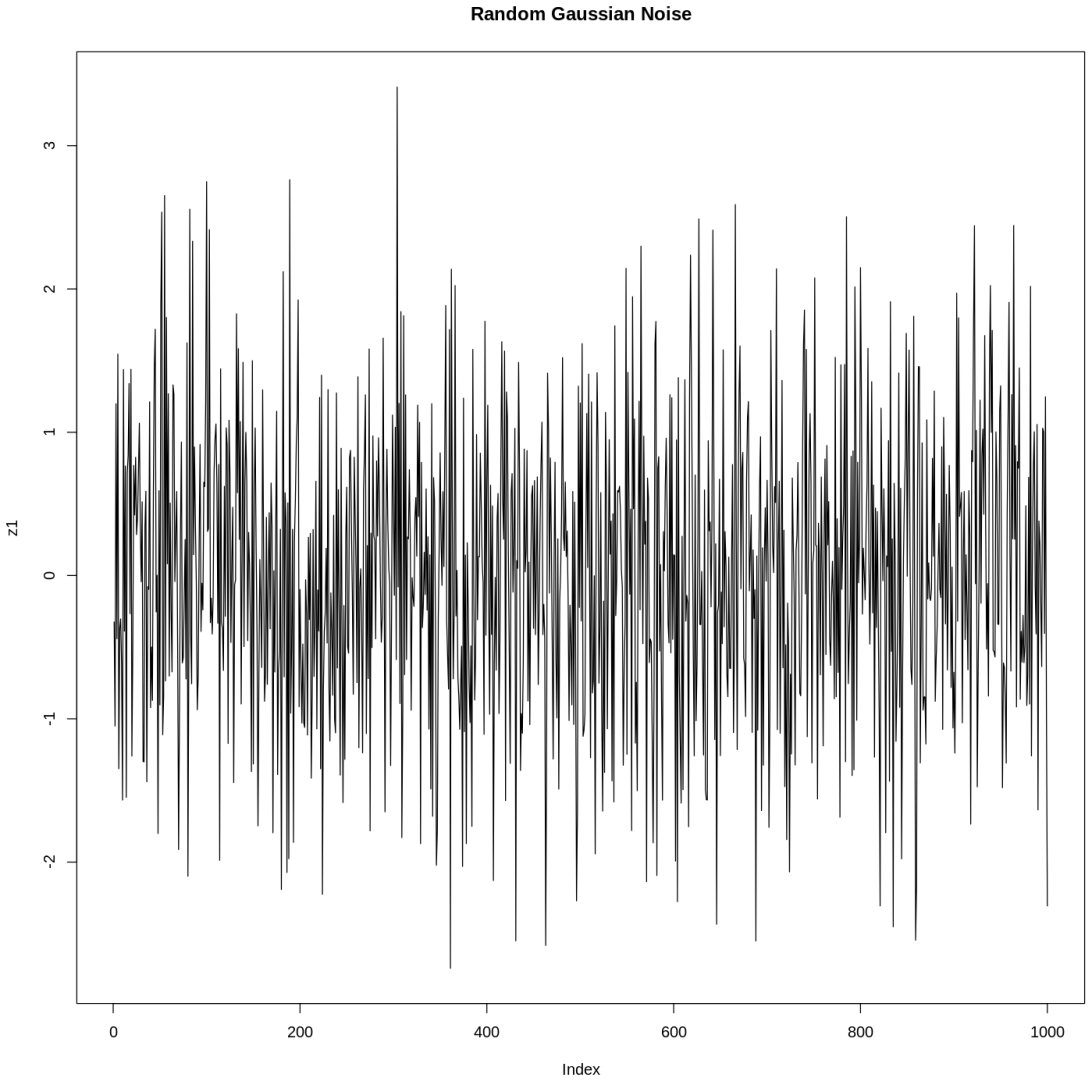


[404]:
```
dim(z)<-12*6
u<-z-est
plot (u,type="o", main="Residuals of the data")
```

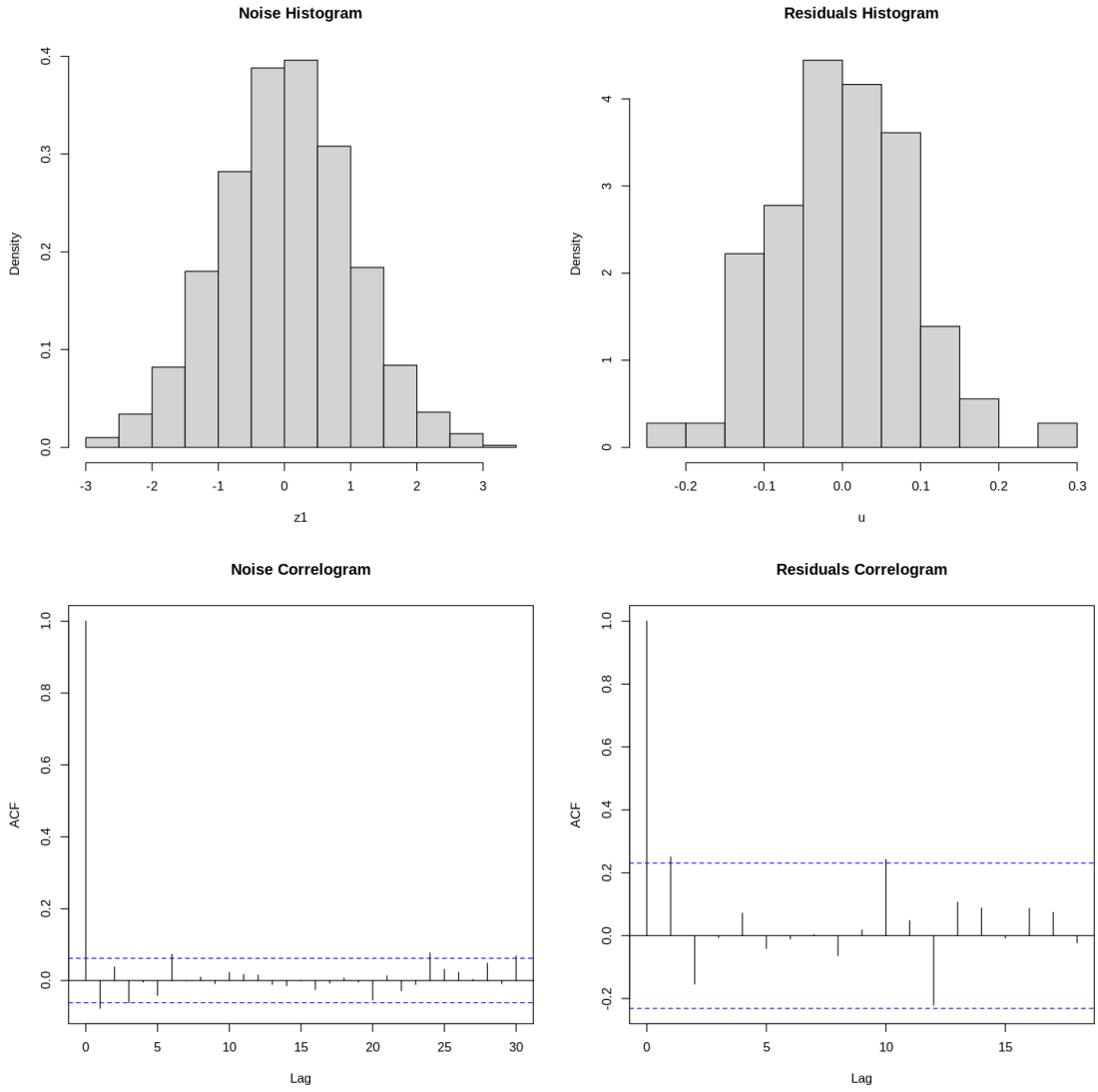


This last plot represents the residuals of the data, let us now check if it truly is IID noise.

[405]:
```
n<-1000
z1<-rnorm(n)
plot(z1,type="l", main="Random Gaussian Noise")
```



[406]:
```
par(mfrow = c(2, 2))
hist(z1,freq=F, main = "Noise Histogram")
hist(u,freq=F, main = "Residuals Histogram")
acf(z1, main = "Noise Correlogram")
acf(u, main = "Residuals Correlogram")
par(mfrow = c(1, 1))
```

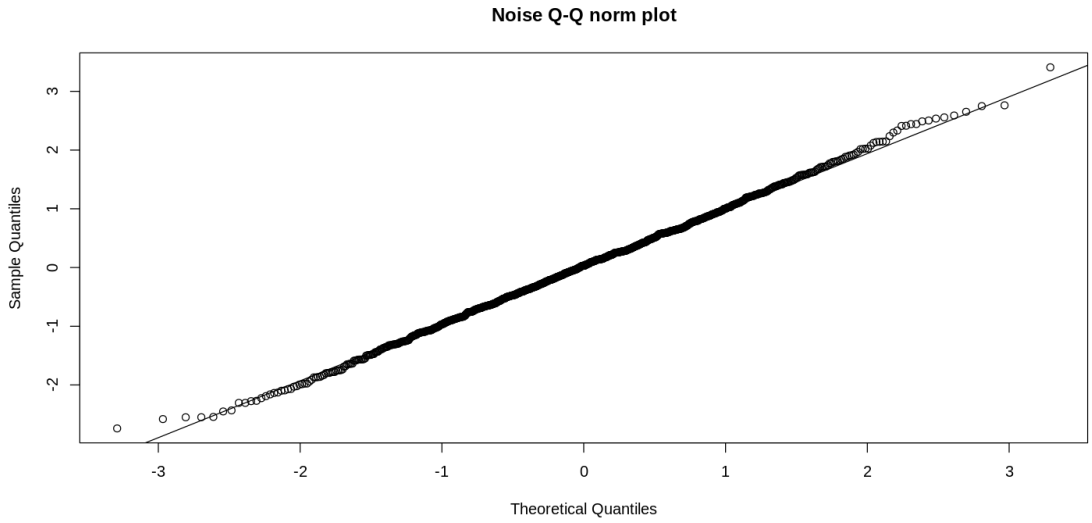


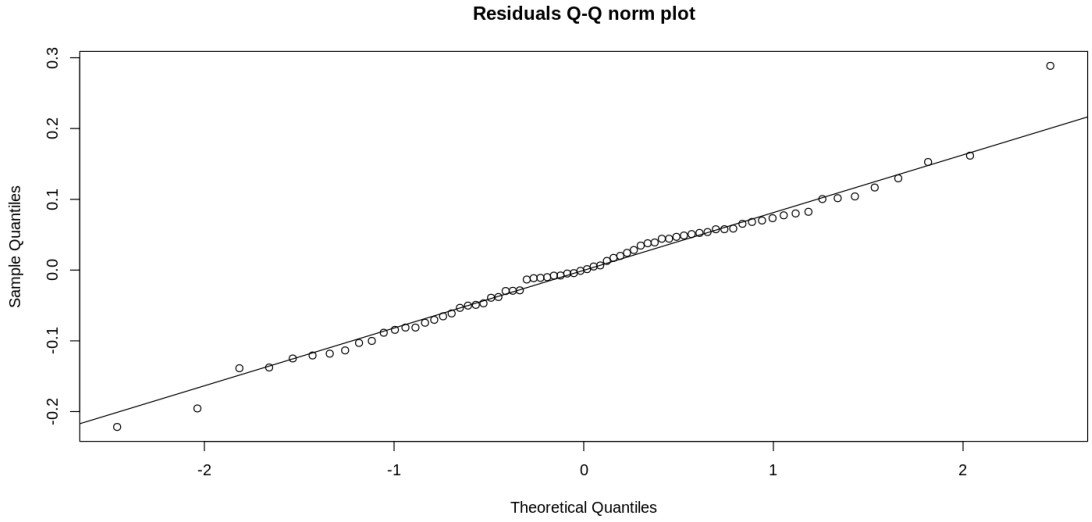
[407]:
```
Box.test (z1, lag=1, type=c("Ljung-Box"))
Box.test (u, lag=1, type=c("Ljung-Box"))
```

```
        Box-Ljung test

data:  z1
X-squared = 5.9906, df = 1, p-value = 0.01438


        Box-Ljung test

data:  u
```

```
X-squared = 4.7035, df = 1, p-value = 0.0301
```

[408]:
```
par(mfrow = c(2,1))
qqnorm(z1, main="Noise Q-Q norm plot")
qqline(z1, main="Noise Q-Q line plot")
qqnorm(u, main="Residuals Q-Q norm plot")
qqline(u, main="Residuals Q-Q line plot")
par(mfrow = c(1,1))
```





[409]:
```
shapiro.test(z1)
shapiro.test(u)
```

```
        Shapiro-Wilk normality test

data:  z1
W = 0.99891, p-value = 0.8259


        Shapiro-Wilk normality test

data:  u
W = 0.98442, p-value = 0.5167
```

As we can see, the correlogram and the tests seem to indicate that the residuals are IID noise. Nevertheless, the residuals still have some autocorrelation after h lags.

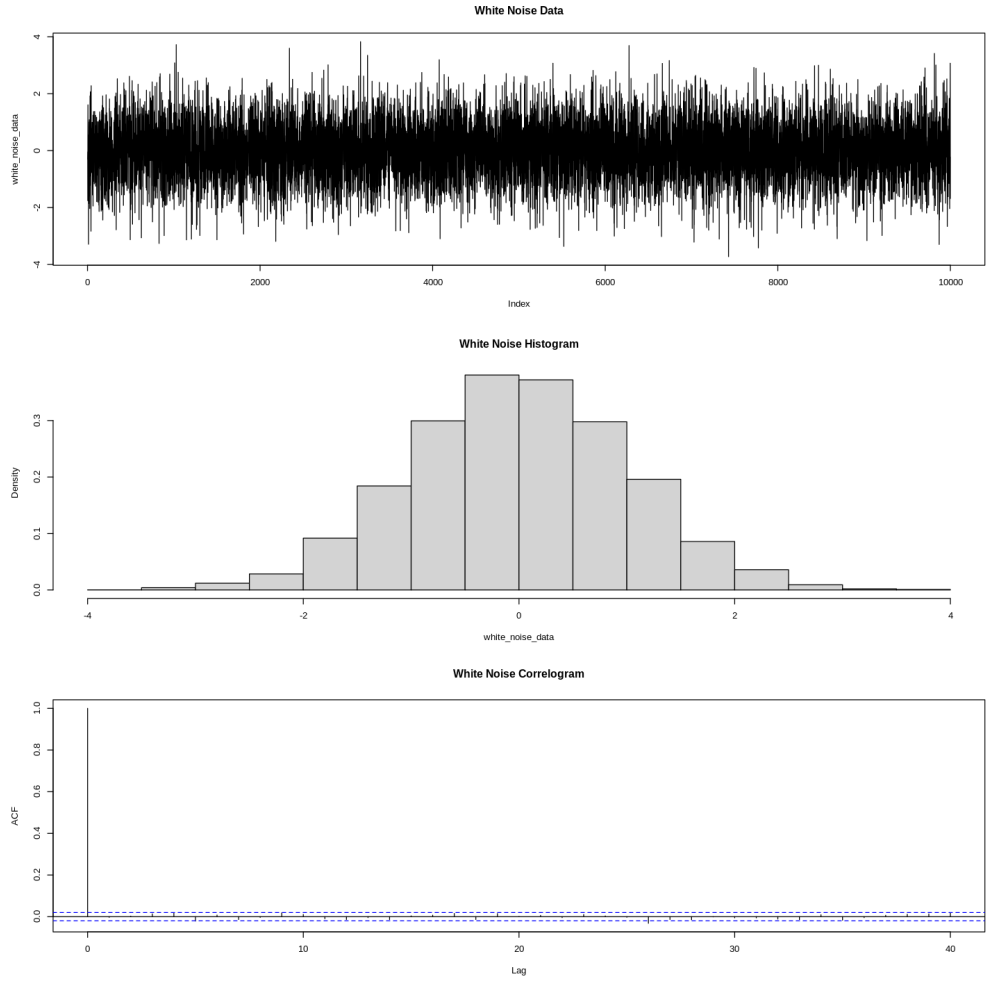
2 Exercise 2

Simulate a Gaussian white noise of $n = 10.000$ data. Verify by testing that it is an IID noise and a Gaussian white noise. Simulate a Gaussian Random Walk. Simulate IID noises of 10.000 data that are not a Gaussian white noise: a Poisson noise and an exponential noise. Test all what you can.

Firstly let us simulate gaussian white noise with $n = 10000$.

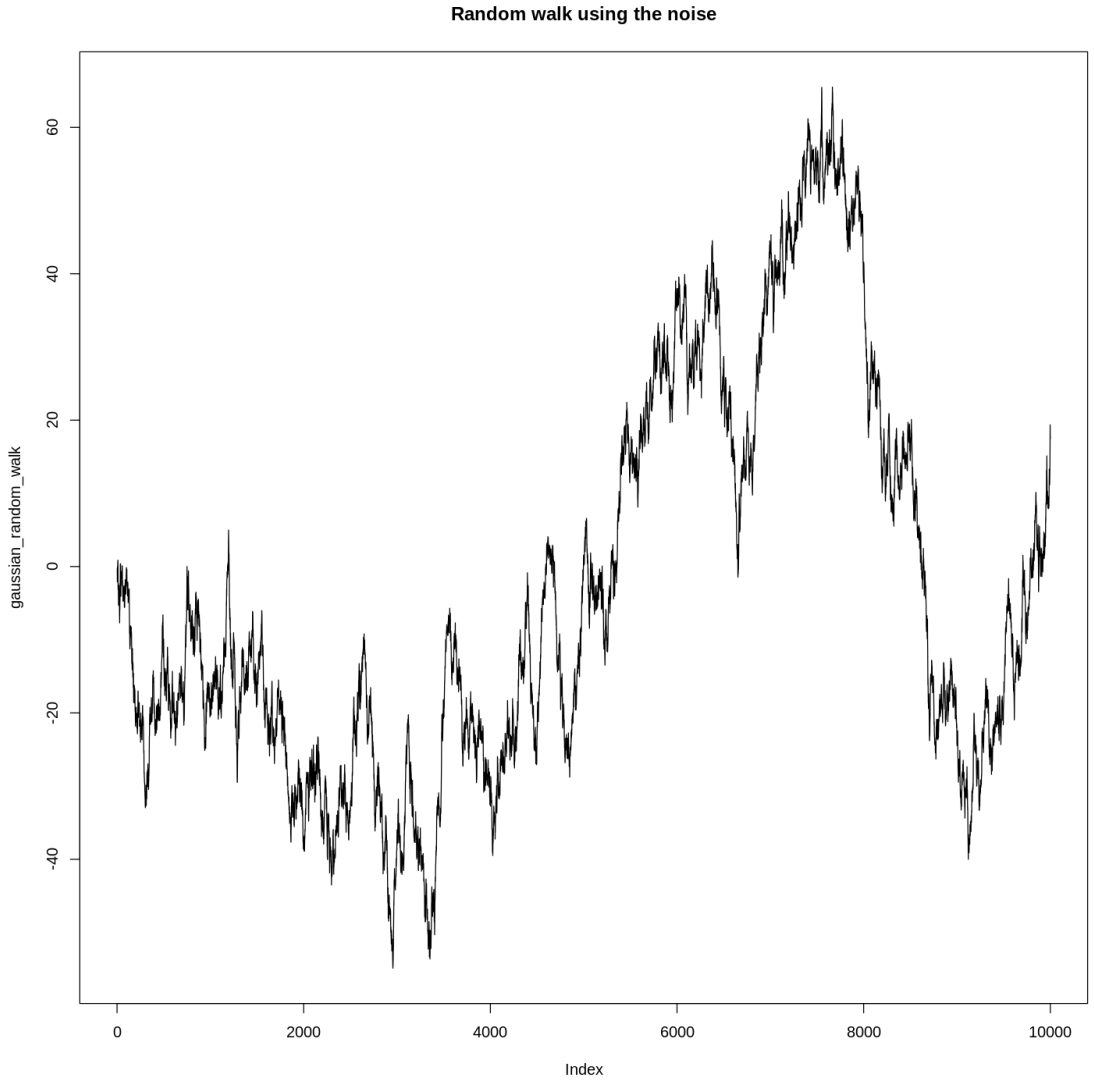
[428]:
```
n<-10000
white_noise_data<-rnorm(n) # Generate the white noise
```

[429]:
```
par(mfrow = c(3, 1))
plot(white_noise_data,type="l", main="White Noise Data")
hist(white_noise_data,freq=F, main="White Noise Histogram")
acf(white_noise_data, main="White Noise Correlogram")
par(mfrow = c(1, 1))
```

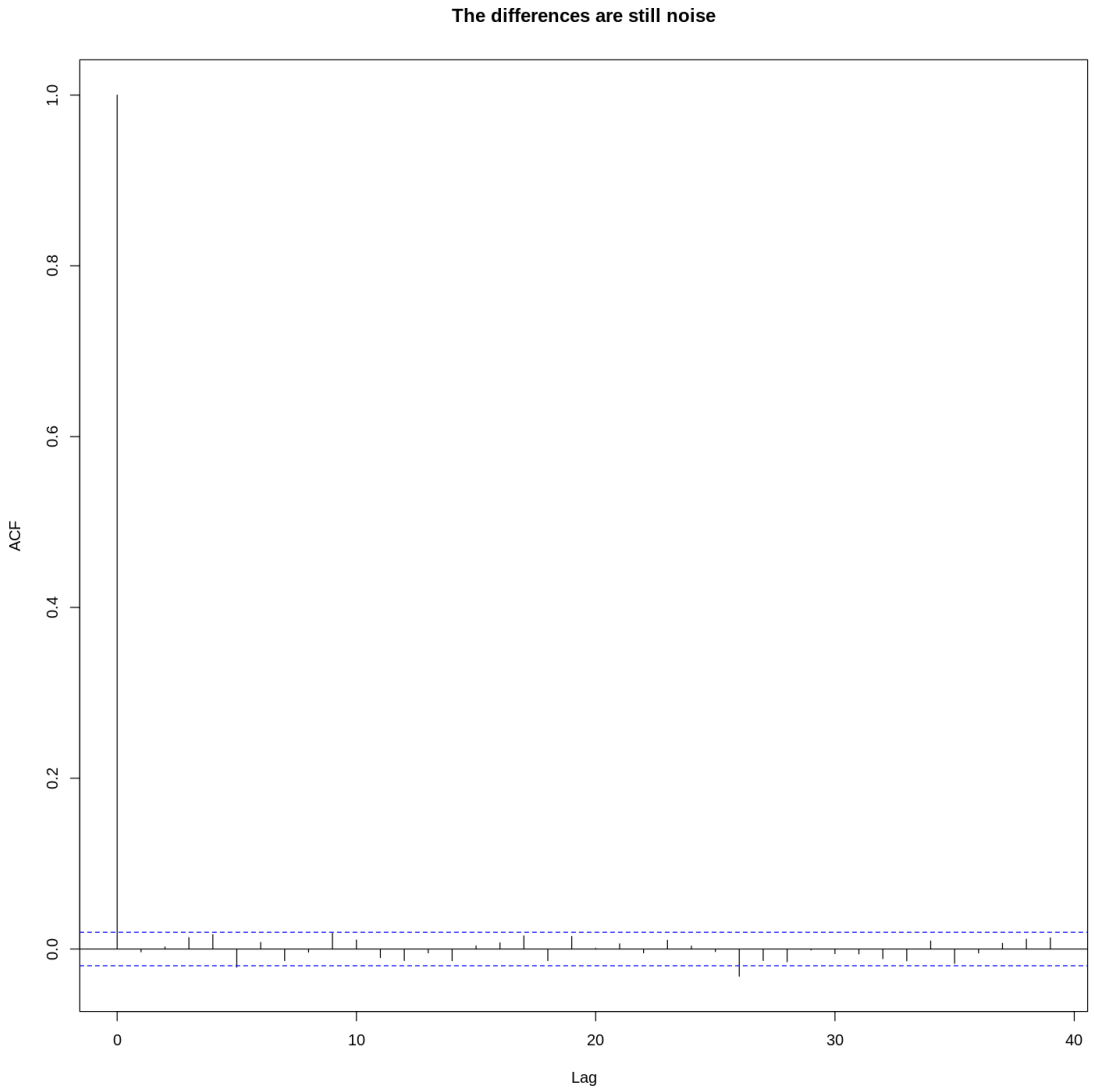


As we can see, this is the typical correlogram of an IID-noise.

[431]:
```
gaussian_random_walk<-cumsum(white_noise_data)
plot(gaussian_random_walk,type="l", main="Random walk using the noise")
```



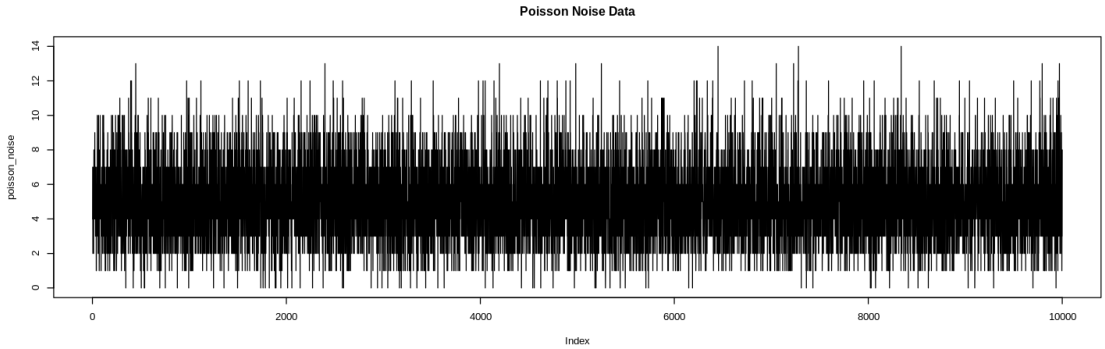
[433]:
```
acf(diff(gaussian_random_walk), main="The differences are still noise")
```

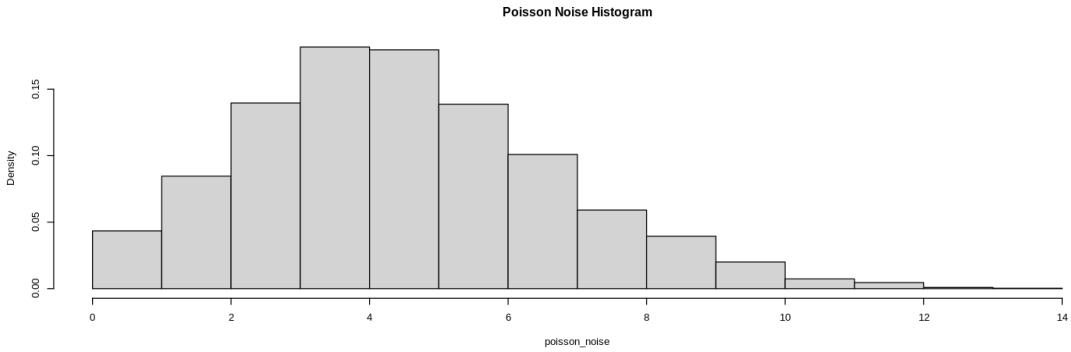


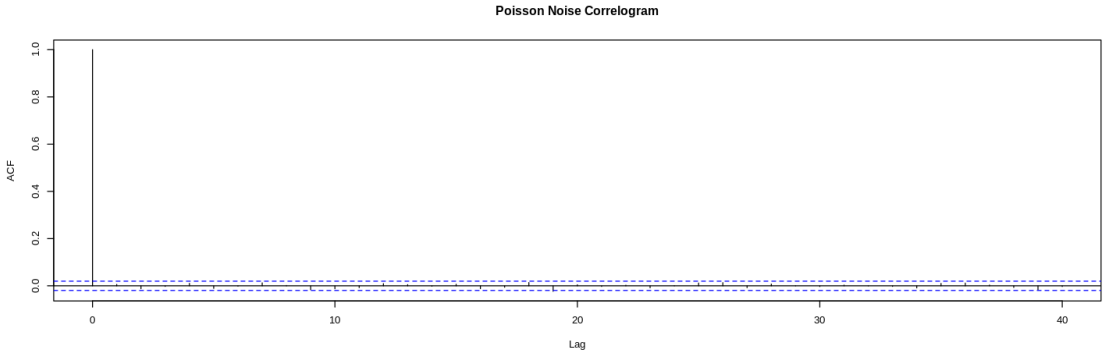
Now we can test noises of non-gaussian sources: Poisson distribution and exponentials.

[307]:
```
n <- 10000
poisson_noise <- rpois (n, lambda=5)
exponential_noise <- rexp(n, rate=10)
```

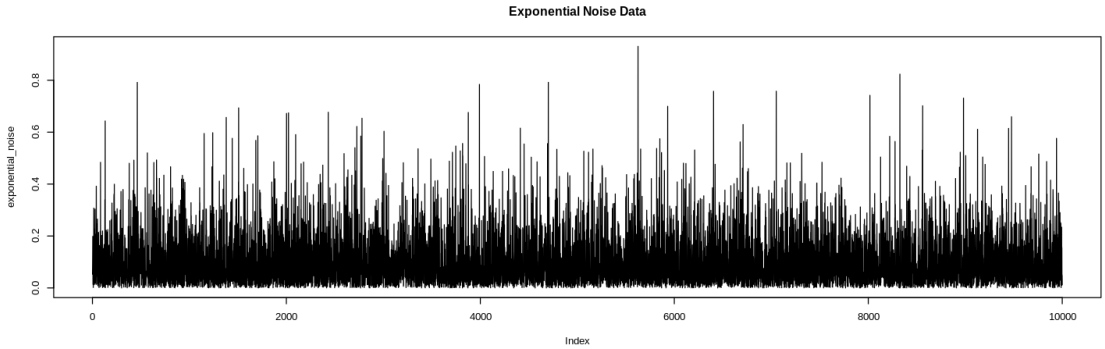
[308]:
```
par(mfrow = c(3, 1))
plot(poisson_noise,type="l", main="Poisson Noise Data")
hist(poisson_noise,freq=F, main="Poisson Noise Histogram")
acf(poisson_noise, main="Poisson Noise Correlogram")
par(mfrow = c(1, 1))
```

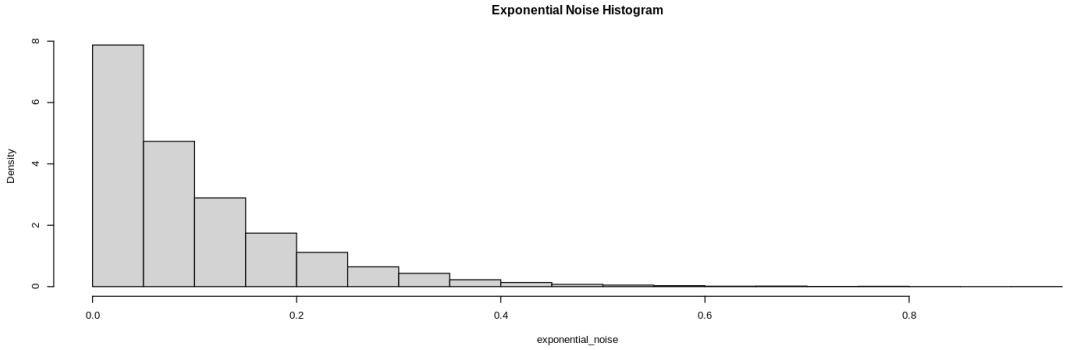


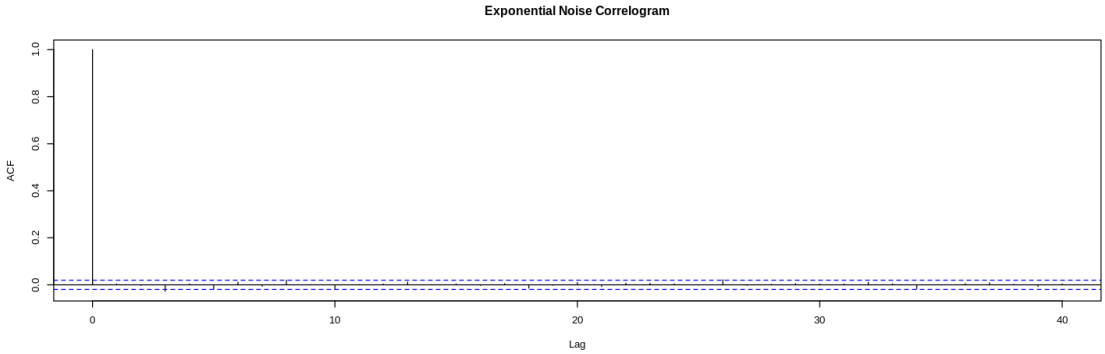




[309]:
```
par(mfrow = c(3, 1))
plot(exponential_noise,type="l", main="Exponential Noise Data")
hist(exponential_noise,freq=F, main="Exponential Noise Histogram")
acf(exponential_noise, main="Exponential Noise Correlogram")
par(mfrow = c(1, 1))
```





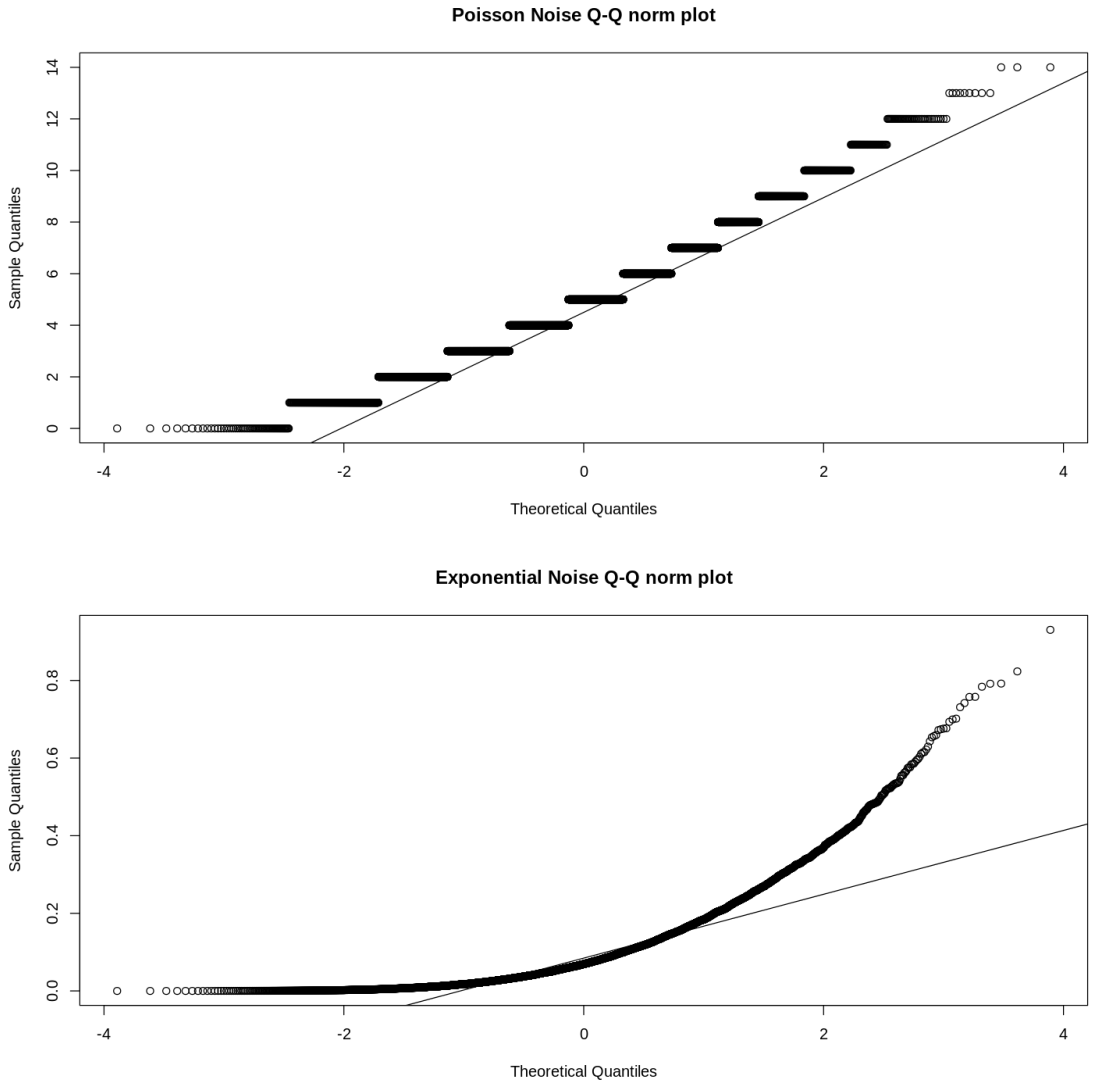


[310]:
```
Box.test (poisson_noise, lag=1, type=c("Ljung-Box"))
Box.test (exponential_noise, lag=1, type=c("Ljung-Box"))
```

```
        Box-Ljung test

data:  poisson_noise
X-squared = 0.33487, df = 1, p-value = 0.5628


        Box-Ljung test

data:  exponential_noise
```

```
X-squared = 0.12587, df = 1, p-value = 0.7228
```

[311]:
```
par(mfrow = c(2,1))
qqnorm(poisson_noise, main="Poisson Noise Q-Q norm plot")
qqline(poisson_noise)
qqnorm(exponential_noise, main="Exponential Noise Q-Q norm plot")
qqline(exponential_noise)
par(mfrow = c(1,1))
```



[312]:
```
shapiro.test(poisson_noise[1:5000])
shapiro.test(exponential_noise[1:5000])
```

```
        Shapiro-Wilk normality test

data:  poisson_noise[1:5000]
W = 0.97202, p-value < 2.2e-16
```

```
        Shapiro-Wilk normality test

data:  exponential_noise[1:5000]
W = 0.81829, p-value < 2.2e-16
```

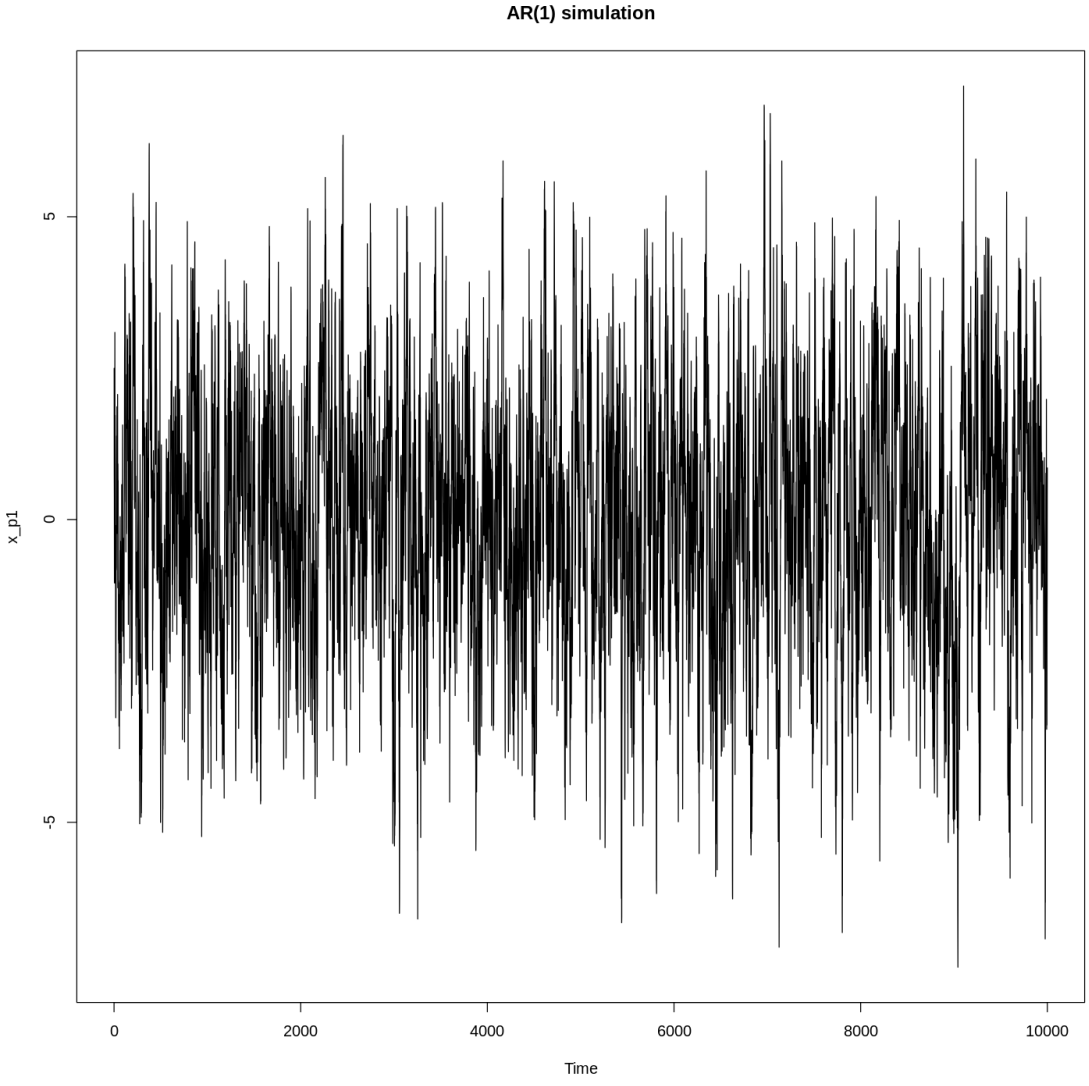
As we can see, both the correlograms show that both of these samples are noise. Nonetheless, using the Q-Q test and the shapiro test we can see that they are not gaussian noise.

3 Exercise 3

Simulate an AR(p) model with 10000 data, for p=1 and p=2. Fit the best model to the data in both cases. Validate the model by showing the residuals are an IID noise.

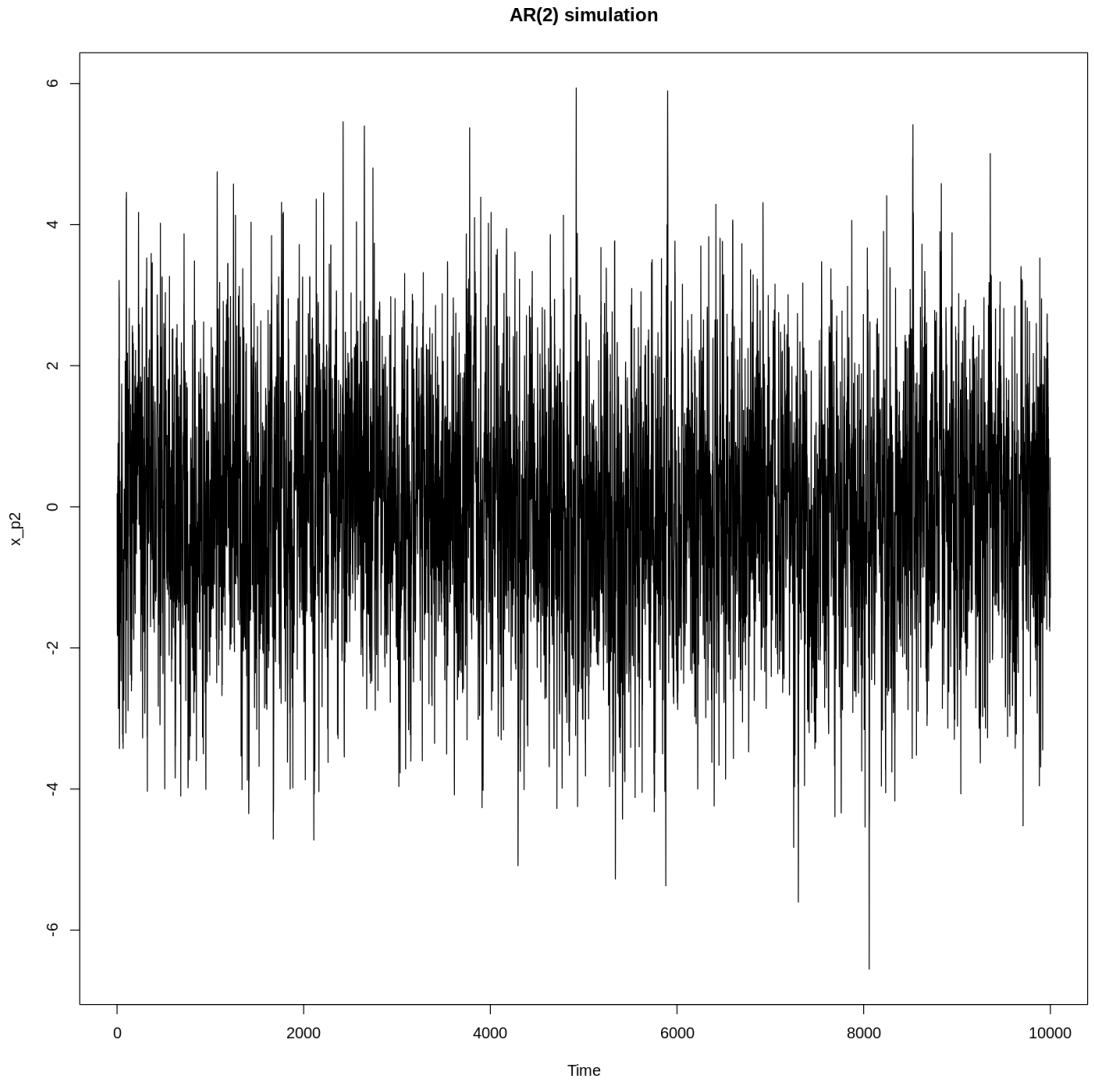
Firstly we will simulate an AR(1) model with $n = 10000$ with parameter equal to 0.8792.

[435]:
```
n<-10000
x_p1<-arima.sim(list(ar = c(0.8792), sd = sqrt(0.1796)), n)
plot(x_p1,type="l", main="AR(1) simulation")
```



Secondly, we will simulate an AR(2) model with $n = 10000$ with parameters equal to $(0.8792, -0.1845)$.

[437]:
```
x_p2<-arima.sim(list(ar = c(0.8792, -0.1845), sd = sqrt(0.1796)), n)
plot(x_p2,type="l", main="AR(2) simulation")
```



We can now fit AR models to both simulations.

[438]:
```
best_s1<-ar(x_p1, order.max=3, method="mle")
best_s1
```

```
Call:
ar(x = x_p1, order.max = 3, method = "mle")

Coefficients:
     1
```

```
0.8758

Order selected 1  sigma^2 estimated as  0.9889
```

[316]:
```
best_s2<-ar(x_p2, order.max=3, method="mle")
best_s2
```

```
Call:
ar(x = x_p2, order.max = 3, method = "mle")

Coefficients:
      1        2
 0.8237  -0.1905

Order selected 2  sigma^2 estimated as  0.9996
```

[317]:
```
s_p1<-arima(x_p1, order=c(1,0,0), method="ML")
s_p1
```

```
Call:
arima(x = x_p1, order = c(1, 0, 0), method = "ML")

Coefficients:
         ar1  intercept
      0.8552    -0.5755
s.e.  0.0166     0.2168

sigma^2 estimated as 0.9969:  log likelihood = -1418.05,  aic = 2842.1
```

[318]:
```
s_p2<-arima(x_p2, order=c(2,0,0), method="ML")
s_p2
```

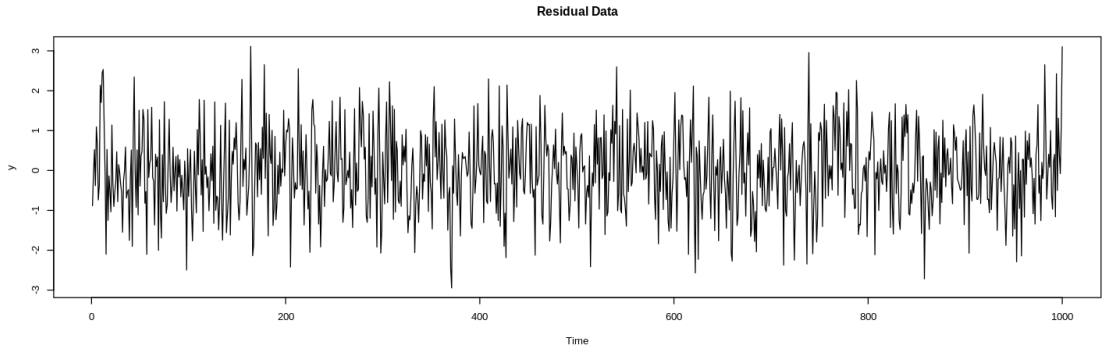
```
Call:
arima(x = x_p2, order = c(2, 0, 0), method = "ML")

Coefficients:
         ar1      ar2  intercept
      0.8237  -0.1905    -0.0691
s.e.  0.0311   0.0311     0.0861

sigma^2 estimated as 0.9996:  log likelihood = -1419.12,  aic = 2846.23
```

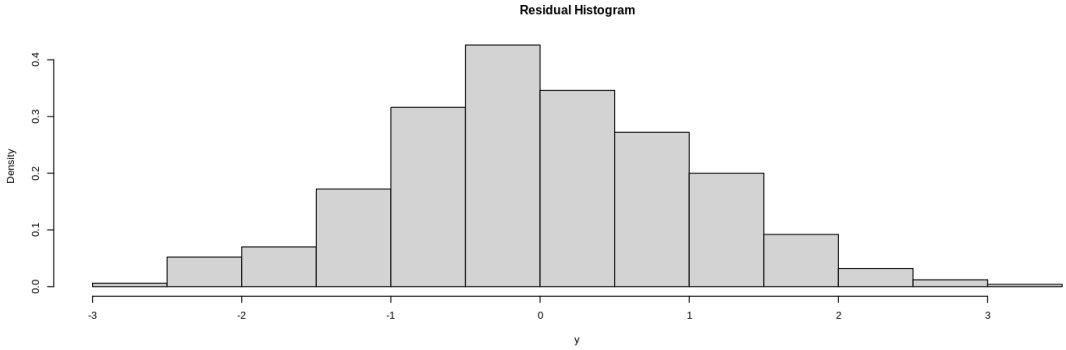
[319]:
```
y<-resid(s_p1)
h<-floor(log(n))
Box.test(x,lag=h, type=c("Ljung-Box"))
Box.test(y,lag=h,type=c("Ljung-Box"))
```

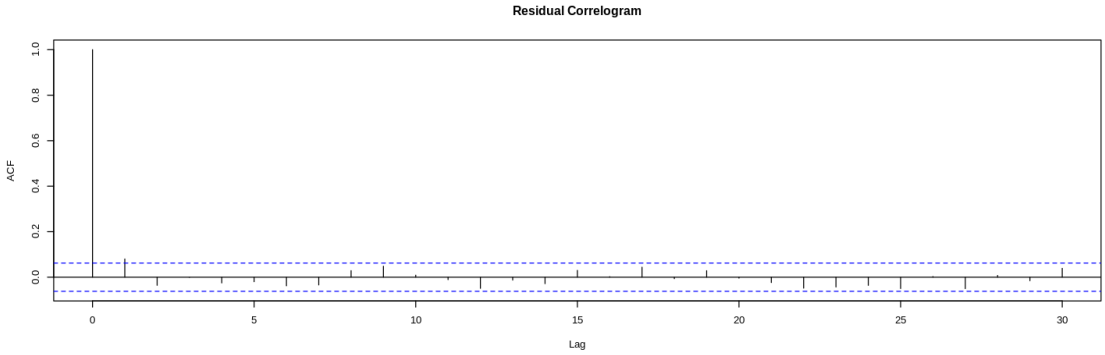
```r
par(mfrow = c(3, 1))
plot(y,type="l", main="Residual Data")
hist(y,freq=F, main="Residual Histogram")
acf(y, main="Residual Correlogram")
par(mfrow = c(1, 1))
```

```
        Box-Ljung test

data:  x
X-squared = 4947.7, df = 6, p-value < 2.2e-16
```

```
        Box-Ljung test

data:  y
X-squared = 10.186, df = 6, p-value = 0.117
```



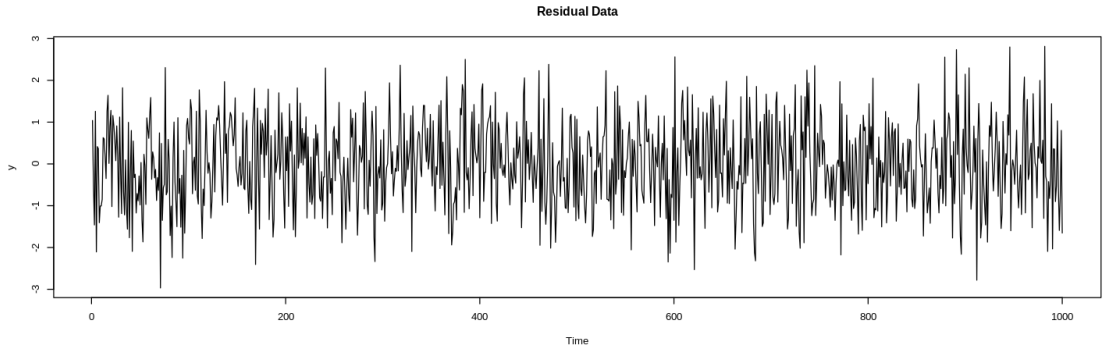


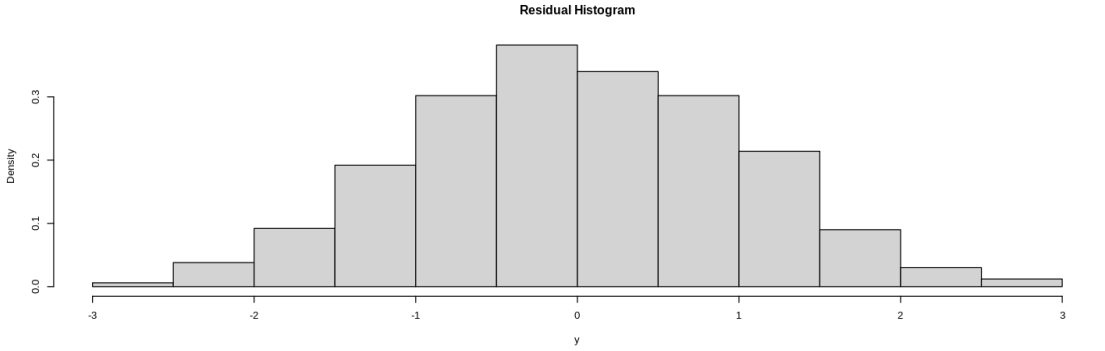


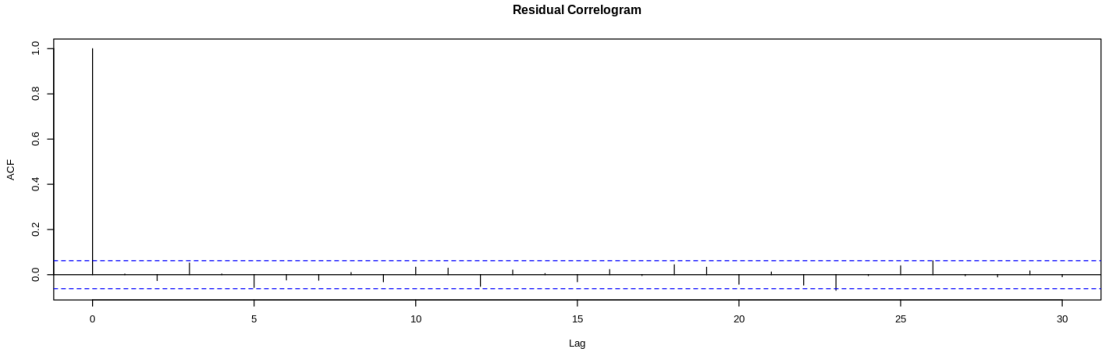
[320]:
```
y<-resid(s_p2)
h<-floor(log(n))
Box.test(x,lag=h, type=c("Ljung-Box"))
Box.test(y,lag=h,type=c("Ljung-Box"))

par(mfrow = c(3, 1))
plot(y,type="l", main="Residual Data")
hist(y,freq=F, main="Residual Histogram")
acf(y, main="Residual Correlogram")
par(mfrow = c(1, 1))
```

```
	Box-Ljung test
```

```
data:  x
X-squared = 4947.7, df = 6, p-value < 2.2e-16


        Box-Ljung test

data:  y
X-squared = 7.1279, df = 6, p-value = 0.3092
```







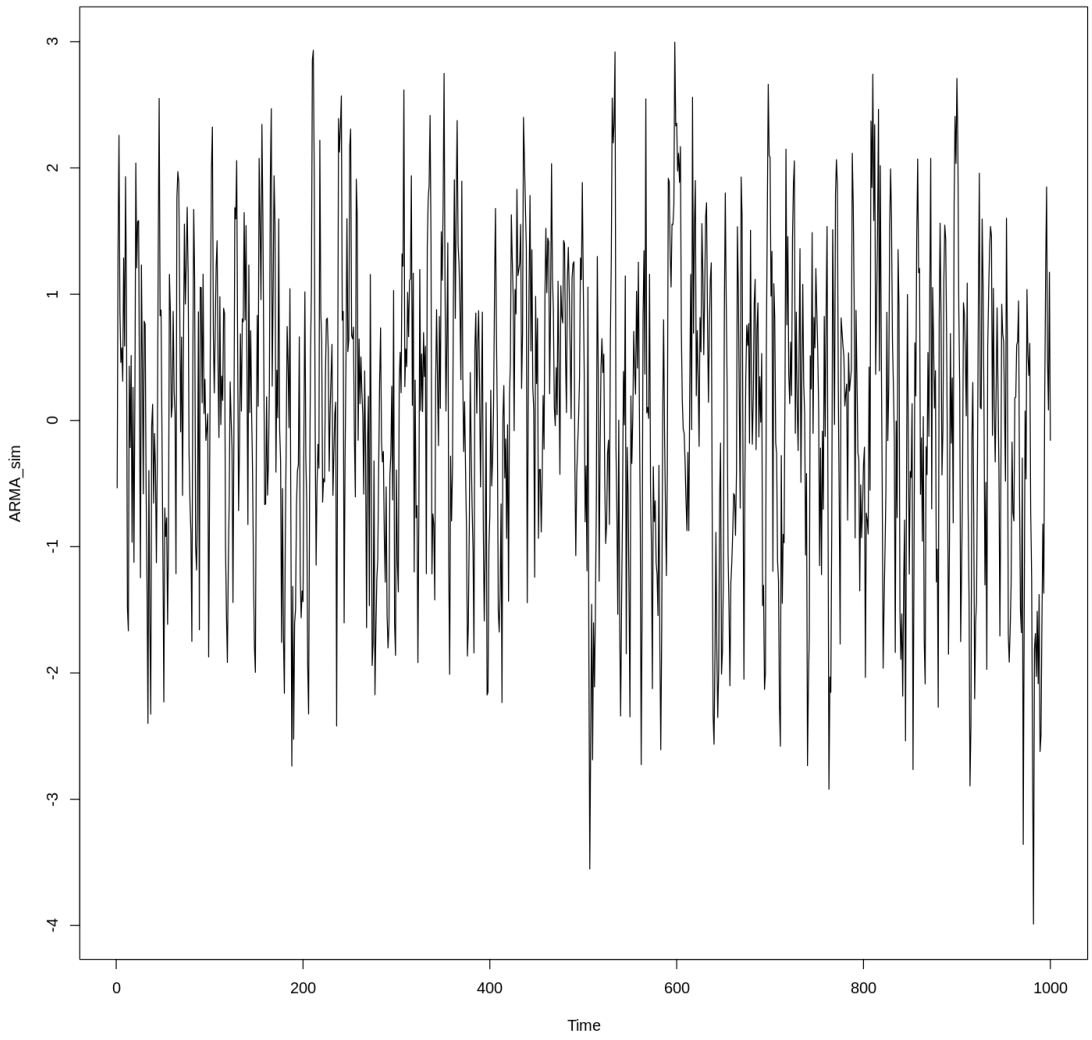
As we can see, residuals for both simulations are IID noise as the correlogram indicate.

4 Exercise 4

Simulate an ARMA (2,1). Compute the autocorrealtion and the partial autocorrelation. Fit the best ARMA model. Validate it. Make the graphical representation of the forecasting.

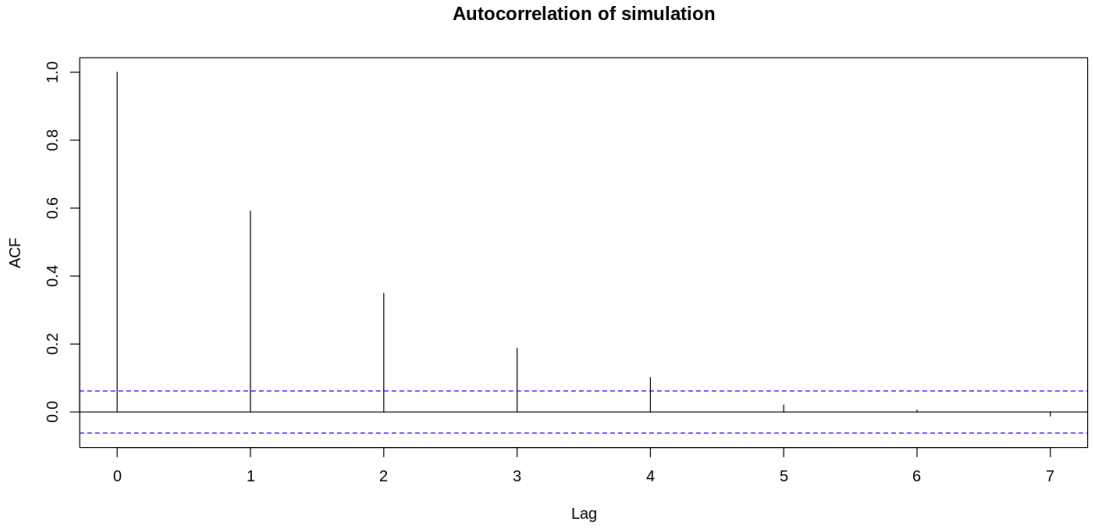
Firstly, let's simulate 1000 points with an ARMA(2,1) model with parameters $AR = (0.25, 0.15)$ and $MA = (0.35)$.

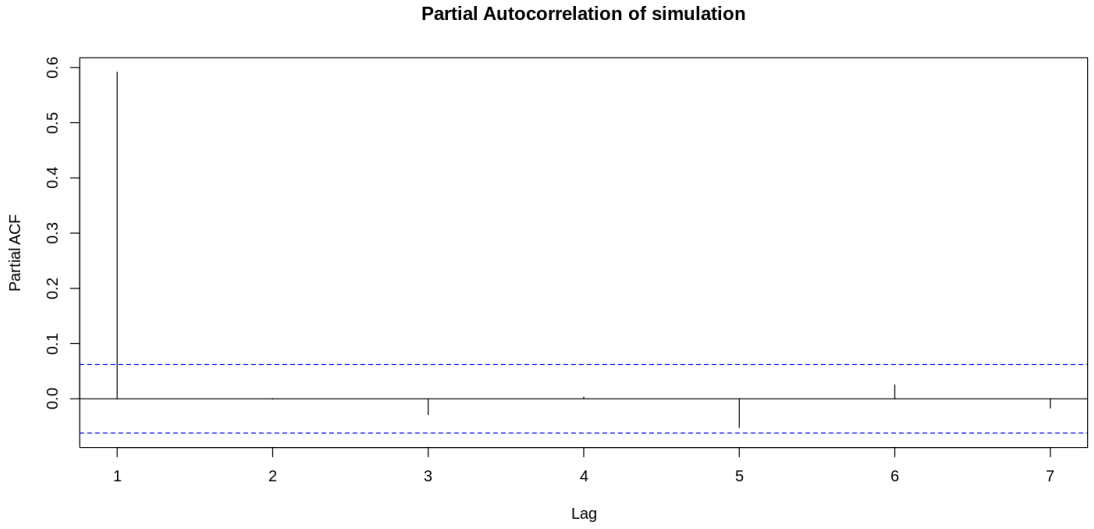
[321]:
```
ARMA_sim<-arima.sim(list(order=c(2,0,1), ar=c(0.25,0.15), ma=0.35), n=1000)
plot(ARMA_sim)
```



Now for the ACF and PACF we will use $k = ln(n) = 6.9$ approximately 7.

[322]:
```
par(mfrow = c(2, 1))
acf(ARMA_sim,7, main="Autocorrelation of simulation")
pacf(ARMA_sim,7, main="Partial Autocorrelation of simulation")
par(mfrow = c(1, 1))
```





Now we can fit the ARMA(2,1) model to the simulation.

[323]:
```
ARMA_model<-arima(ARMA_sim, order=c(2,0,1), method="ML")
ARMA_model
```

```
Call:
arima(x = ARMA_sim, order = c(2, 0, 1), method = "ML")
```

```
Coefficients:
          ar1     ar2     ma1  intercept
      -0.2586  0.5184  0.8365     0.0749
s.e.   0.1167  0.0673  0.1185     0.0768

sigma^2 estimated as 0.9622:  log likelihood = -1399.87,  aic = 2809.73
```

[324]:
```
Box.test(ARMA_sim,lag=7,type=c("Ljung-Box"))
```

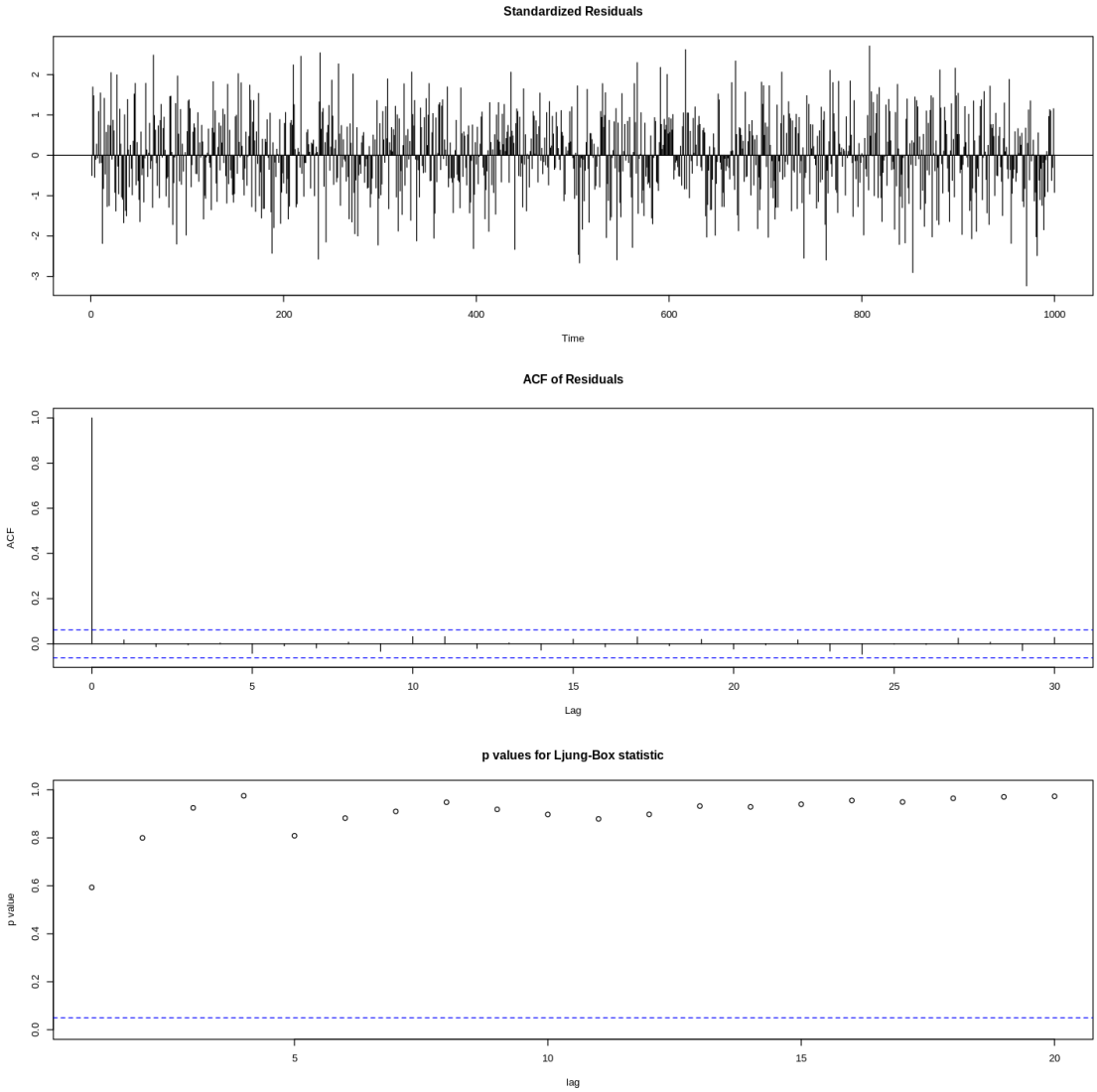
```
        Box-Ljung test

data:  ARMA_sim
X-squared = 519.65, df = 7, p-value < 2.2e-16
```

[325]:
```
y<-resid(ARMA_model)
```

[326]:
```
Box.test(y,lag=7,type=c("Ljung-Box"))
```

```
        Box-Ljung test

data:  y
X-squared = 2.7099, df = 7, p-value = 0.9105
```

[327]:
```
tsdiag(ARMA_model, gof.lag=20)
```



We can see that the residuals are IID noise since the p-values and correlograms fall into the IID noise patterns.

Let's do forecasting now

[328]:
```
install.packages('forecast')
```

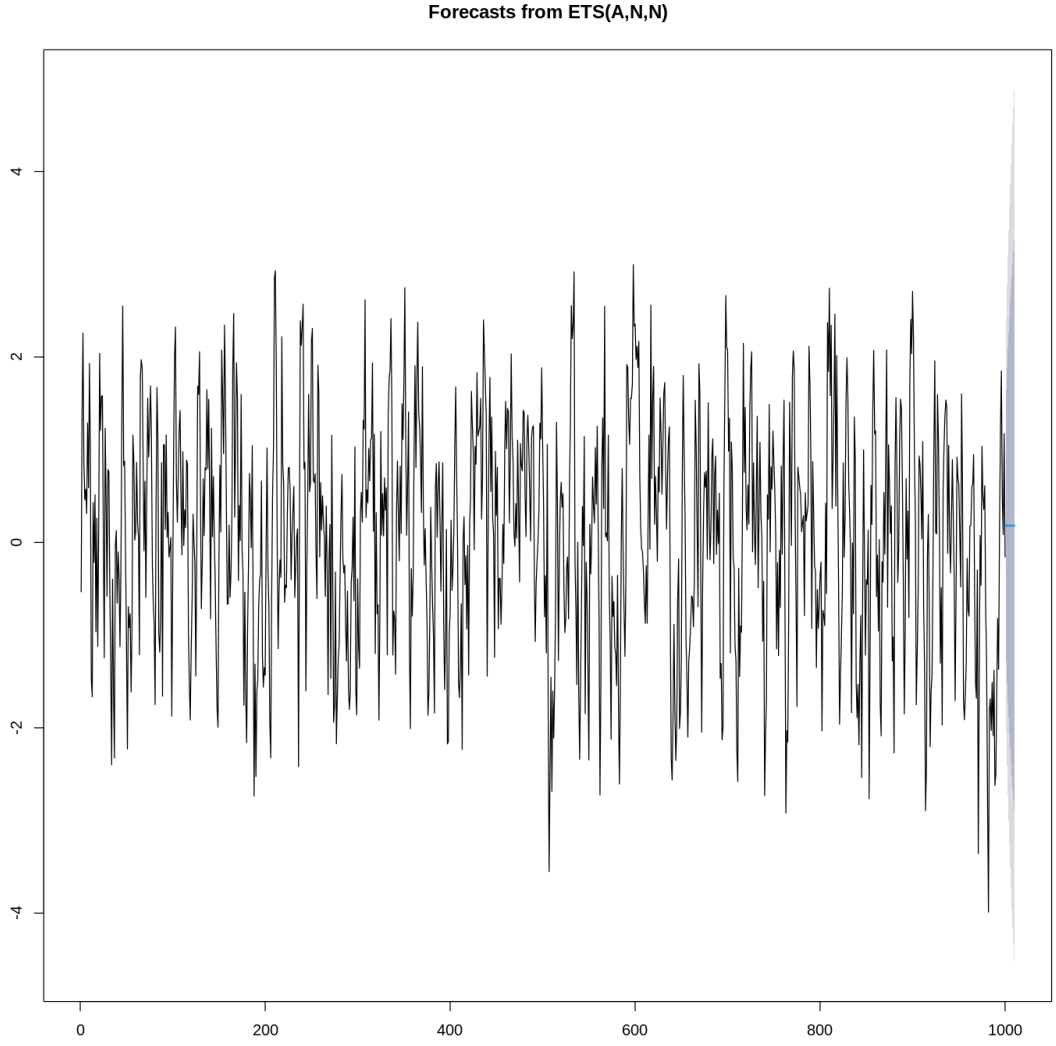
```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘lmtest’, ‘Rcpp’, ‘timeDate’, ‘urca’,
‘RcppArmadillo’
```

[329]:
```
require(forecast)
y<-auto.arima(ARMA_sim, max.p=2, max.d=0,max.q=2)
y
```

```
Loading required package: forecast


Series: ARMA_sim
ARIMA(1,0,0) with zero mean

Coefficients:
         ar1
      0.5926
s.e.  0.0254

sigma^2 = 0.9661:  log likelihood = -1401.4
AIC=2806.81   AICc=2806.82   BIC=2816.62
```

[330]:
```
yz<-resid(y)
Box.test(yz, lag=7, type=c("Ljung-Box"))
xfc<-forecast(ARMA_sim)
plot(xfc, type="l")
```

```

        Box-Ljung test

data:  yz
X-squared = 5.0994, df = 7, p-value = 0.6478
```


Forecasts from ETS(A,N,N)
4
2
0
-2
-4
0
200
400
600
800
1000

5 Exercise 5

Take the file Nile in datasets. Fit the best ARIMA model to the process. Validate it. Make the graphical representation of the forecasting.

Let's load the Nile dataset into our workspace.

[439]:
```
nile_data <- Nile
nile_data
```

A Time Series:
1. 1120 2. 1160 3. 963 4. 1210 5. 1160 6. 1160 7. 813 8. 1230 9. 1370 10. 1140 11. 995 12. 935 13. 1110 14. 994 15. 1020 16. 960 17. 1180 18. 799 19. 958 20. 1140 21. 1100 22. 1210 23. 1150 24. 1250 25. 1260 26. 1220 27. 1030 28. 1100 29. 774 30. 840 31. 874 32. 694 33. 940 34. 833 35. 701 36. 916 37. 692 38. 1020 39. 1050 40. 969 41. 831 42. 726 43. 456 44. 824 45. 702 46. 1120 47. 1100 48. 832 49. 764 50. 821 51. 768 52. 845 53. 864 54. 862 55. 698 56. 845 57. 744 58. 796 59. 1040 60. 759 61. 781 62. 865 63. 845 64. 944 65. 984 66. 897 67. 822 68. 1010 69. 771 70. 676 71. 649 72. 846 73. 812 74. 742 75. 801 76. 1040 77. 860 78. 874 79. 848 80. 890 81. 744 82. 749 83. 838 84. 1050 85. 918 86. 986 87. 797 88. 923 89. 975 90. 815 91. 1020 92. 906 93. 901 94. 1170 95. 912 96. 746 97. 919 98. 718 99. 714 100. 740

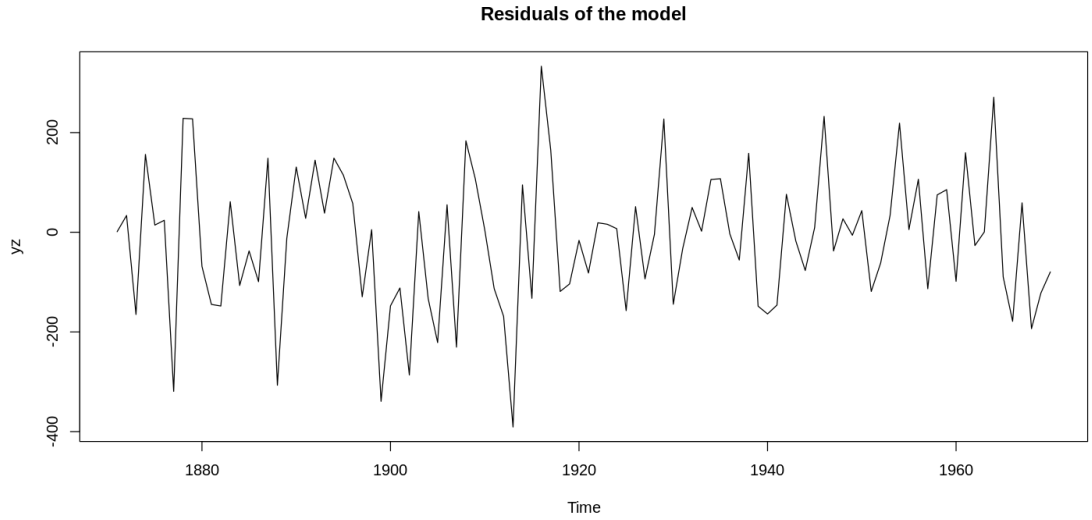
And now let's find the best ARIMA model to the data.

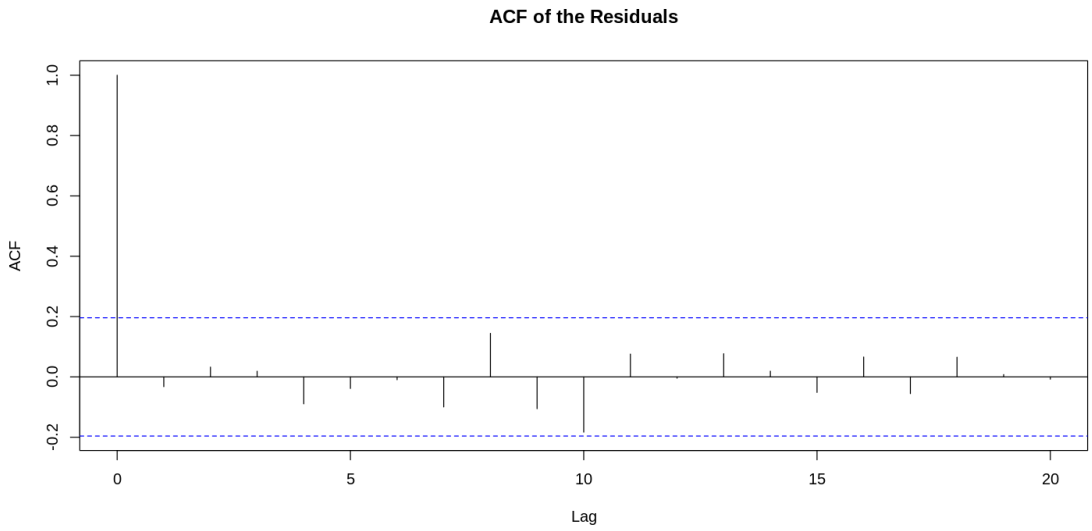
[440]:
```
ARMA_nile<-auto.arima(nile_data)
ARMA_nile
```

```
Series: nile_data
ARIMA(1,1,1)

Coefficients:
         ar1      ma1
      0.2544  -0.8741
s.e.  0.1194   0.0605

sigma^2 = 20177:  log likelihood = -630.63
AIC=1267.25   AICc=1267.51   BIC=1275.04
```

[443]:
```
par(mfrow = c(2, 1))
yz<-resid(ARMA_nile)
plot(yz, main="Residuals of the model")
acf(yz, main="ACF of the Residuals")
Box.test(yz, type=c("Ljung-Box"))
par(mfrow = c(1, 1))
```

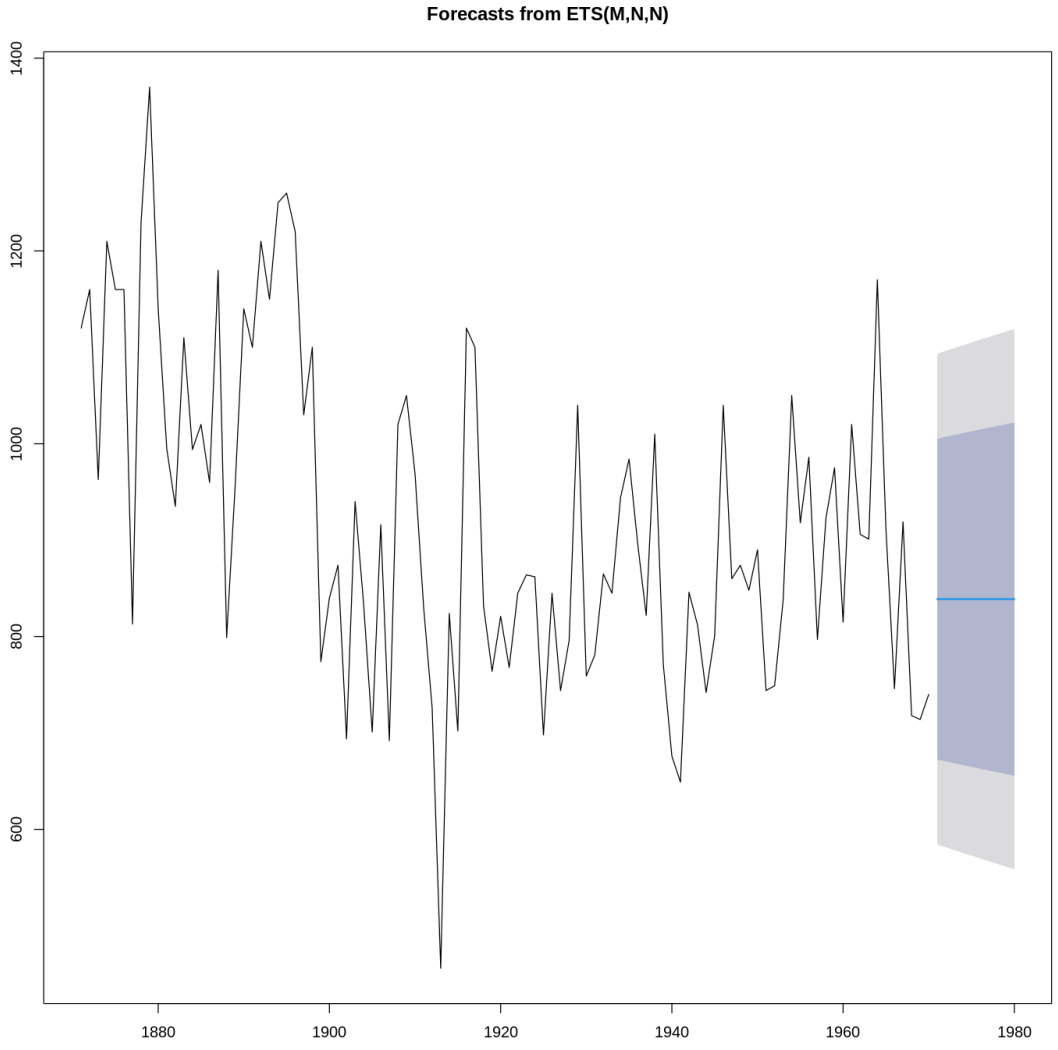
```
        Box-Ljung test

data:  yz
X-squared = 0.10739, df = 1, p-value = 0.7431
```





As we can see, the residuals are IID noise, so the model is a good fit to the data.

[334]:
```
xfc<-forecast(nile_data)
plot(xfc, type="l")
```


Forecasts from ETS(M,N,N)
1400
1200
1000
800
600
1880
1900
1920
1940
1960
1980

6 Exercise 6

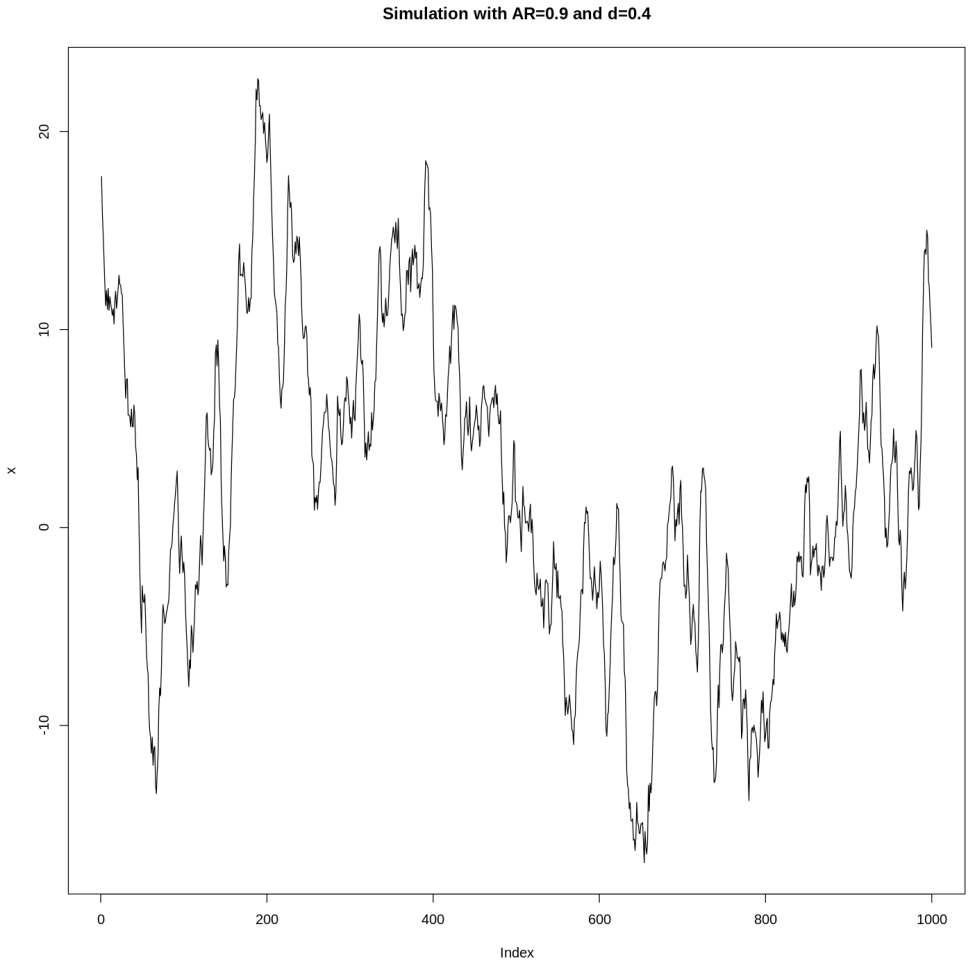
Simulate a FARIMA time series. Fit it the best model and test that the residuals of the fitted model are a white noise. Fit a FARIMA model to Nile data in datasets. Check that the fitted model is a good model.

[335]:
```
install.packages("fracdiff")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

To begin, let's simulate 1000 data points with an AR of 0.9 and a d parameter equal to 0.4.

[445]:
```
n<-1000
sample<-fracdiff.sim(n, ar=0.9, d=0.4)
x<-sample$series
plot(x, type="l", main="Simulation with AR=0.9 and d=0.4")
```



Now we can fit the ARFIMA model to the simulation and get the coefficients.

[337]:
```
library(fracdiff)
fit1<-fracdiff(x, nar=1, drange=c(0,0.5))
summary(fit1)
```

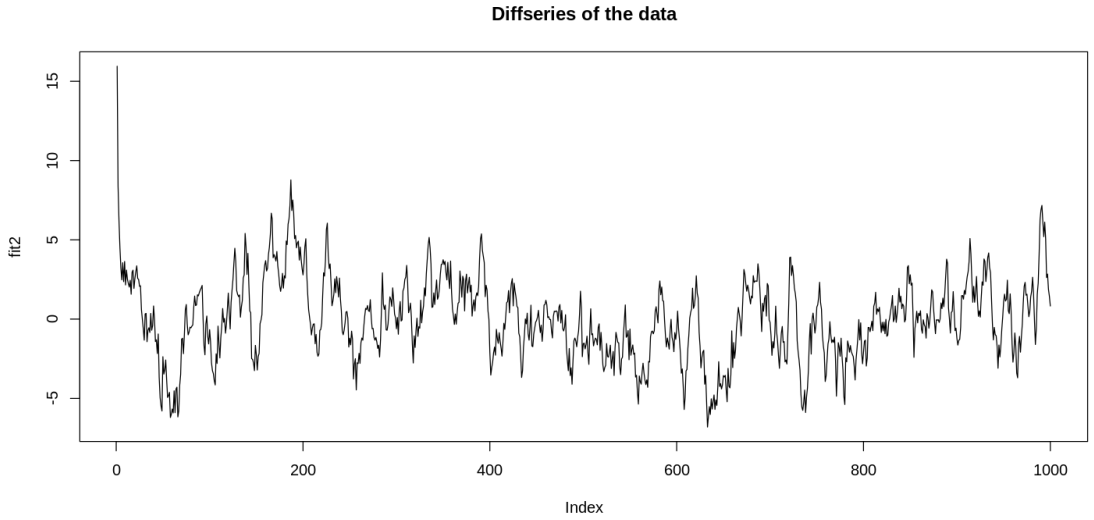
```
Call:
  fracdiff(x = x, nar = 1, drange = c(0, 0.5))

Coefficients:
   Estimate Std. Error z value Pr(>|z|)
d   0.35647    0.01191   29.94   <2e-16 ***
ar  0.92014    0.01317   69.87   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
sigma[eps] = 1.001226
[d.tol = 0.0001221, M = 100, h = 1.498e-05]
Log likelihood: -1421 ==> AIC = 2847.348 [3 deg.freedom]
```

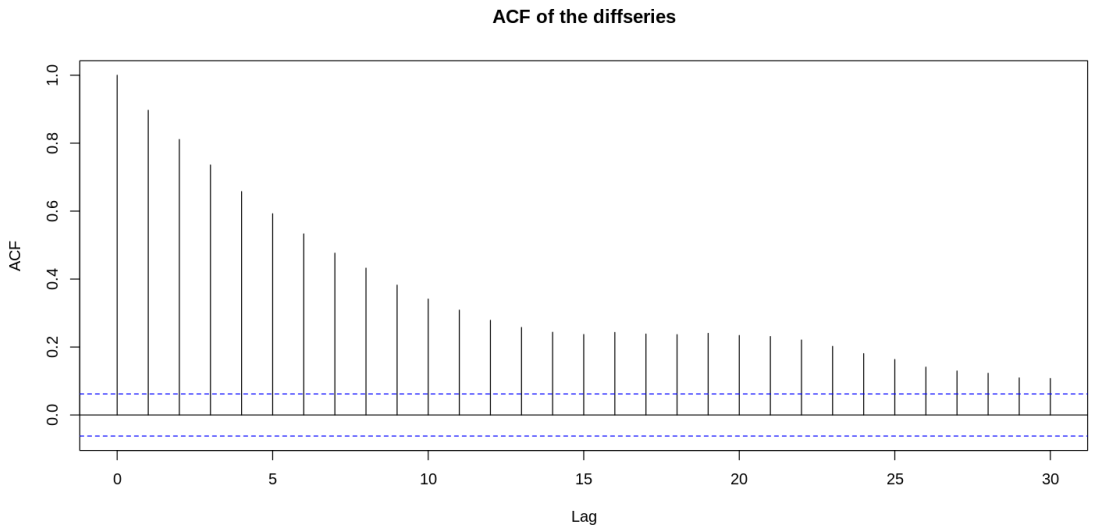
[338]:
```
d<-fit1$d
d
```

0.356470668074661

We see that both parameters are resonably close with the d equal to 0.36, close to 0.4. And the parameter of the AR model is approximately 0.9 as well.

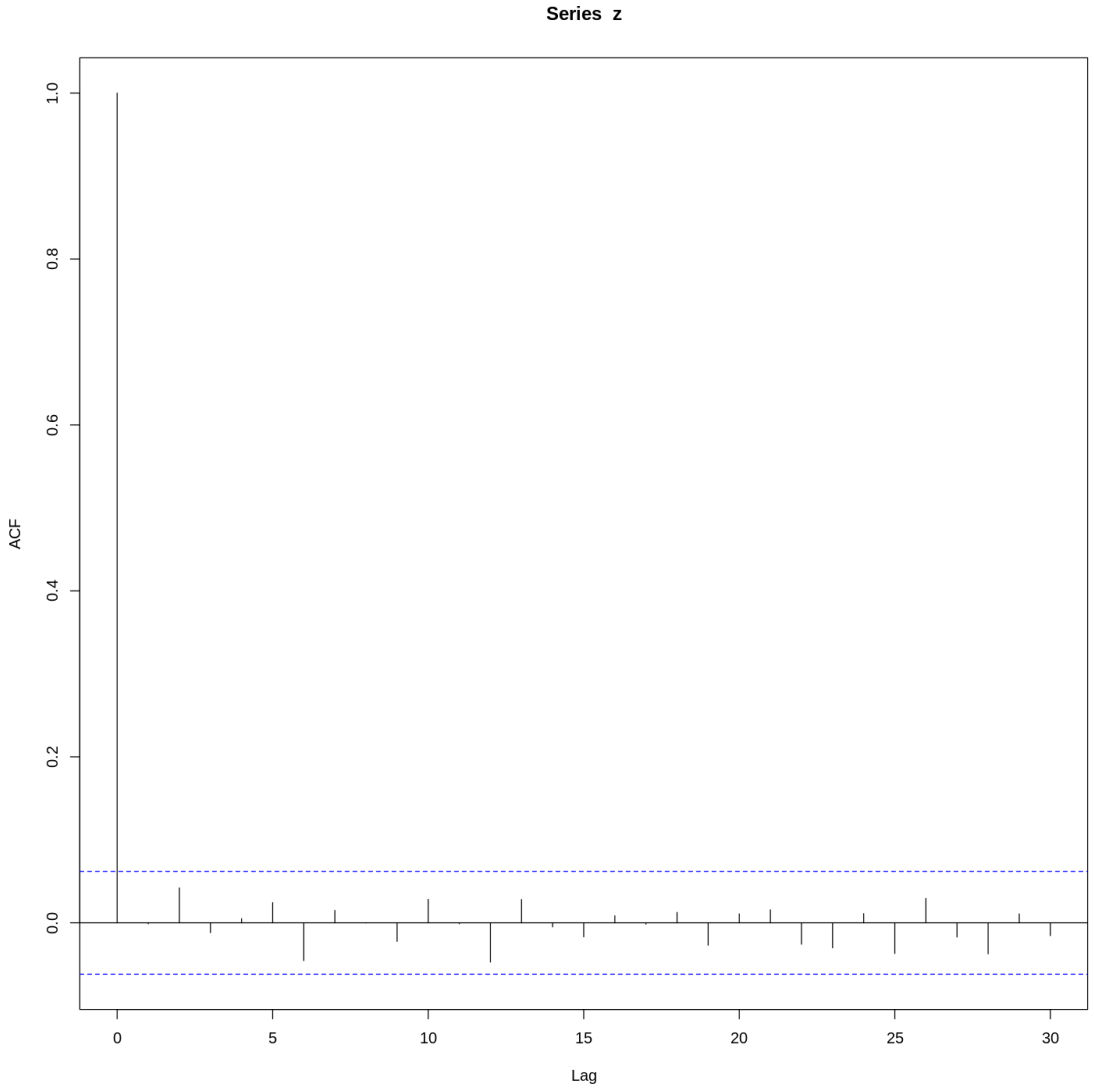
[446]:
```
par(mfrow = c(2, 1))
fit2<-diffseries(x,d)
plot(fit2, type="l", main="Diffseries of the data")
acf(fit2, main="ACF of the diffseries")
par(mfrow = c(1, 1))
```





[340]:
```
fit3<-arima(fit2,order=c(1,0,0))
z<-resid(fit3)
acf(z)
Box.test(z,log(n),type=c("Ljung-Box"))
```

```
        Box-Ljung test

data:  z
X-squared = 4.6285, df = 6.9078, p-value = 0.6956
```



We can see that this model is a good fit since the resulting residuals are IID noise.

Now let us test it on the Nile Dataset.

[447]:
```
nile_data <- Nile # Load the data
nile_data
```

A Time Series:
1. 1120 2. 1160 3. 963 4. 1210 5. 1160 6. 1160 7. 813 8. 1230 9. 1370 10. 1140 11. 995 12. 935 13. 1110 14. 994 15. 1020 16. 960 17. 1180 18. 799 19. 958 20. 1140 21. 1100 22. 1210 23. 1150 24. 1250 25. 1260 26. 1220 27. 1030 28. 1100 29. 774 30. 840 31. 874 32. 694 33. 940 34. 833 35. 701 36. 916 37. 692 38. 1020 39. 1050 40. 969 41. 831 42. 726 43. 456 44. 824 45. 702 46. 1120 47. 1100 48. 832 49. 764 50. 821 51. 768 52. 845 53. 864 54. 862 55. 698 56. 845 57. 744 58. 796 59. 1040 60. 759 61. 781 62. 865 63. 845 64. 944 65. 984 66. 897 67. 822 68. 1010 69. 771 70. 676 71. 649 72. 846

73. 812 74. 742 75. 801 76. 1040 77. 860 78. 874 79. 848 80. 890 81. 744 82. 749 83. 838 84. 1050 85. 918 86. 986 87. 797 88. 923 89. 975 90. 815 91. 1020 92. 906 93. 901 94. 1170 95. 912 96. 746 97. 919 98. 718 99. 714 100. 740

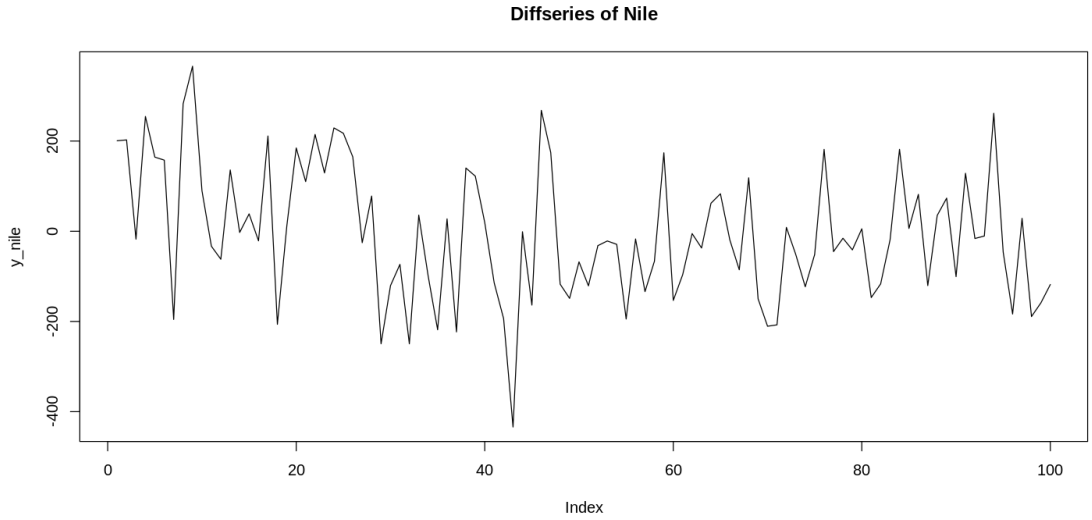
[448]:
```
fit_nile<-fracdiff(nile_data, nar=1,nma=1) # Fit the ARFIMA model
summary(fit_nile)
```

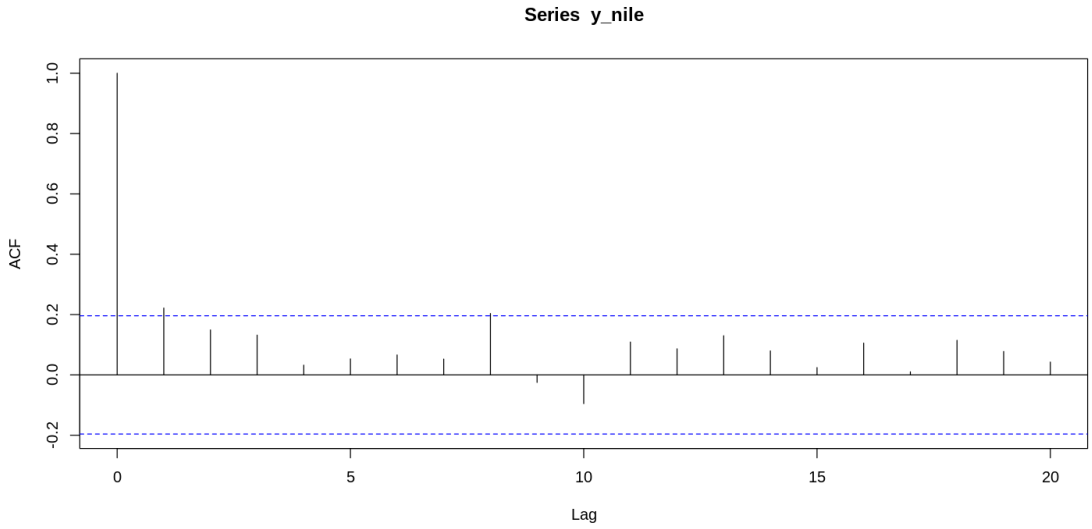
```
Call:
  fracdiff(x = nile_data, nar = 1, nma = 1)

Coefficients:
   Estimate Std. Error z value Pr(>|z|)
d   0.19005    0.01684  11.288   <2e-16 ***
ar  0.90474    0.09397   9.628   <2e-16 ***
ma  0.83657    0.06026  13.882   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
sigma[eps] = 139.8735
[d.tol = 0.0001221, M = 100, h = 6.713e-06]
Log likelihood: -636.1 ==> AIC = 1280.181 [4 deg.freedom]
```

[449]:
```
d_nile<-fit_nile$d
d_nile
```

0.190054053215489

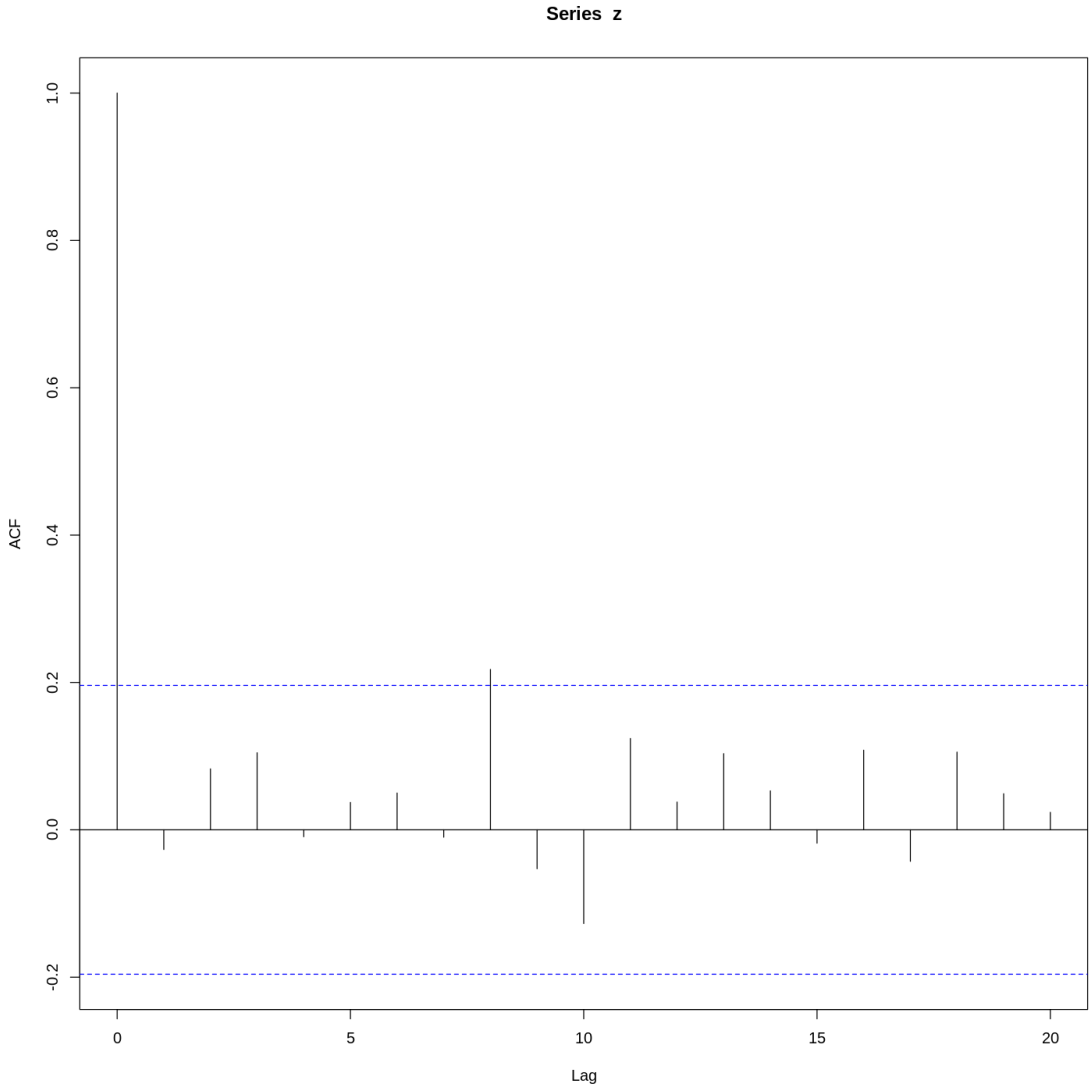
[450]:
```
par(mfrow = c(2, 1))
y_nile<-diffseries(nile_data,d_nile)
plot(y_nile, type="l", main="Diffseries of Nile")
acf(y_nile)
par(mfrow = c(1, 1))
```





[345]:
```
yz_nile<-arima(y_nile,order=c(1,0,0))
z<-resid(yz_nile)
acf(z)
Box.test(z,log(length(nile_data)),type=c("Ljung-Box"))
```

```
        Box-Ljung test

data:  z
X-squared = 1.9453, df = 4.6052, p-value = 0.8187
```

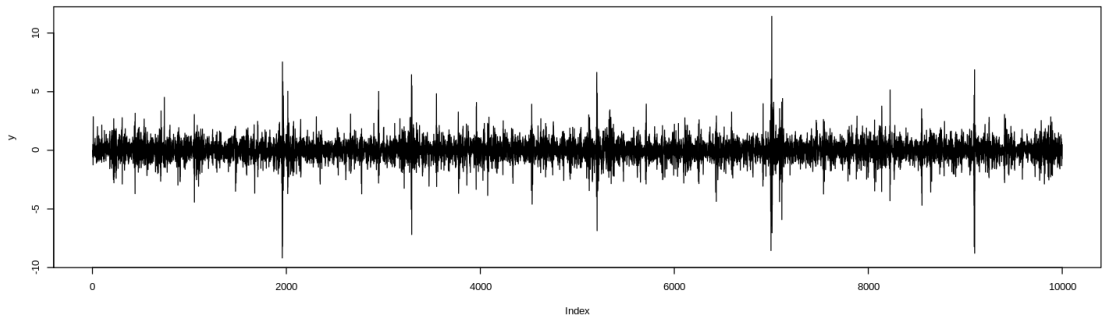


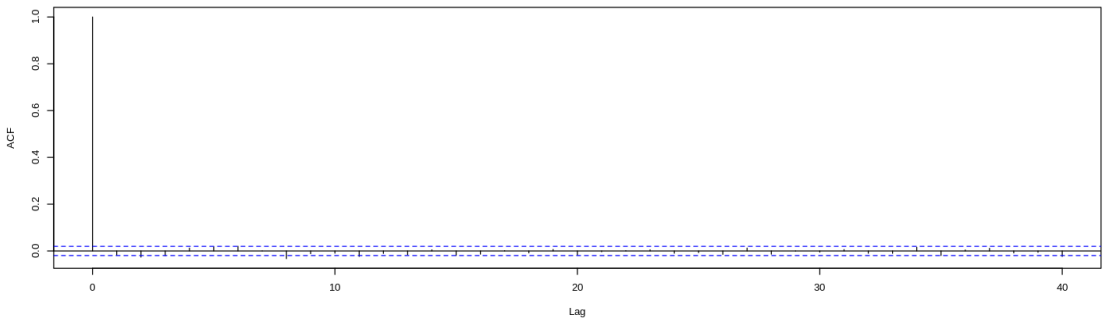
The resulting residuals for this series is IID noise as the model is a good fit for it.

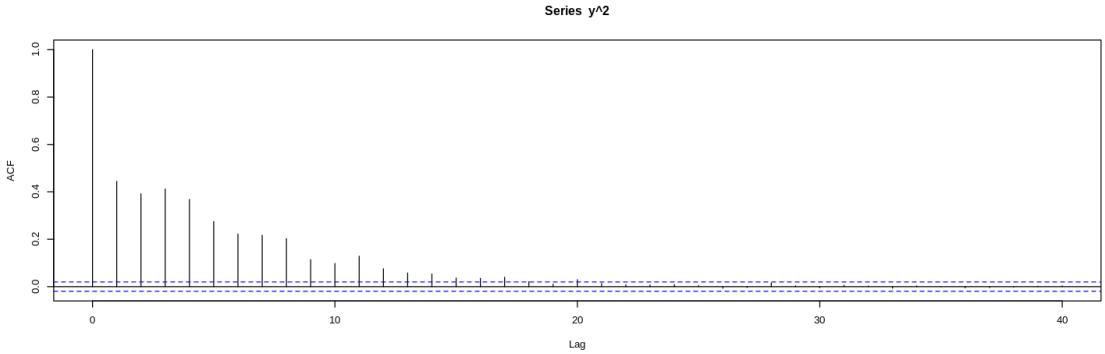
7 Exercise 7

Simulate a GARCH(1,1) time series. Fit the best model to this series. Check that the fitted model it is a good model. Fit a GARCH model to the logarithmic transformation of series in EuStockMarkets of datasets. Check the stylized facts (un-correlation, correlation of the squares, heavy tails, volatility clustering). Check that the fitted model is a good model.

[346]:
```
par(mfrow = c(3, 1))
n<-10000
a0<-0.1
a1<-0.4
b1<-0.5
w<-rnorm(n)
y<-rep(0,n)
h<-rep(0,n)
for(i in 2:n){
  h[i]<-a0+a1*(y[i-1]^2)+b1*h[i-1]
  y[i]<-w[i]*sqrt(h[i])
}
plot(y, type="l")
acf(y)
acf(y^2)
par(mfrow = c(1, 1))
```







[347]:
```
install.packages("tseries")
library(tseries)
```

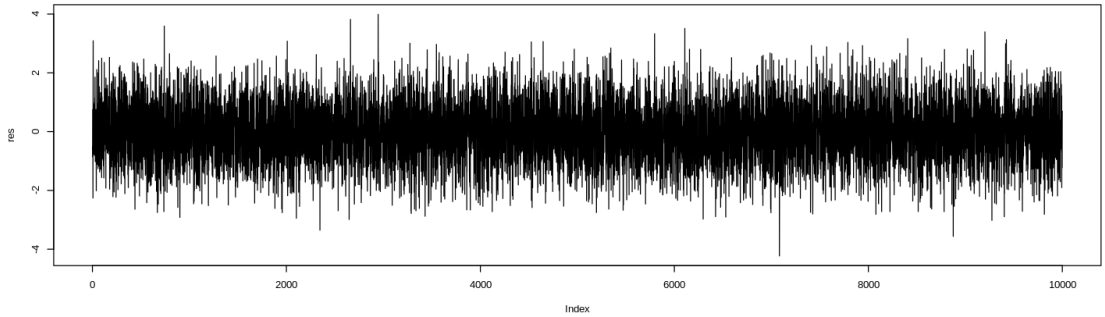
```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

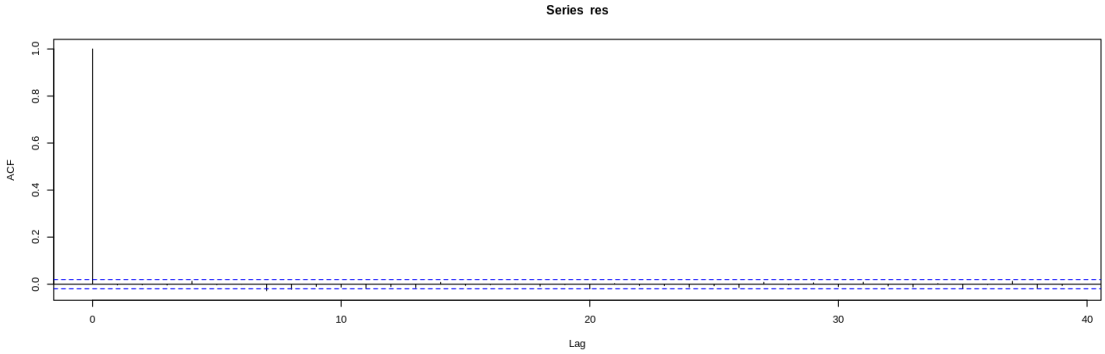
[348]:
```
par(mfrow = c(3, 1))
sp<-garch(y, trace=F)
res<-sp$res[-1]
plot(res,type="l")
acf(res)
acf(res^2)
```

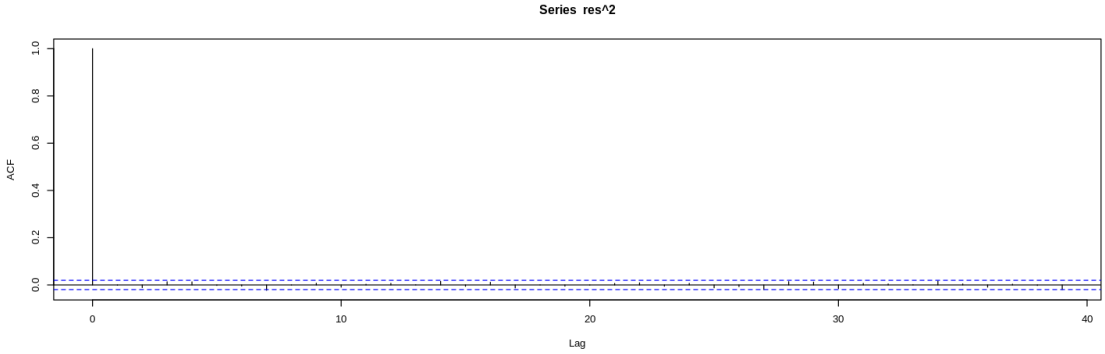
```
confint(sp)
par(mfrow = c(1, 1))
```

A matrix: 3 × 2 of type dbl

	2.5 %	97.5 %
a0	0.09267028	0.1192382
a1	0.37696719	0.4470673
b1	0.44212258	0.5114359







Let us try the EU stock markets dataset:

[411]:
```
eu_data <- EuStockMarkets
length(eu_data)
```

7440

[412]:
```
log_eu <- log(eu_data)
summary(log_eu)
```

```
      DAX              SMI              CAC              FTSE       
 Min.   :7.246   Min.   :7.370   Min.   :7.385   Min.   :7.732  
 1st Qu.:7.464   1st Qu.:7.680   1st Qu.:7.536   1st Qu.:7.953  
 Median :7.669   Median :7.936   Median :7.597   Median :8.085  
 Mean   :7.763   Mean   :8.023   Mean   :7.682   Mean   :8.145  
 3rd Qu.:7.909   3rd Qu.:8.246   3rd Qu.:7.729   3rd Qu.:8.292  
 Max.   :8.730   Max.   :9.037   Max.   :8.387   Max.   :8.729  
```

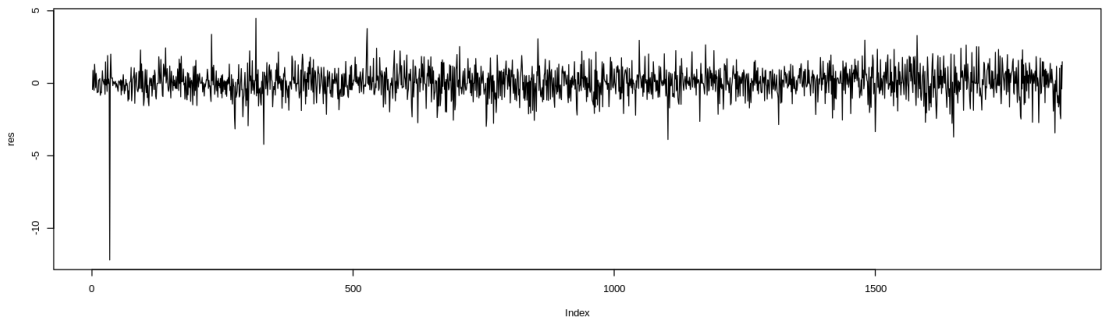
[422]:
```
DAX <- diff(log_eu[, 1])
SMI <- diff(log_eu[, 2])
CAC <- diff(log_eu[, 3])
FTSE <- diff(log_eu[, 4])
```

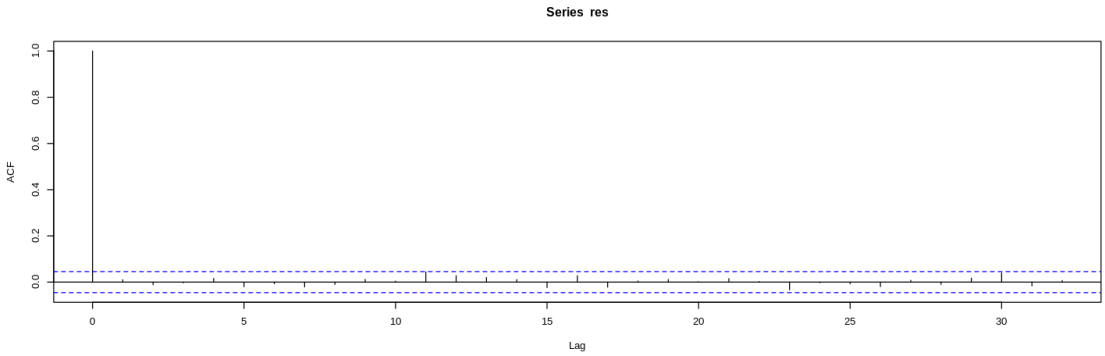
[423]:
```
par(mfrow = c(3, 1))
sp<-garch(DAX, trace=F,order=c(1,1))
res<-sp$res[-1]
plot(res,type="l")
acf(res)
acf(res^2)
confint(sp)
par(mfrow = c(1, 1))
garch(DAX,order=c(1,1), trace=F)
```

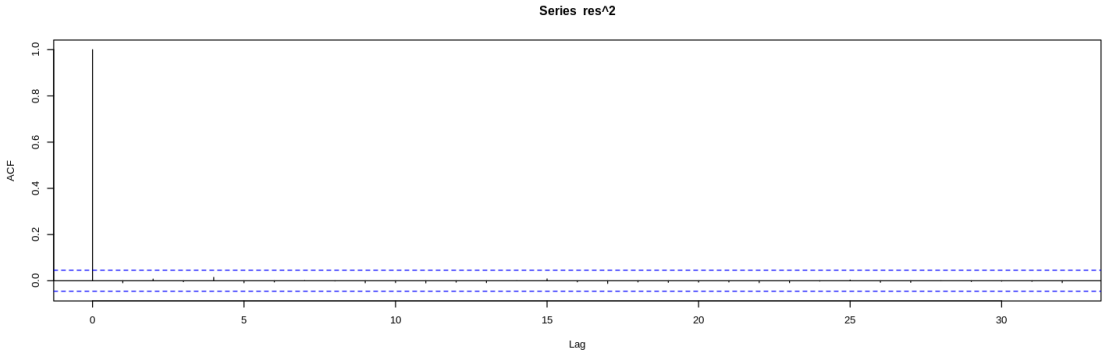
A matrix: 3 × 2 of type dbl

	2.5 %	97.5 %
a0	3.157588e-06	6.120989e-06
a1	4.627785e-02	9.037965e-02
b1	8.566877e-01	9.214456e-01

```
Call:
garch(x = DAX, order = c(1, 1), trace = F)

Coefficient(s):
       a0         a1         b1  
4.639e-06  6.833e-02  8.891e-01  
```





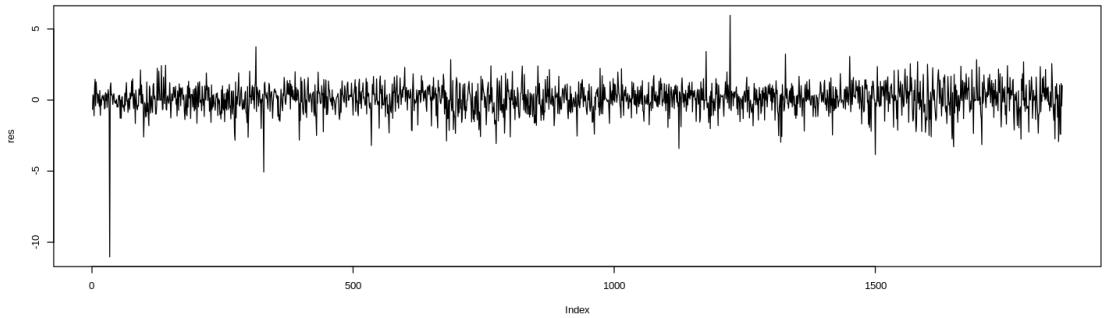


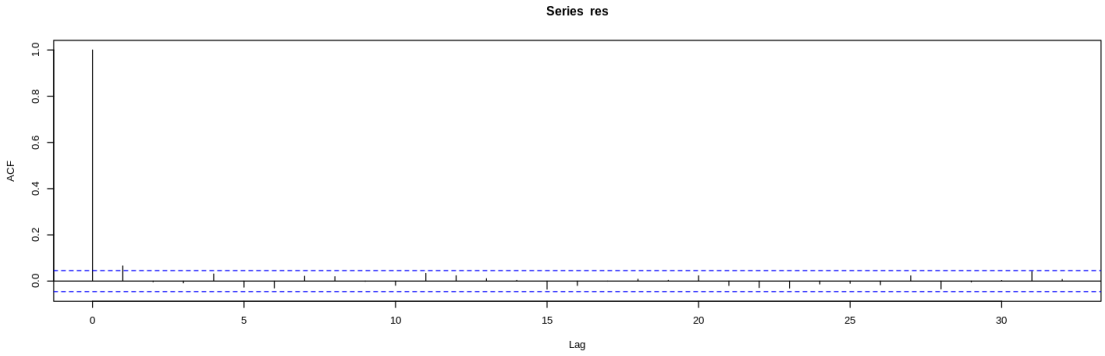
[452]:
```
par(mfrow = c(3, 1))
sp<-garch(SMI, trace=F)
res<-sp$res[-1]
plot(res,type="l")
acf(res)
acf(res^2)
confint(sp)
par(mfrow = c(1, 1))
garch(SMI,order=c(1,1), trace=F)
```

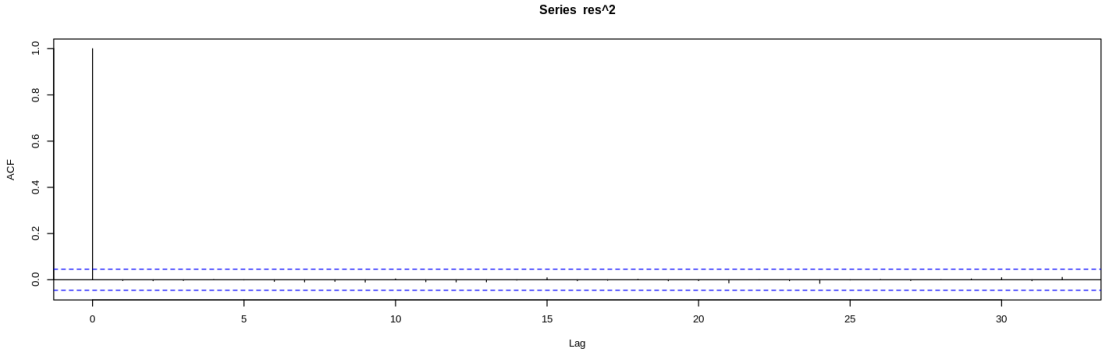
A matrix: 3 × 2 of type dbl

	2.5 %	97.5 %
a0	8.974129e-06	1.441805e-05
a1	7.640451e-02	1.525530e-01
b1	6.899628e-01	8.146311e-01

```
Call:
garch(x = SMI, order = c(1, 1), trace = F)

Coefficient(s):
       a0         a1         b1
0.0000117  0.1144787  0.7522969
```





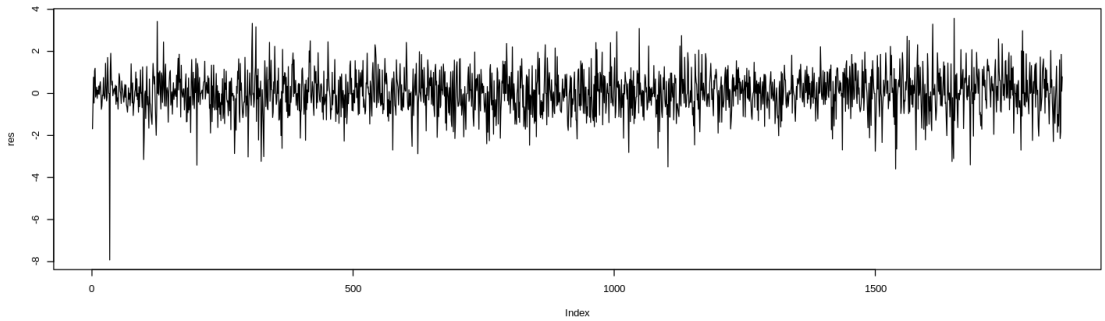


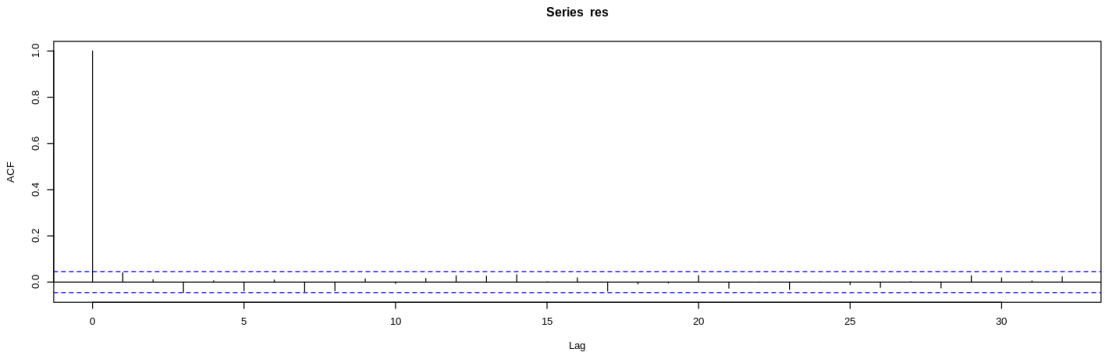
[425]:
```
par(mfrow = c(3, 1))
sp<-garch(CAC, trace=F)
res<-sp$res[-1]
plot(res,type="l")
acf(res)
acf(res^2)
confint(sp)
par(mfrow = c(1, 1))
garch(CAC,order=c(1,1), trace=F)
```

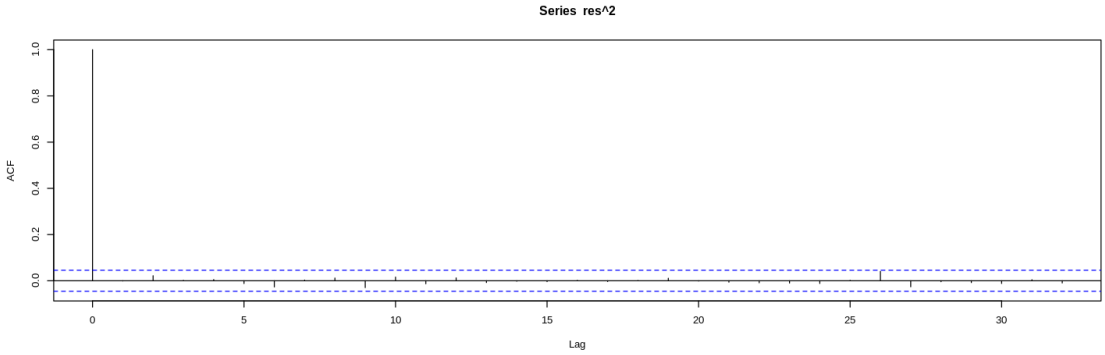
A matrix: 3 × 2 of type dbl

	2.5 %	97.5 %
a0	6.542420e-06	1.702857e-05
a1	3.712088e-02	8.119886e-02
b1	7.821879e-01	9.056679e-01

```
Call:
garch(x = CAC, order = c(1, 1), trace = F)

Coefficient(s):
       a0         a1         b1
1.179e-05  5.916e-02  8.439e-01
```





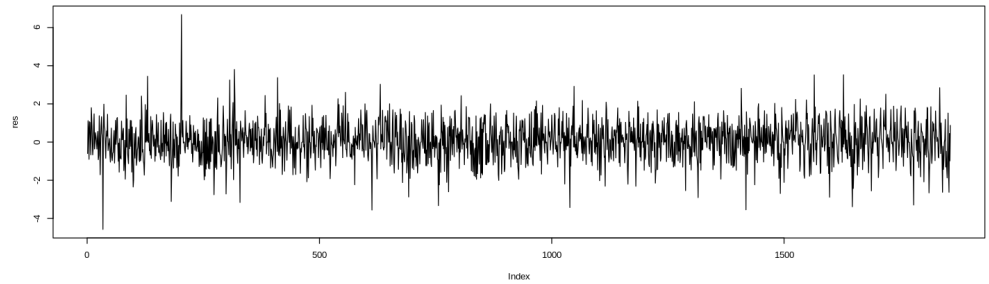


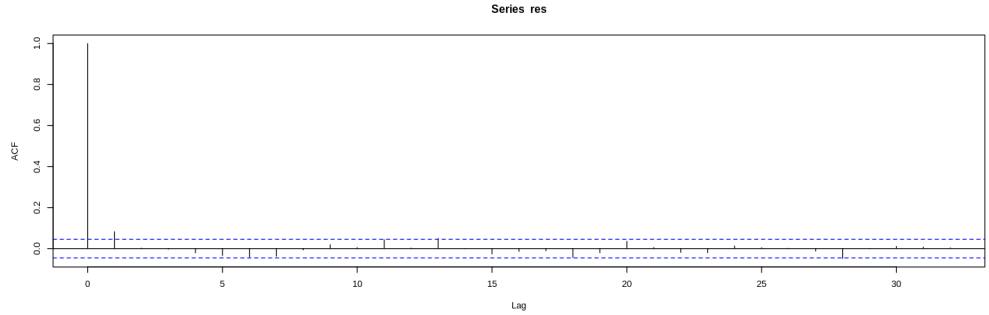
[426]:
```
par(mfrow = c(3, 1))
sp<-garch(FTSE, trace=F)
res<-sp$res[-1]
plot(res,type="l")
acf(res)
acf(res^2)
confint(sp)
par(mfrow = c(1, 1))
garch(FTSE,order=c(1,1), trace=F)
```

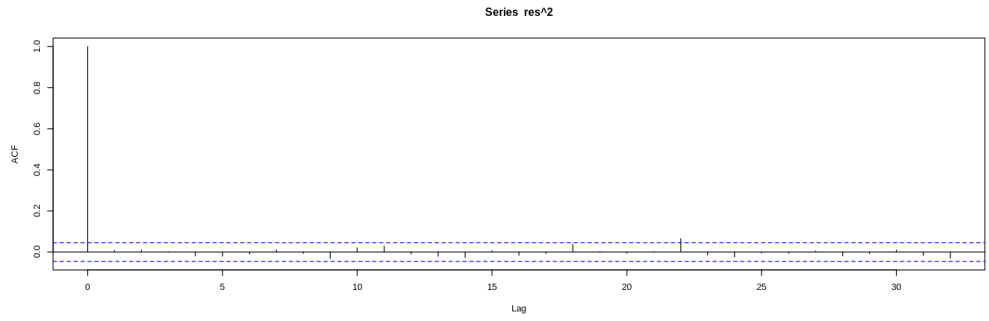
A matrix: 3 × 2 of type dbl

	2.5 %	97.5 %
a0	2.678668e-07	1.476560e-06
a1	3.201160e-02	5.863026e-02
b1	9.219037e-01	9.618274e-01

```
Call:
garch(x = FTSE, order = c(1, 1), trace = F)

Coefficient(s):
       a0         a1         b1
8.722e-07  4.532e-02  9.419e-01
```







From the correlograms of the residuals we can conclude that the residuals are uncorrelated, thus being IID noise. We can also see that the correlograms of the squares are also IID noise