Time Series Analysis - A Tutorial

Rosskopf,E.; Cordes, M.; Lumiko, J.

November 24, 2014

Abstract

Tutorial for time series analysis in R...

Contents

1 Introduction

This tutorial assumes that the reader has some basic knowledge of time series analysis, and the principal focus of the tutorial is not to explain time series analysis, but rather to explain how to carry out these analyses using R. If you are new to time series analysis, and want to learn more about any of the concepts presented here, We would highly recommend the Open University book "Time series" (product code M249/02), available from from the Open University Shop.

2 Getting started

2.1 Packages

Before we get started, please make sure to set a working directory and download the necessary packages listed below.

Useful packages for time series analysis:

```
> library(tseries)
> library(nlme)
> library(car)
> library(knitr)
> library(xtable)
> library(SweaveListingUtils)
> library(stats)
> library(forecast)
> library(AICcmodavg)
> library(TTR)
```

2.2 Dataset (CO2-Concentrations)

The first dataset we will work with consists of monthly CO2-concentrations [ppm] in the atmosphere, measured over time at the famous Mauna Loa Station on Hawaii.
To download this dataset, just use the code provided below.

```
> url<-"ftp://aftp.cmdl.noaa.gov/products/trends/co2/co2_mm_mlo.txt"
> dest<-"C:/Users/Mallypop/Desktop/FINAL SWEAVE VERSION/myts.txt"
> download.file(url, dest )
> co2month=read.table(dest, skip=72)
> co2month
```

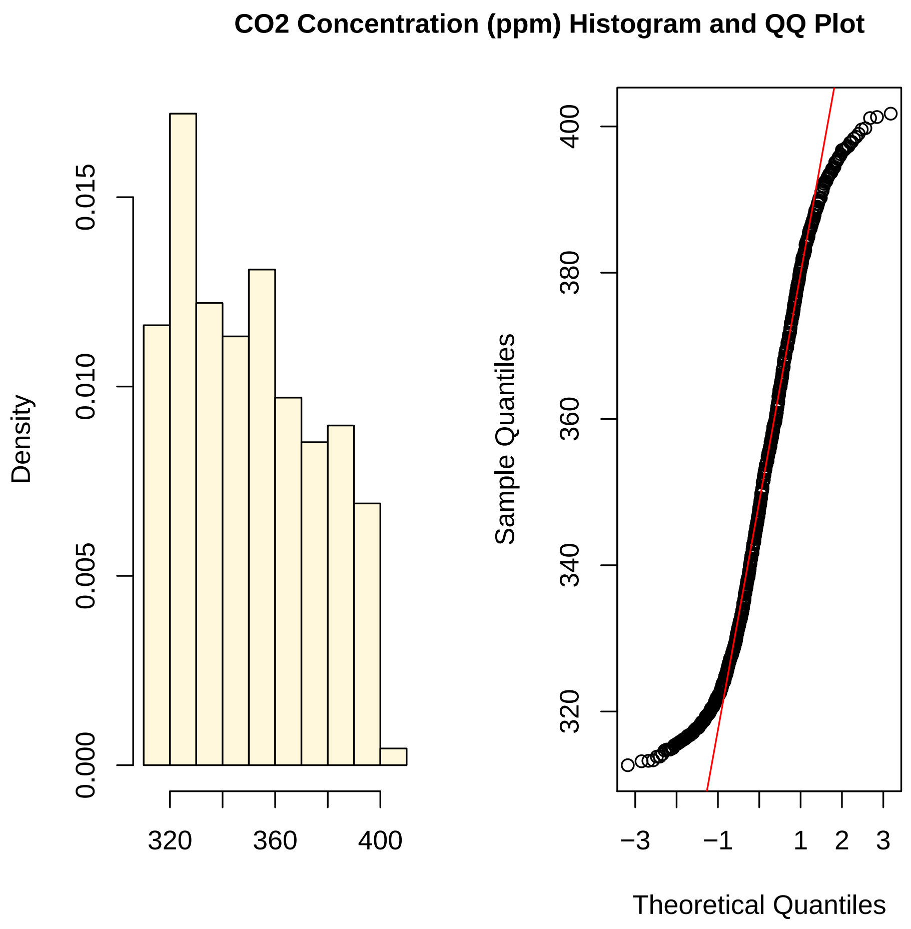
Note:"dest" represents a randomly chosen name for a text file in which the CO2-dataset will be saved. Feel free to adjust the name and directory.

2.3 Dataset Visualization

It can be really useful to visualize your dataset before you transform it into a timeseries (ts) in order to detect potential errors.

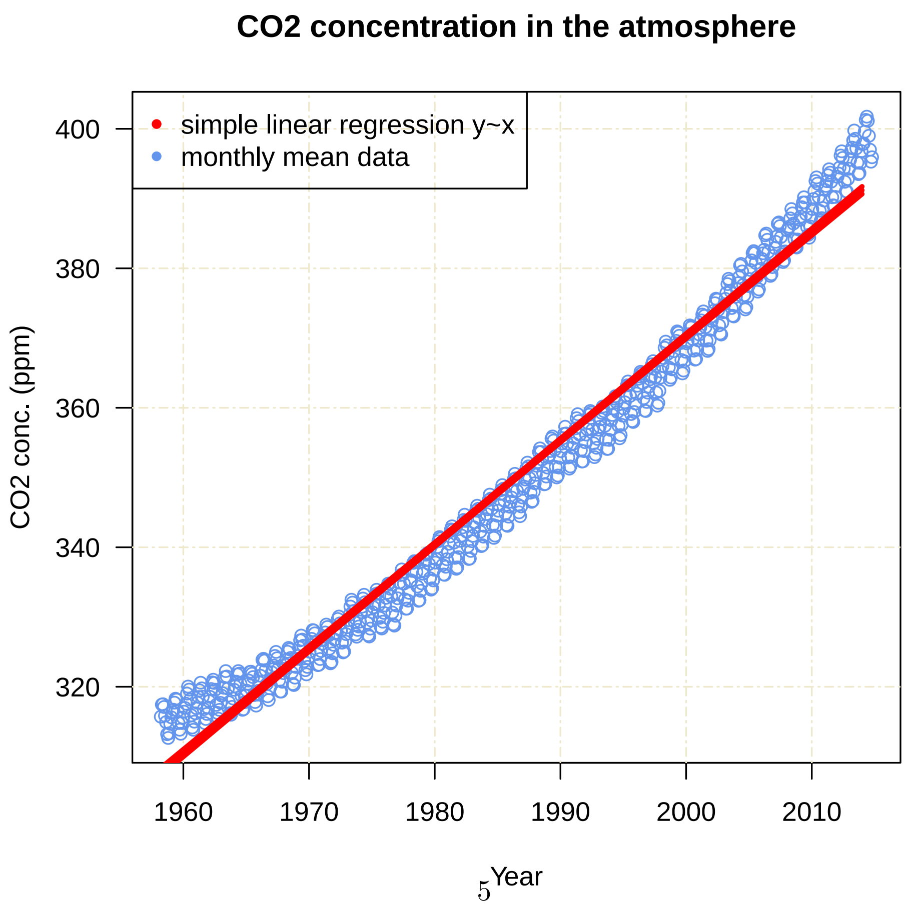
2.3.1 Histogram & QQ-Plot

```
> data = co2month[,c(3,5)]
> colnames(data)= c("year", "co2")
> attach(data)
> x = co2
> op = par(mfrow = c(1,2),
+           mar = c(5,4,1,2)+.1,
+           oma = c(0,0,2,0))
> hist(co2, freq=F, col = "cornsilk",xlab = "", main = "")
> qqnorm(x, main = ""); qqline(x,col = 'red')
> par(op)
> mtext("CO2 Concentration (ppm) Histogram and QQ Plot", line = 2.5,font = 2,cex = 1.0)
```



2.3.2 Plotting the fitted values

```
> # Run a linear model
> #
> datalm = lm( co2 ~ year)
> #
> # Fit predict values
> #
> MyData=data.frame(year=seq(from=(1958),to=2014, by=0.1))
> pred=predict(datalm, newdata=MyData, type="response", se=T) #poission: type ="LINK",
> #
> # Plot the fitted values
> #
> plot(year, co2, type="n",las=1, xlab="Year", ylab="CO2 conc. (ppm)", main="CO2 concer
> grid (NULL,NULL, lty = 6, col = "cornsilk2")
> points(year, co2, col="cornflowerblue" )
> #
> # Write confidence interval
> #
> F=(pred$fit)
> FSUP=(pred$fit+1.96*pred$se.fit) # make upper conf. int.
> FSLOW=(pred$fit-1.96*pred$se.fit) # make lower conf. int.
> lines(MyData$year, F, lty=1, col="red", lwd=3)
> lines(MyData$year, FSUP,lty=1, col="red", lwd=3)
> lines(MyData$year, FSLOW,lty=1, col="red", lwd=3)
> legend("topleft",c("simple linear regression y~x", "monthly mean data"),
+        pch=c(20,20), col=c("red", "cornflowerblue"))
>
```



This plot can be used to observe if there are outliers which could possibly bias the model. However the poly-1 linear regression is not accurate in fitting the CO2-dataset. This is due to the present autocorrelation that not yet has been taken into account. Neglecting this factor will always effect the accuracy of the model results. The standard errors are lower than their true values thus giving high statistical significance with a p-value lower than it should be. The clue in statistical modelling is to present the correct statistical evidence, which would be highly biased with a linear model.

2.4 Dataset Transformation

It is essential to transform your dataset into a timeseries (ts) if you seek for an accurate and extensive analysis of the data.
The data stored as a dataframe needs to be transformed with the important columns into the class of a time series to continue working on it properly. If you have monthly data you have to set the deltat of the function ts() to deltat=1/12 describing the sampling period parts between successive values xt and xt+1. Your time series should somehow look like table 1.

Original Data

```
> xtable(head(data), caption="Original CO2-Data")
```

	year	co2
1	1958.21	315.71
2	1958.29	317.45
3	1958.38	317.50
4	1958.46	317.10
5	1958.54	315.86
6	1958.62	314.93

Table 1: Original CO2-Data

Transformation

```
> yourts=ts(co2, c(1958,3),c(2014,10), deltat=1/12)
> class(yourts)
```

[1] ”ts”

2.5 Time-Series Visualization

It is important to get a quick overview of your data. Some simple plots for visualization are quite helpful.

2.5.1 Time-Series Plot

```
> par(mfrow=c(1,1))
> plot.ts(yourts,las=1, xlab="Year", ylab="CO2 conc. (ppm)", main="CO2 concentration in
> grid (NULL,NULL, lty = 6, col = "cornsilk2")
> points(yourts, col="cornflowerblue" )
> k <- 5
> lines(year,filter(co2, rep(1/k,k) ),col = 'red', type="l", lwd = 3)
> legend("topleft",c("simple moving average", "monthly mean data"),
+ pch=c(20,20), col=c("red", "cornflowerblue"))
```

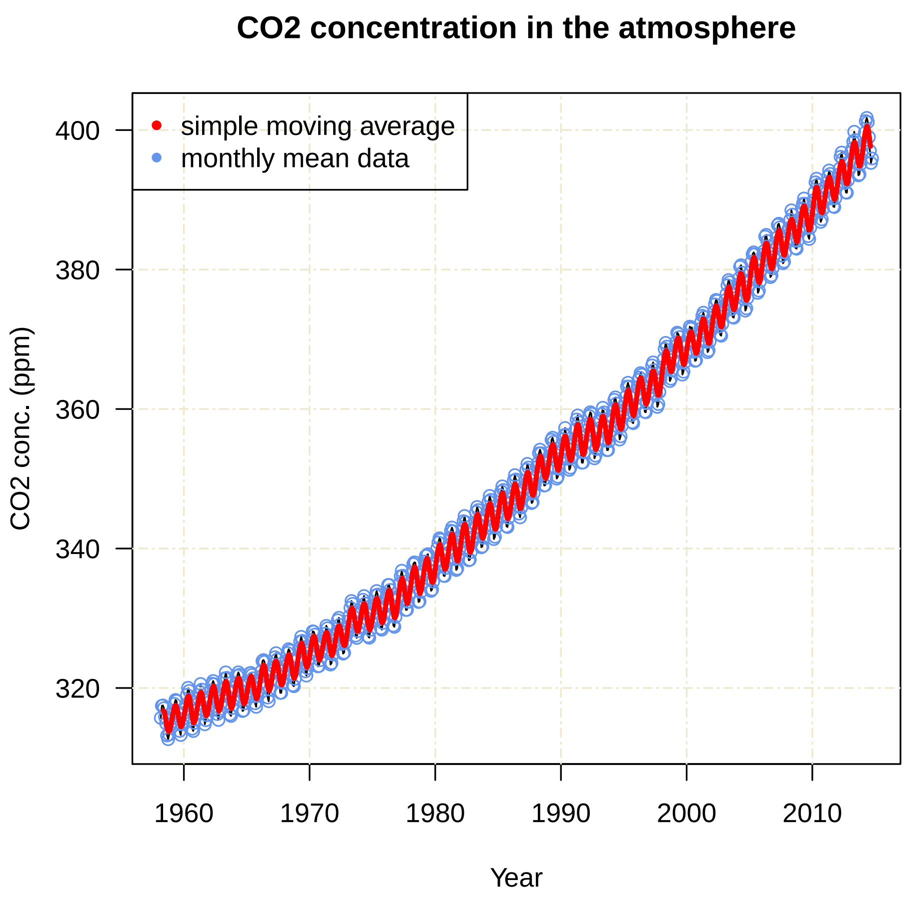


Figure 1: Visualization of the CO2 Concentrations

Note: The red line in plot 1 was computed with a simple moving average. It is not enough to just run a MA.

2.5.2 ACF, PACF, SPECTRUM

Since time-series data usually violates the independence assumption of the model, the standard error is potentially too small. As the data are regularly spaced in time, we can easily employ the autocorrelation function to investigate residuals correlations in the model errors. **The acf() function** can be used for that, which produces a plot of the correlogram.
Another nice procedure is to run the **autocorrelation function** with its complementary **partial acf** and the **spectrum** showing the spectral density of your time series at frequencies corresponding to the possibly approx. Fourier frequencies.

```
> op <- par(mfrow = c(3,1),
+           mar = c(2,4,1,2)+.1,
+           oma = c(0,0,2,0))
> acf(x, xlab = "")
> pacf(x, xlab = "")
> spectrum(x, xlab = "", main = "")
> par(op)
> mtext("CO2 Concentration (ppm) correlogram",
+       line = 2.5,
+       font = 2,
+       cex = 0.8)
> op <- par(mfrow = c(3,1),
+           mar = c(2,4,1,2)+.1,
+           oma = c(0,0,2,0))
> acf(resid(datalm), xlab = "")
> pacf(resid(datalm),xlab = "")
> spectrum(resid(datalm), xlab = "", main = "")
> par(op)
> mtext("Model residual correlogram",
+       line = 2.5,
+       font = 2,
+       cex = 1.2)
```

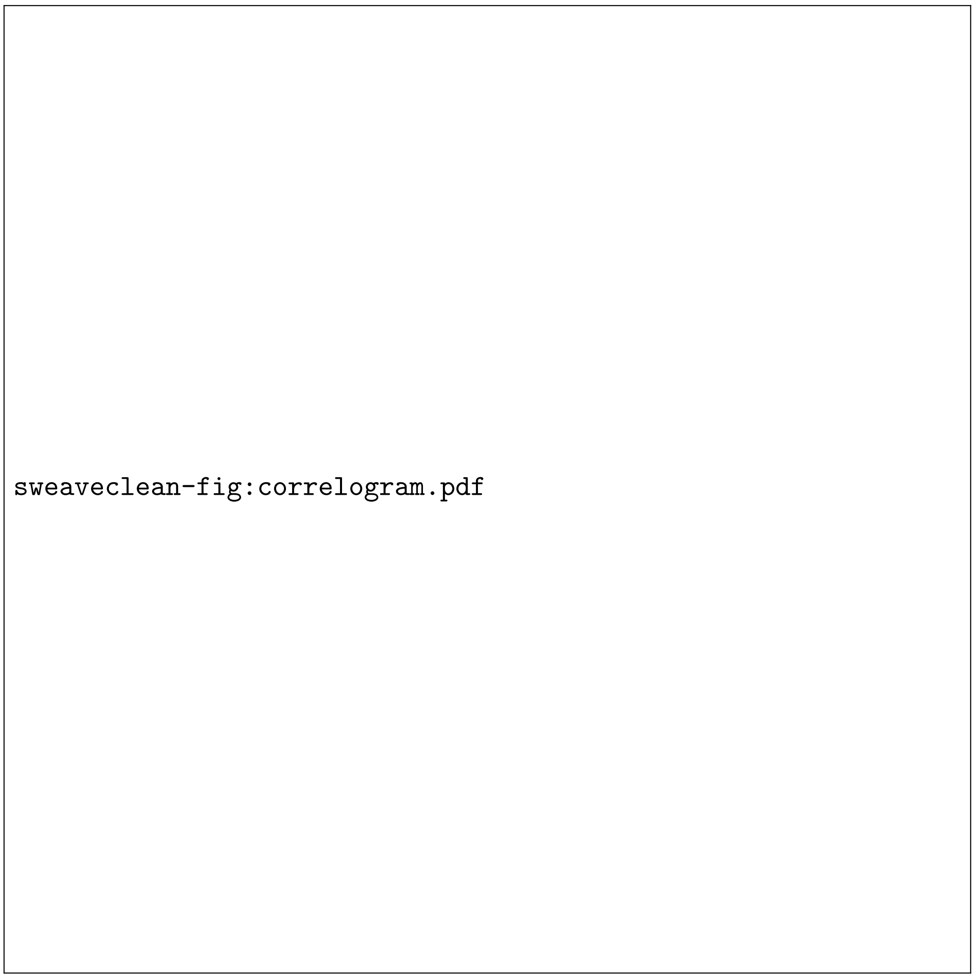


Figure 2: Correlogram of time series and residuals of lm

Explain the acf , pacf, spectrum here.
ACF & PACF:
The generated correlogram reveals that there are major autocorrelations.
There is a strong correlation at lag 1,a weaker correlation at lag 2, and a noticeable correlation at lag 3. Such a correlation pattern is typical for an autoregressive process where most of the sequential dependence can be explained as a flow-on effect from a dependence at lag 1.

In an autoregressive time series, an independent error component, or ”innovation” is associated with each time point. For an order p autoregressive time series, the error for any time point is obtained by taking the innovation for that time point, and adding a linear combination of the innovations at the p previous time points. (For the present time series, initial indications are that p = 1 might capture most of the correlation structure.)”
(autosmooth.pdf)

Spectrum: easier to interpret the acf / log scaled / strong cycles where spectrum max. occurs Here the highest maximum is at about 0.75. 1/0.75 = 12 meaning a 12 month cycle is occuring here .

The residuals in a time series are serially correlated. The ACF is waving and decreases only slowly, which could be an identification of non-stationarity (If the ACF would drop to zero quickly, the time series would be stationary). We stop all diagnostics here for our clearly wrong model and go on to investigate the different components of our time series.

3 Decomposition of Time Series

A time serie consists of 3 components; a trend component, an irregular (random) component and (if it is a seosonal time series) seasonal component.

3.1 Decomposing Seasonal Data

We can decompose the ts and plot these components:

```
> plot(decompose(yourts))
>
```

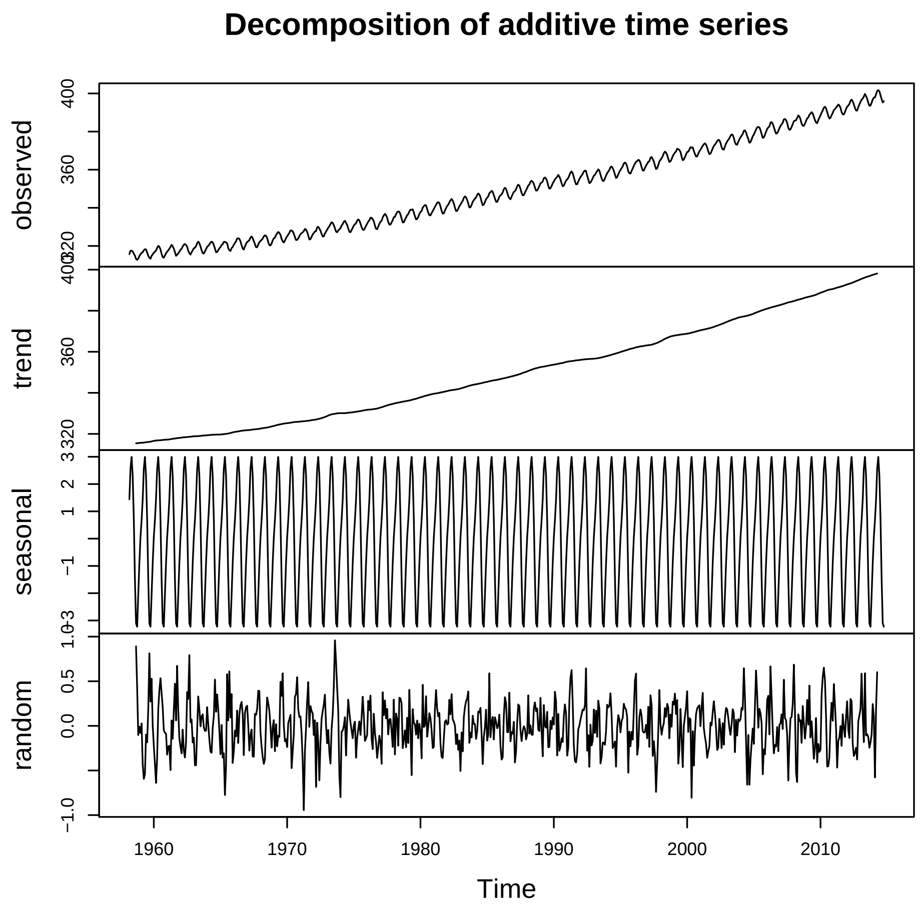


Figure 3: Decomposition of the CO2 Time Series

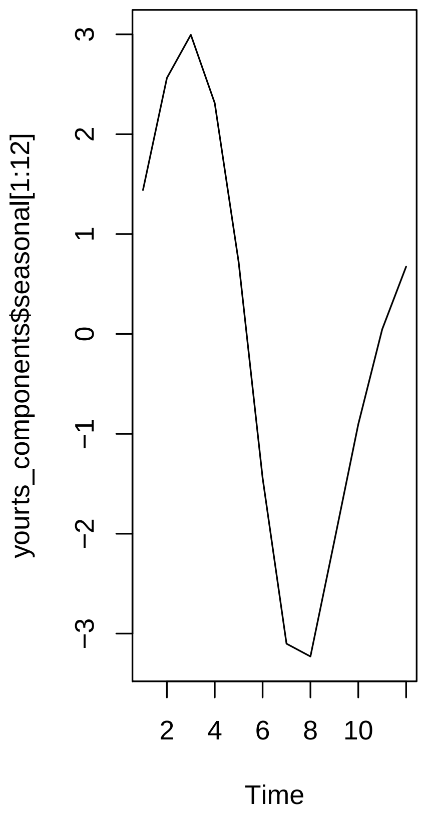
We can see each component with:

```
> yourts_components<- decompose(yourts)
```

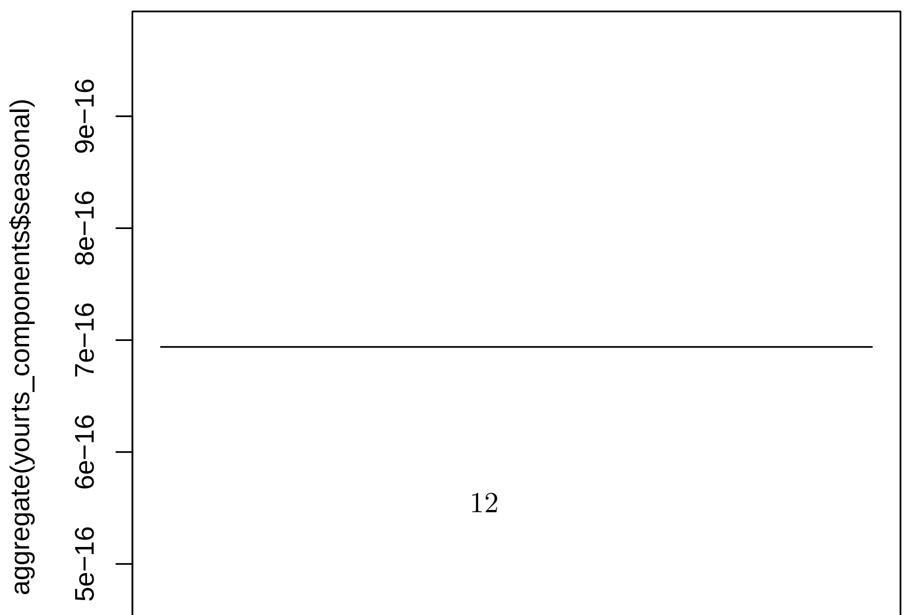
```
> yourts_components$seasonal
```

It seems that our seasonal component is positiv until the sommer months, were it turns to be negativ and turning to be positiv again in the winter

```
> par(mfrow=c(1,2))
> #we can see the trend for the first year:
> ts.plot(yourts_components$seasonal[1:12])
>
```



```
> ts.plot(aggregate(yourts_components$seasonal))
> #and we can see that this seasonal component is constant over all the years
>
> yourts_seasonallyadjusted <- yourts - yourts_components$seasonal
> #We can then plot the seasonally adjusted time series using "plot()"
```



It seems that our data can probably be described using an additive model, since the random fluctuations in the data are roughly constant in size over time (constant seasonal component)

3.2 Decomposing Non-Seasonal Data (Annual Flow Of The River Nile)

Our second dataset contains the measurements of the annual flow of the river Nile at Ashwan.(1871–1970) This dataset can be found in the packagedatasets.

First we visualize our data:

```
> str(Nile)

 Time-Series [1:100] from 1871 to 1970: 1120 1160 963 1210 1160 1160 813 1230 1370 114

> plot(Nile, main="Annual flow of teh Nile", ylab="Flow [V/a]")
```

Figure 5: Annual Flow of the Nile

A non-seasonal time series consists of a trend and an irregular component. To estimate the trend component of a non-seasonal time series that can be described using an additive model, it is common to use a smoothing method, such as calculating the simple moving average of the time series.
The SMA() function in the "TTR" R package can be used to smooth time series data using a simple moving average.

```
> library(TTR)
> plot(SMA(Nile,n=20))
```

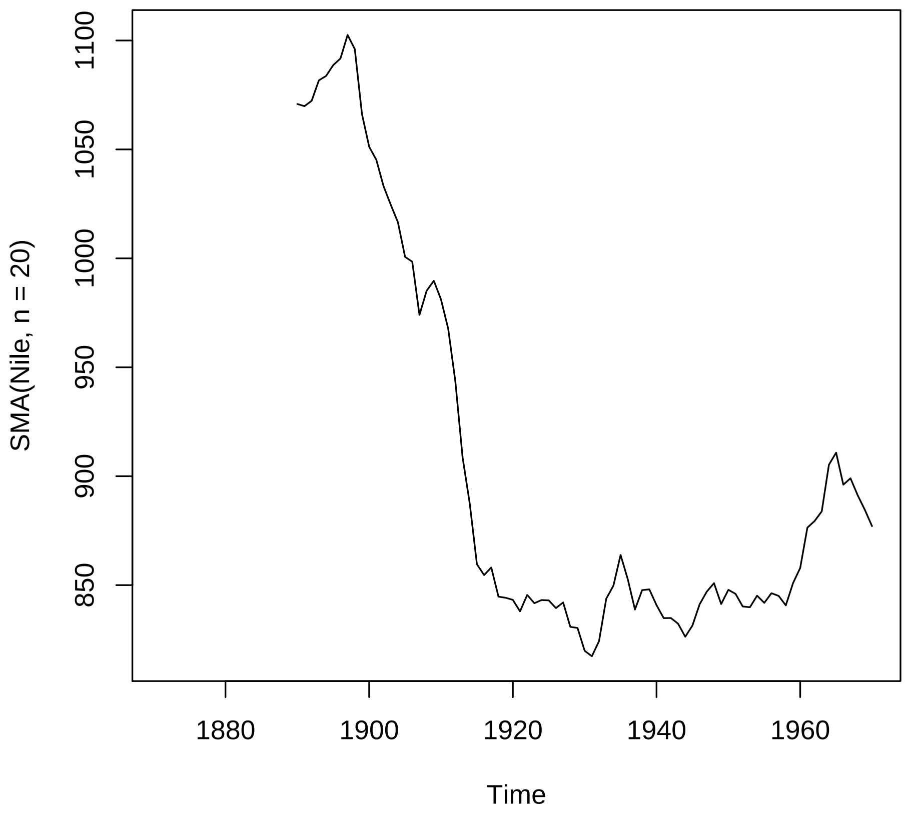


Figure 6: Trend of the Annual Flow of the Nile

We can see that there was a negative trend until 1920 and that it becomes positiv since then. However it is difficult to make a clear statement of the trends, since there are high fluctuations in the data.
Now we can procede plotting the correlograms:

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

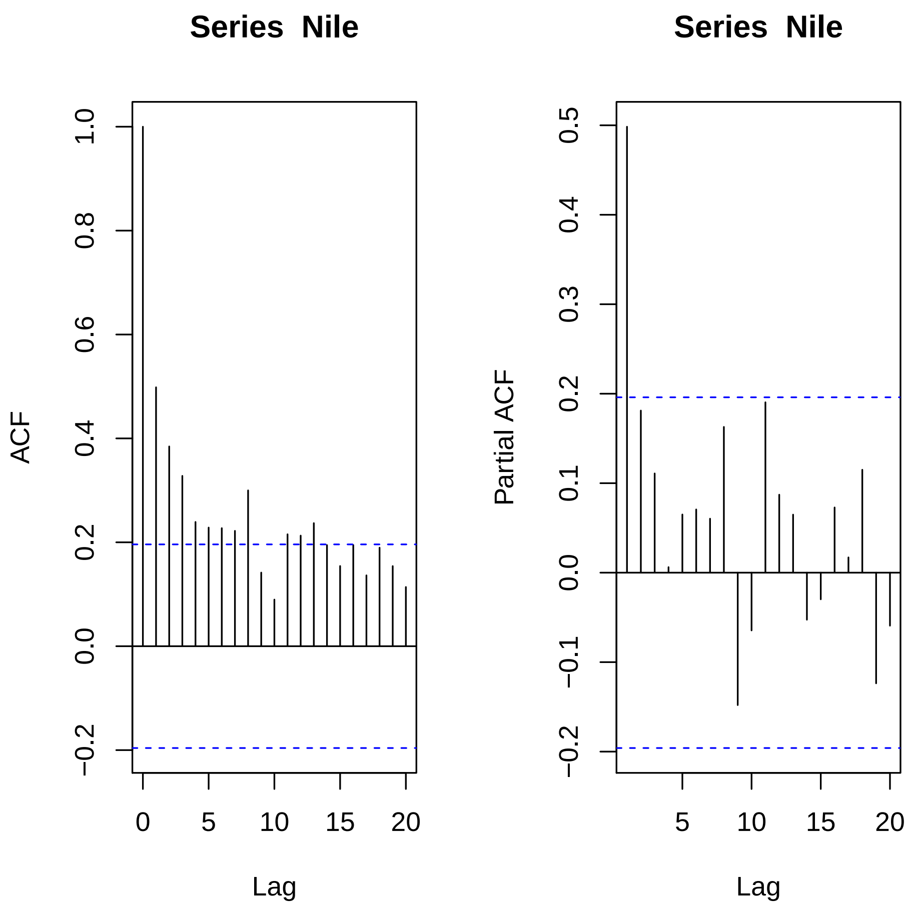


Figure 7: Autocorrelations and Partial autocorrelations.

We find a significant autocorrelation at the first lags only. This means that the autocorrelation is not constant over time. The partial autocorrelation plot does not show any significant autocorrelation

4 Modelling the time series

The best model for a time series needs to have residuals as white noise terms. There are different ways to approach this task.

5 Forecasts

We have three different options to make (up to now)

1. predict
2. Holt Winters
3. Arima forecasts

5.1 Holt-Winters Exponential Smoothing

If we have a time series that can be described using an additive model,we can short-time forecast using exponential smoothing.
Preconditions:forecast errors are uncorrelated and are normally distributed with mean zero and constant variance.

```
> hw<-HoltWinters(yourts)
> #the alpha value tells us the weight of the previous values for the forecasting
> #values of alpha that are close to 0 mean that little weight is placed on the most recen
> #gamma is for the seasonality
> plot(hw)
>
```

Holtwinters just makes forecasts for the time period covered by the original data.If we want to forecast for the future, we need the packeged ”forecast”.

```
> hw1<- forecast.HoltWinters(hw, h=12)
> #for the next year
> plot.forecast(hw1, main="Prediction for the next year",shadecols = "oldstyle")
> #for next 10 years
> hw10<- forecast.HoltWinters(hw, h=120)
> plot.forecast(hw10, main="Prediction for the next 10 years", shadecols = "oldstyle")
```

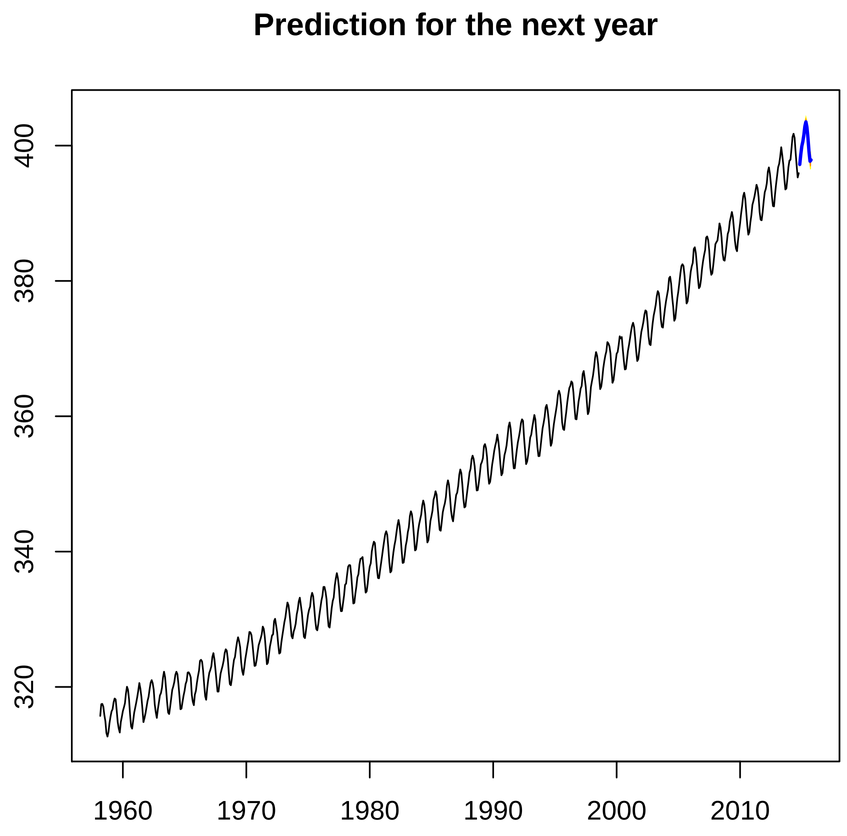


Figure 8: Forecasting using Holt Winters exponential smoothing

Here the forecasts for 1913-1920 are plotted as a blue line, the 80

The 'forecast errors' are calculated as the observed values minus the predicted values, for each time point. We can only calculate the forecast errors for the time period covered by our original time series. One measure of the accuracy of the predictive model is the sum-of-squared-errors (SSE) for the in-sample forecast errors.
To calculate a correlogram of the in-sample forecast errors for the CO2 Time series data for lags 1-20, we type:

```
> acf(hw10$residuals, lag.max=20)
```

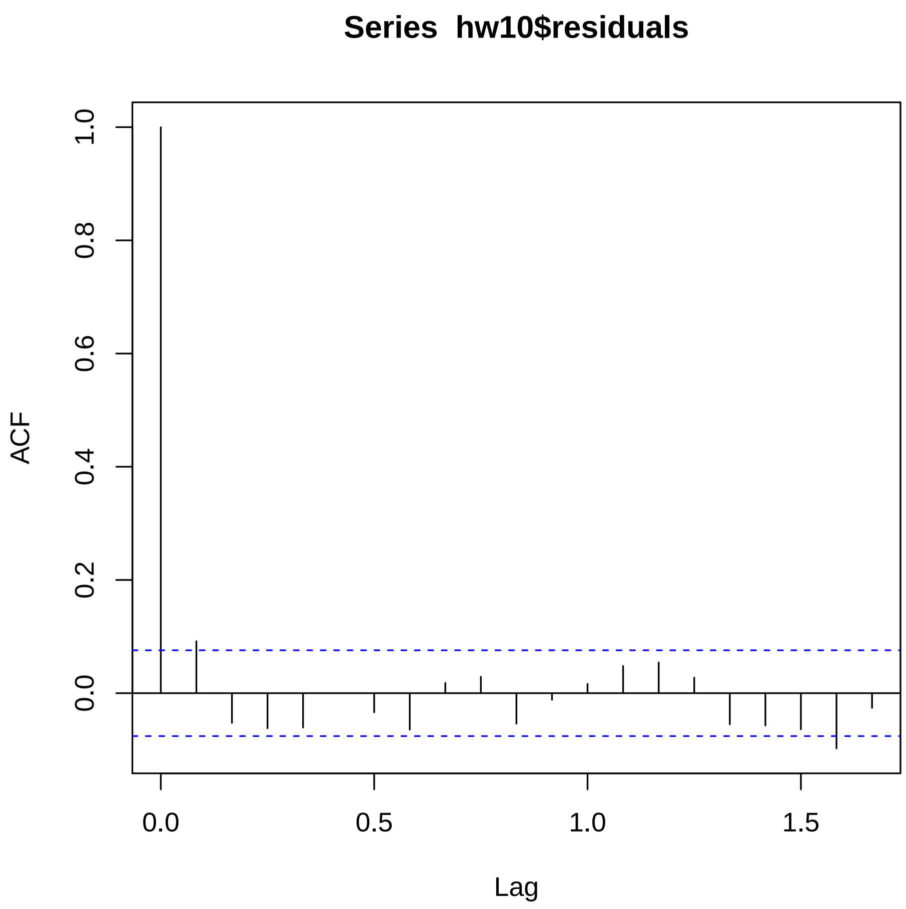


Figure 9: Correlogram of the residuals.

```
> Box.test(hw10$residuals, lag=20, type="Ljung-Box")

        Box-Ljung test

data:  hw10$residuals
X-squared = 37.5445, df = 20, p-value = 0.01006
```

There is little evidence of non-zero autocorrelations at lags 1-20.

```
> plotForecastErrors <- function(forecasterrors)
+   {
+      # make a histogram of the forecast errors:
+      mybinsize <- IQR(forecasterrors)/4
+      mysd   <- sd(forecasterrors)
+      mymin  <- min(forecasterrors) - mysd*5
+      mymax  <- max(forecasterrors) + mysd*3
+      # generate normally distributed data with mean 0 and standard deviation mysd
+      mynorm <- rnorm(10000, mean=0, sd=mysd)
+       mymin2 <- min(mynorm)
+       mymax2 <- max(mynorm)
+       if (mymin2 < mymin) { mymin <- mymin2 }
+       if (mymax2 > mymax) { mymax <- mymax2 }
+       # make a red histogram of the forecast errors, with the normally distributed data o
+       mybins <- seq(mymin, mymax, mybinsize)
+       hist(forecasterrors, col="red", freq=FALSE, breaks=mybins)
+       # freq=FALSE ensures the area under the histogram = 1
+       # generate normally distributed data with mean 0 and standard deviation mysd
+       myhist <- hist(mynorm, plot=FALSE, breaks=mybins)
+       # plot the normal curve as a blue line on top of the histogram of forecast errors:
+       points(myhist$mids, myhist$density, type="l", col="blue", lwd=2)
+    }
```

```
> plotForecastErrors(hw10$residuals)
```

Figure 10: Histogram of the errors

The histogram of the time series shows that the forecast errors are roughly normally distributed and the mean seems to be close to zero.

5.2 Seasonal Decomposition of Time Series by Loess

Forecasting using stl objects:

```
> plot(stlf(yourts, lambda=0, h =120))
> (tslm(yourts~time(yourts)))

Call:
lm(formula = formula, data = "yourts", na.action = na.exclude)

Coefficients:
 (Intercept)  time(yourts)
   -2618.494         1.494
```

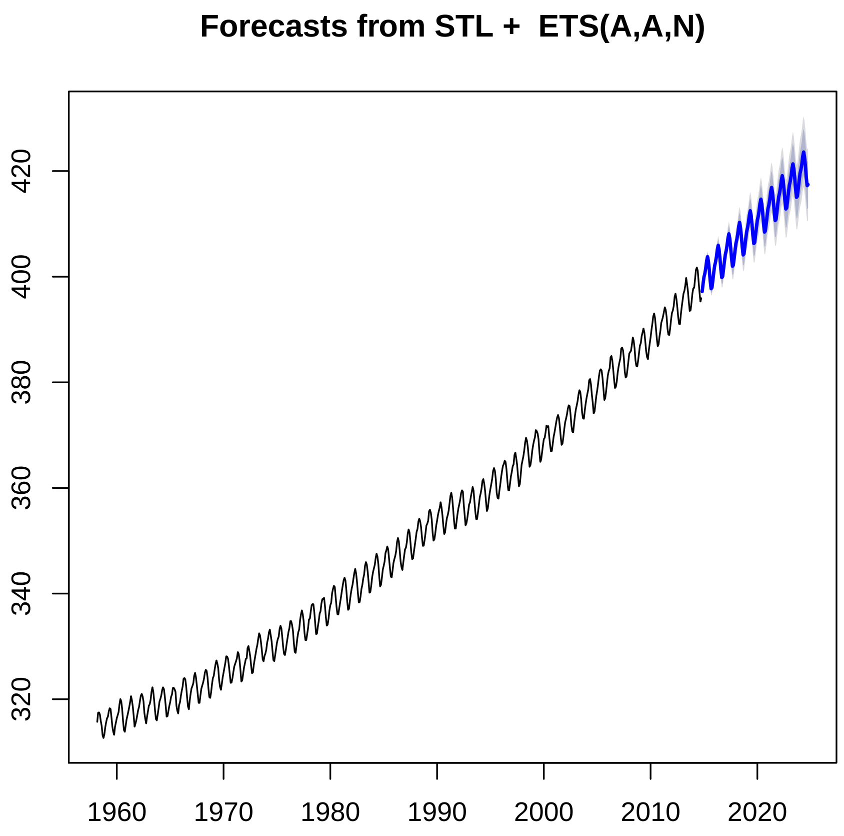
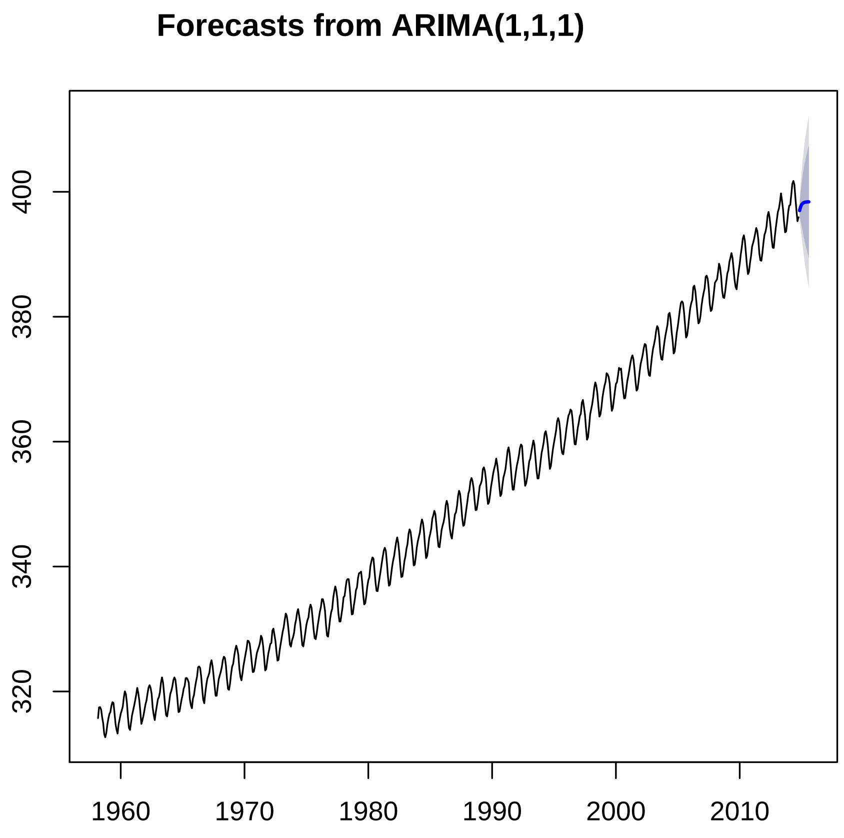


Figure 11: Forecasting using Loess

5.3 ARIMA Models

For the CO2 time series:

```
> au=auto.arima(yourts, ic = "bic")
> arima1=arima(yourts, order = c(au$arma[1],au$arma[6],au$arma[2]))
> fore1=forecast.Arima(arima1,h=10)
> plot.forecast(fore1)
>
```



We can first fit an autoregression model to the Nile Time Series:

```
> ar(Nile) # selects order 2

Call:
ar(x = Nile)

Coefficients:
     1       2
0.4081  0.1812

Order selected 2  sigma^2 estimated as  21247

> cpgram(ar(Nile)$resid)
> arima(Nile, c(2, 0, 0))

Series: Nile
ARIMA(2,0,0) with non-zero mean

Coefficients:
         ar1     ar2  intercept
      0.4096  0.1987   919.8397
s.e.  0.0974  0.0990    35.6410

sigma^2 estimated as 20291:  log likelihood=-637.98
AIC=1283.96   AICc=1284.38   BIC=1294.38

>
```

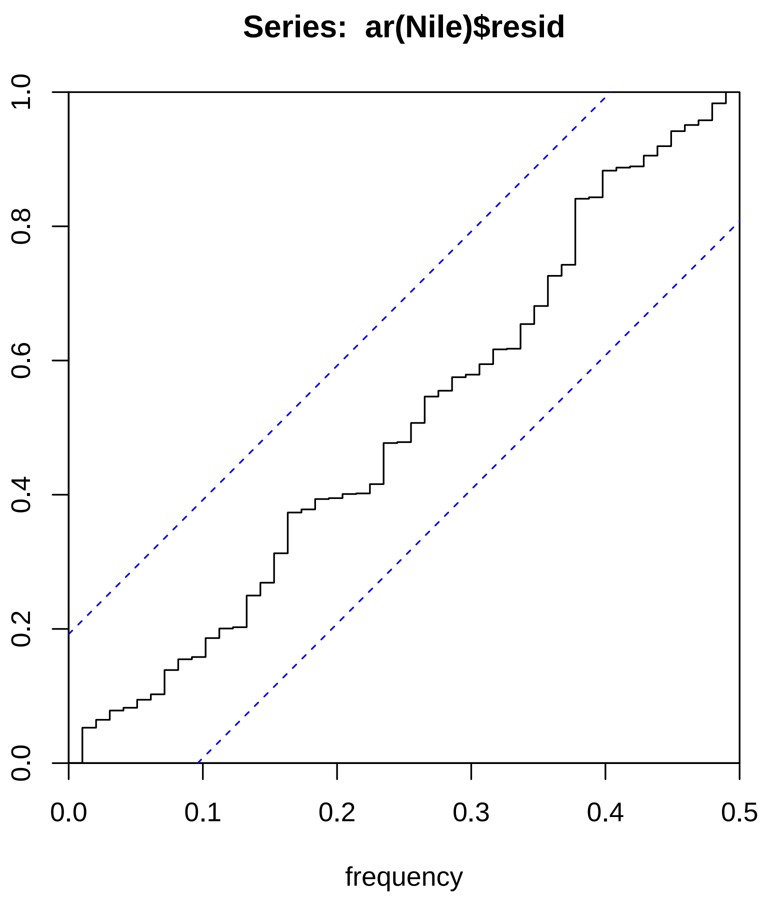


Figure 12: Cumulative Peridiogram of the residuals

Fitting a autoregressive model, we can see that the residuals are well placed.
Structural time series models:

```
> par(mfrow = c(3, 1))
> plot(Nile)
> ## local level model
> (fit <- StructTS(Nile, type = "level"))

Call:
StructTS(x = Nile, type = "level")

Variances:
  level  epsilon
   1469    15099

> lines(fitted(fit), lty = 2)              # contemporaneous smoothing
> lines(tsSmooth(fit), lty = 2, col = 4)   # fixed-interval smoothing
> plot(residuals(fit)); abline(h = 0, lty = 3)
> ## local trend model
> (fit2 <- StructTS(Nile, type = "trend")) ## constant trend fitted

Call:
StructTS(x = Nile, type = "trend")

Variances:
  level    slope  epsilon
   1427        0    15047

> pred <- predict(fit, n.ahead = 30)
> ## with 50% confidence interval
> ts.plot(Nile, pred$pred,
+         pred$pred + 0.67*pred$se, pred$pred -0.67*pred$se, col=1:3)
>
>
```

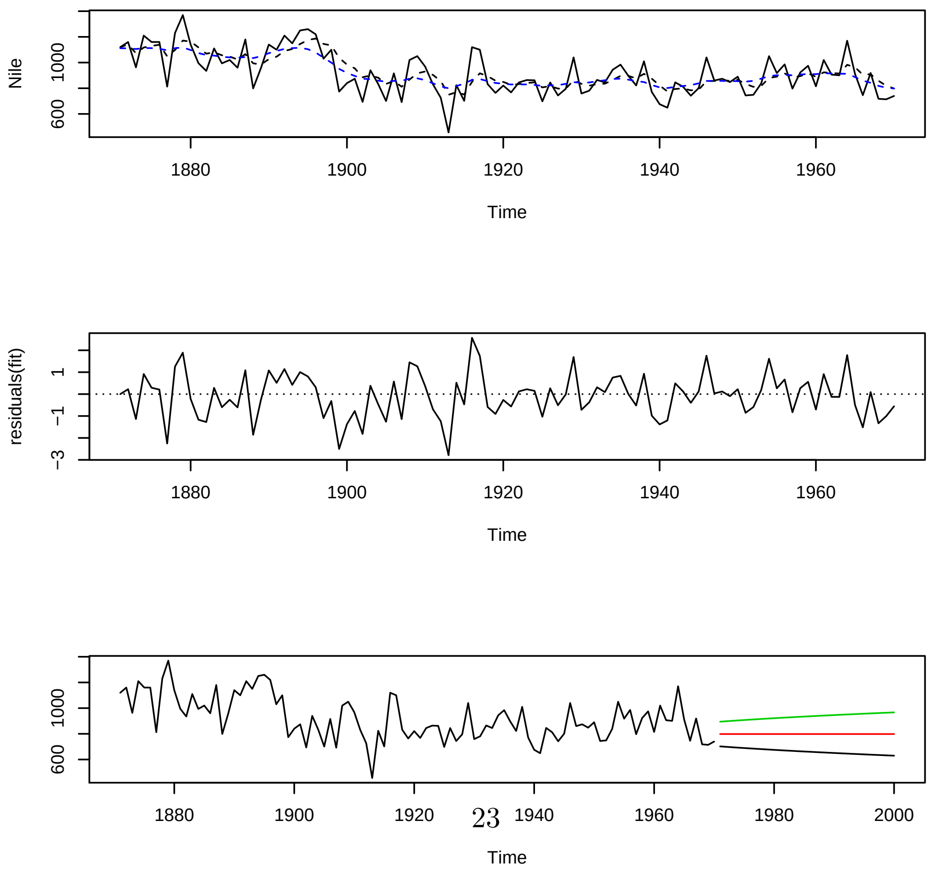


Figure 13: Fitting a Structural Model

5.4 Selecting a Candidate ARIMA Model

TEXT

5.5 Forecasting Using an ARIMA Model

compare both functions of forecasting:

```
> par(mfrow=c(1,2))
> plot(forecast.arima, xlim=c(2010,2025), ylim=c(385,430))
> plot.forecast(forecasts2 ,xlim=c(2010,2025), ylim=c(385,430))
```

Figure 14: Model Comparisons

6 Links and Further Reading

and in:

http://cran.r-project.org/web/views/TimeSeries.html

Another Exemple Datasets are avaliable at:

http://www.comp-engine.org/timeseries/browse-data-by-category
https://datamarket.com/data/list/?q=provider:tsdl

7 Acknowledgements

Don't forget to thank TeX and R and other opensource communities if you use their products! The correct way to cite R is shown when typing "`citation()`", and "`citation("mgcv")`" for packages.