Time Series Analysis and Forecasting

October 15 – 16, 2018

John Carlo P. Daquis
Assistant Professor 4
UP Diliman School of Statistics

Chapter Outline

Chapter Outline

II. Introduction to RStudio

1. The RStudio Interface
2. Basic Data Management and Functions in RStudio
3. Creating Line Graphs

The RStudio Interface

The RStudio Interface

What is  ?

- R is an open-source software environment for statistical computing and graphics.
- It has a worldwide repository system called the Comprehensive R Archive Network (CRAN) http://cran.r-project.org to download the software and user-contributed add-on packages.
- The R language is **case sensitive**.

The RStudio Interface

What is RStudio?

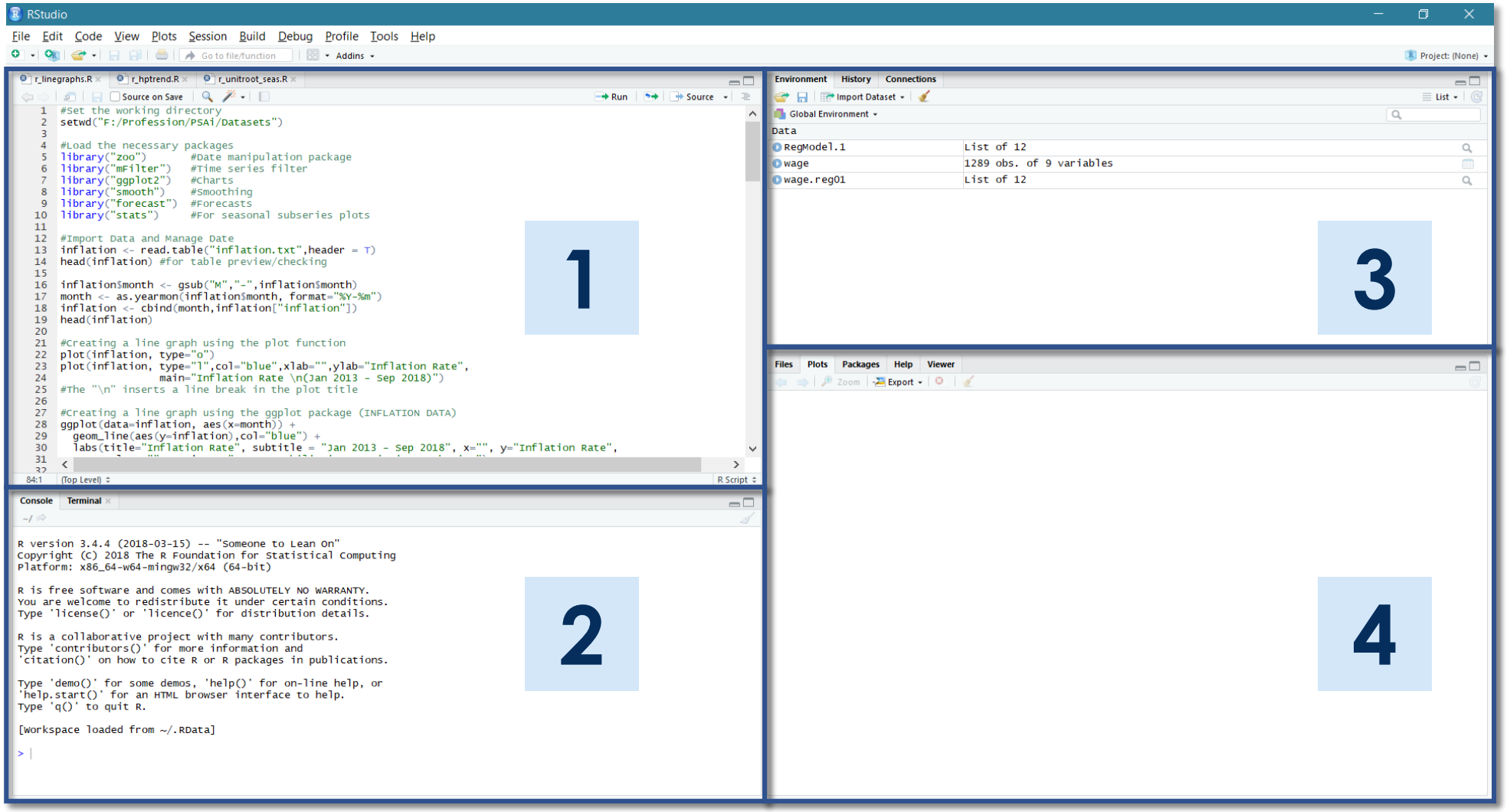
- RStudio is an Integrated Development Environment (IDE) that facilitates issuing of commands in R interactively.
- It has many convenient and easy-to-use tools for coding, plot management, object browsing, package management, among other things.
- To download Rstudio: https://www.rstudio.com/

The RStudio Interface

The RStudio 4-Panel Component Layout

- **Upper left:** Script window for editing R script files.
- **Lower left**: Console window for directly interacting with an R process.
- **Upper right:** Workspace and history browser
- **Lower right:** File browser, plot browser, package management, help browser.

The RStudio Interface



The RStudio Interface

The RStudio Script Window

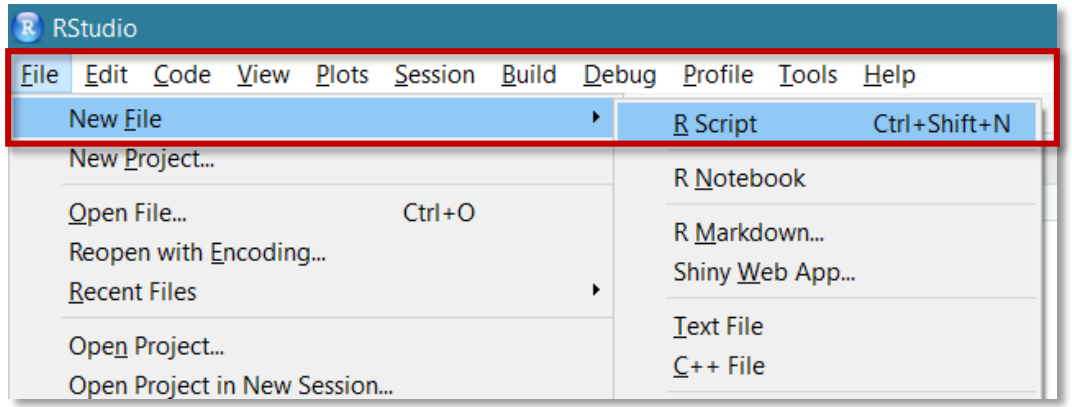
- RStudio's source editor includes a variety of productivity enhancing features including
  - syntax highlighting
  - selection running
  - code completion
  - multiple script file editing, and
  - find/replace.

Basic Data Management and Functions in RStudio

Basic Data Management and Functions in RStudio

The RStudio Script Window

- **Scripts.** The RStudio code is saved as a script with an **.R** extension.
- To create a new script, press the shortcut Ctrl+Shift+n or



Basic Data Management and Functions in RStudio

The RStudio Script Window

- **Working directories.** To set a working directory that contains the datasets to be used in the program, use the **setwd()** command. An example is:

```
#Set the working directory
setwd("F:/Profession/PSAi/Datasets")
```

- When copy-pasting directories, make sure to change \ to /.

Basic Data Management and Functions in RStudio

The RStudio Script Window

- **Comments.** These make source code easier for us to understand, and are generally ignored by the computer.

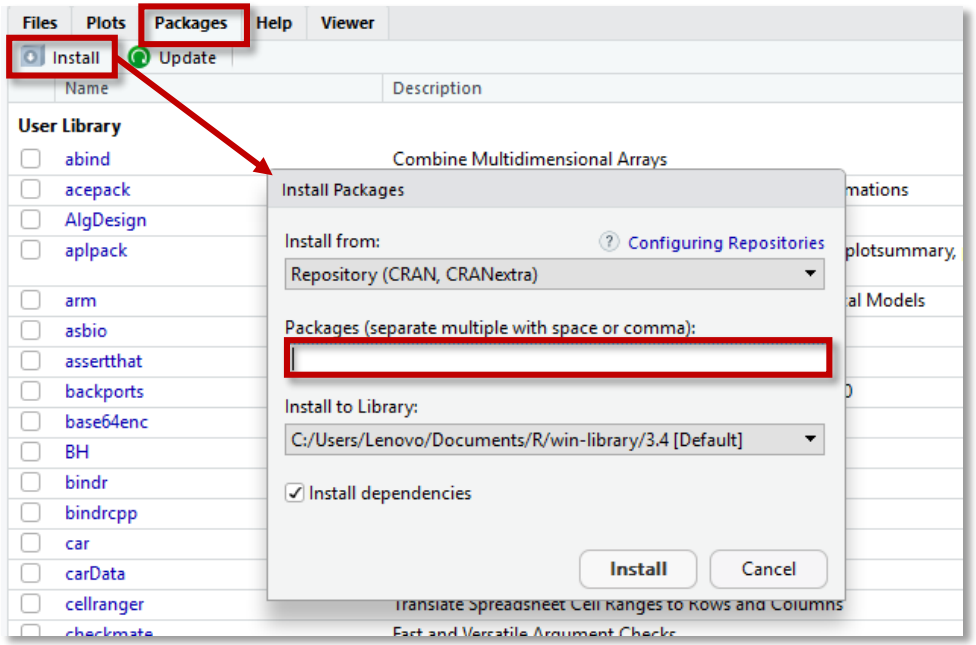
```
#Set the working directory
setwd("F:/Profession/PSAi/Datasets")
```

- Comments are preceded by the # symbol.

Basic Data Management and Functions in RStudio

Installing Packages

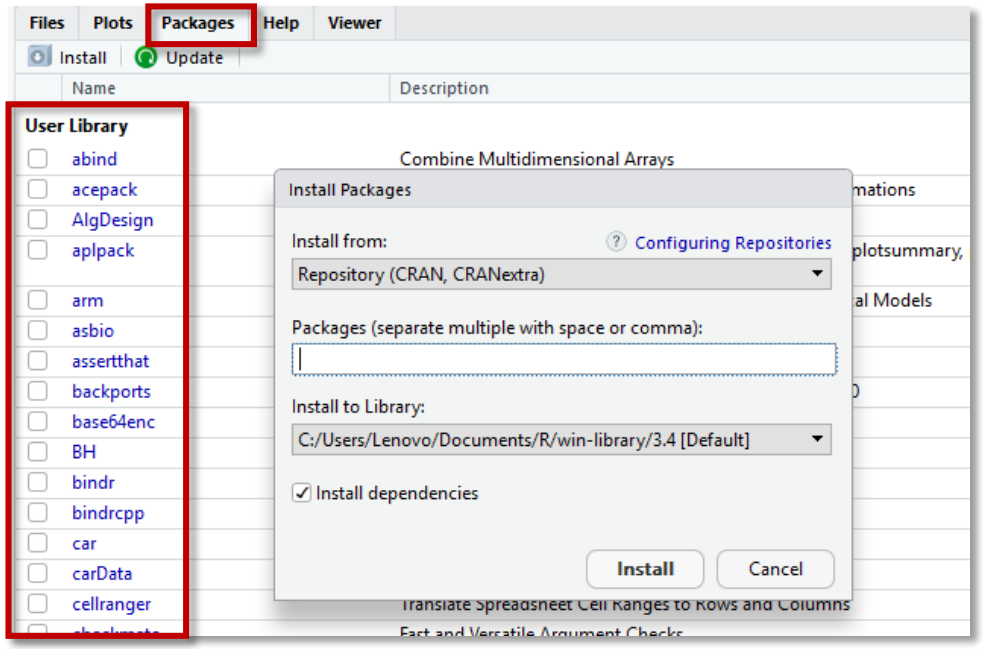
- The **packages** tab in the lower right panel can be used to install and load different packages.



Basic Data Management and Functions in RStudio

Installing Packages

- Loaded **packages** are marked with a check ✓ symbol.



Basic Data Management and Functions in RStudio

Installing Packages

- Alternatively, packages can be installed and loaded using **install.packages()** and **library()** commands, respectively.

```
install.packages("forecast")
library("forecast")
```

- While installation is a one-off process, needed packages are be loaded every time R is opened.

Basic Data Management and Functions in RStudio

Other Remarks

- More syntax will be added as we progress with the training.
- R syntax is **case-sensitive** (e.g., Variables x and X are different. Thus, creating objects with the same name but different case is highly discouraged.
- It is likely that a process can be coded and done in a number of ways in R.

Basic Data Management and Functions in RStudio

Other Remarks

- **Assignment operator.** R expressions at the right can be assigned to objects at its left using the assignment operator <-. Here, a dataset from the text file "inflation.txt" is assigned to an object at the left named "inflation".

```
inflation <- read.table("inflation.txt",header = T)
```

Creating Line Graphs

Line Graphs

What are line graphs?

- Line graphs or line charts are plots with time indices (e.g. months, quarters, years) on the x-axis and the time series values on the y axis;
- Line graphs are used to visualize the value of a variable over time.
- RStudio can output single, multiple and seasonal line graphs.

Line Graphs

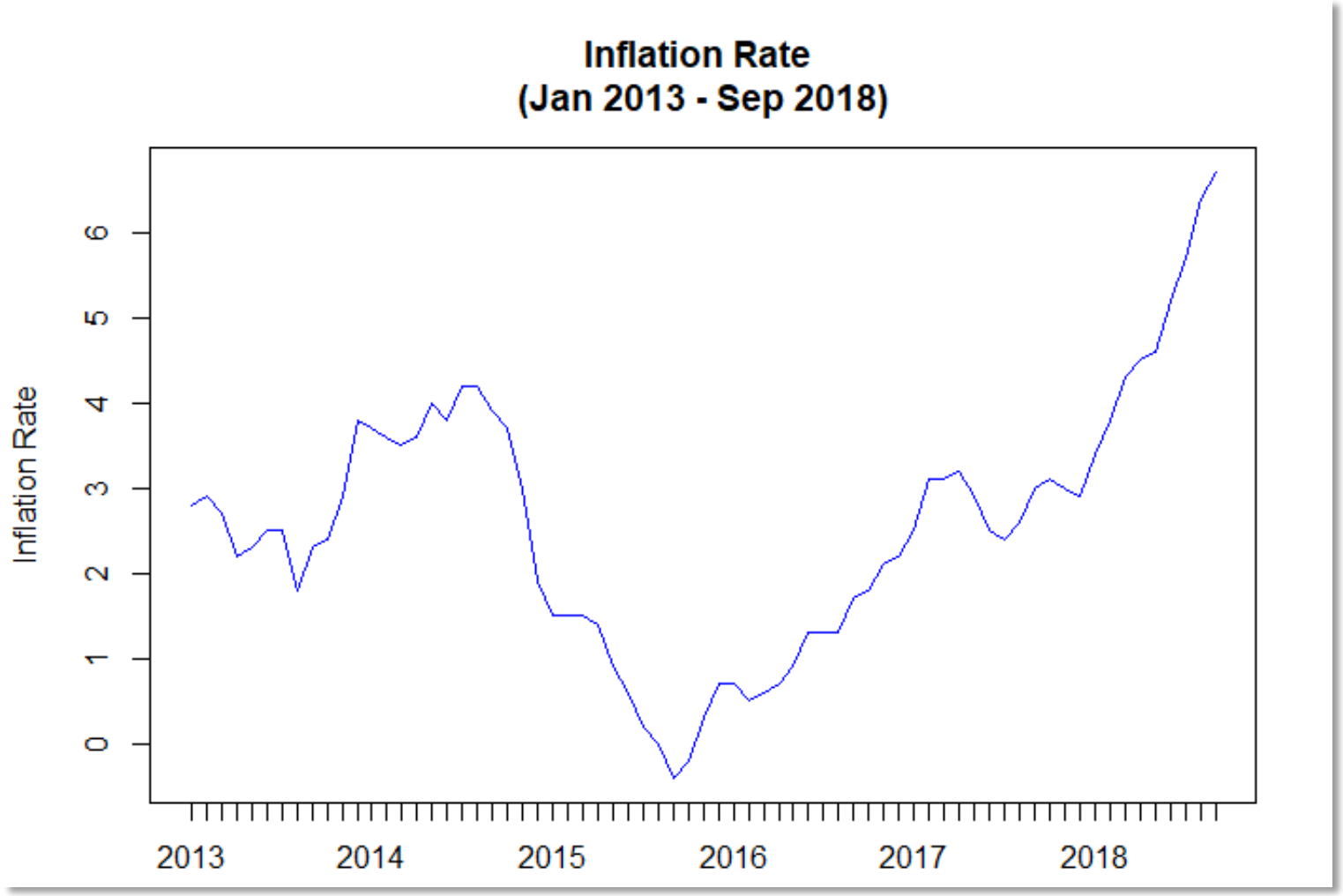
What are line graphs?

- These are used to obtain information about the movement and behavior of a time series.
- Multiple graphs allow for comparison across different time series.
- Seasonal graphs help the researcher know the seasonal behaviors of the time series and be able to analyze it across years.

Line Graphs

Graphs

Here's a single line graph of the time series.



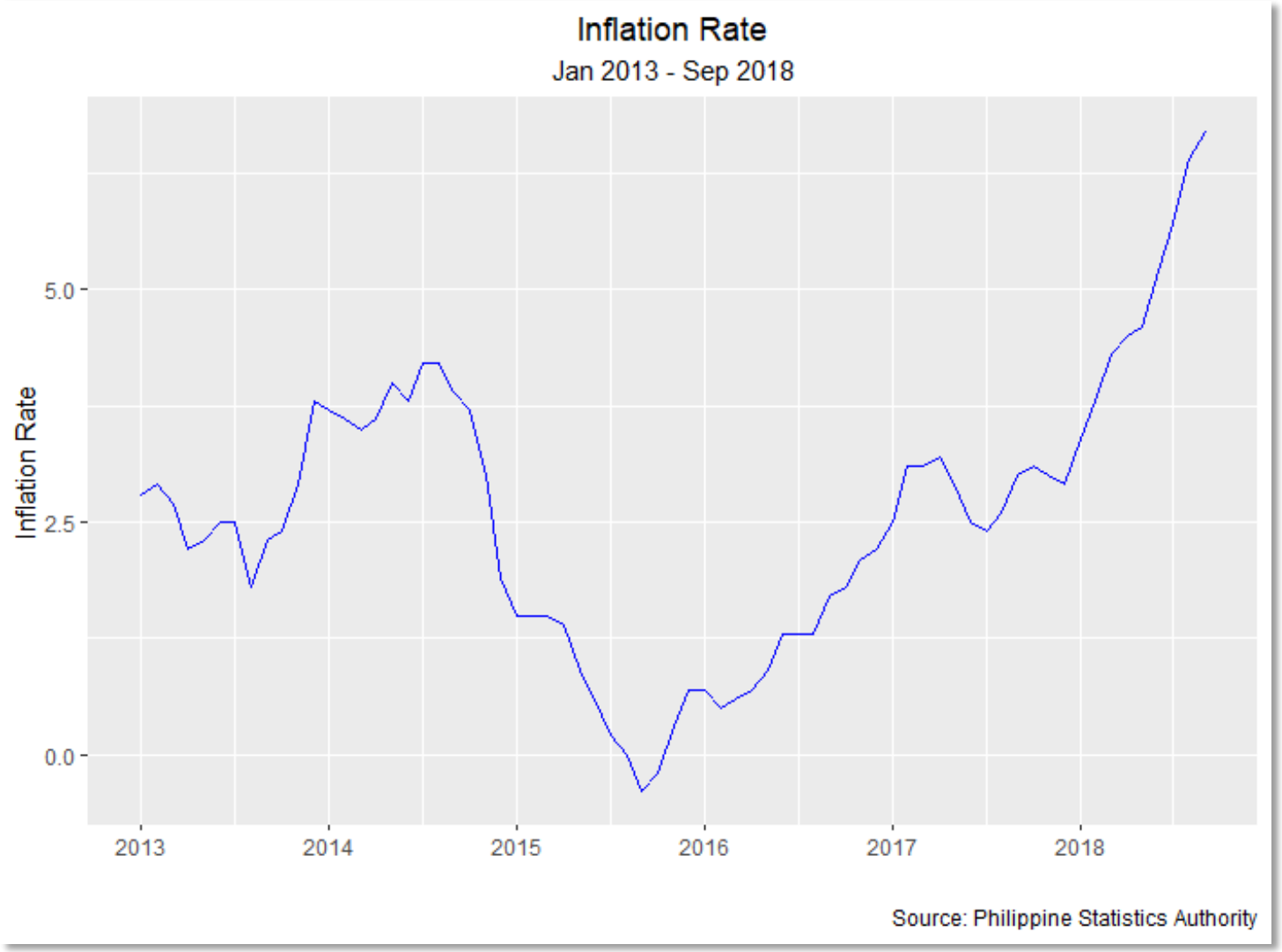
Line Graphs

```
#Import Data and Manage Date
inflation <- read.table("inflation.txt",header = T)
head(inflation) #for table preview/checking

inflation$month <- gsub("M","-",inflation$month)
month <- as.yearmon(inflation$month, format="%Y-%m")
inflation <- cbind(month,inflation["inflation"])
head(inflation)

#Creating a line graph using the plot function
plot(inflation, type="o")
plot(inflation, type="l",col="blue",xlab="",ylab="Inflation Rate",
                main="Inflation Rate \n(Jan 2013 - Sep 2018)")
#The "\n" inserts a line break in the plot title
```

Line Graphs

Graphs

Here's the same data, graphed using R's **ggplot2** package



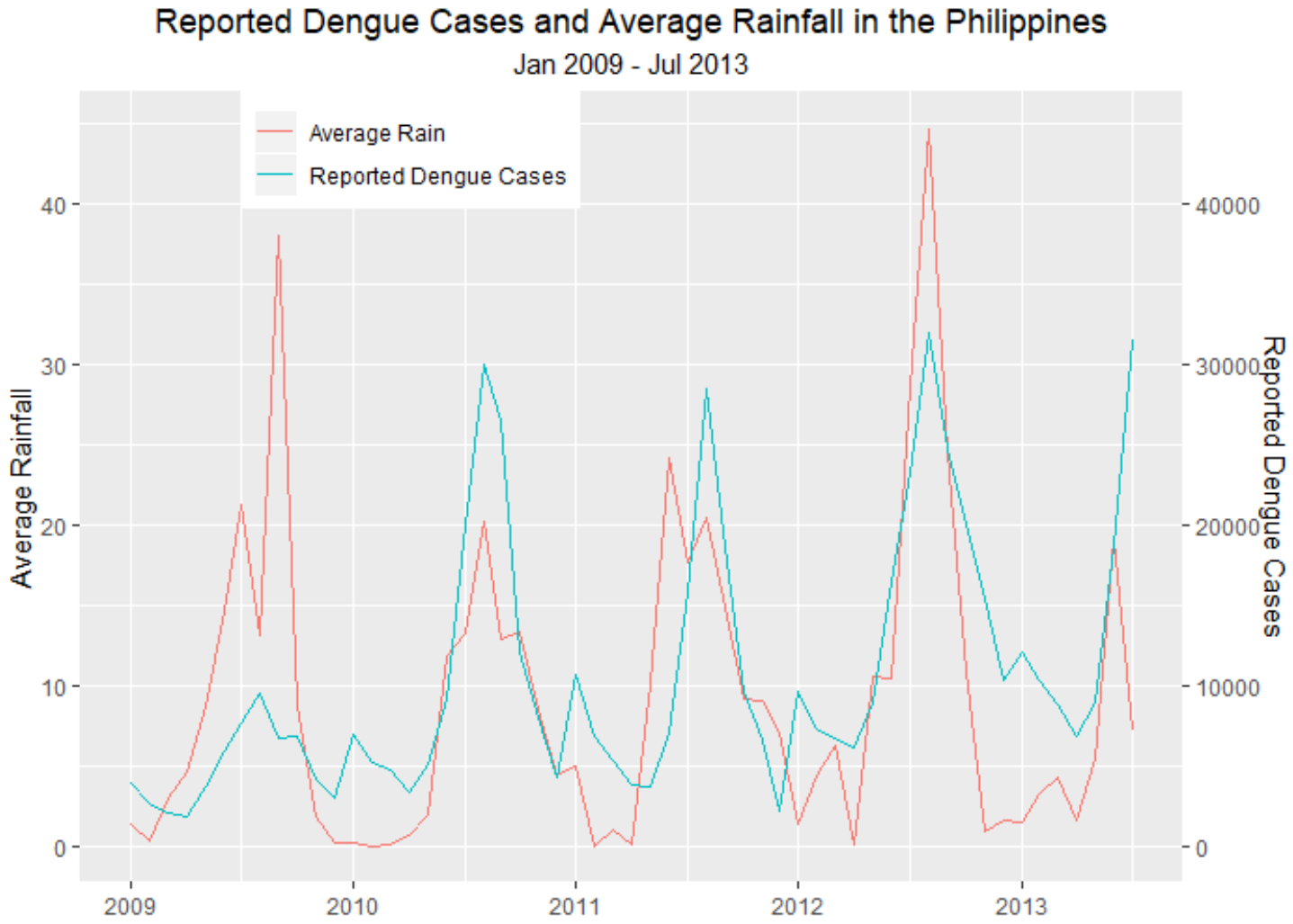
Line Graphs

```
#Creating a line graph using the ggplot package (INFLATION DATA)
ggplot(data=inflation, aes(x=month)) +
  geom_line(aes(y=inflation),col="blue") +
  labs(title="Inflation Rate", subtitle = "Jan 2013 - Sep 2018", x="",
       y="Inflation Rate",
       colour="",caption = "Source: Philippine Statistics Authority") +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```

Line Graphs

Graphs

Multiple graphs allow for comparison among different series.



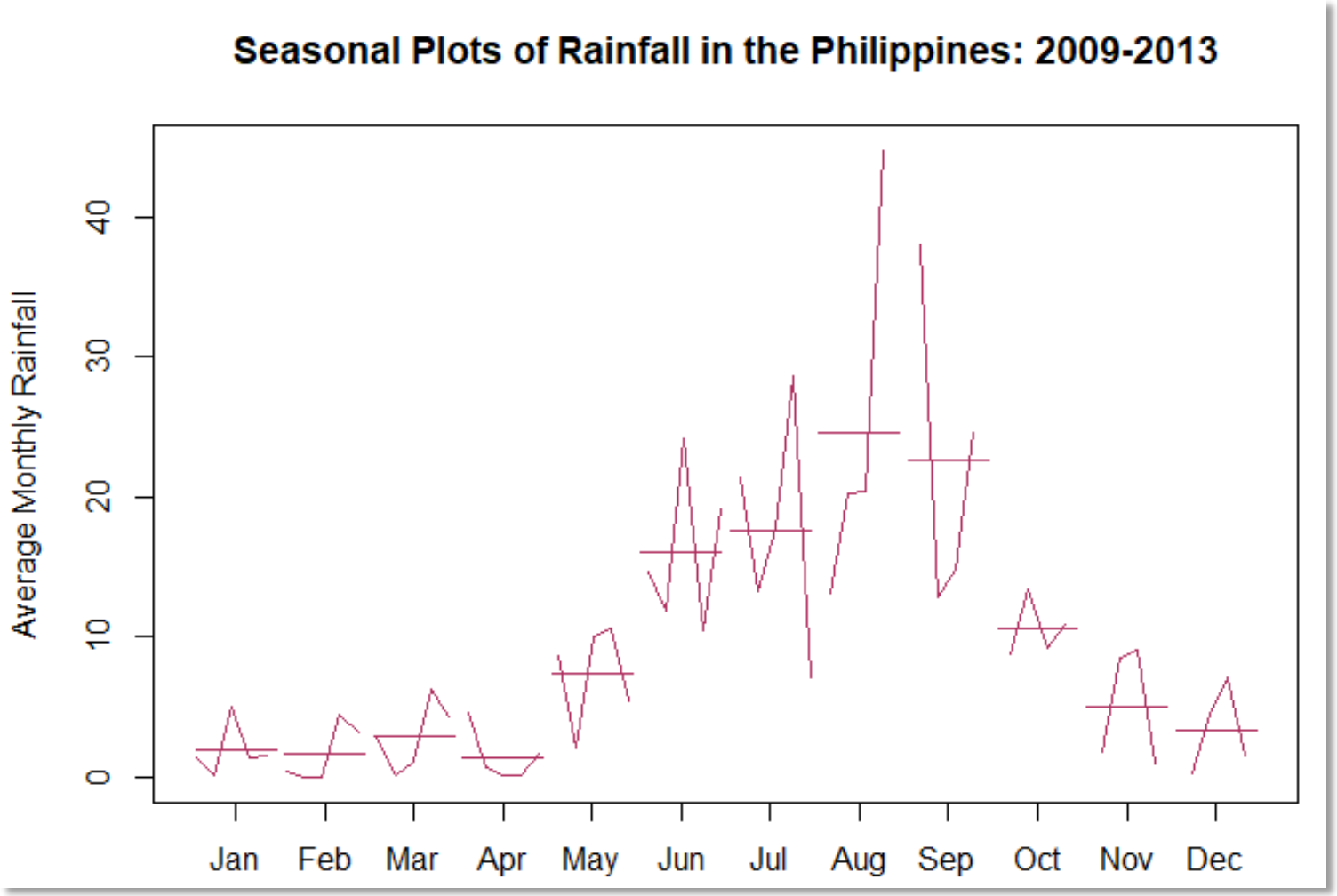
Line Graphs

```
#rainfall data (IMPORT AND LINE GRAPH)
rainfall <- read.table("rainfall.txt",header = T)
head(rainfall) #for table preview/checking

rainfall$month <- gsub("M","-",rainfall$month)
month <- as.yearmon(rainfall$month, format="%Y-%m")
rainfall <- cbind(month,rainfall["ave_rain"],rainfall["dengue"])
head(rainfall)

is.data.frame(rainfall)
ggplot(data=rainfall, aes(x=month,group = 1)) +
  geom_line(aes(y=ave_rain,colour="Average Rain")) +
  geom_line(aes(y=dengue*1/1000,colour="Reported Dengue Cases")) +
  labs(title="Reported Dengue Cases and Average Rainfall in the Philippines",
       subtitle = "Jan 2009 - Jul 2013", x="", y="Average Rainfall", colour="") +
  theme(plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        legend.position = c(.3, .95)) +
  scale_y_continuous(sec.axis = sec_axis(~.*1000/1,
                     name = "Reported Dengue Cases"))
```

Line Graphs

Graphs

Seasonal graphs reveal the seasonal behavior along with its mean (horizontal line).



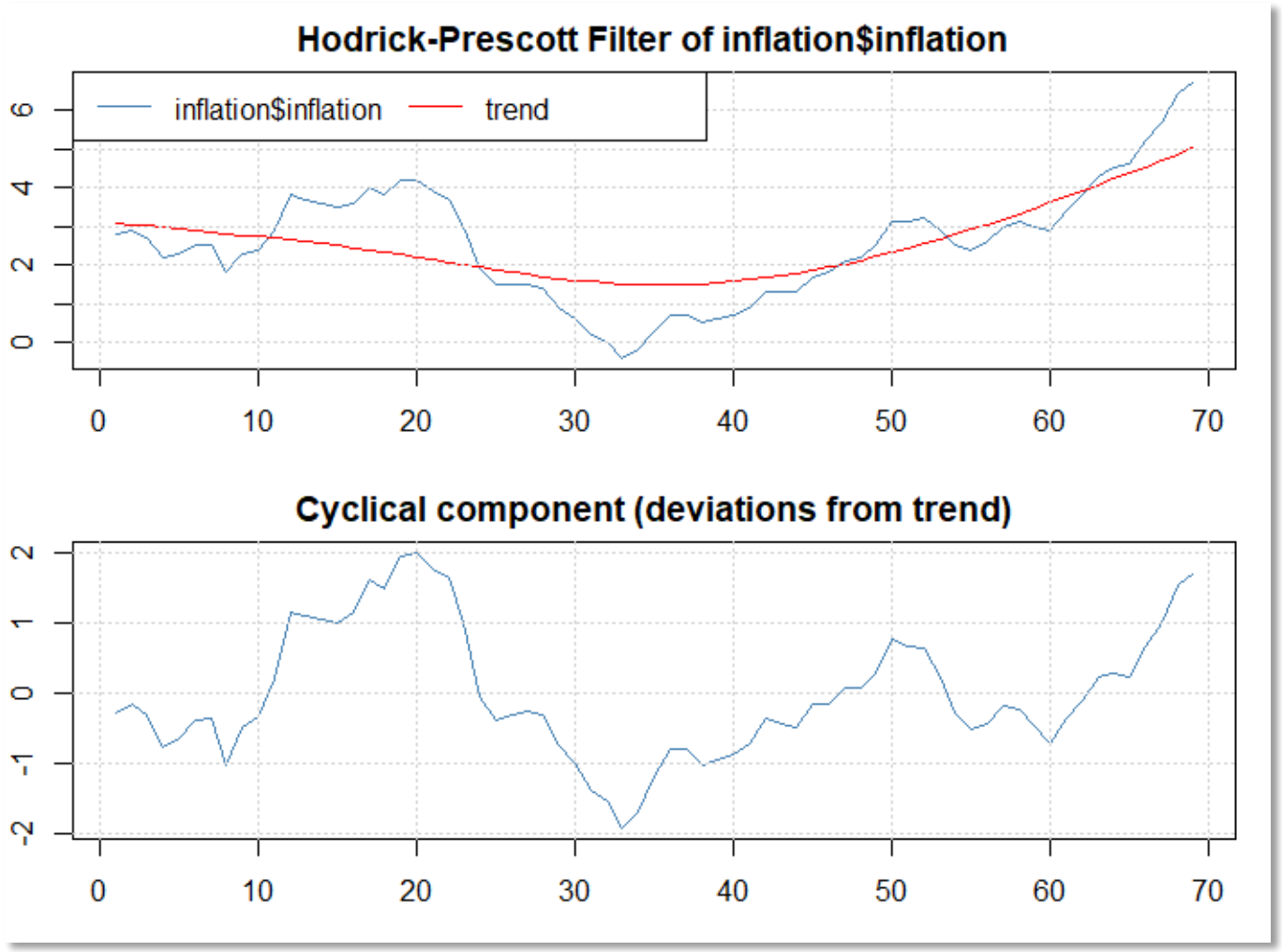
Line Graphs

```
#Rainfall data (for seasonal plots)
rain <- read.table("rain.txt",header = T)
rain <- ts(rain,frequency=12,start=c(2009,1),end=c(2013,7))
head(rain) #for table preview/checking

plot(rain,col="maroon",
     main="Average Rainfall in the Philippines: 2009-2013",
     xlab="",
     ylab="Average Monthly Rainfall")
monthplot(rain,labels = month.abb,col="maroon",
          main="Seasonal Plots of Rainfall in the Philippines: 2009-2013",
          ylab="Average Monthly Rainfall")
```

Line Graphs

Graphs

The **Hodrick-Prescott Filter** is a method that is used to obtain a smooth estimate of the long-term trend of a series.



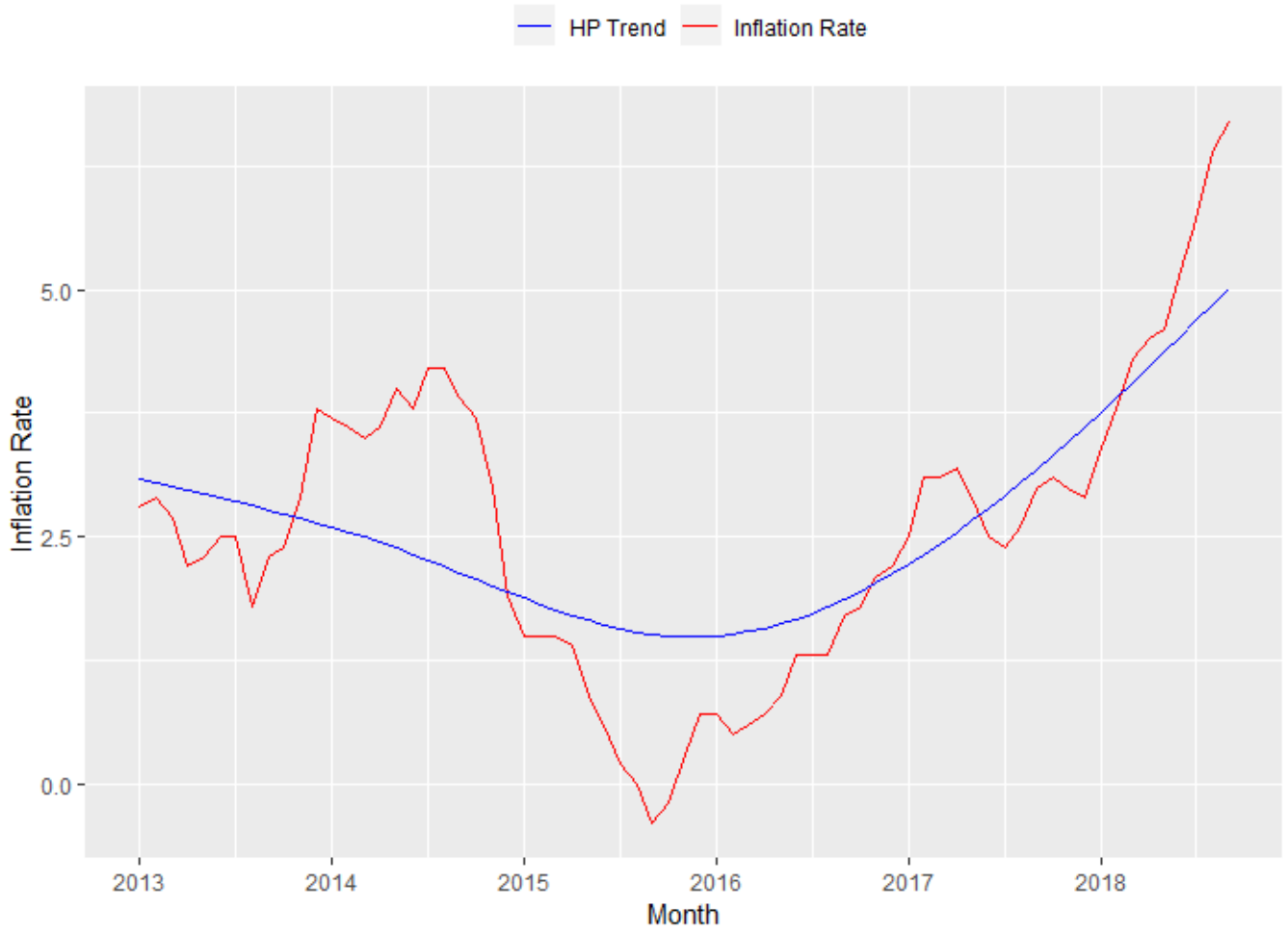
Line Graphs

```
#Hodrick-Prescott Filter
inflation.hp <- hpfilter(inflation$inflation, freq=14400, type="lambda")
#for freq values: monthly-1440, quarterly-1600, annual-100
plot(inflation.hp)

inflation <- cbind(inflation,inflation.hp$trend)
head(inflation)
colnames(inflation)[3] <- "hp_trend"
head(inflation)
```

Line Graphs

Graphs

Here's the same HP filter output dataset with added **ggplot2** formatting.



Line Graphs

```
#ggplot input: Dataframe
is.data.frame(inflation)
ggplot(data=inflation, aes(x=month)) +
  geom_line(aes(y=inflation, colour="Inflation Rate")) +
  geom_line(aes(y=hp_trend, colour="HP Trend")) +
  labs(x="Month", y="Inflation Rate", colour="") +
  theme(legend.position="top") +
  scale_colour_manual(values=c("blue","red"))
```

Workshop

Line Graphs

Workshop 1: Time Series Decomposition

Run the program below and comment on what happens.

```
#put the data into a time series
rgdp.ts <- ts(read.table("rgdp2.txt",header =
T),frequency=4,start=c(1998,1),end=c(2018,2))

#line plot with elements
plot(rgdp.ts, main="Quartely GDP of the Philippines: Q1 1998 - Q2 2018
               (at constant 2000 prices)",
     ylab="Real GDP (in Million Pesos)", xlab="")
```

Line Graphs

Workshop 1: Time Series Decomposition

Run the program below and comment on what happens.

```
#time series decomposition
decomp_rgdp <- decompose(rgdp.ts,"additive")

#plot components
par(mfrow=c(2,2)) #run this par if you want a single panel of four graphs
plot(rgdp.ts,ylab="Observed",xlab="")
plot(decomp_rgdp$trend,ylab="Trend-Cycle",xlab="")
plot(decomp_rgdp$seasonal,ylab="Seasonality",xlab="")
plot(decomp_rgdp$random,ylab="Irregular",xlab="")
par(mfrow=c(1,1)) #run this par if you to revert to a single panel of one graph
```

Line Graphs

Workshop 2: Hodrick-Prescott Filter

Write an R code that outputs a Hodrick-Prescott filter on *exports* data. Output graphs similar to that of slides #30 and #32.


Thank
You