Flow of The Nile

2025-04-21

The dataset Nile contains measurements of the annual flow of the Nile River at Aswan (formerly Assuan), Egypt, spanning the years 1871 to 1970 — a total of 100 consecutive observations.

- Units: 10^8 cubic meters (CMS) per year
- Time Span: 1871–1970
- Length: 100 observations (annually recorded)

```
library(urca)
library(tseries)

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

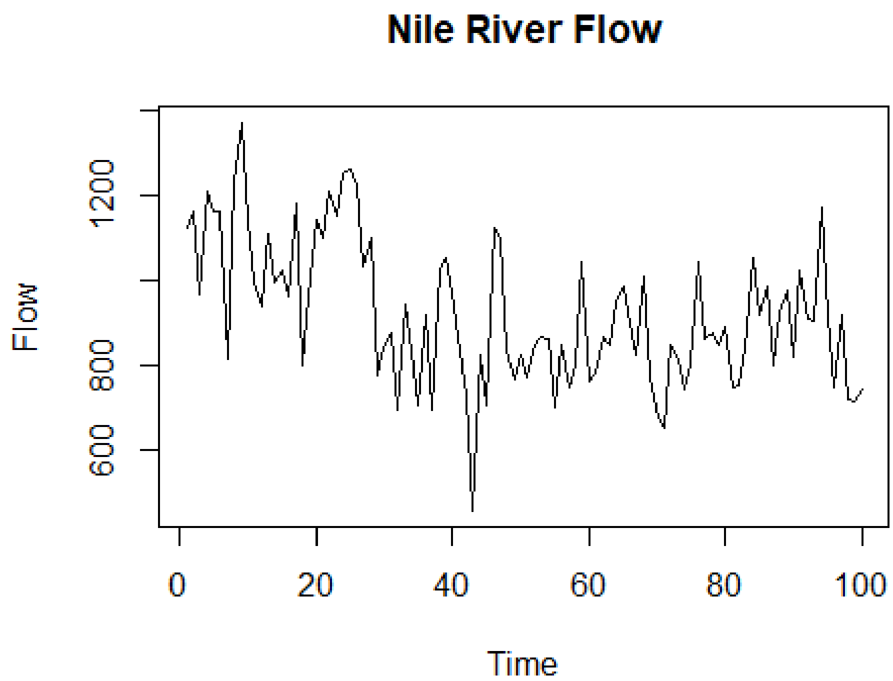
###Loading the data

```
my_data<-read.csv("Nile.csv")

my_data

##        X time Nile
## 1      1 1871 1120
## 2      2 1872 1160
## 3      3 1873  963
## 4      4 1874 1210
## 5      5 1875 1160
## 6      6 1876 1160
## 7      7 1877  813
## 8      8 1878 1230
## 9      9 1879 1370
## 10    10 1880 1140
## 11    11 1881  995
## 12    12 1882  935
## 13    13 1883 1110
## 14    14 1884  994
## 15    15 1885 1020
## 16    16 1886  960
## 17    17 1887 1180
## 18    18 1888  799
## 19    19 1889  958
## 20    20 1890 1140
## 21    21 1891 1100
## 22    22 1892 1210
```

```
## 23   23 1893 1150
## 24   24 1894 1250
## 25   25 1895 1260
## 26   26 1896 1220
## 27   27 1897 1030
## 28   28 1898 1100
## 29   29 1899  774
## 30   30 1900  840
## 31   31 1901  874
## 32   32 1902  694
## 33   33 1903  940
## 34   34 1904  833
## 35   35 1905  701
## 36   36 1906  916
## 37   37 1907  692
## 38   38 1908 1020
## 39   39 1909 1050
## 40   40 1910  969
## 41   41 1911  831
## 42   42 1912  726
## 43   43 1913  456
## 44   44 1914  824
## 45   45 1915  702
## 46   46 1916 1120
## 47   47 1917 1100
## 48   48 1918  832
## 49   49 1919  764
## 50   50 1920  821
## 51   51 1921  768
## 52   52 1922  845
## 53   53 1923  864
## 54   54 1924  862
## 55   55 1925  698
## 56   56 1926  845
## 57   57 1927  744
## 58   58 1928  796
## 59   59 1929 1040
## 60   60 1930  759
## 61   61 1931  781
## 62   62 1932  865
## 63   63 1933  845
## 64   64 1934  944
## 65   65 1935  984
## 66   66 1936  897
## 67   67 1937  822
## 68   68 1938 1010
## 69   69 1939  771
## 70   70 1940  676
## 71   71 1941  649
## 72   72 1942  846
```

```
## 73   73 1943  812
## 74   74 1944  742
## 75   75 1945  801
## 76   76 1946 1040
## 77   77 1947  860
## 78   78 1948  874
## 79   79 1949  848
## 80   80 1950  890
## 81   81 1951  744
## 82   82 1952  749
## 83   83 1953  838
## 84   84 1954 1050
## 85   85 1955  918
## 86   86 1956  986
## 87   87 1957  797
## 88   88 1958  923
## 89   89 1959  975
## 90   90 1960  815
## 91   91 1961 1020
## 92   92 1962  906
## 93   93 1963  901
## 94   94 1964 1170
## 95   95 1965  912
## 96   96 1966  746
## 97   97 1967  919
## 98   98 1968  718
## 99   99 1969  714
## 100 100 1970  740
```

```
plot.ts(my_data$Nile, main = "Nile River Flow", ylab = "Flow", xlab = "Time")
```



The original time series of Nile River flow shows **a slight downward trend**, we apply **first differencing** to remove this trend.

```
adf0=ur.df(my_data$Nile,type="trend",lags=1)
summary(adf0)

##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -396.28  -93.67  -11.34   87.67  323.01
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 569.0062   123.8960   4.593 1.36e-05 ***
## z.lag.1      -0.5471     0.1142  -4.791 6.19e-06 ***
## tt           -1.4059     0.5865  -2.397   0.0185 *
## z.diff.lag   -0.1285     0.1021  -1.259   0.2113
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 140.9 on 94 degrees of freedom
## Multiple R-squared:  0.3263, Adjusted R-squared:  0.3048
## F-statistic: 15.18 on 3 and 94 DF,  p-value: 3.933e-08
##
##
## Value of test-statistic is: -4.7908 7.7108 11.4787
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

Running the dickey-fuller test to verify what we concluded from the graph that the series is not stationary. The p-value for z.lag.1 is very small compared to the 0.05 significance level, so we reject the null hypothesis that

$$\delta$$

= 0 which indicates that the series might be stationary but the tt p-value is also smaller than 0.05 yeilding a rejection to the null that

$$\beta = 0$$

. Thus, the two conclusions contradict.

```
augmented0<-adf.test(my_data$Nile,k=1)
```

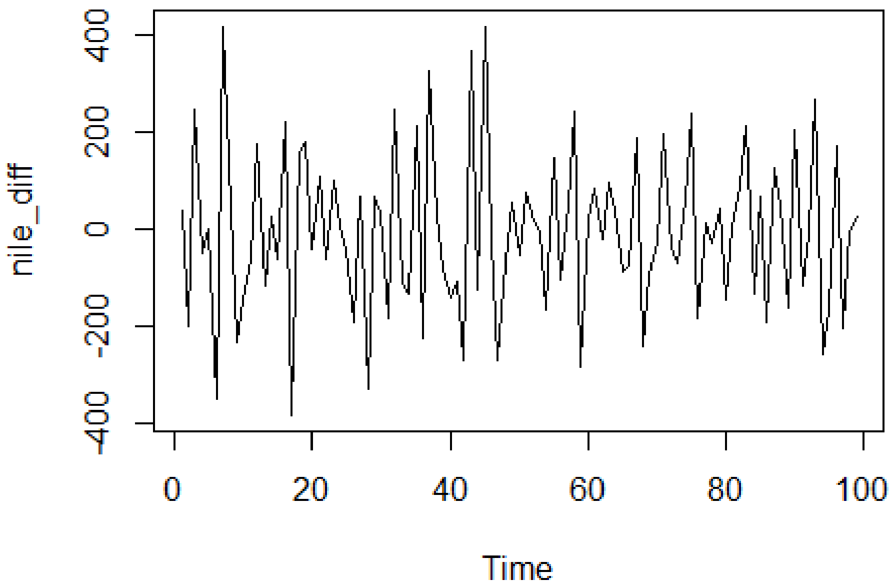
```
## Warning in adf.test(my_data$Nile, k = 1): p-value smaller than printed p-
value
```

```
augmented0
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  my_data$Nile
## Dickey-Fuller = -4.7908, Lag order = 1, p-value = 0.01
## alternative hypothesis: stationary
```

The augmented test shows a p-value of 0.01 which is less than the significance level so we reject the null that the series is not stationary, we will try differencing to remove the trend and do the tests again.

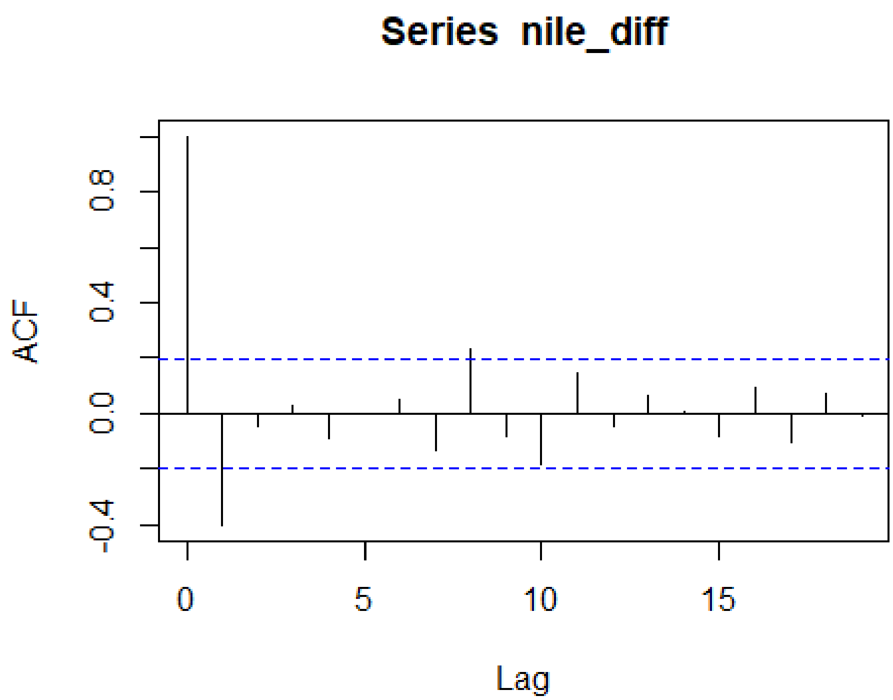
```
nile_diff <- diff(my_data$Nile, 1)
plot.ts(nile_diff)
```



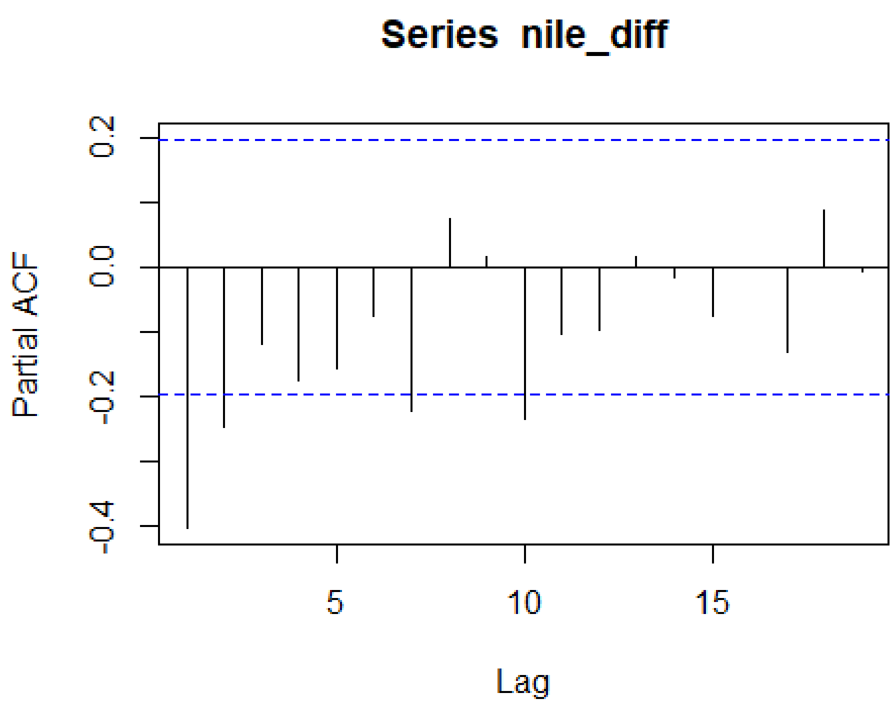
The differenced series fluctuates around a constant mean with approximately stable variance, indicating that the time series is now stationary.

Checking ACF and PACF

```
dif1_acf<-acf(nile_diff)
```




```
dif1_pacf<-pacf(nile_diff)
```



Looking at the ACF graph has significant spikes at lags 0 and 1 and the other spikes are ignored since they are at very late lags so we consider it cutting off after lag 1. The PACF shows significant spikes at 1 and 2 and then it starts going up and down which we might consider cutting off as well ignoring the late lags. This is inconclusive to determine the values of p and q in ARIMA(p,1,q) but we will be looking at different models and compare their AIC values to decide.

###Running ADF again to check for stationarity

```
adf=ur.df(nile_diff,type="trend",lags=1)
summary(adf)

##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -356.84 -101.85   -3.09   90.91  452.41
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.02440   31.60935  -0.064   0.9491
## z.lag.1     -1.74214    0.16718 -10.421   <2e-16 ***
## tt          -0.07152    0.55141  -0.130   0.8971
## z.diff.lag   0.24445    0.09981   2.449   0.0162 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 152.1 on 93 degrees of freedom
## Multiple R-squared:  0.721,	Adjusted R-squared:  0.712
## F-statistic: 80.13 on 3 and 93 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -10.4208 36.2115 54.3087
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47

augmented<-adf.test(nile_diff,k=1)
```

```
## Warning in adf.test(nile_diff, k = 1): p-value smaller than printed p-
value
```

```
augmented
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  nile_diff
## Dickey-Fuller = -10.421, Lag order = 1, p-value = 0.01
## alternative hypothesis: stationary
```

After running the test again, we look at the p-values for z.lag.1 and tt. The p-value for z.lag.1 is smaller than 0.05 so we reject the null hypothesis that

$$\delta = 0$$

or the series has a unit root and conclude that series is stationary. Furthermore, the p-value for tt shows a value much larger than the significant level so we fail to reject the null hypothesis that

$$\beta = 0$$

and now we are more likely to confirm that the series is stationary. Finally, looking at augmented test we find p-value is smaller than 0.05 so we reject the null that series is not stationary and thus we do not need further differencing. ### Estimating Model Parameters

```
m1<-arima(nile_diff,order=c(0,1,1))
arima1<-arima(nile_diff,order=c(1,1,1))
arima2<-arima(nile_diff,order=c(2,1,1))
m1
```

```
##
## Call:
## arima(x = nile_diff, order = c(0, 1, 1))
##
## Coefficients:
##           ma1
##       -1.0000
## s.e.   0.0254
##
## sigma^2 estimated as 28269:  log likelihood = -643.58,  aic = 1291.16
```

```
arima1
```

```
##
## Call:
## arima(x = nile_diff, order = c(1, 1, 1))
##
## Coefficients:
##           ar1      ma1
```

```
##       -0.3924  -1.0000
## s.e.   0.0921   0.0261
##
## sigma^2 estimated as 23697:  log likelihood = -635.35,  aic = 1276.7
```

```
arima2
```

```
##
## Call:
## arima(x = nile_diff, order = c(2, 1, 1))
##
## Coefficients:
##           ar1      ar2      ma1
##       -0.4887  -0.2353  -1.0000
## s.e.   0.0979   0.0979   0.0268
##
## sigma^2 estimated as 22264:  log likelihood = -632.56,  aic = 1273.12
```

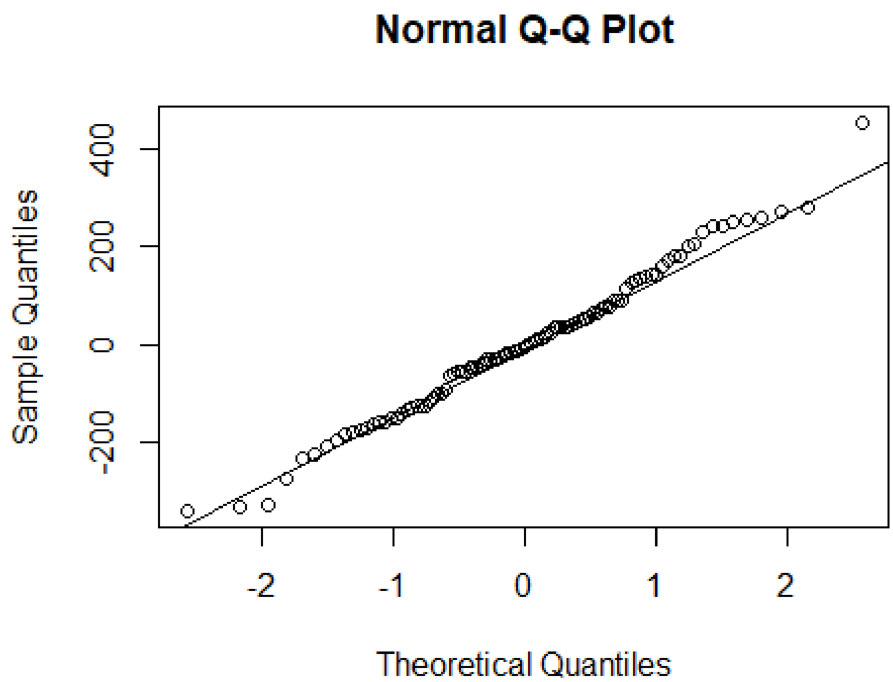
Checking the AIC values for different models hypothesized from the ACF and PACF graphs, we looked for ARIMA(1,1,1) and ARIMA(2,1,1) since they have the lowest values. We thought that ARIMA(2,1,1) would be overfitting the model since the change in AIC wasn't that significant alongside the coefficients of ma1 and ar1 and their standard errors; however, looking at the coefficient of ar2, we observe that it is not close to zero, so we decided to keep the ARIMA(2,1,1)

```
final_model=arima(nile_diff,order=c(2,1,1))
final_model
```

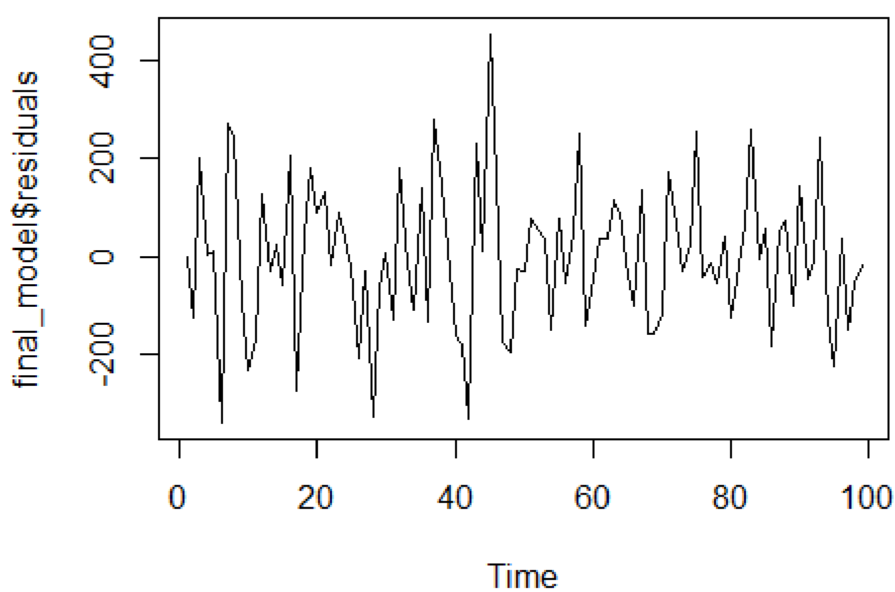
```
##
## Call:
## arima(x = nile_diff, order = c(2, 1, 1))
##
## Coefficients:
##           ar1      ar2      ma1
##       -0.4887  -0.2353  -1.0000
## s.e.   0.0979   0.0979   0.0268
##
## sigma^2 estimated as 22264:  log likelihood = -632.56,  aic = 1273.12
```

Checking Model Residual Assumptions (NICE Assumptions)

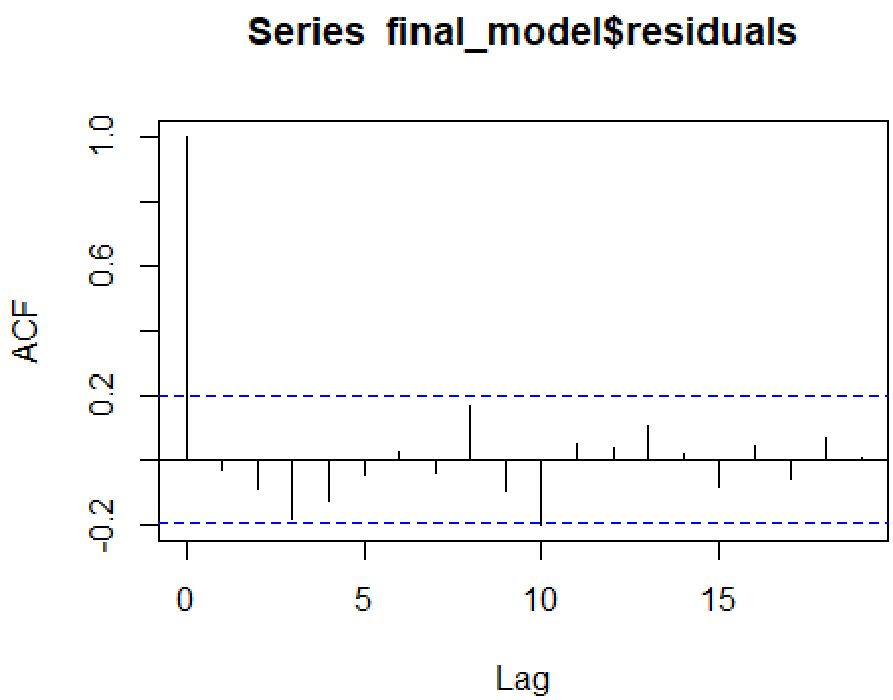
```
qqnorm(final_model$residuals)
qqline(final_model$residuals)
```




```
plot.ts(final_model$residuals)
```




```
acf(final_model$residuals)
```




```r
Box.test(final_model$residuals,lag=20,fitdf = 1)

##
##  Box-Pierce test
##
## data:  final_model$residuals
## X-squared = 16.909, df = 19, p-value = 0.5961
```

- Q-Q Plot: Residuals are approximately normally distributed, aligning closely with the diagonal reference line.
- Residual Time Series Plot: No obvious structure or trend remains. Residuals are randomly distributed around zero.
- ACF of Residuals: No significant autocorrelation is present; all autocorrelations fall within the 95% confidence bounds.
- Box-Pierce Test: Null Hypothesis: Residuals are white noise (i.e., uncorrelated).

Alternative Hypothesis: Residuals are not white noise (i.e., there is autocorrelation).

Since p-value is 0.5961, we fail to reject the null hypothesis. This suggests that the residuals are not significantly autocorrelated.

Therefore, ARIMA(2,1,1) is an appropriate model for the differenced Nile River flow data.

ARIMA(2,1,1) equation

The ARIMA(2,1,1) model can be expressed as:

$$(1 - \phi_1 B - \phi_2 B^2)(1 - B)Y_t = (1 + \theta_1 B)\varepsilon_t$$

Where:
Y_t: observed time series (e.g., Nile River flow)
B: backshift operator, i.e., $BY_t = Y_{t-1}$
ϕ_1, ϕ_2: autoregressive (AR) parameters
θ_1: moving average (MA) parameter
ε_t: white noise error term

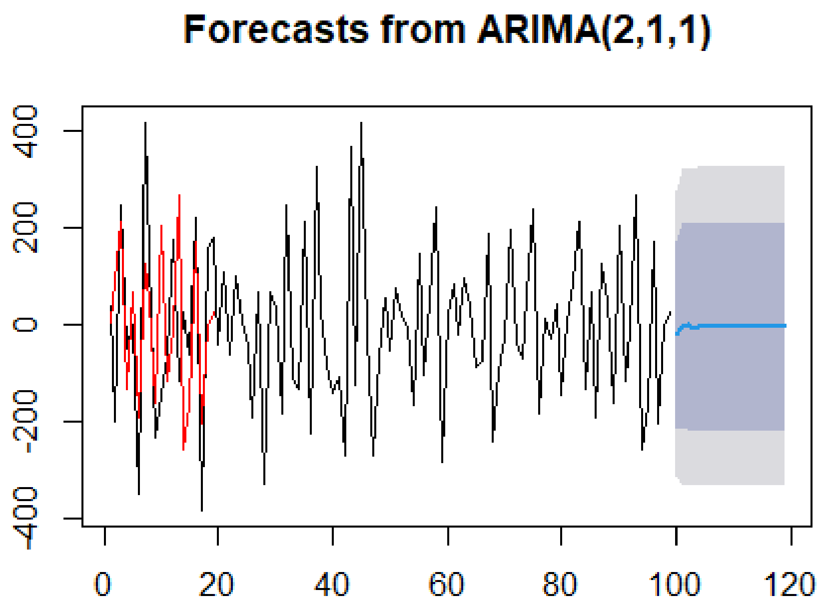
Using the parameters from the output it becomes: $$

(1 + 0.4887 B + 0.2353 B^2)(1 - B)Y_t = (1 - 1.0000 B)_t $$

$$\Delta Y_t = -0.4887 \cdot \Delta Y_{t-1} - 0.2353 \cdot \Delta Y_{t-2} + \varepsilon_t - 1.0000 \cdot \varepsilon_{t-1}$$

Forecasting

```
library(forecast)
train_values <- nile_diff[1:80]
test_values <- nile_diff[81:100]
l_ahead <- length(test_values)
fc <- forecast(final_model, h = l_ahead)
plot(fc)
lines(test_values, col = "red")
```




```
error_metrics <- accuracy(fc, test_values)
print(error_metrics)

##                       ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -0.04208607 148.4560 115.7861      Inf      Inf 0.5016819
## Test set      4.33413300 153.2859 129.8912 111.4898 111.4898 0.5627969
##                     ACF1
## Training set -0.03383688
## Test set              NA
```

We splitted the data into 80:20 as training and testing respectively. Looking at the error values representing the performance metrics of our predictions, the mean error is very close to zero which shows minimal error in prediction whereas RMSE values are a bit large depending on the data scale. Other values close to infinity might have appeared because of very small values causing division by zeroes. Overall, the model performance is moderately well and acceptable.

Spectral Analysis

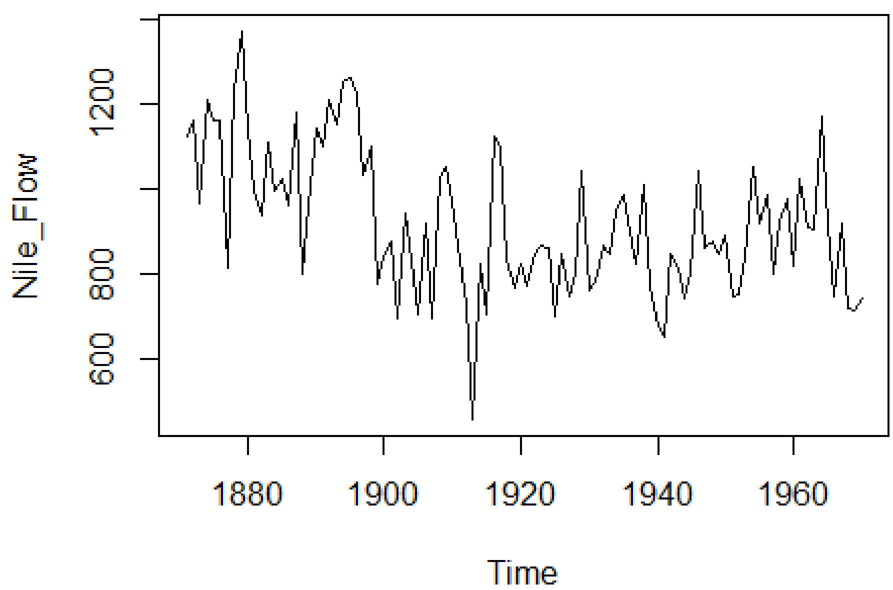
The goal of using Spectral Analysis is to view the time series from a frequency perspective. This involves decomposing the time series into a linear combination of sine and cosine functions at different frequencies.

```
if (!require(readr)) install.packages("readr")

## Loading required package: readr
```

```r
if (!require(forecast)) install.packages("forecast", dependencies = TRUE);
library(readr)
library(forecast)
```

Load and visualize the Nile River dataset

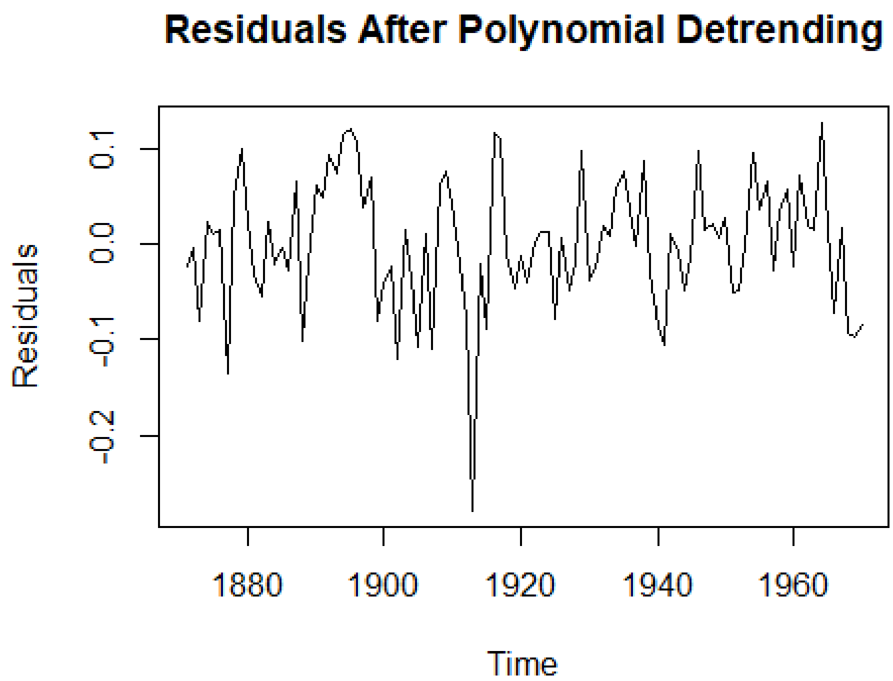
```r
my_data<-read.csv("Nile.csv")

Nile_Flow <- ts(my_data$Nile,frequency=1,start=c(1871))
plot.ts(Nile_Flow)
```



Log-transform and detrend the series manually

Firstlt, the series should be detrended before a spectral analysis. Since the data was recorded annually, we did not proceed with decompose() function as no seasonality. The frequency in this case is 1. Therefore, we are going to proceed with Log-transform and detrend the serious manually.

```r
log_Nile <- log10(Nile_Flow)
t <- time(log_Nile)
trend_model <- lm(log_Nile ~ poly(t, 2))  # Polynomial trend (degree 2)
residuals_ts <- ts((trend_model$residuals), start = 1871)

# Plot residuals
plot(residuals_ts, main = "Residuals After Polynomial Detrending", ylab =
"Residuals")
```



This plot shows the Nile River flow after removing long-term trends via a quadratic regression on the log-transformed data. The residuals fluctuate around zero, indicating that the trend has been successfully removed.

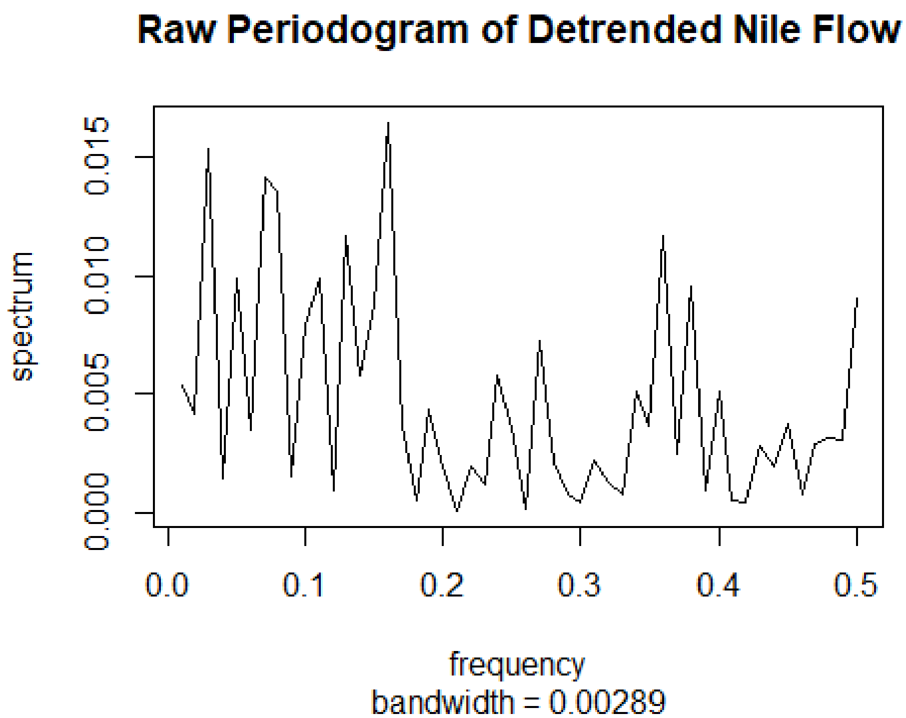
The periodogram is calculated using the Discrete Fourier Transform (DFT).

Periodogram shows the contribution of different frequencies to the variance of the time series.

- The **x-axis** represents frequency, typically in cycles per time unit.
- Since our data is annual (frequency=1), the unit is cycles per year.
- A peak in the periodogram indicates a frequency at which the time series has a strong periodic component.

Raw Periodogram: Frequency Decomposition

```
spec.pgram(residuals_ts, detrend = FALSE, log = "no", main = "Raw Periodogram of Detrended Nile Flow")
```



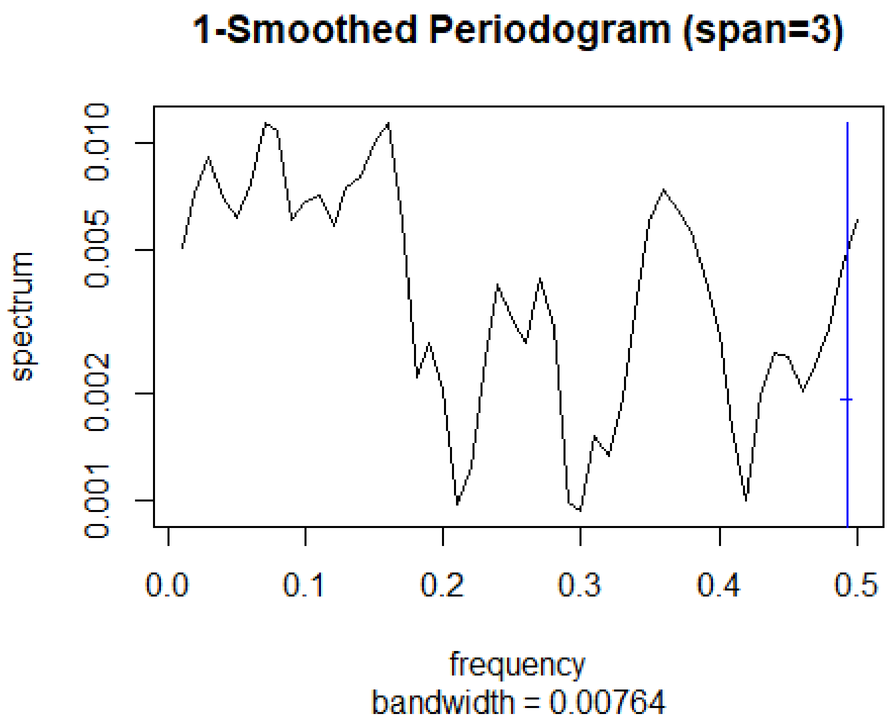
The periodogram displays the strength of different frequencies present in the detrended series. Peaks in the

spectrum suggest dominant periodic components. Notably, the peak around frequency 0.15 implies a possible 15-year cycle, as Period=$$$$. This supports the presence of medium-term oscillations in river flow, beyond the removed trend.

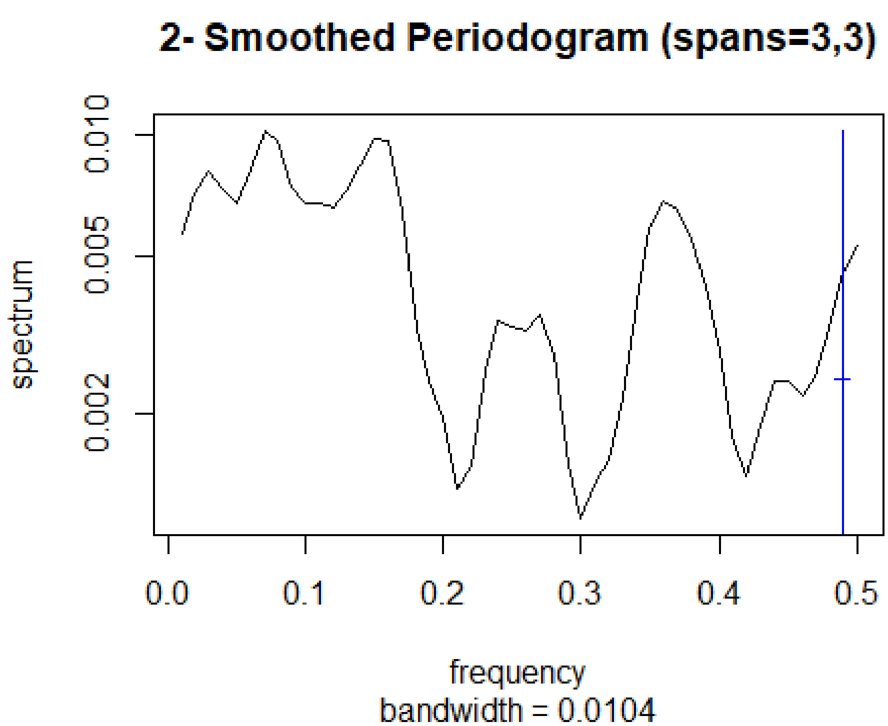
Smoothed Periodograms for Stability

We applied different smoothing spans to the periodogram to assess spectral stability. Lower spans (e.g., 3) preserve fine detail but are more sensitive to noise, while higher spans (e.g., 7,7) yield smoother, more stable spectra that highlight dominant periodicities clearly.

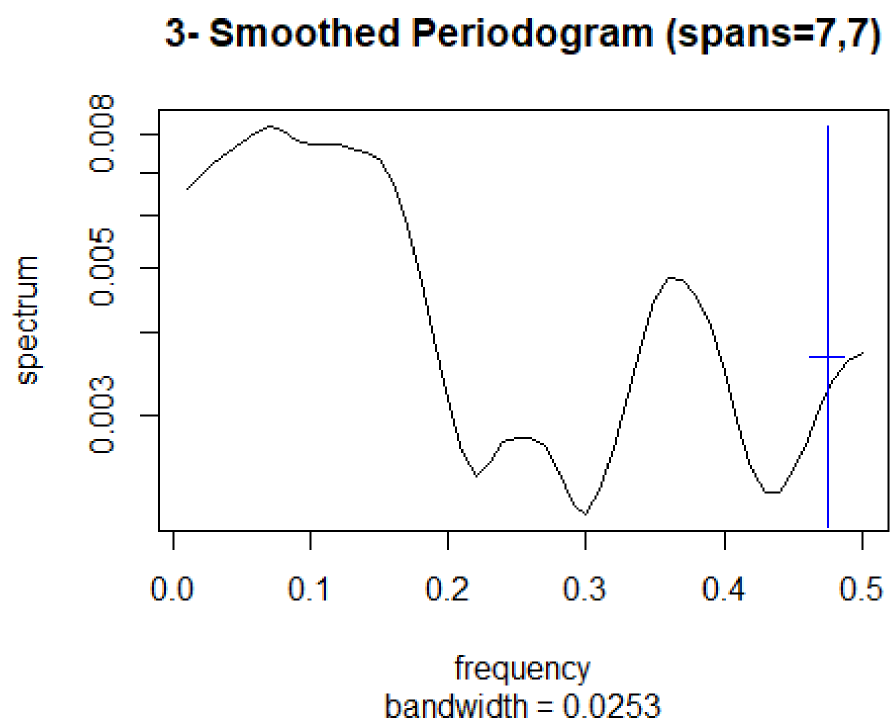
```
spec.pgram(residuals_ts, spans = c(3), detrend = FALSE, main = "1-Smoothed
Periodogram (span=3)")
```




```r
spec.pgram(residuals_ts, spans = c(3, 3), detrend = FALSE, main = "2-
Smoothed Periodogram (spans=3,3)")
```



```
spec.pgram(residuals_ts, spans = c(7, 7), detrend = FALSE, main = "3-
Smoothed Periodogram (spans=7,7)")
```



(1) Smoothed Periodogram (span = 3) This smoothed periodogram retains much of the detail in the raw spectrum while reducing noise. The dominant peak around frequency 0.15 still stands out, corresponding to a cycle of approximately 6.67 years.

(3)Smoothed Periodogram (spans = 7,7) With greater smoothing, the periodogram becomes more stable. The dominant peak around frequency 0.15. However; peaks are less than the one with span=3 which shows primary periodic components.

The dominant frequency used in harmonic regression was extracted from a **moderately smoothed periodogram** (`spans = c(3,3)`) to ensure robustness while preserving spectral resolution.

```
spec <- spec.pgram(residuals_ts, spans = c(3,3), plot = FALSE)
dominant_freq <- spec$freq[which.max(spec$spec)]
dominant_period <- 1 / dominant_freq
cat("Dominant period (years):", round(dominant_period, 2), "\n")

## Dominant period (years): 14.29
```

This suggests that the Nile River flow exhibits a long-term oscillation approximately every 14 years after removing the long-term trend.

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

Harmonic Regression: Fit the Dominant Cycle

```r
# Use the previously found dominant frequency
f <- dominant_freq

# Time vector
t_vals <- time(residuals_ts)

# Create sine and cosine components
harmonic_data <- data.frame(
  y = as.numeric(residuals_ts),
  sin_term = sin(2 * pi * f * t_vals),
  cos_term = cos(2 * pi * f * t_vals)
)

# Fit harmonic regression
harmonic_model <- lm(y ~ sin_term + cos_term, data = harmonic_data)

# Summary of model
summary(harmonic_model)
```

```
##
## Call:
## lm(formula = y ~ sin_term + cos_term, data = harmonic_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.25331 -0.03074 -0.00059  0.04076  0.12989
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.032e-18  6.597e-03   0.000   1.0000
## sin_term     7.331e-03  9.329e-03   0.786   0.4339
## cos_term    -2.580e-02  9.329e-03  -2.766   0.0068 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06597 on 97 degrees of freedom
## Multiple R-squared:  0.07852,    Adjusted R-squared:  0.05953
## F-statistic: 4.133 on 2 and 97 DF,  p-value: 0.01894
```

Although harmonic regression revealed a significant ~14-year cycle, the model explained less than 8% of the residual variance and lacks predictive strength on its own. Therefore, we do not proceed with standalone forecasting using this model. Instead, an ARIMA model or LSTM is recommended for time series forecasting.

###LSTM After applying LSTM (code notebook and results attached below), it yielded the following measures of mean squared and root mean squared errors Mean Squared Error (MSE) on test data: 25371.77545236458 Root Mean Squared Error (RMSE) on test data:

159.28520161133795 We conclude that errors are relatively very high compared to other error measures from spectral analysis and ARIMA. Thus, we stick to fitting the ARIMA model since it performed the best at predictions which might suggest that our data was originally simple and linear.

Appendix and Sources Used: Dataset: https://www.kaggle.com/datasets/lsind18/flow-of-the-river-nile

Spectral Analysis:https://vlyubchich.github.io/tsar/l13_spectral.html

LSTM Colab Notebook: https://colab.research.google.com/drive/1S81xWdGR0kIT6MVbncRjzAnZkbfKs9rP?usp=sharing

ChatGPT: Asked about performing harmonic regression, LSTM and their codes. History is found at: https://chatgpt.com/share/683592f8-d0cc-8006-9e7e-f679894198d7