MidTermExam

Lince Rumainum

March 7, 2019

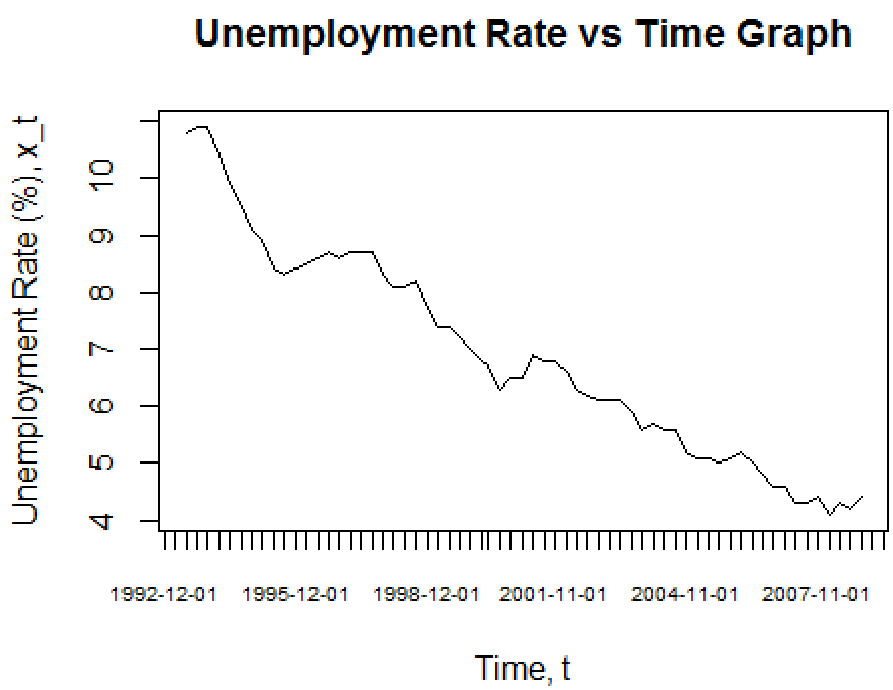
R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

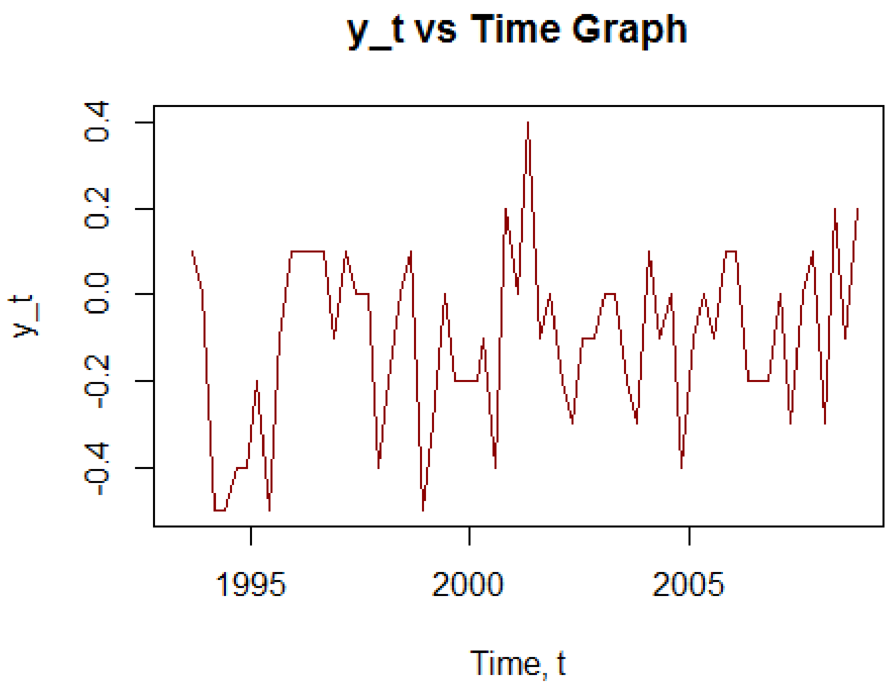
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
#------------------------------------------------------------------------------------------
# Problem #2
# Download two time series from the website
# that exhibit seasonality and trend
# data URLs:
# https://www.rba.gov.au/statistics/xls/unemployment-rate-by-horizon.xls
# http://www.bom.gov.au/tmp/cdio/IDCJAC0010_066196_2017.pdf - data no longer available
# http://www.bom.gov.au/tmp/cdio/IDCJAC0010_066196_2016.pdf
# outside sources: (reading xls and pdf file)
# https://stackoverflow.com/questions/41368628/read-excel-file-from-a-url-using-the-
readxl-package
# https://datascienceplus.com/extracting-tables-from-pdfs-in-r-using-the-tabulizer-
package/
#------------------------------------------------------------------------------------------
#list of libraries needed for problem #2
#install.packages("httr")
#install.packages('tabulizer')
library(DataCombine) # for slide function, i.e.: x_t-1
library(mgcv)

## Loading required package: nlme

## This is mgcv 1.8-26. For overview type 'help("mgcv-package")'.

library(readxl) # for reading excel file
library(httr)
packageVersion("readxl")

## [1] '1.3.0'

library(tabulizer) # for reading pdf file
library(dplyr)

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:nlme':
##
##     collapse
```

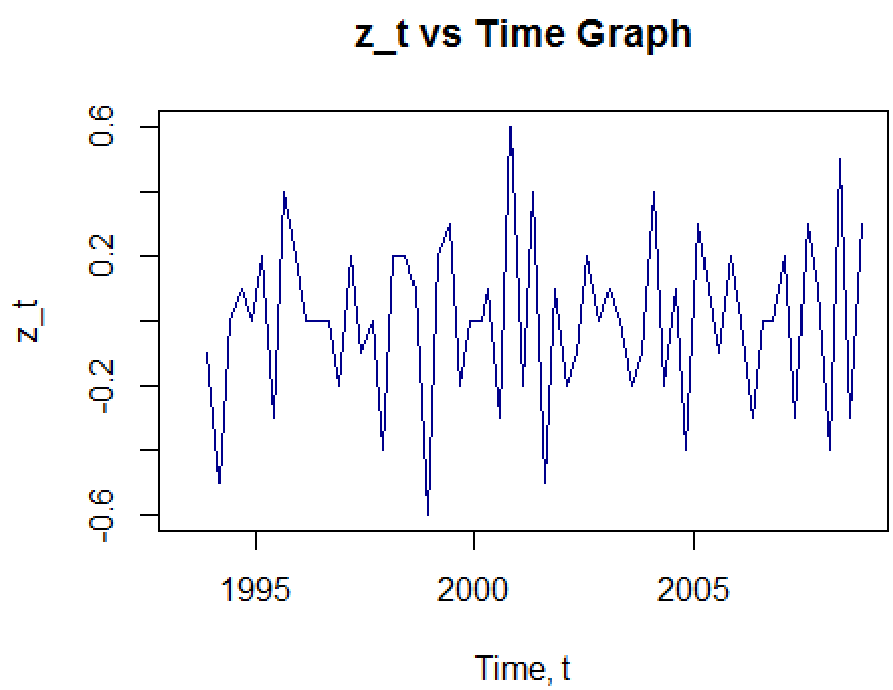
```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

# First time series is the unemployment rate data from the Australian government website
dataURL <- "https://www.rba.gov.au/statistics/xls/unemployment-rate-by-horizon.xls"
GET(dataURL, write_disk(tf <- tempfile(fileext = ".xls")))

## Response [https://www.rba.gov.au/statistics/xls/unemployment-rate-by-horizon.xls]
##   Date: 2019-03-08 03:51
##   Status: 200
##   Content-Type: application/octet-stream
##   Size: 243 kB
## <ON DISK>  C:\Users\Lince\AppData\Local\Temp\RtmpKiy49a\file26a446b9d374f.xls

df_unemployment <- read_excel(tf, sheet = 3, col_names = TRUE, col_types = NULL, na = "",
skip = 1)

## New names:
## * `t-2` -> `t-2..11`
## * `t-2` -> `t-2..12`

df_unemployment$`Forecast date` <- as.Date(df_unemployment$`Forecast date`)

# Get unemployment data from June 1993 to November 2008
t <- df_unemployment$`Forecast date`[18:80]
unemploymentRate <- df_unemployment$t[18:80]

# Plot the original series, x_t
plot(t,unemploymentRate, type = "l", main="Unemployment Rate vs Time Graph", xaxt = "n",
     xlab="Time, t", ylab="Unemployment Rate (%), x_t")
# Fix the x-axis so it would show dates
axis(1, df_unemployment$`Forecast date`, format(df_unemployment$`Forecast date`),
cex.axis = 0.6)
```



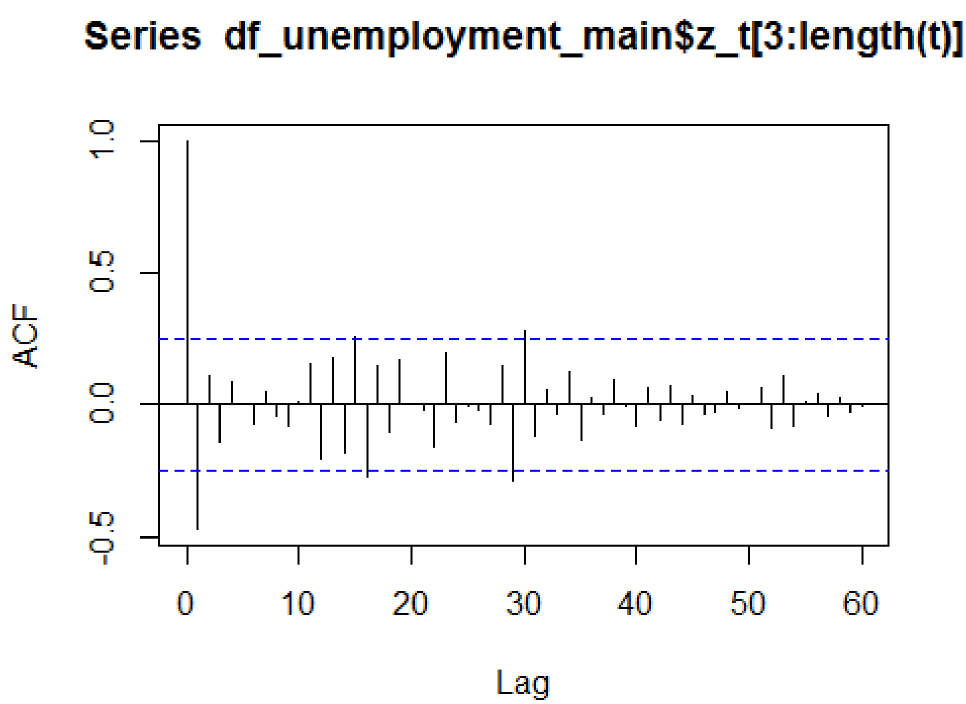

```r
# Create a new dataframe without the unnecessary data
df_unemployment_main <- data.frame(t)
for (i in 1:length(t)){
  df_unemployment_main$x_t[i] <- unemploymentRate[i]
}

# Problem 1-b: Plot y_t after removing the seasonality in x_t
df_unemployment_main<- slide(df_unemployment_main,"x_t", "t", NewVar="xtLag1", slideBy =
-1)

##
## Lagging x_t by 1 time units.

# removing the seasonality from x_t
for (i in 1:length(t)){
  if (!is.na(df_unemployment_main$x_t[i]) & !is.na(df_unemployment_main$xtLag1[i])){
    df_unemployment_main$y_t[i] <- df_unemployment_main$x_t[i] -
df_unemployment_main$xtLag1[i]
  }
  else{
    df_unemployment_main$y_t[i] <- NA
  }
}

# plot yt vs t
y_t <- df_unemployment_main$y_t
t <- df_unemployment_main$t
plot(t, y_t, type = "l", col= colors()[100], main="y_t vs Time Graph", xlab="Time, t",
ylab="y_t")
```




```
# Problem 1-c: Plot z_t after removing the trend in y_t

df_unemployment_main <- slide(df_unemployment_main,"y_t", "t", NewVar="ytLag1", slideBy = -1)

##
## Lagging y_t by 1 time units.

# removing the trend in y_t
for (i in 1:length(t)){
  df_unemployment_main$z_t[i] <- df_unemployment_main$y_t[i] - df_unemployment_main$ytLag1[i]
}

# plot zt vs t
z_t <- df_unemployment_main$z_t
t <- df_unemployment_main$t
plot(t, z_t, type = "l", col= colors()[30], main="z_t vs Time Graph", xlab="Time, t", ylab="z_t")
```




```
#Problem 1-d: autocorrelation, ACF
acf (df_unemployment_main$z_t[3:length(t)], lag.max = length(t),
type=c("correlation"),plot=TRUE)
```



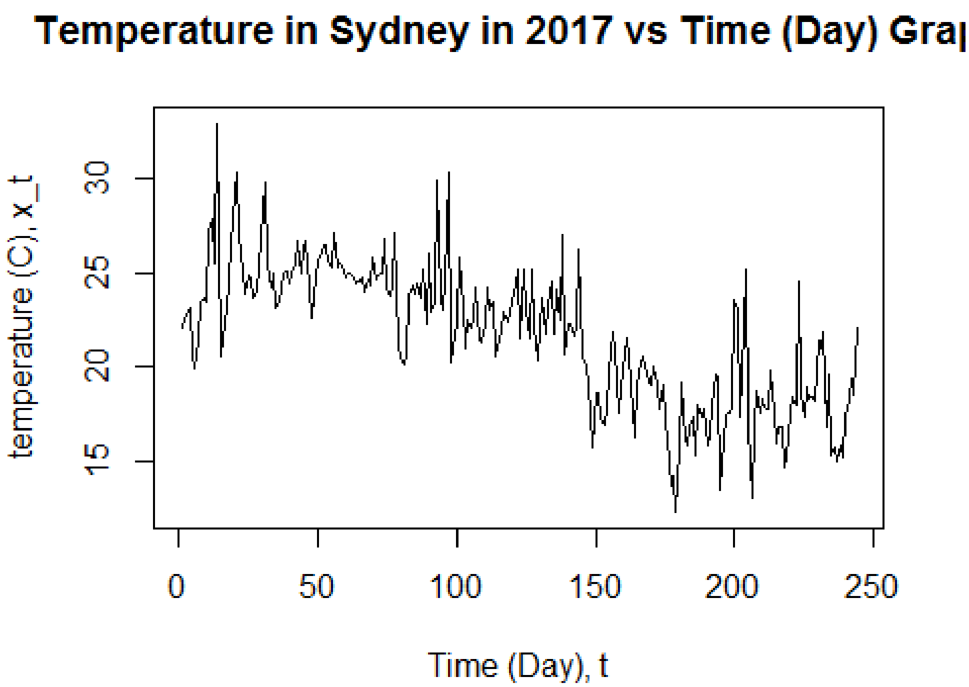

```
#-----------------------------------------------------------------------------------------------
---------------------------


# Second time series is the temperature data on 2017 from the Australian government
website
```

```r
#dataURL2 <- 'http://www.bom.gov.au/tmp/cdio/IDCJAC0010_066196_2017.pdf'
dataURL2 <- 'http://www.bom.gov.au/tmp/cdio/IDCJAC0010_066196_2016.pdf'

# Extract the table
out <- extract_tables(dataURL2)

df_temperature <- do.call(rbind, out[-length(out)])

# table data start at second row
df_temperature <- as.data.frame(df_temperature[2:nrow(df_temperature), ],
stringsAsFactors=F)
levels(droplevels((df_temperature[,]))) # drop levels from the dataframe
```

```
## NULL
```

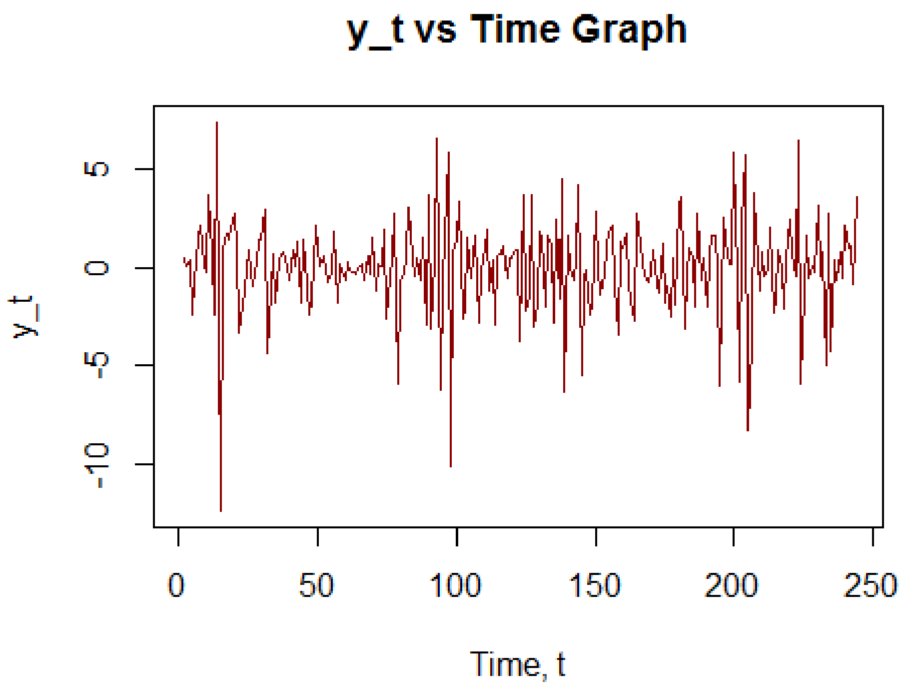
```r
# Column names
headers <- c('2017', 'Jan', 'Feb', 'Mar', 'Apr', 'May','Jun',
             'Jul', 'Aug', 'Sep','Oct','Nov','Dec')

# Apply custom column names
names(df_temperature) <- headers

# Create a new dataframe without the unnecessary data
# Data from 1 January 2016 to 31 August 2016 (Day 1 to Day 244)
# 2016 is a leap year - 29 days in Feb
maxDay <- 244
df_temperature_main <- data.frame(t = seq(1, maxDay, by = 1))

#current row of temperature for the new main dataframe
currTempRow <- 1

# goes from Jan column to Aug column
for(currCol in 2:9) {
  # iterate through row 1 to 31 and put the data into new main dataframe
  for (currRow in 1:nrow(df_temperature)){
    if ((df_temperature[currRow,currCol]!= "") & currRow <= 31){
      df_temperature_main$x_t[currTempRow] <- df_temperature[currRow,currCol]
      currTempRow = currTempRow + 1
    }
    else if ((currCol == 2 || currCol == 4 || currCol == 6 || currCol == 8 || currCol ==
9)
             & (df_temperature[currRow,currCol] == "") & currRow <= 31){
      df_temperature_main$x_t[currTempRow]  <- NA
      currTempRow = currTempRow + 1
    }
    else if ((currCol == 5 || currCol == 7) & (df_temperature[currRow,currCol] == "") &
currRow <= 30){
      df_temperature_main$x_t[currTempRow]  <- NA
      currTempRow = currTempRow + 1
    }
    else if ((currCol == 3) & (df_temperature[currRow,currCol] == "") & currRow <= 29){
      df_temperature_main$x_t[currTempRow]  <- NA
      currTempRow = currTempRow + 1
    }
  }
```

```
}

t <- df_temperature_main$t[1:maxDay]
temperature <- df_temperature_main$x_t[1:maxDay]

#convert the temperature data from character type to numeric type
df_temperature_main$x_t = as.numeric(as.character(df_temperature_main$x_t))

# Plot the original series, x_t
plot(t,temperature, type = "l", main="Temperature in Sydney in 2017 vs Time (Day) Graph",
xlab="Time (Day), t", ylab="temperature (C), x_t")
```

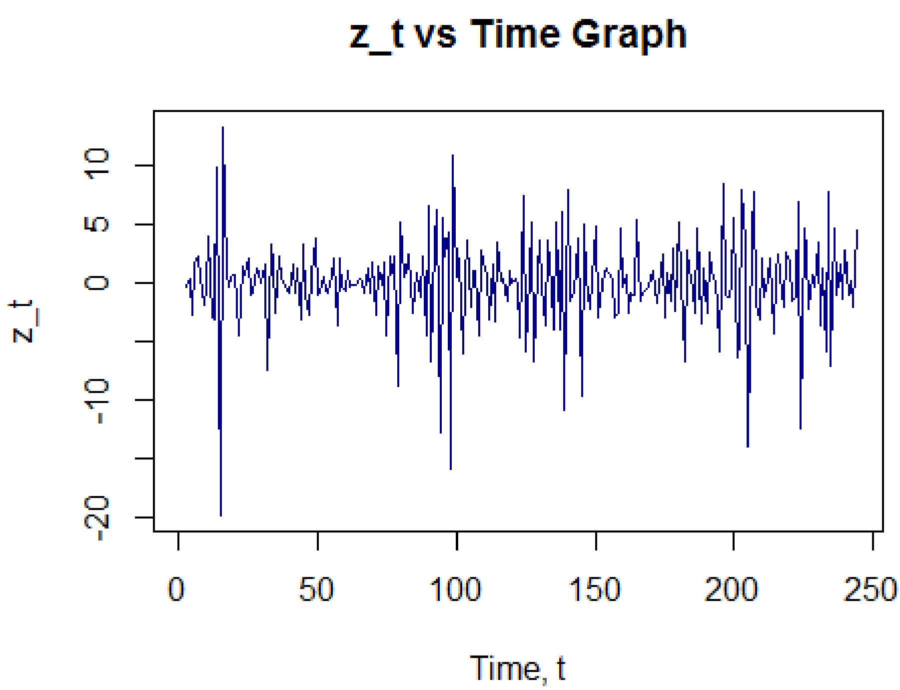



```
# Problem 1-b: Plot y_t after removing the seasonality in x_t
df_temperature_main<- slide(df_temperature_main,"x_t", "t", NewVar="xtLag1", slideBy = -
1)

##
## Lagging x_t by 1 time units.

# removing the seasonality from x_t
for (i in 1:length(t)){
  if (!is.na(df_temperature_main$x_t[i]) & !is.na(df_temperature_main$xtLag1[i])){
    df_temperature_main$y_t[i] <- df_temperature_main$x_t[i] -
df_temperature_main$xtLag1[i]
  }
  else{
    df_temperature_main$y_t[i] <- NA
  }
}

# plot yt vs t
y_t <- df_temperature_main$y_t
```

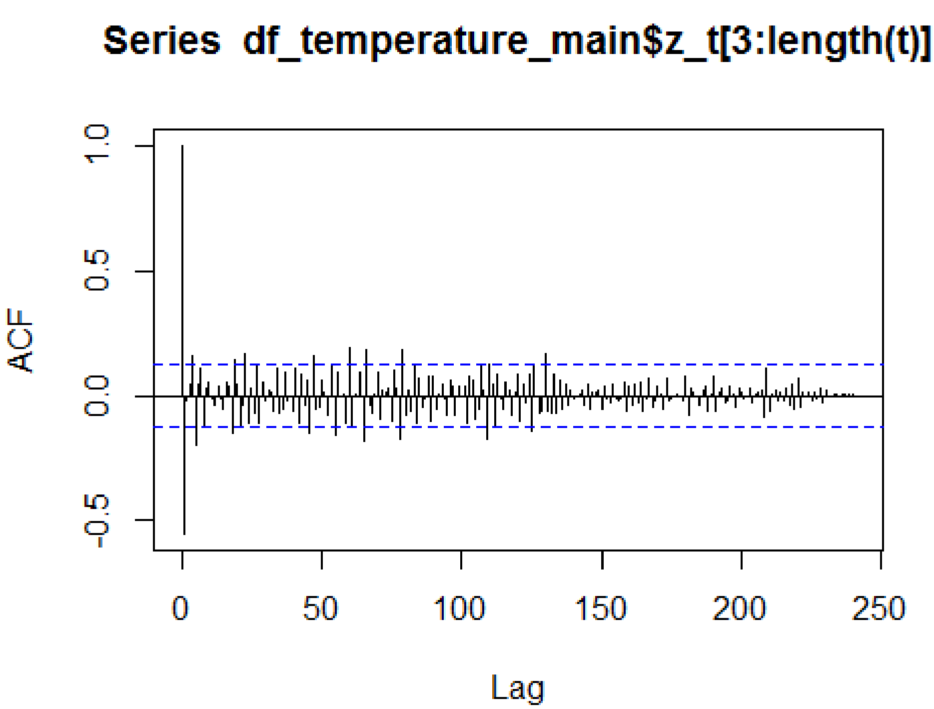
```
t <- df_temperature_main$t
plot(t, y_t, type = "l", col= colors()[100], main="y_t vs Time Graph", xlab="Time, t",
ylab="y_t")
```




```
# Problem 1-c: Plot z_t after removing the trend in y_t

df_temperature_main <- slide(df_temperature_main,"y_t", "t", NewVar="ytLag1", slideBy = -
1)

##
## Lagging y_t by 1 time units.

# removing the trend in y_t
for (i in 1:length(t)){
  df_temperature_main$z_t[i] <- df_temperature_main$y_t[i] -
df_temperature_main$ytLag1[i]
}

# plot zt vs t
z_t <- df_temperature_main$z_t
t <- df_temperature_main$t
plot(t, z_t, type = "l", col= colors()[30], main="z_t vs Time Graph", xlab="Time, t",
ylab="z_t")
```

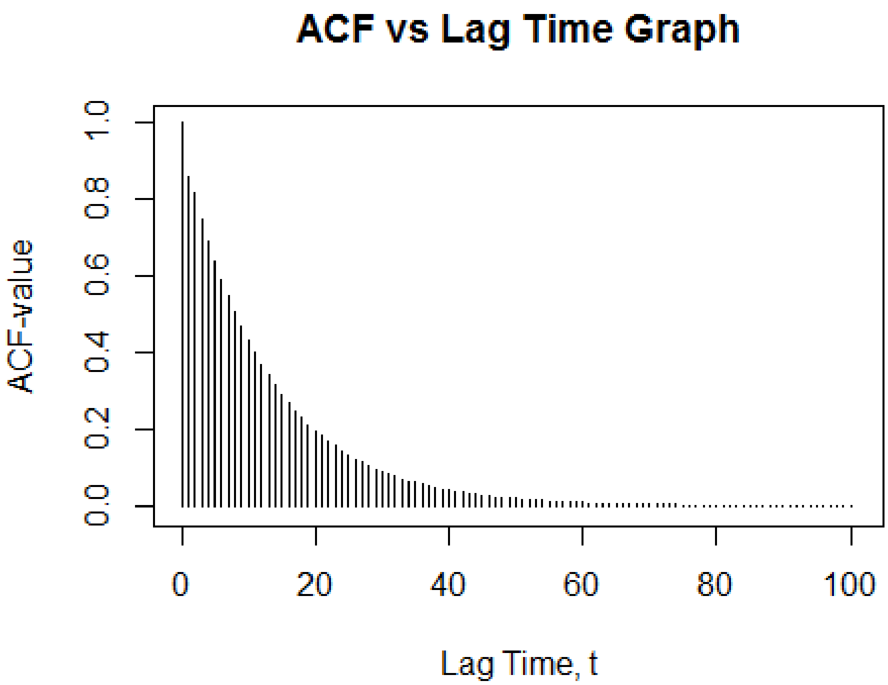



```
#Problem 1-d: autocorrelation, ACF
acf (df_temperature_main$z_t[3:length(t)], lag.max = length(t),
type=c("correlation"),plot=TRUE)
```




```
#-----------------------------------------------------------------
# END OF PROBLEM #2
#-----------------------------------------------------------------


#-----------------------------------------------------------------
```

```r
# Problem 3-b
# from equation 0.6y_t-1 + 0.3y_t-2 + epsilon_t
# we calculated ACF value from the derived function of AR(2) model
# and then plot those values
#-------------------------------------------------------------------
# Create time sequences from 0 to 10
mainDf3 <- data.frame(t = seq(0, 100, by = 1))

# set constants for phi1 and phi2
# from equation 0.6y_t-1 + 0.3y_t-2 + epsilon_t
phi_1 <- 0.6
phi_2 <- 0.3

# calculate and plot ACF-value from given constants
# ACF value at lag = 0
rho_0 <- 1
# ACF value at lag = 1
rho_1 <- phi_1 / (1-phi_2)
# ACF value at lag = 2
rho_2 <- (phi_1 * rho_1) + phi_2

# put those calculated values to data frame
# and then plot a line for the current acf data
mainDf3$rho_t[1] <- rho_0
x <- c(mainDf3$t[1],mainDf3$t[1])
y <- c(0,mainDf3$rho_t[1])
plot(x,y, type = "l", main="ACF vs Lag Time Graph", xlab="Lag Time, t", ylab="ACF-value",
     xlim=c(0,length(mainDf3$t)), ylim=c(-0.01, max(mainDf3$rho_t)))

mainDf3$rho_t[2] <- rho_1
x <- c(mainDf3$t[2],mainDf3$t[2])
y <- c(0,mainDf3$rho_t[2])
lines(x,y, type = "l")

mainDf3$rho_t[3] <- rho_2
x <- c(mainDf3$t[3],mainDf3$t[3])
y <- c(0,mainDf3$rho_t[3])
lines(x,y, type = "l")

# Create the rest of the table for rho_t
for(i in 4:length(mainDf3$t)){
  # Calculate data of rho_t
  # ACF value at current lag t value
  mainDf3$rho_t[i] <- (phi_1 * mainDf3$rho_t[i-1]) + (phi_2 * mainDf3$rho_t[i-2])
  # plot a line for current acf data
  x <- c(mainDf3$t[i],mainDf3$t[i])
  y <- c(0,mainDf3$rho_t[i])
  lines(x,y, type = "l")
}
```




```
#------------------------------------------------------------------
# END OF PROBLEM #3
#------------------------------------------------------------------
```