Week 1 - Importing, Simulating and Plotting Data

Jonathan Thong

2023-02-21

Importing Data From A .csv File

Let's suppose that we were interested in exploring some real GDP data from the United States. We can obtain the data from the FRED database at the following link https://fred.stlouisfed.org/series/GDPC1. Downloading the data in .csv format as *GDPC.csv* into our specified working directory, we can import it into our R environment and store it in a data frame called **gdpdata** using the following line of code:

```
gdpdata <- read.csv("GDPC1.csv")
```

We can have a look at the entirety our data by clicking on the spreadsheet icon in the R Studio Environment tab or alternatively, we can print the first few rows of our data frame using the following command:

```
head(gdpdata)
```

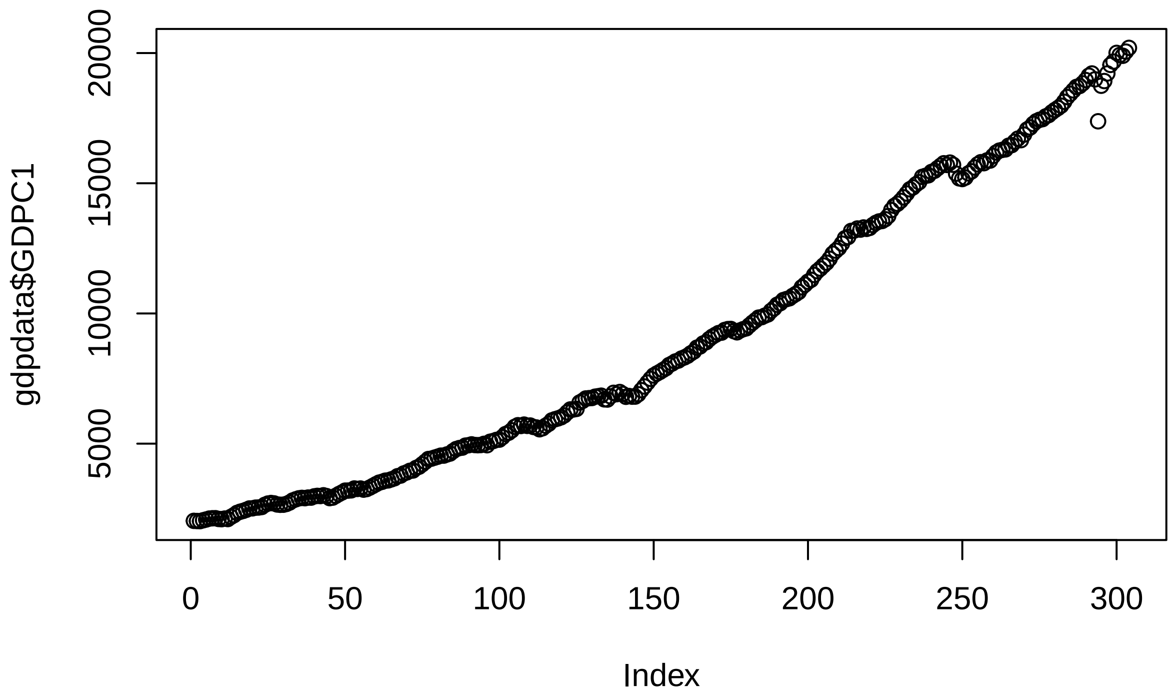
```
##         DATE    GDPC1
## 1 1947-01-01 2034.450
## 2 1947-04-01 2029.024
## 3 1947-07-01 2024.834
## 4 1947-10-01 2056.508
## 5 1948-01-01 2087.442
## 6 1948-04-01 2121.899
```

Note that the data is observed at a quarterly frequency.

Generating A Simple Time Series Plot

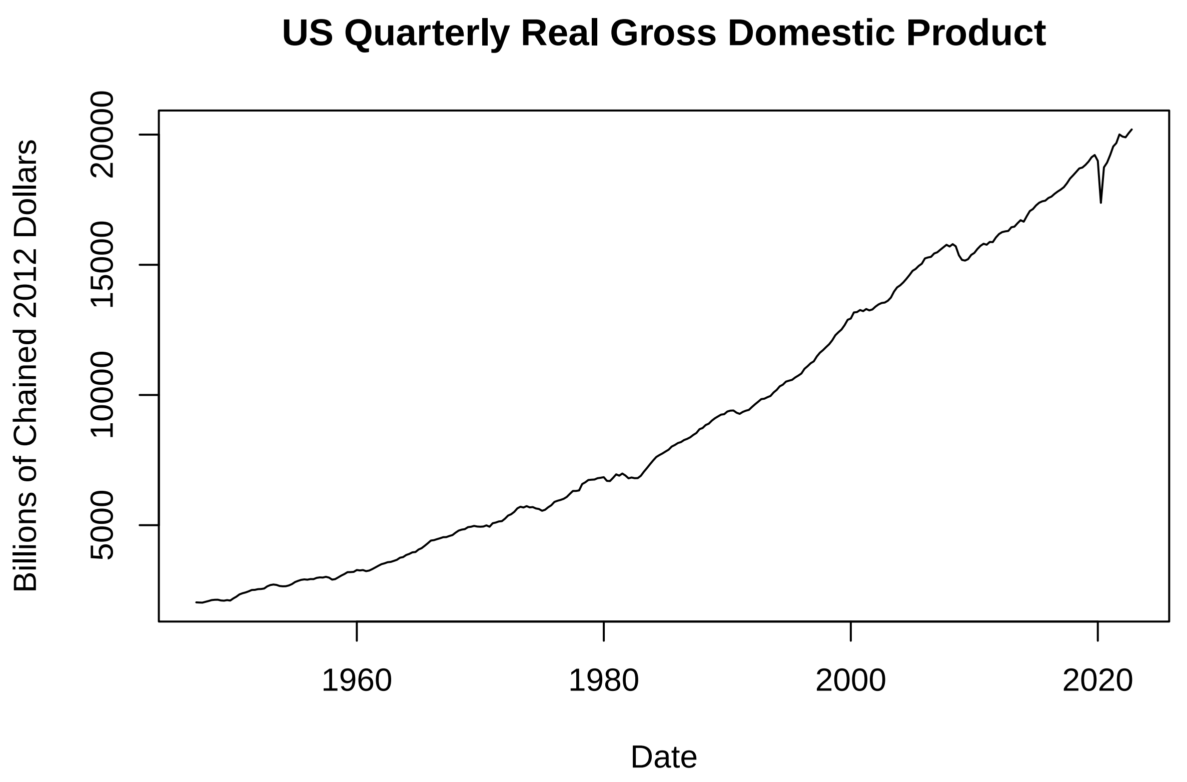
Let's now proceed to generate a plot of the data. Since the observations are contained in the column named **GDPC1**, we can simply use the following function to generate a simple plot of the data:

```
plot(gdpdata$GDPC1)
```



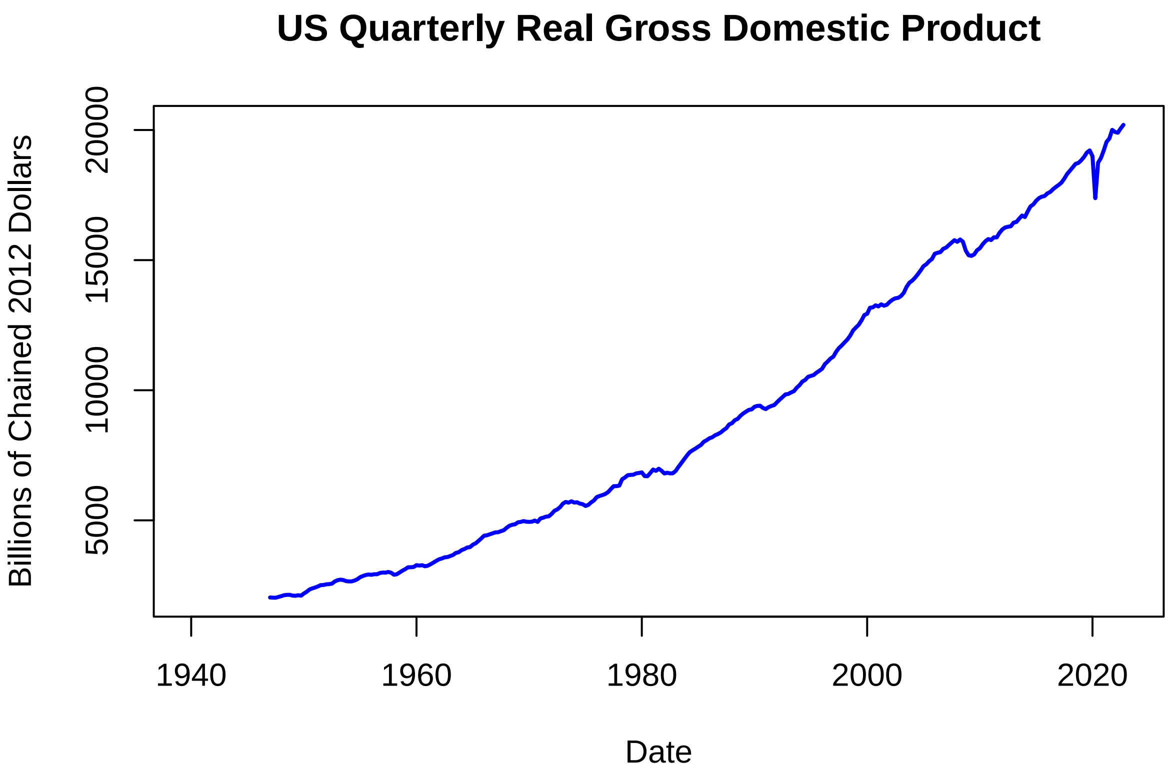
Notice however that this is not a very good visual representation of the data, the chart has no title, the axes are not very informative and the data points are not depicted in a manner typically appropriate for time series data. We can fix this by adding a few more details into our **plot** command:

```
date <- as.Date(gdpdata$DATE)
plot(date, gdpdata$GDPC1,
     main = "US Quarterly Real Gross Domestic Product",
     xlab = "Date",
     ylab = "Billions of Chained 2012 Dollars",
     type = "l")
```



We can add some further details such as changing the range of the horizontal axis, drawing a slightly thicker line width and different line colour:

```
date <- as.Date(gdpdata$DATE)
plot(date, gdpdata$GDPC1,
     main = "US Quarterly Real Gross Domestic Product",
     xlab = "Date",
     xlim = as.Date(c("1940-01-01","2023-01-01")),
     ylab = "Billions of Chained 2012 Dollars",
     type = "l",
     lwd = 2.0,
     col = "blue")
```

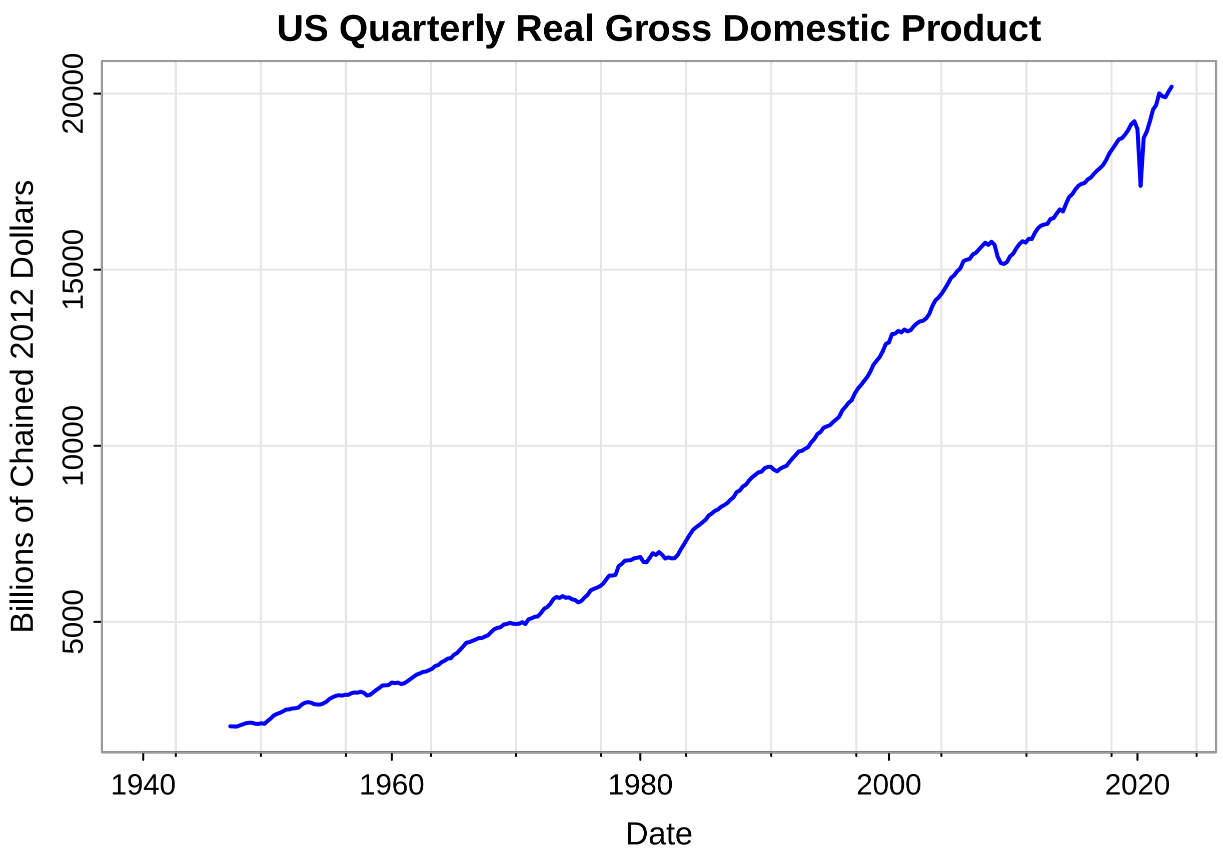


While the basic plot function does the job, it can be a little cumbersome to use. Fortunately there are many packages available that can streamline the process. One is is the **astsa** package. We can install the package using the **install.packages("astsa")** command, and load the package by executing the following line:

```
library(astsa)
```

Then we can generate the time series plot using the following code:

```
tsplot(date, gdpdata$GDPC1,
       main = "US Quarterly Real Gross Domestic Product",
       xlab = "Date",
       xlim = as.Date(c("1940-01-01","2023-01-01")),
       ylab = "Billions of Chained 2012 Dollars",
       lwd = 2.0,
       col = "blue")
```



Computing Percentage Change

Let's now proceed to compute a new data series called **gdpgr** that is the percentage change in quarterly real GDP. That is, we want to perform the following mathematical computation:

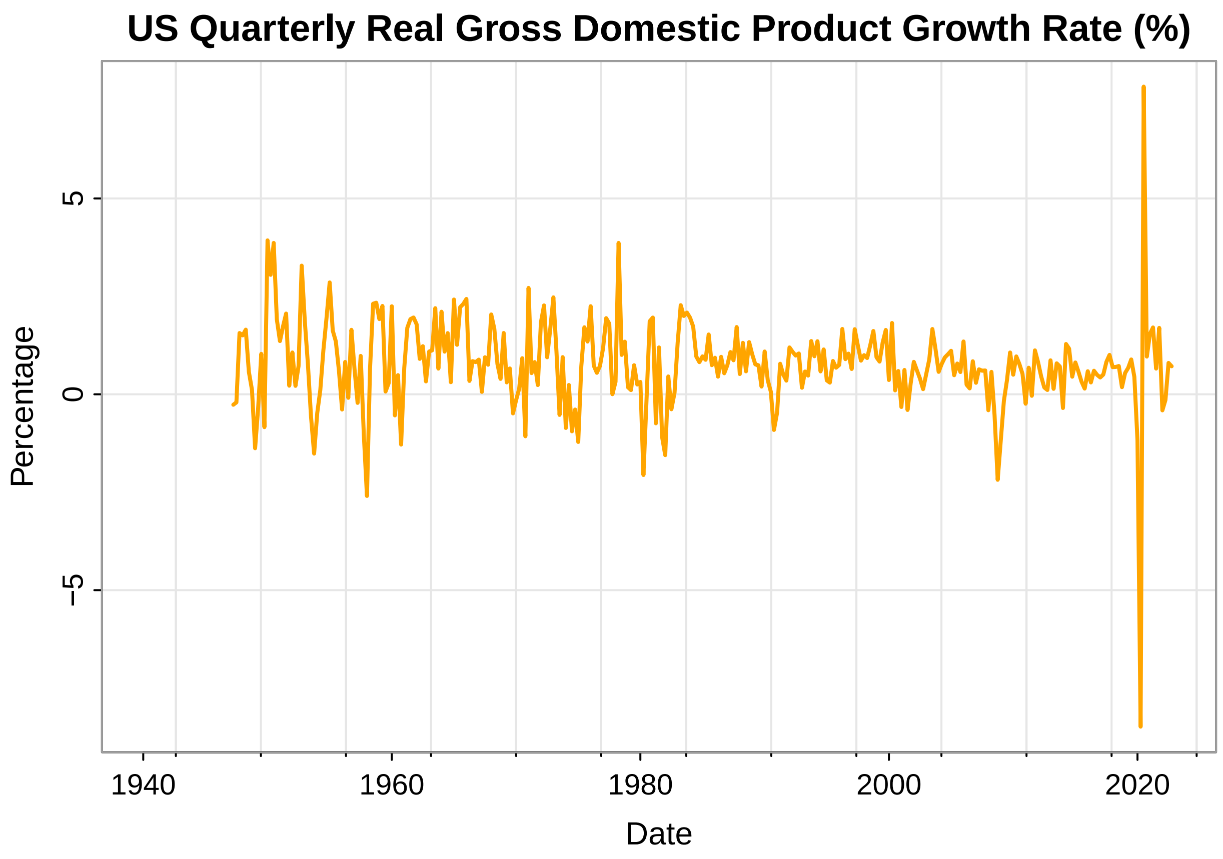
$\frac{RGDP_{t+1}-RGDPt}{RGDP_t} \times 100$

To do this, we run the following line of code:

```
n = length(date)
gdpgr = 100*(gdpdata$GDPC1[2:n] - gdpdata$GDPC1[1:n-1])/gdpdata$GDPC1[1:n-1]
```

Having generated the real gdp growth series, we can proceed to plot it using the following code:

```
tsplot(date[2:n], gdpgr,
       main = "US Quarterly Real Gross Domestic Product Growth Rate (%)",
       xlab = "Date",
       xlim = as.Date(c("1940-01-01","2023-01-01")),
       ylab = "Percentage",
       lwd = 2.0,
       col = "orange")
```



Simulating Time Series Data

Putting the GDP data aside, let's now turn our attention to the simulated data that I presented in the lecture. We begin with the white noise process defined as

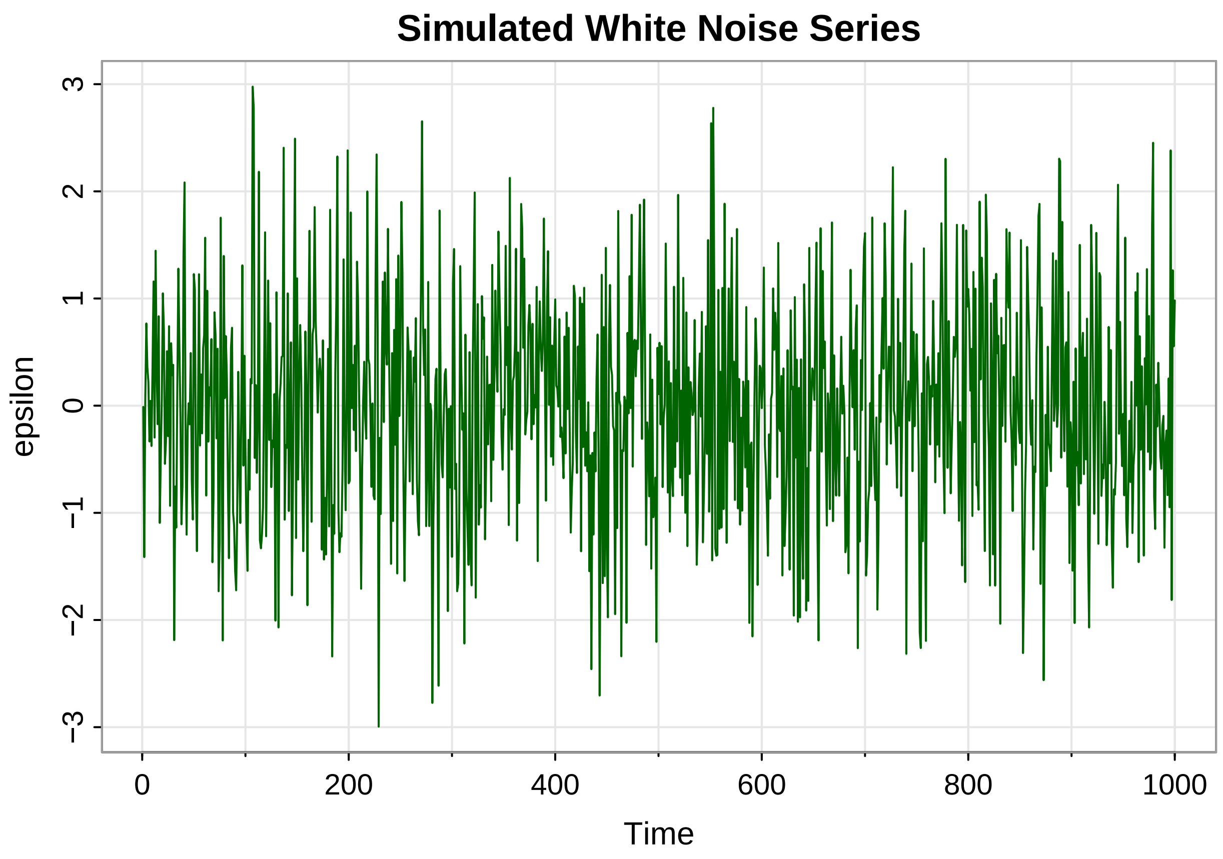
$$\epsilon_t \underset{\text{i.i.d.}}{\sim} N(0,1)$$

We can simulate a 1000 observations from this process using the following code:

```
epsilon = rnorm(1000, mean = 0, sd = 1)
```

To generate the plot, we run the following:

```
tsplot(epsilon,
       main = "Simulated White Noise Series",
       col = "darkgreen")
```



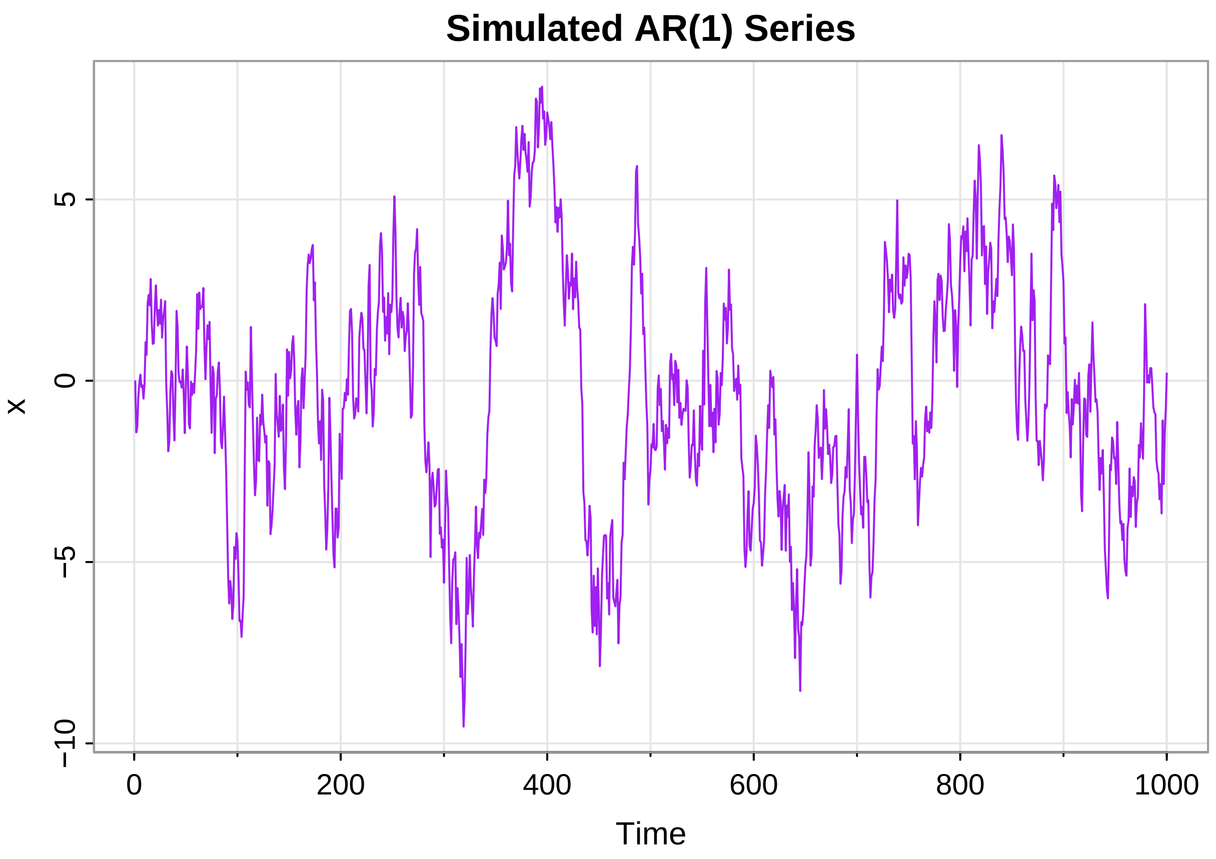
Having generated the white noise series, we can proceed to construct the autoregressive AR(1) series defined by the following formula

$X_t = 0.95X_{t-1} + \epsilon_t$

Due to the recursive nature of the AR(1) process, we can use a for loop to construct the series:

```r
t = length(epsilon)
x = rep(0,t)
x[1] = epsilon[1]
for(i in 2:t){
  x[i] = 0.95*x[i-1] + epsilon[i]
}
```

```r
tsplot(x,
       main = "Simulated AR(1) Series",
       col = "purple")
```


Simulated AR(1) Series
x
Time
5
0
−5
−10
0
200
400
600
800
1000