Lab 1

PSTAT W 174/274

R and Rstudio guidance

1. Download the latest version of Rstudio from the link: https://posit.co/downloads/

2. There is a cloud service available free from the link: https://posit.cloud

   RStudio Cloud provides about 15 hours per month of free and functional computing time in a virtual RStudio environment. The hours may not be enough to do all the class labs/projects, but can work in the case of emergencies.

3. Please make sure your downloaded library and data file is in the same working directory where project is run. Otherwise, you may not be able to load the data or use the functions in the library.

Plotting time series

1. Read in the data

   In this lab, we use a data set of the number of births per month in New York city, from January 1946 to December 1959 (originally collected by Dr. Newton). This data is available in the file http://robjhyndman.com/tsdldata/data/nybirths.dat We can read the data into R, and store it as a time series object, by typing:

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

2. Convert it into time series format.

```
births_ts = ts(births, start = c(1946,1), frequency = 12)
births_ts
```

```
##         Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
## 1946 26.663 23.598 26.931 24.740 25.806 24.364 24.477 23.901 23.175 23.227
## 1947 21.439 21.089 23.709 21.669 21.752 20.761 23.479 23.824 23.105 23.110
## 1948 21.937 20.035 23.590 21.672 22.222 22.123 23.950 23.504 22.238 23.142
## 1949 21.548 20.000 22.424 20.615 21.761 22.874 24.104 23.748 23.262 22.907
## 1950 22.604 20.894 24.677 23.673 25.320 23.583 24.671 24.454 24.122 24.252
```

```
## 1951 23.287 23.049 25.076 24.037 24.430 24.667 26.451 25.618 25.014 25.110
## 1952 23.798 22.270 24.775 22.646 23.988 24.737 26.276 25.816 25.210 25.199
## 1953 24.364 22.644 25.565 24.062 25.431 24.635 27.009 26.606 26.268 26.462
## 1954 24.657 23.304 26.982 26.199 27.210 26.122 26.706 26.878 26.152 26.379
## 1955 24.990 24.239 26.721 23.475 24.767 26.219 28.361 28.599 27.914 27.784
## 1956 26.217 24.218 27.914 26.975 28.527 27.139 28.982 28.169 28.056 29.136
## 1957 26.589 24.848 27.543 26.896 28.878 27.390 28.065 28.141 29.048 28.484
## 1958 27.132 24.924 28.963 26.589 27.931 28.009 29.229 28.759 28.405 27.945
## 1959 26.076 25.286 27.660 25.951 26.398 25.565 28.865 30.000 29.261 29.012
##         Nov    Dec
## 1946 21.672 21.870
## 1947 21.759 22.073
## 1948 21.059 21.573
## 1949 21.519 22.025
## 1950 22.084 22.991
## 1951 22.964 23.981
## 1952 23.162 24.707
## 1953 25.246 25.180
## 1954 24.712 25.688
## 1955 25.693 26.881
## 1956 26.291 26.987
## 1957 26.634 27.735
## 1958 25.912 26.619
## 1959 26.992 27.897
```

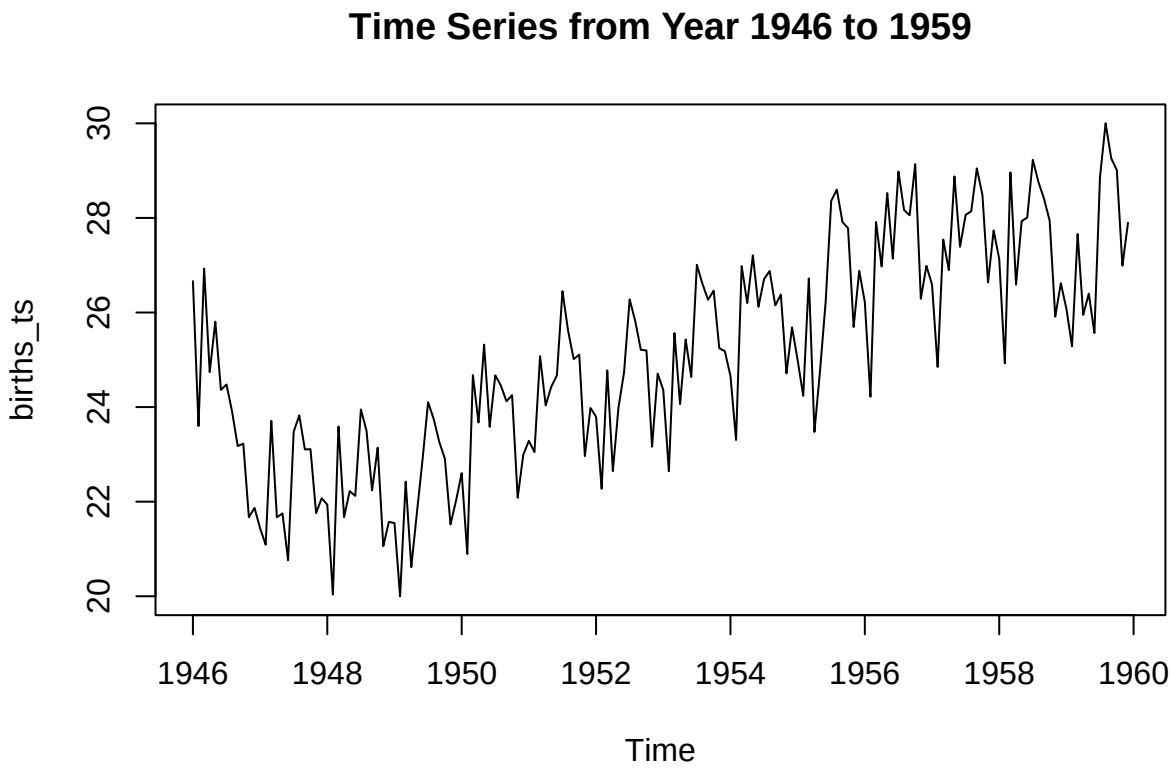
Function arguments:

start: the time of the first observation. Either a single number or a vector of two numbers (the second of which is an integer), which specify a natural time unit and a (1-based) number of samples into the time unit.

frequency: the number of observations per unit of time.

3. Plot the data versus time.

```
plot(births_ts, main = "Time Series from Year 1946 to 1959")
```



4. Plot the aggregated data versus time

Method: apply.yearly function in xts library

Apply Function over Calendar Periods: we have apply.daily, apply.weekly, apply.monthly, apply.quarterly and apply.yearly to aggregate our time series data to daily, weekly, monthly, quarterly and yearly. The common aggregated functions are mean, sum and var, which calculate the mean, sum and variance of the data in distinct period in a given time series.

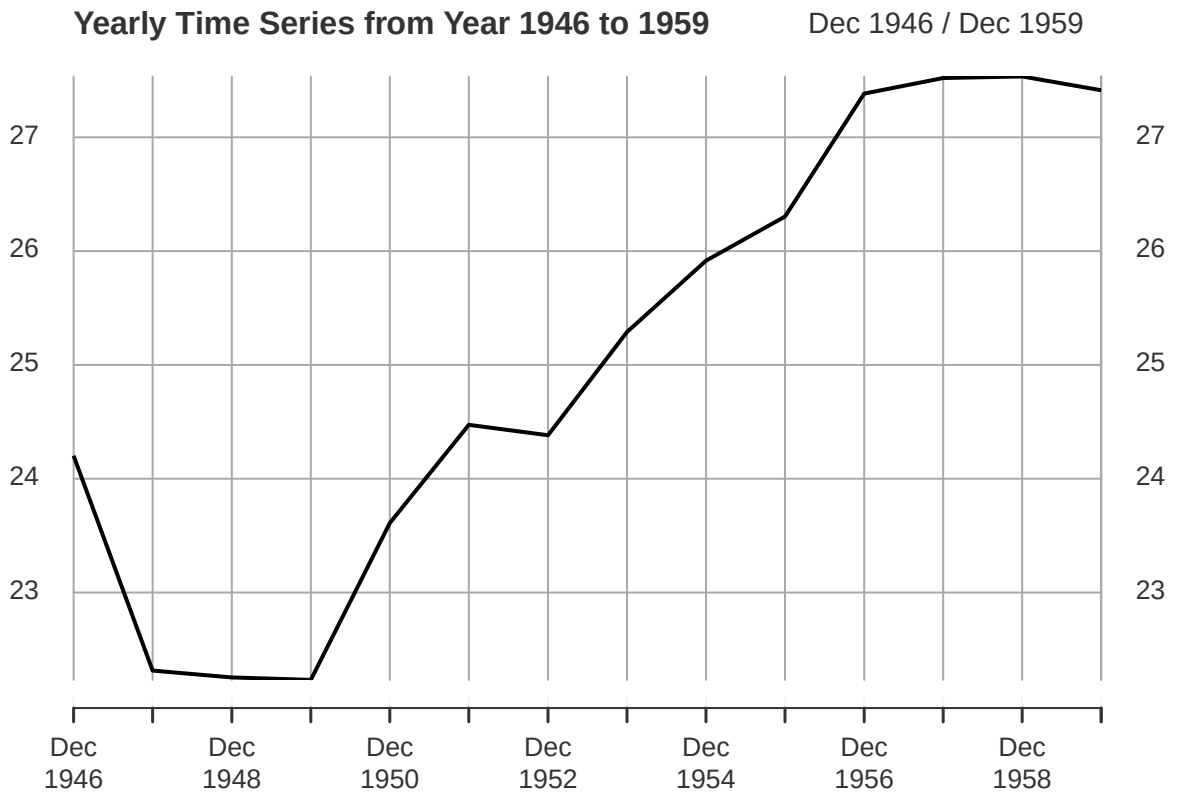
```
library(xts)
ts.yearly <- apply.yearly(as.xts(births_ts),FUN=mean)
ts.yearly
```

```
##              [,1]
## Dec 1946 24.20200
## Dec 1947 22.31408
## Dec 1948 22.25375
## Dec 1949 22.23225
## Dec 1950 23.61042
## Dec 1951 24.47367
## Dec 1952 24.38200
## Dec 1953 25.28933
## Dec 1954 25.91575
## Dec 1955 26.30358
## Dec 1956 27.38425
## Dec 1957 27.52092
```

```
## Dec 1958 27.53475
## Dec 1959 27.41358
```

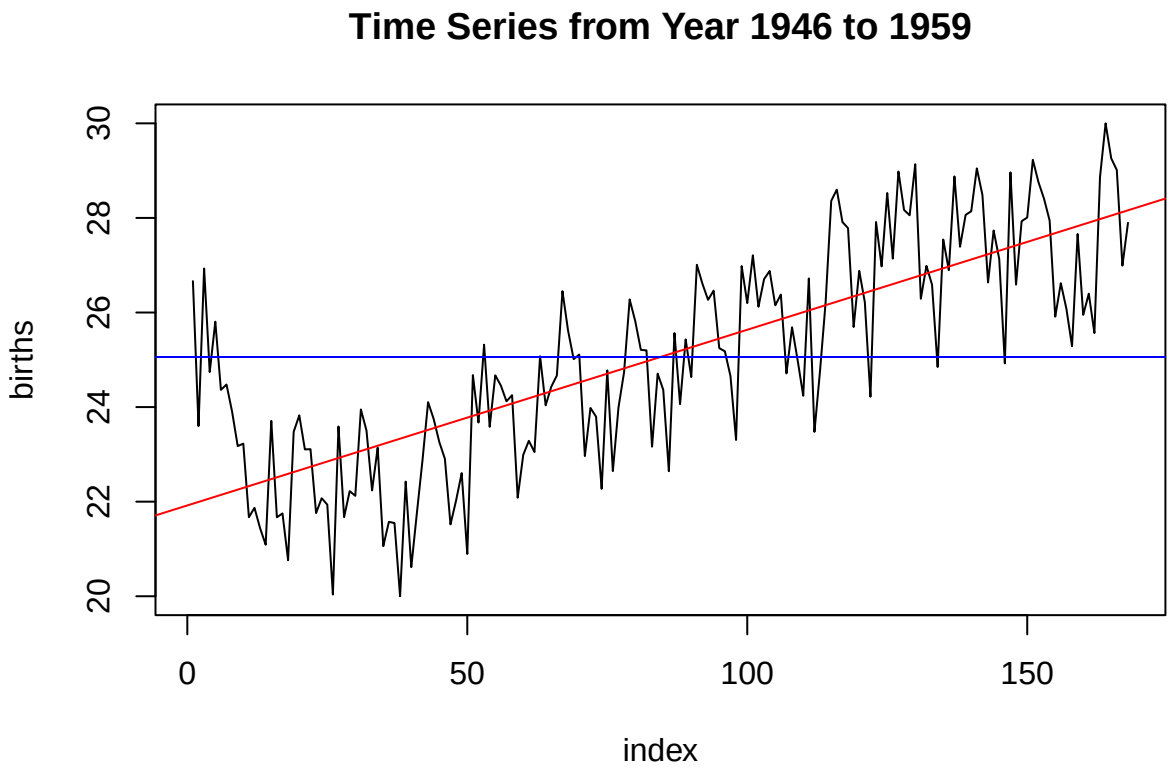
Plot the Aggregated yearly data:

```
plot(ts.yearly, main = "Yearly Time Series from Year 1946 to 1959")
```



5. Plot the data versus time index number from 1, 2, . . . n. Add regression line and mean line to the plot and print sample size n.

```
plot(1:length(births),births, main =
   "Time Series from Year 1946 to 1959", type = 'l',xlab='index')
index = 1: length(births)
trend <- lm(births ~ index)
abline(trend, col="red")
abline(h=mean(births) , col='blue')
```



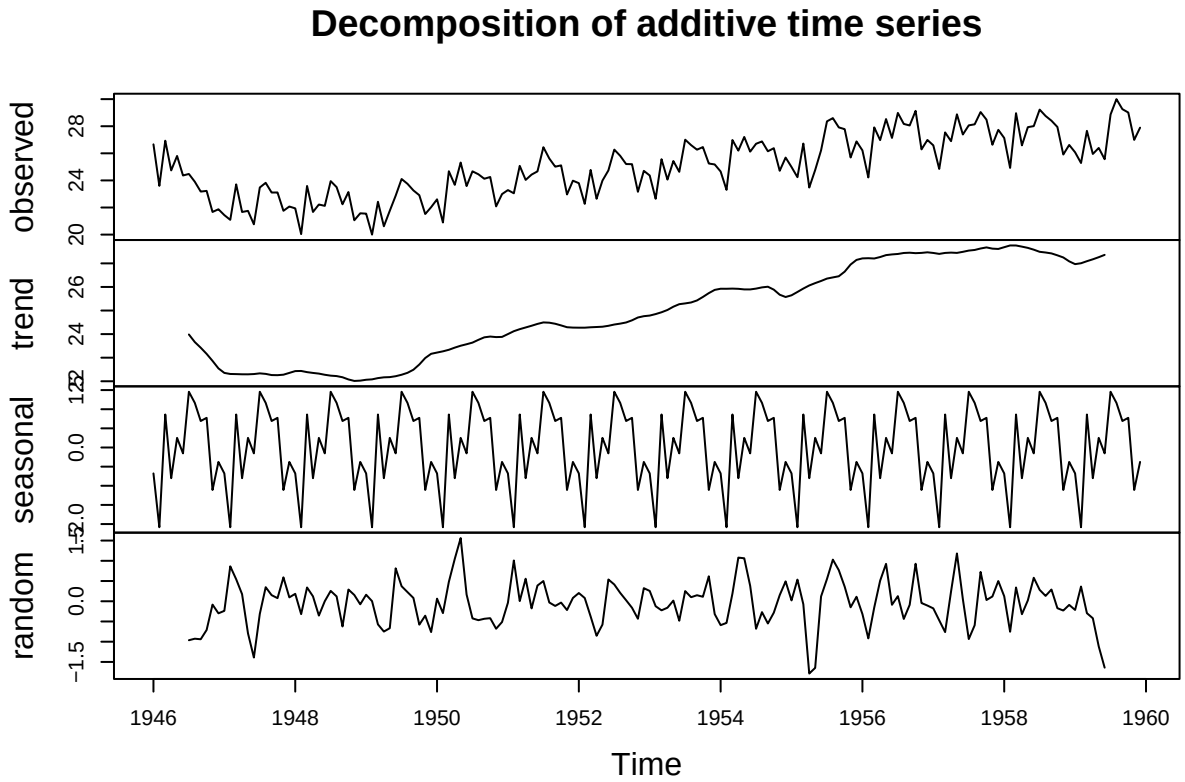
For the sample size of our data, we can use 'length' function to find the sample size:

```
length(births)
```

```
## [1] 168
```

6. State whether the data looks stationary. If not, is there a trend or is variance stable or is there a seasonal pattern.

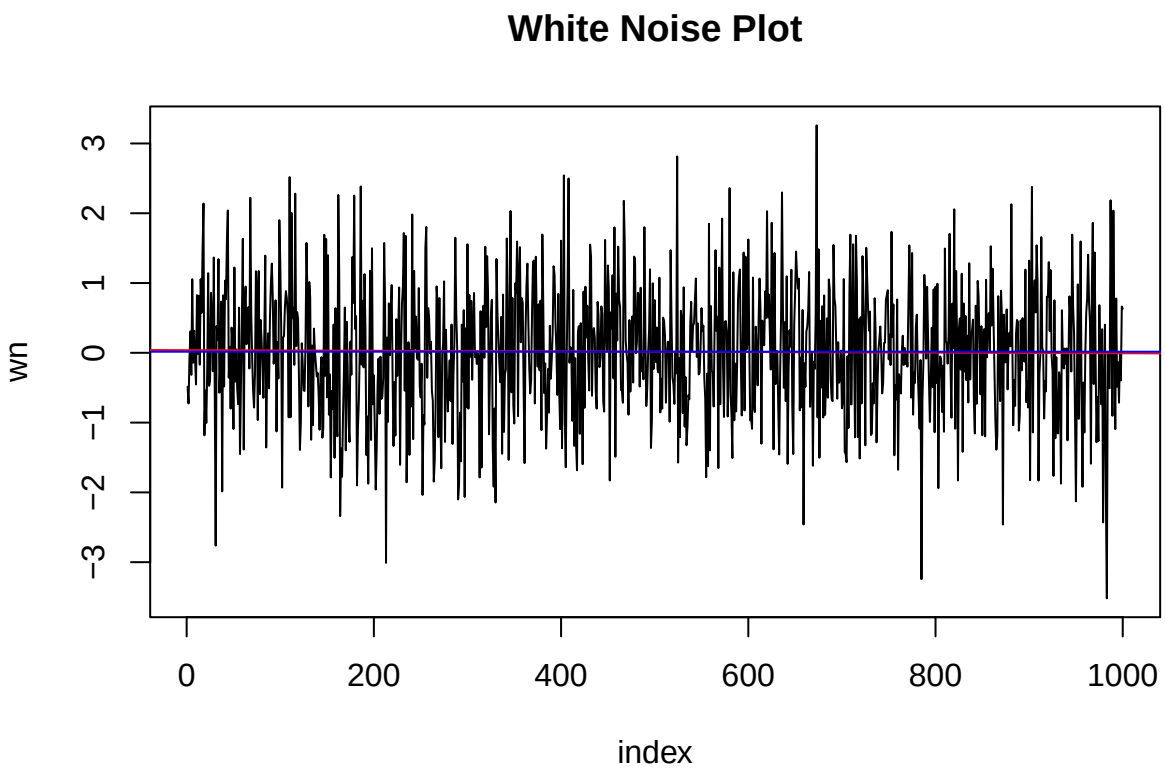
```
birthstimeseriescomponents <- decompose(births_ts)
plot(birthstimeseriescomponents)
```



From the plot above, we can see a downward trend between Jan 1946 to Dec 1947 and an upward trend from Jan 1948 to the end of the data. Also, a clear seasonal pattern is presented in the plot, as we can see the observations regularly go up and down. The variance is relatively stable across time. Therefore, this time series is not stationary.

Comparison Between a stationary process:

```
wn = rnorm(1000,0,1)
plot(1:length(wn),wn, main =
   "White Noise Plot", type = 'l',xlab='index')
trend <- lm(wn ~ time(wn))
abline(trend, col="red")
abline(h=mean(wn) , col='blue')
```



From the plot above, there is no obvious trend and the variance is stable across time. We do not observe obvious seasonality. Therefore, this is a stationary time series.

7. Does 1946-1948 data have the same pattern as the rest of the data? If no, please cut the start of the data e.g. your data will start at Jan, 1948.

   The data between Jan 1946 to Dec 1947 has a downward trend whereas the rest of the data has an upward trend. The pattern is different. Also, we need to reserve one year at the end of the data for testing purposes. Therefore, we cut the data and start at Jan, 1948 and end at Dec 1959.

```
select <- c(1:24)
#eliminate data from year Jan 1946 to Dec 1947 and we therefore now have 12 years(144 obse
birth_split <- births[-c(select)]
#use the first 11 years(132 observations) for training
training = birth_split[1:132]
#use the last 1 year(12 observations) for testing
testing = birth_split[133:144]
```

   We will be using the training data to train the time series model and use the last one year as test if our model performs well.

8. Generating Uniform distribution and calculate the mean.

   To generate from uniform distribution, we will use runif() function in r:

   runif(n, min, max)

n: number of observations.

min, max: lower and upper limits of the distribution. Must be finite.

In this example, we generate 1000 observations from uniform distribution U(-1,1) and calculate the mean:

```
observations = runif(1000,-1,1)
mean(observations)
```

```
## [1] -0.01055006
```