Toronto Business College

Data Visualization

Assignment (4)

Time series and forecasting

|  |  |
| --- | --- |
| **Course** | 2021F-T2 AISC2001 - Data Visualization 01 |
| **Instructor** | Gamal Ali |
| **Students** |  |
|  | Charitha Priya Dongari – 500191110 |
|  | Katterapalli Venkata Satya Narayana Reddy – 500190693 |
|  | Rumana Banu Iliyas Ahmed – 500186725 |
|  | Rohan Bhatt – 500187633 |
|  | Shivam Jolly – 500196452 |

Table of Contents

[Introduction 2](#_Toc88084061)

[Reading and loading data 2](#_Toc88084062)

[Convert to Time Series Object 3](#_Toc88084063)

[Plotting NY Births Time Series 4](#_Toc88084064)

[Exploratory Data Analysis 5](#_Toc88084065)

[Exploring the Trend 5](#_Toc88084066)

[Detect Empty Nodes 6](#_Toc88084067)

[Boxplot to examine seasonal effects across months 7](#_Toc88084068)

[Decomposing Seasonal Data 8](#_Toc88084069)

[Seasonal Adjusting 11](#_Toc88084070)

[Exponential Smoothening 12](#_Toc88084071)

[Predictions for 240 months to show the exponentially growing confidence interval 15](#_Toc88084072)

[References 16](#_Toc88084073)

# Introduction

A time series is a collection of data points that are arranged in chronological order w.r.t. time. Time series analysis is a technique for analysing previous data over a set period of time in order to forecast the future.

Some of the advantages of Time Series Analysis are below,

1. **Identifying Pattern**: The pattern of the data may be plainly seen by plotting the time series data, whether it is moving higher or decreasing every year or follows a regular pattern.
2. **Cleaning the Data**: The missing data can be easily identified by looking for a gap in the plotted time series graph, which can then be used to impute the missing values.
3. **Predicting the Future**: Time series can aid in the prediction of the future, allowing us to make more educated decisions.

Using the R programming language, we will perform time-series analysis and forecasting using the dataset from http://robjhyndman.com/tsdldata/data/nybirths.dat. From January 1946 through December 1959, this dataset contains data on the number of births per month in New York City.

# Reading and loading data

***Code:***

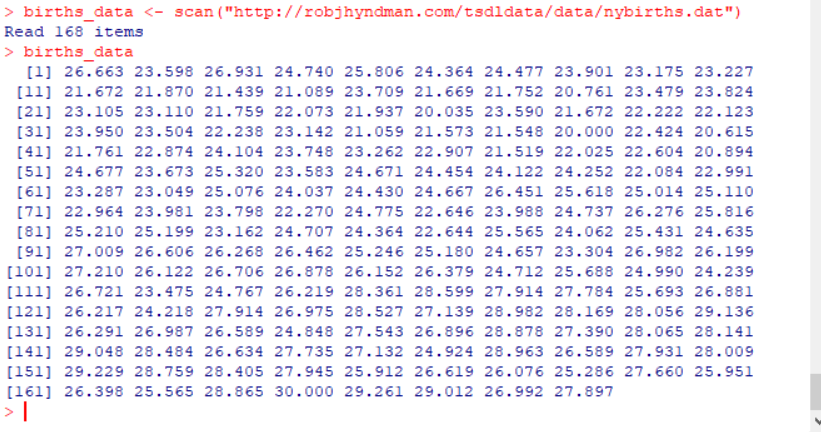
> library(forecast)

> births\_data <- scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")

> births\_data

* **Forecast library**: It is R package that includes methods and tools for visualising and analysing univariate time series forecasts, such as exponential smoothing using state space models and automatic ARIMA modelling.
* First include the forecast library that is needed later to predict the future data.
* Load the NY City births data using the **scan**() function of R. This function reads the data into a vector or list.
* Below is the output of the data.

***Output:***



# Convert to Time Series Object

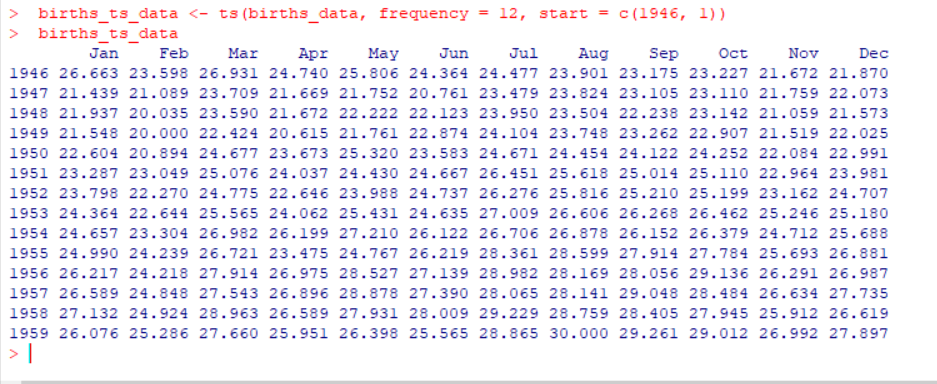
* A time series object is created with ts().
* This function creates a R time series object from a numeric vector.
* Format of this function is below,  
   **ts(data, start, end, frequency)**  
   where,  
   ***data***: Vector or matrix containing the time series values  
   ***start***: start time for the first observation in time series  
   ***end***: end time for the last pbservation in time series  
   ***frequency***: specifies the number of observations per unit time

***Code:***

> births\_ts\_data <- ts(births\_data, frequency = 12, start = c(1946, 1))

> births\_ts\_data

***Output:***



In the above output we can see the time series data of the births in the NY city for the months across the years.

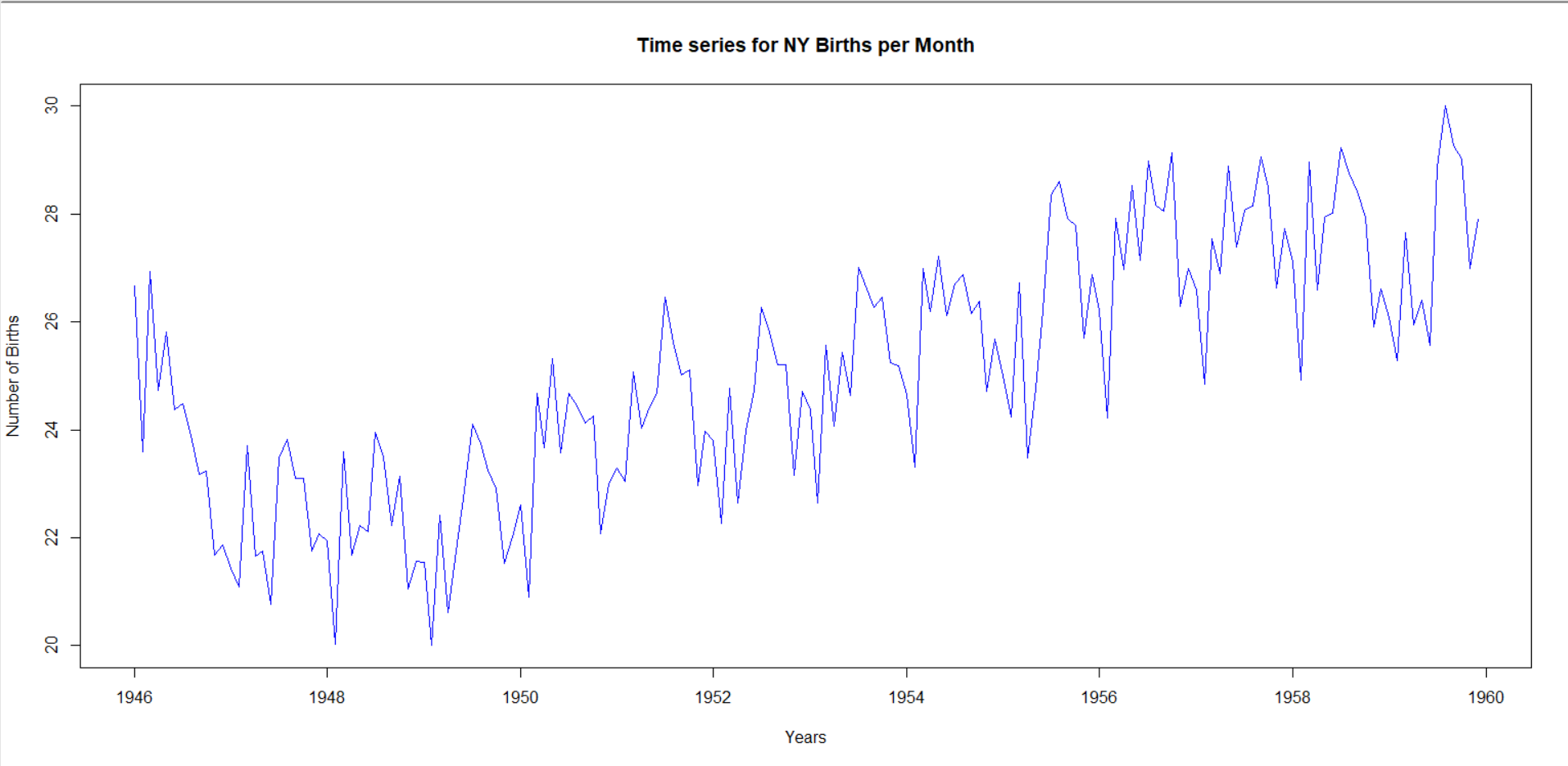
# Plotting NY Births Time Series

* The next step is to plot the time series data using R's plot() function.
* Format of plot() depends based on the input. The format for time series data input is as follows,  
   **plot(timeSeries, xlab,ylab,main,sub,type,asp)**  
   where,  
   ***timeSeries***: time series object  
   ***xlab***: title for x-axis  
   ***ylab***: title for y-axis  
   ***main***: title for the plot  
   ***sub***: sub-title for the plot  
   ***type***: type of plot like ‘l’ for lines, ‘p’ for points, ’h’ for histogram, etc.  
   ***asp***: aspect ratio

***Code:***

> plot(births\_ts\_data, xlab= "Years", ylab = "Number of Births", main="Time series for NY Births per Month", col="blue")

***Output:***



* Looking at the above time series, the seasonal fluctuations appear to be fairly constant in size over time suggesting that this could be described using an additive model
* The number of births each month exhibits some seasonal variation in this time series, with a maximum in the summer and a minimum in the winter.

# Exploratory Data Analysis

* It is used to understand a dataset in order to prepare for further phases of data analysis or to answer to a data-related query.
* It can aid in the detection of evident errors, as well as a better understanding of data patterns and the detection of outliers

## Exploring the Trend

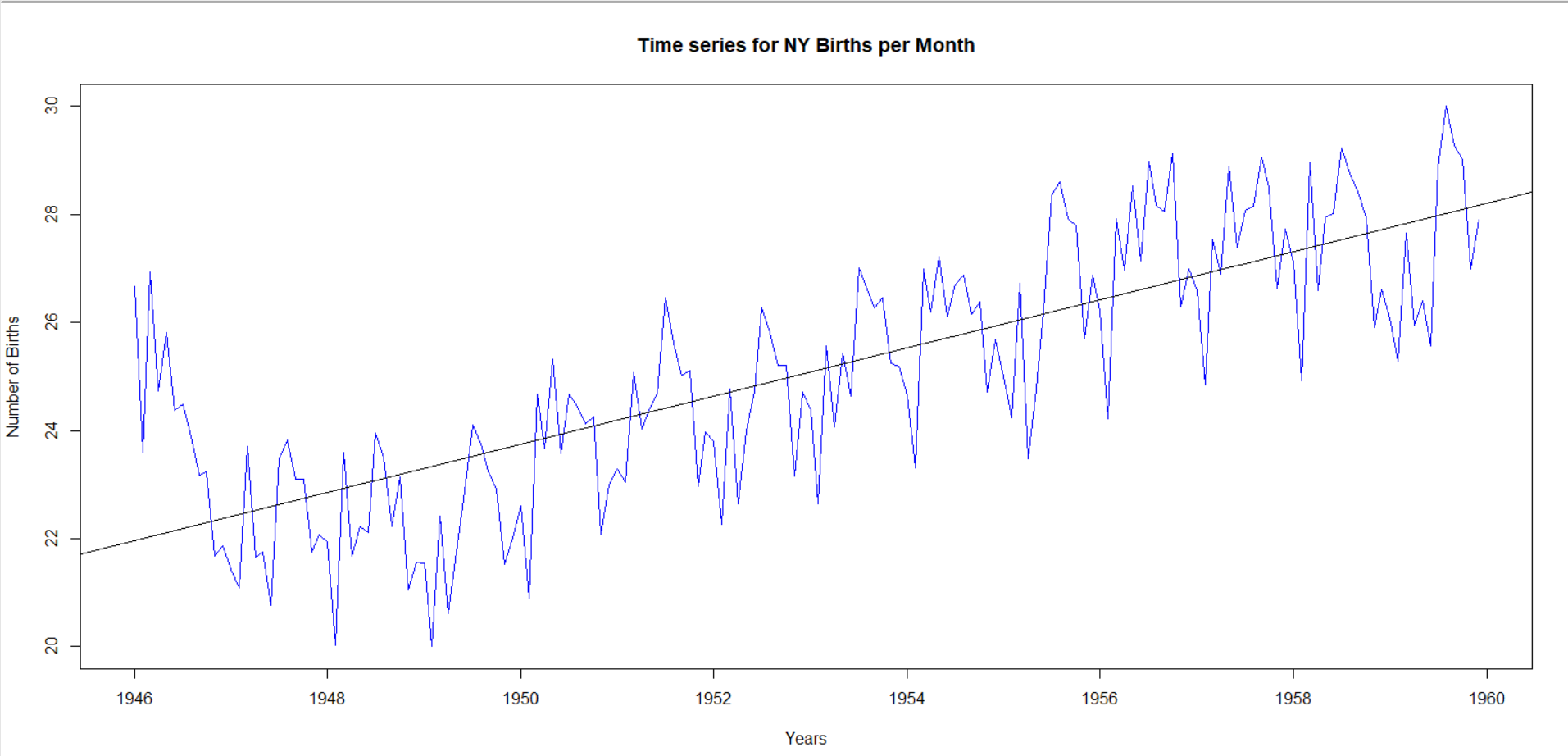
* The linear model can be used to investigate the trend.
* Next, we'll check the trend by plotting the best fit line on the linear model.
* The lm() function will be used to create a linear model, and the albline() function will be used to depict the best fit line.  
  Format for albline(),  
   **abline(a = NULL, b = NULL, h = NULL, v = NULL,…)**  
   where,  
    ***a***: intercept  
   ***b***: slope  
   ***h***: y-value of horizontal line  
   ***v***: x-values of vertical line

***Code:***

> plot(births\_ts\_data, xlab= "Years", ylab = "Number of Births", main="Time series for NY Births per Month", col="blue")

> abline(reg=lm(births\_ts\_data~time(births\_ts\_data)))

***Output:***



From the above output we can infer that,

* We can notice that the number of births increases in a linear fashion as the year progresses.
* After 1946, the number of births declined in 1947, and the trend continued until 1950.
* After 1950, the number of births began to slowly rise.

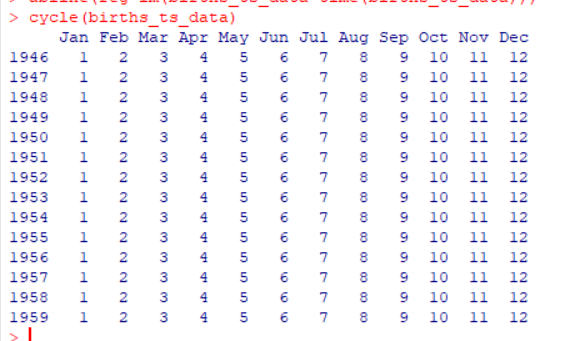
## Detect Empty Nodes

* Detecting empty nodes is one of the steps in the EDA process.
* When the data is large, manually checking for empty nodes becomes difficult.
* To detect the empty nodes in a time series data we use the **cycle()** function.
* The cycle() gives the positions in the cycle of each observation.  
  Format of cycle,  
   **cycle(x)** where,  
    ***x***: a univariate or multivariate time-series, or a vector or matrix.

***Code:***

> cycle(births\_ts\_data)

***Output:***



From the above output we can infer that there are no empty nodes.

## Boxplot to examine seasonal effects across months

We observed that the NY births data exhibits seasonal variation when plotting it, so let's use Boxplot to look at seasonal effects across months.

Format for boxplot,

**boxplot(formula, data = NULL, xlab, ylab, main, col, border, width,outline,…**)

where,

***formula***: a formula, such as x ~ group

***data***: list or data frame from which the variables in formula are taken

***xlab***: title for x-axis  
 ***ylab***: title for y-axis  
 ***main***: title for the plot  
 ***col***: color for the bodoes of the boxplot

***border***: color for the outlines of the boxplots

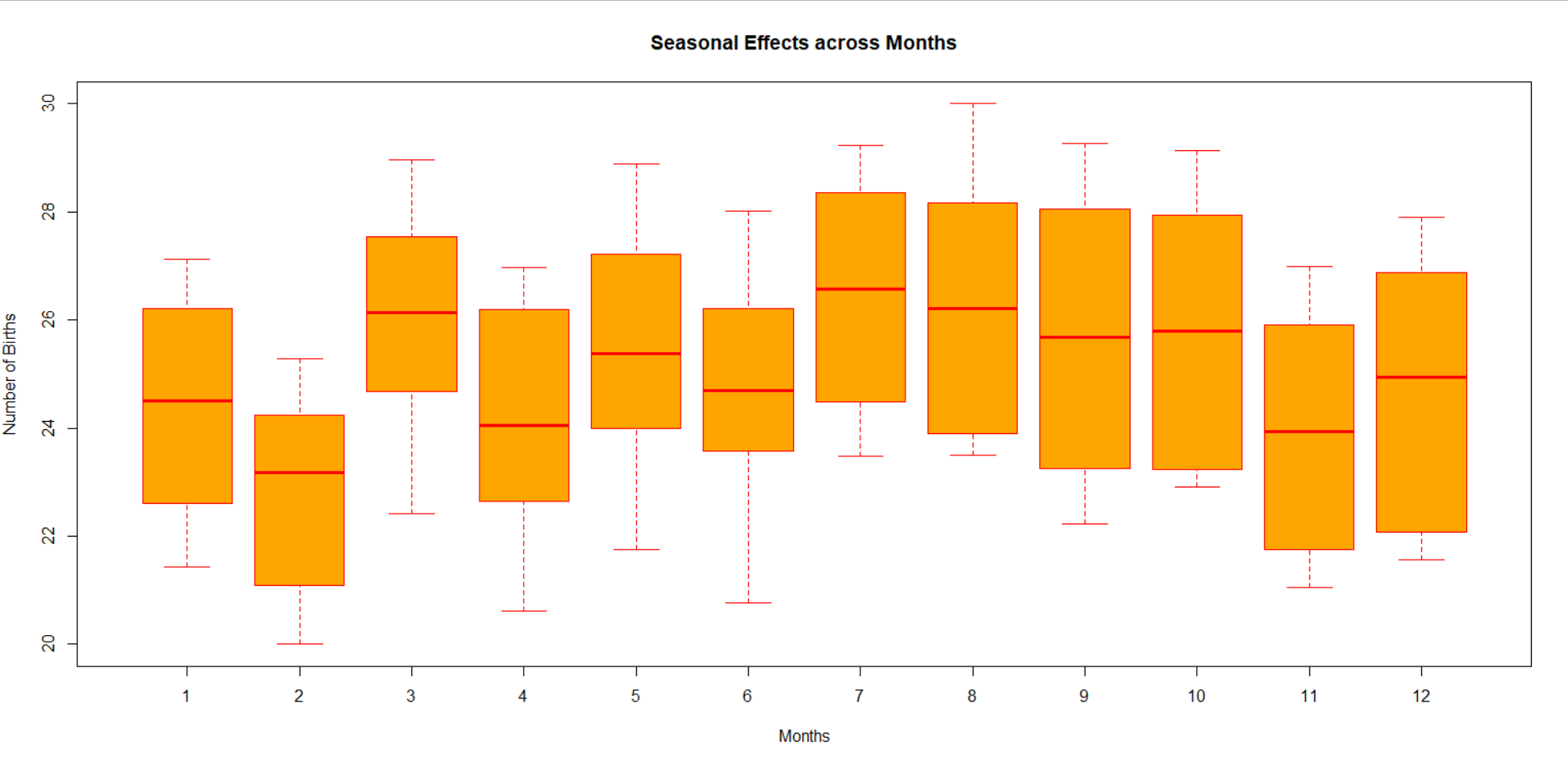
***width***: width of the boxes of the boxplot

***outline***: If this is not true then outliers are not drawn

***Code:***

> boxplot(births\_ts\_data~cycle(births\_ts\_data),xlab="Months",ylab="Number of Births", main="Seasonal Effects across Months",col="orange",border="red")

***Output:***



From the above boxplot we can conclude that this time series shows that there is some seasonal variation in the number of births per month, with a maximum in the summer and a minimum in the winter.

# Decomposing Seasonal Data

* Decomposing a time series involves breaking it down into the three components; i.e., trend, seasonal and irregular components.
* We'll now use an additive model to decompose the time series

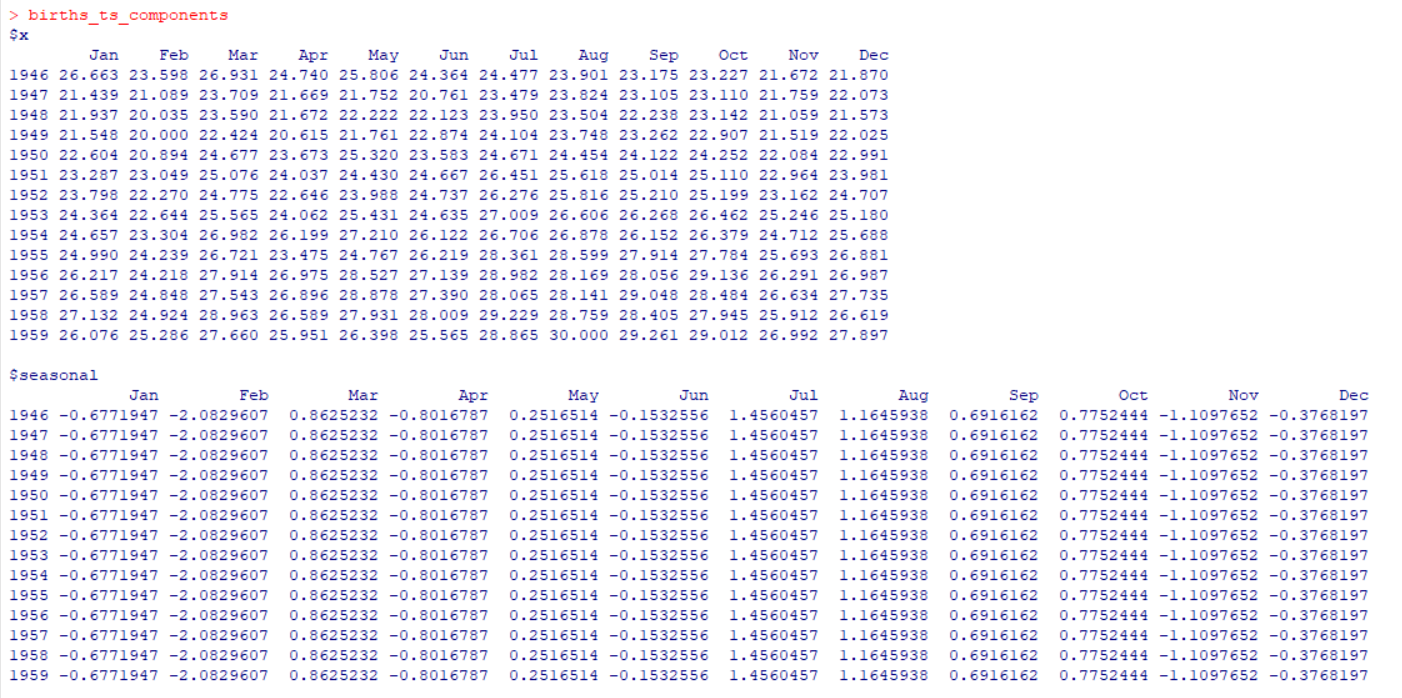
The "**decompose**()" function in R can be used to estimate the trend component and seasonal component of a seasonal time series with an additive model.  
Format for decompose(),  
 **decompose(x, type = c("additive", "multiplicative"), filter = NULL)**  
 where,  
 ***x***: time series  
 ***type***: seasonal component type  
 ***filter***: vector of filter co-efficients.

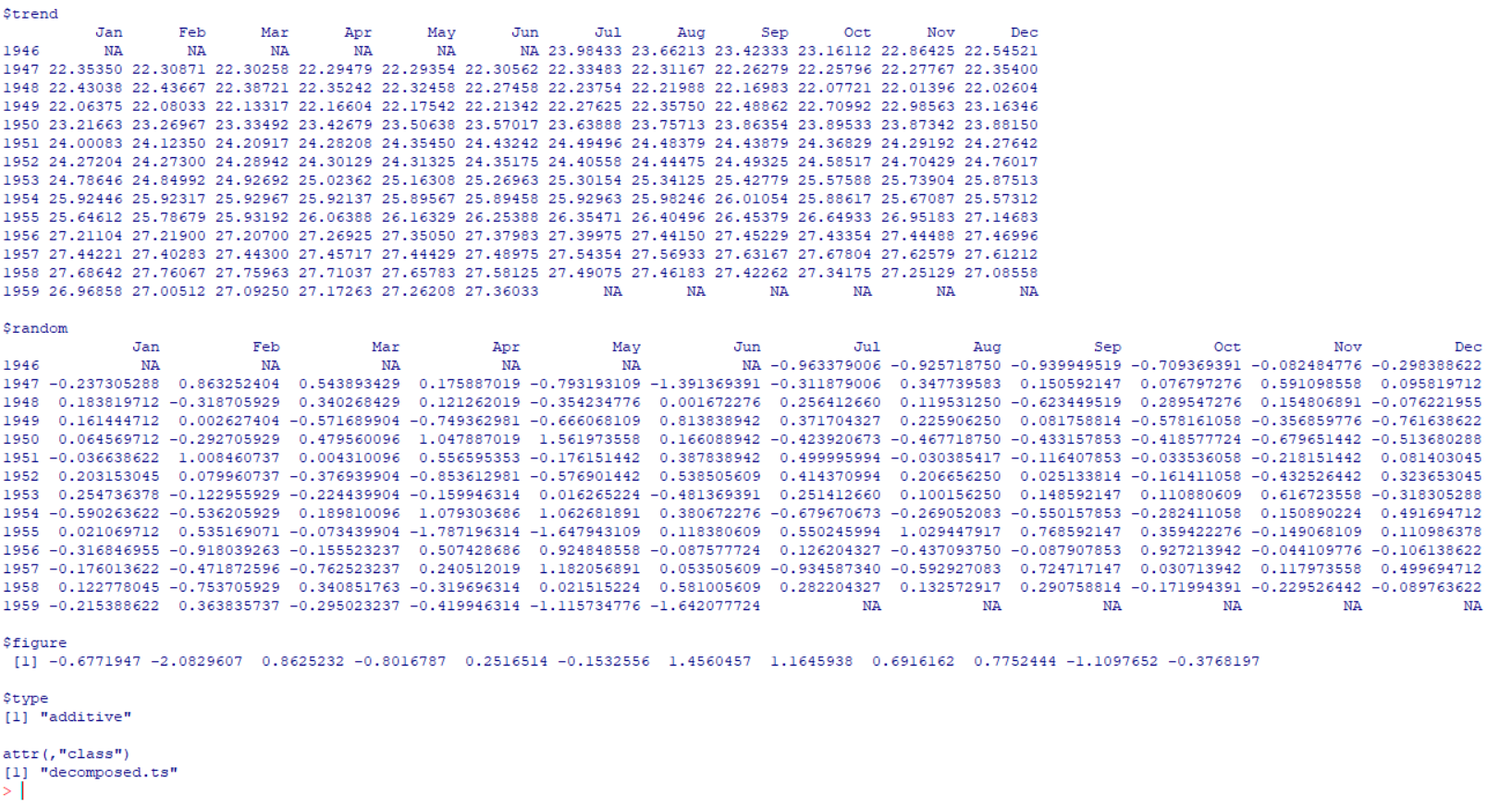
***Code:***

> births\_ts\_components <- decompose(births\_ts\_data)

> births\_ts\_components

***Output:***





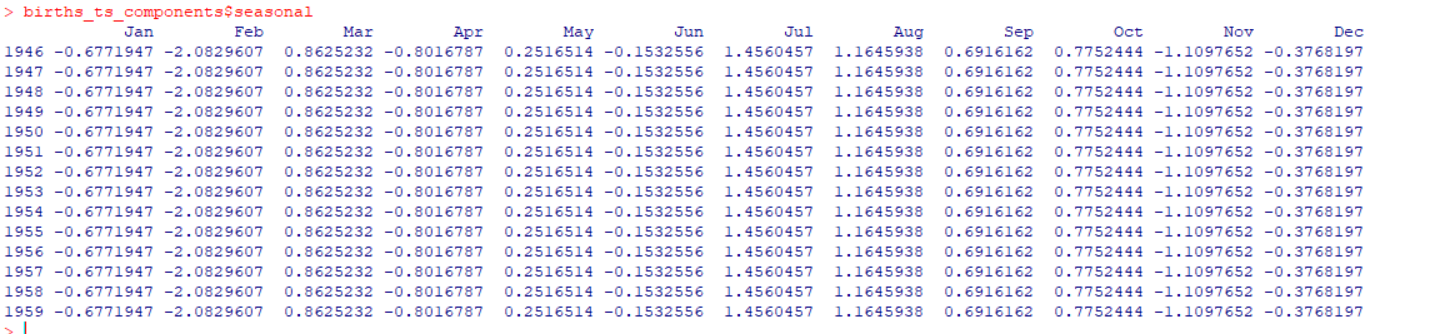
Seasonal, trend, and irregular component estimated values are now stored in variables - births\_ts\_components$seasonal,births\_ts\_components$trend and births\_ts\_components$random.

Let's look at the seasonal component's estimated values.

***Code:***

> births\_ts\_components$seasonal

***Output:***

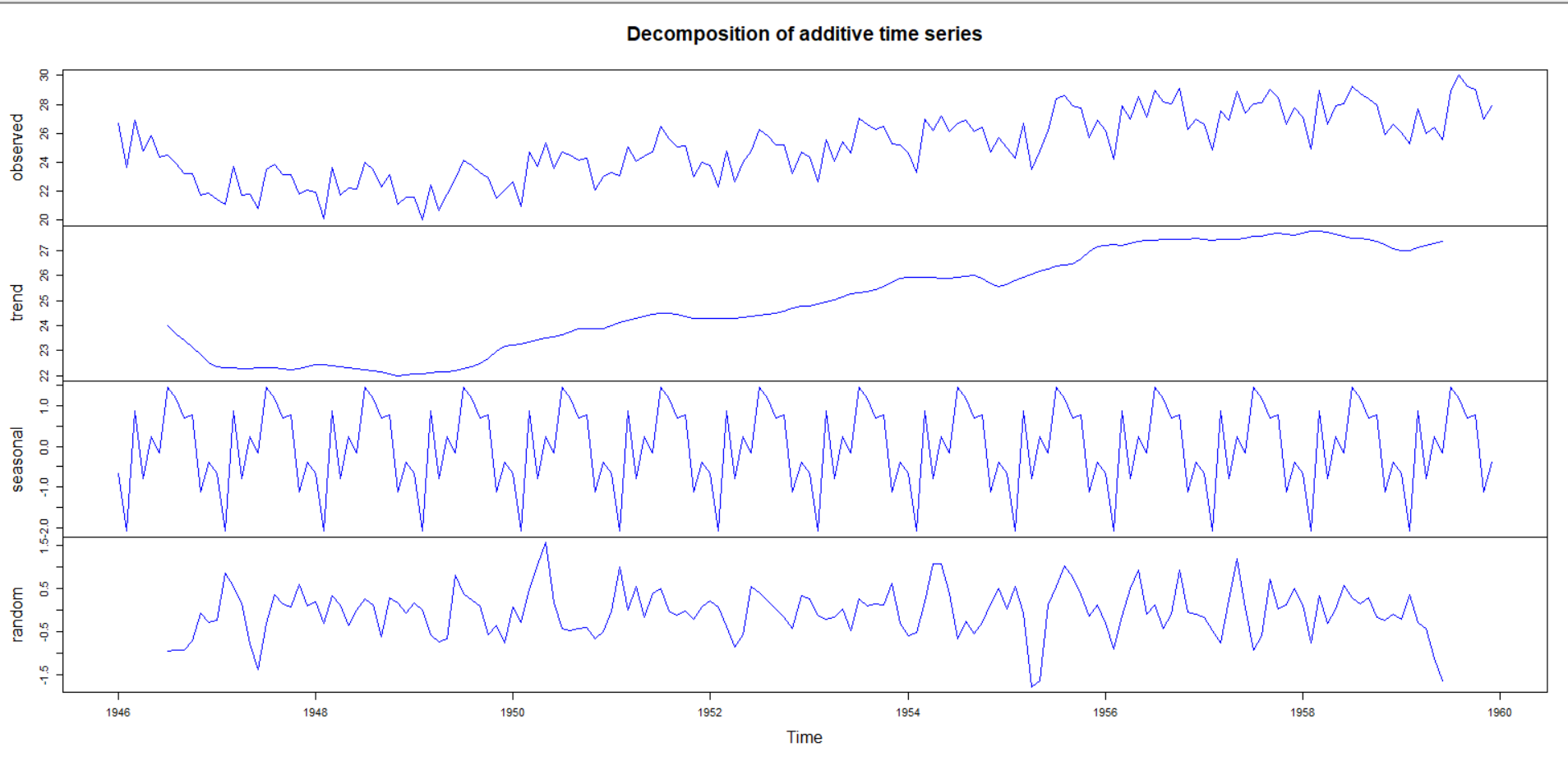


* As seen in the above output, the seasonal component for all of the months is the same.
* We can determine from the above output that the highest seasonal factor is for July about 1.46, and the lowest is for February about -2.08, implying that each year there appears to be a spike in births in July and a fall in births in February.
* Using the plot() function to plot the estimated trend, seasonal, and irregular components of the time series,

***Code:***

> plot(births\_ts\_components,col="blue")

***Output:***



* The original time series is at the top of the plot, followed by the estimated trend component, then the estimated seasonal component, and finally the estimated irregular component.
* The estimated trend component shows a minor dip from approximately 24 in 1947 to approximately 22 in 1948, followed by a steady climb from there on to approximately 27 in 1959.

# Seasonal Adjusting

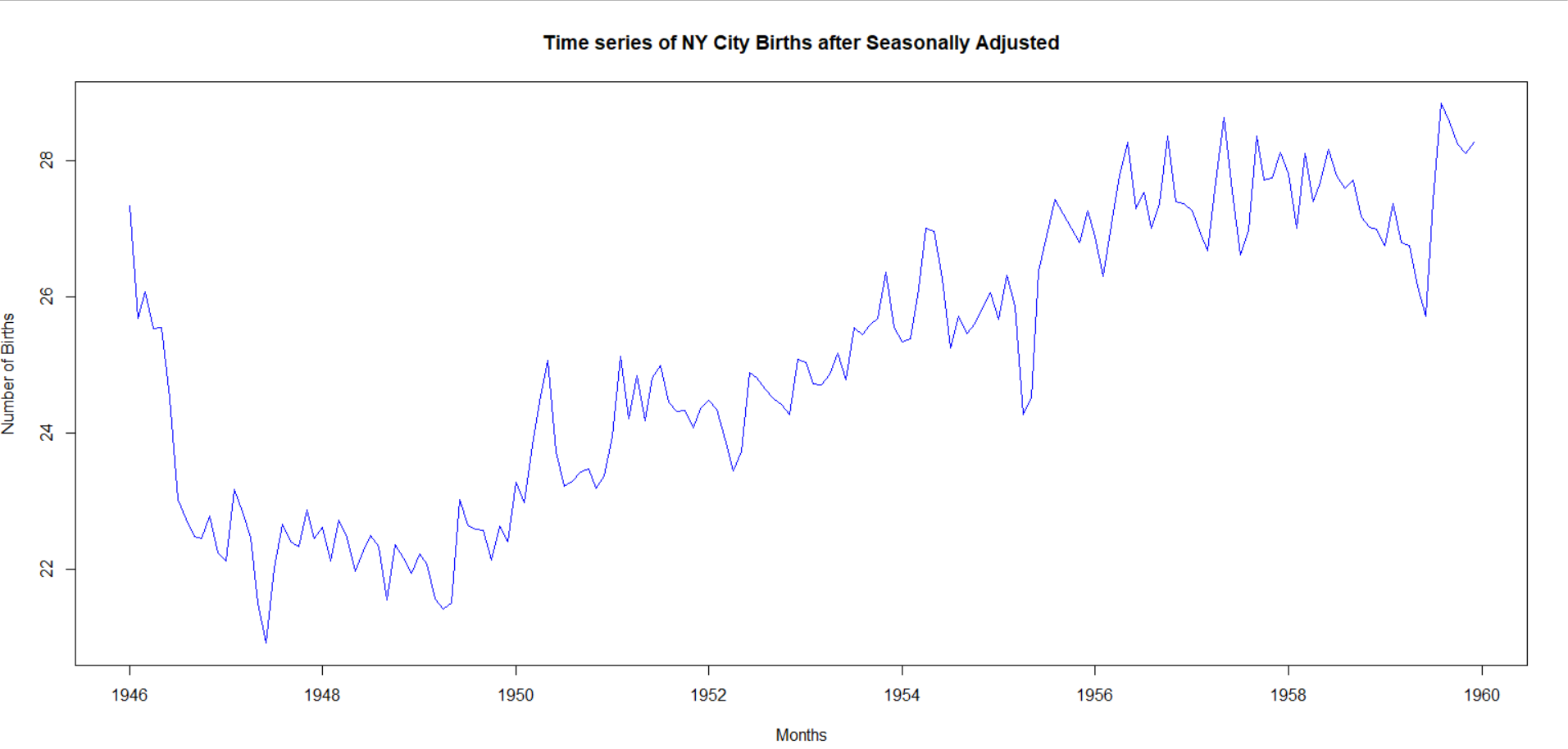
* A monthly or quarterly time series that has been adjusted to remove the effects of seasonal and calendar variables is known as a seasonally adjusted time series.
* By calculating the seasonal component and subtracting the estimated seasonal component from the original time series, we can seasonally adjust the time series.
* We can use "decompose()" to estimate the seasonal component of the number of births each month in New York City, and then subtract the seasonal component from the original time series to seasonally adjust it

***Code:***

> births\_ts\_components <- decompose(births\_ts\_data)

> births\_ts\_components\_seasonallyadjusted <- births\_ts\_data - births\_ts\_components$seasonal

> plot(births\_ts\_components\_seasonallyadjusted,xlab="Months", ylab="Number of Births",main ="Time series of NY City Births after Seasonally Adjusted", col = "blue")

***Output:***  


We can see that the seasonal fluctuation has been removed, leaving only the trend component and an irregular component in the seasonally adjusted time series.

# Exponential Smoothening

* When time series data is exponentially smoothed, the weights for the newest to oldest observations are assigned in an exponentially decreasing order
* Holt-Winters exponential smoothing calculates the current time point's level, slope, and seasonal component.
* All of these factors are between 0 and 1, and values close to 0 indicate that the most recent observations are given less weight when forecasting future values.
* Format for HoltWinters() function,  
   **HoltWinters(x, alpha = NULL, beta = NULL, gamma = NULL,  
   seasonal = c("additive", "multiplicative"),**

**start.periods = 2, l.start = NULL, b.start = NULL,**

**s.start = NULL,**

**optim.start = c(alpha = 0.3, beta = 0.1, gamma = 0.1),**

**optim.control = list())**  
 Where,  
 ***x***: time series object  
 ***alpha***: parameter for Holt-Winters filter  
 ***beta***: if set to FALSE, function will perform exponential smoothening  
 ***seasonal***: to indicate the seasonal model, values to be either additive or multiplicative.  
 ***start.periods***: used in auto-deduction of start values  
 ***l.start***: start value for level  
 ***b.start***: start value for trend

***s.start***: start values for seasonal component  
 ***optim.start***: Vector with named components  
 ***optim***.***control***: Optional additional control parameters  
  
***Code:***

> births\_ES <- HoltWinters(births\_ts\_data,beta=FALSE, seasonal="additive") # We set beta to perform exponential smoothening

> births\_ES

***Output:***  
  
  
  
From the above output,

* the value of alpha (0.45) is low, indicating that the level estimate at the current time point is based on both recent observations and some observations from the distant past.
* The value of gamma (0.52), on the other hand, is relatively high, indicating that the seasonal component estimate at this time point is only based on very recent observations.

**Forecast Errors**

Check the forecast errors for the time period covered by our original time series as follows,

***Code:***

> births\_ES $SSE

***Output:***



The forecast errors have a sum-of-squared-errors of 86.63.

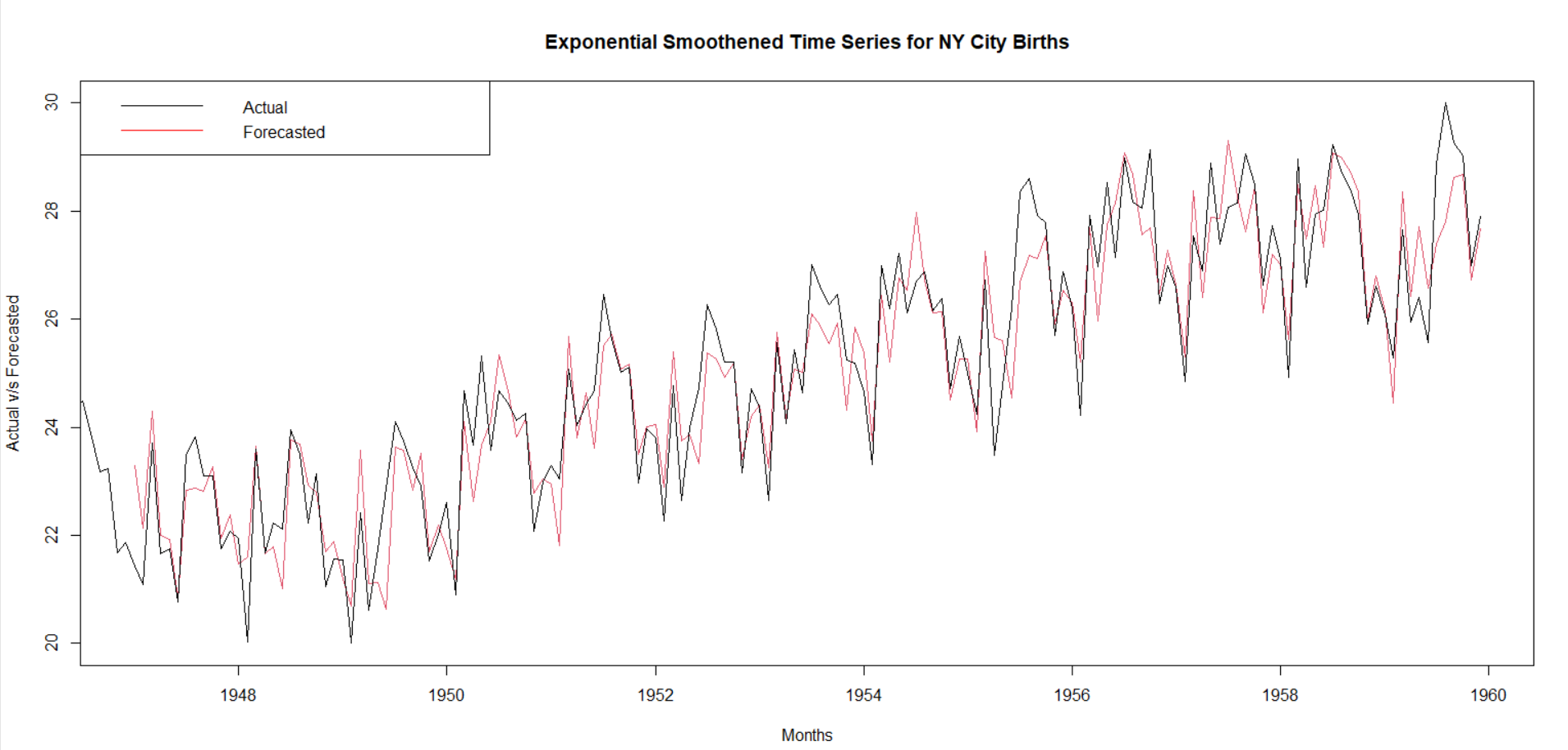
We plot the original time series as a black line, with the forecasted values as a red line on top, using the plot() function.

***Code:***

> plot(births\_ES, xlab="Months", ylab = "Actual v/s Forecasted", main = "Exponential Smoothened Time Series for NY City Births")

> legend("topleft", c("Actual", "Forecasted"), lty=c(1, 1),col=c("black", "red"))

***Output:***



The forecasts, as seen in the graph, correlate rather well with the observed values, though they do tend to lag behind the observed values slightly.

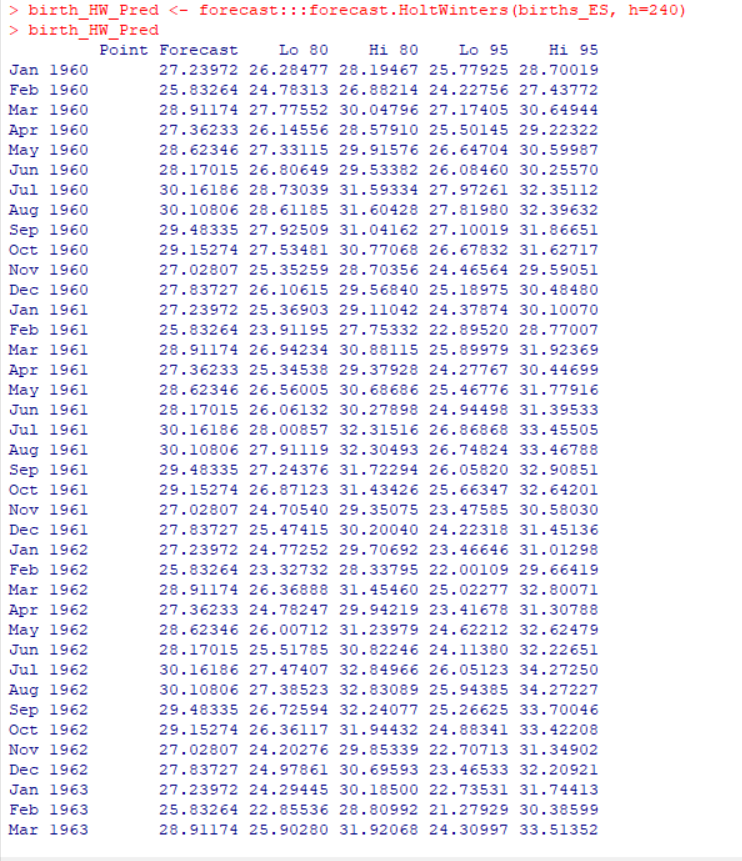
# Predictions for 240 months to show the exponentially growing confidence interval

* We utilise the "**forecast.HoltWinters()**" method in the "forecast" package to produce forecasts for future times not included in the original time series.
* The data used here is from from January 1946 to December 1959, we now predict the data 240 months from January 1960.

***Code:***

> birth\_HW\_Pred <- forecast:::forecast.HoltWinters(births\_ES, h=240)

> birth\_HW\_Pred

***Output:***

* From the above we can see that the forecast.Holtwinters() returned the Forecast forecast for a month along wit 80% and 95% prediction level.
* The births forecasted for January 1960 is 27.24 with 95% prediction interval of 25.78 and 28.7.

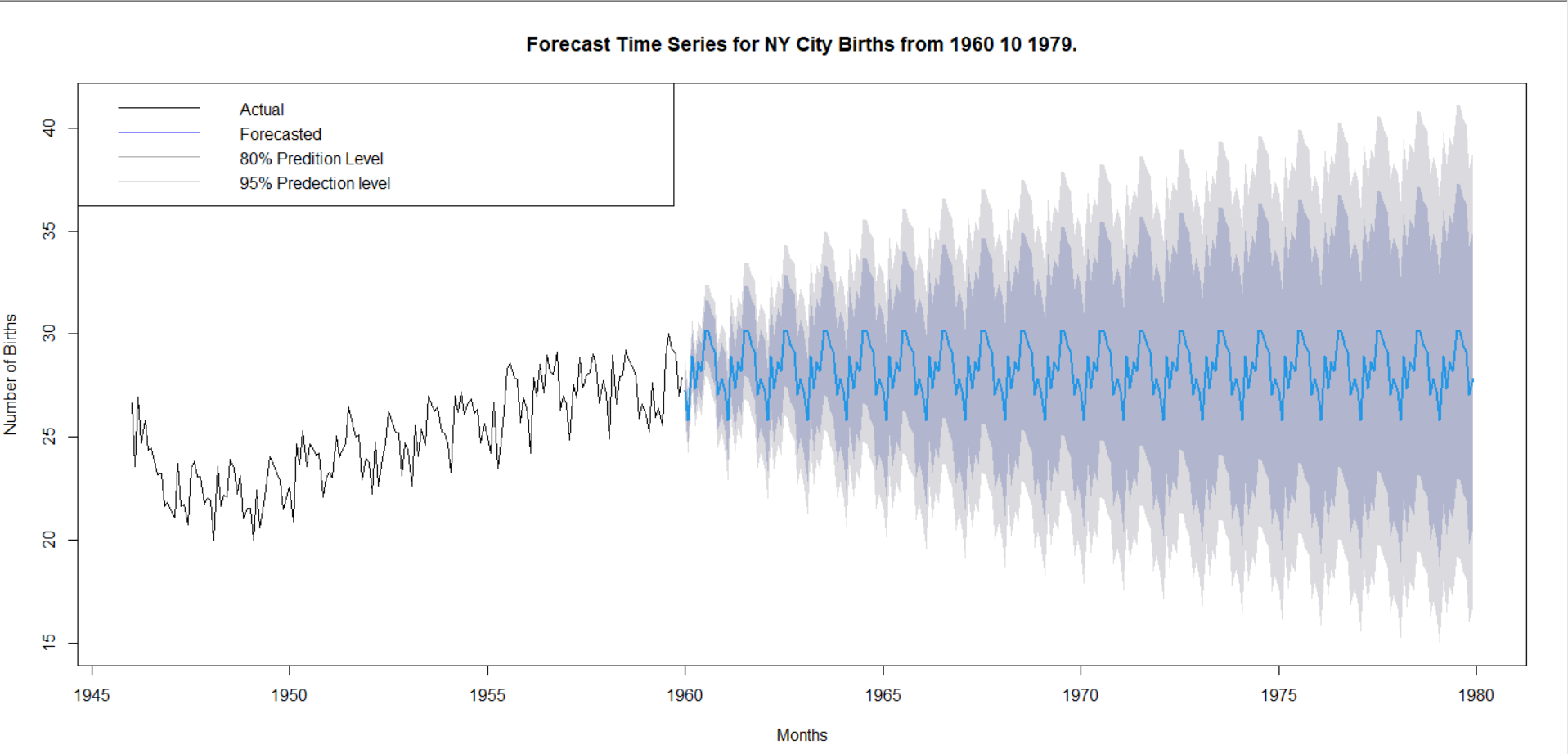
Plotting the forecast,

***Code:***

> plot(birth\_HW\_Pred,xlab="Months",ylab="Number of Births",main="Forecast Time Series for NY City Births from 1960 10 1979.")

> legend("topleft",c("Actual","Forecasted","80% Predition Level","95% Predection level"),lty=c(1,1),col=c("black","blue","dark gray","light gray"))

***Output:***



Forecasts for 1960-1979 are shown as a blue line, with the 80 percent prediction interval coloured in light grey and the 95 percent prediction interval shaded in dark grey.

# References

* <https://www.rdocumentation.org>
* Course Material