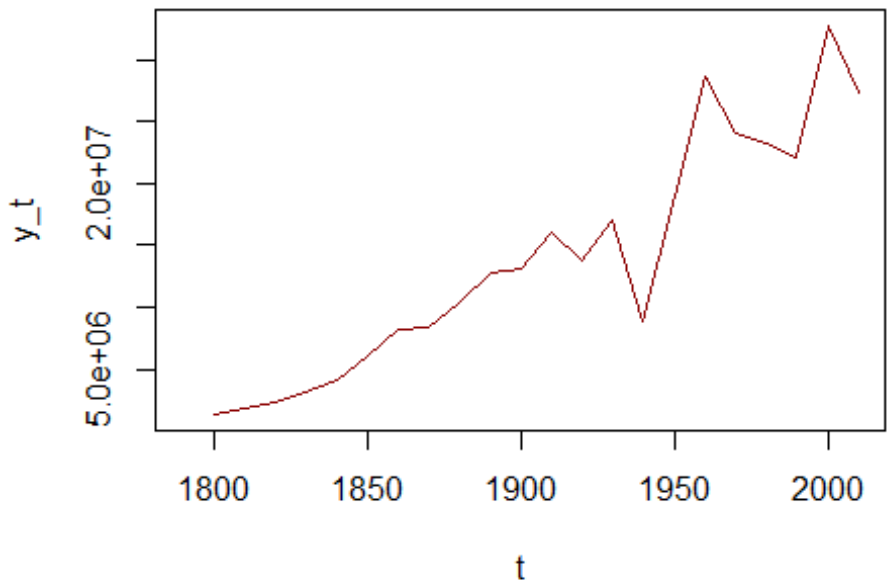
TimeSeries-HW3

Lince Rumainum

February 4, 2019

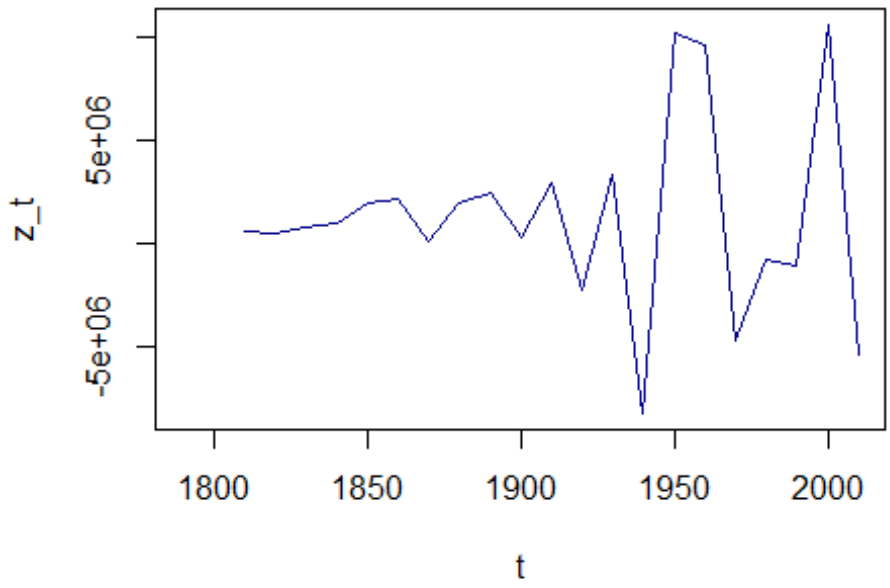
```
# Lince Rumainum
# Time Series Analysis
# HW3

# libraries list
# install.packages("DataCombine")
# install.packages("nlme")
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")'.

####################################
# Problem 1
####################################
mainDf1<- data.frame(t = seq(1, 23, by = 1))

population <- list(3929214,5308483,7239881,9638453,12866020,
  17069453,23191876,31443321,39818449,50155783,62947714,
  75994575,91972266, 105710620,122775046,131669275,
  150697361,179323175,203302031,226545805,248709873,
  281421906,308745538)

mainDf1$year[1] <- 1790
mainDf1$x_t [1] <- population[1]

#create data frame with year and population data
for (i in 2:23){
  mainDf1$year[i] <- mainDf1$year[i-1]+10
  mainDf1$x_t [i] <- population[i]
}

# create new column for the x lag of t-1
#do.call(rbind.data.frame, mainDf1$x_t)
mainDf1 <- as.data.frame(lapply(mainDf1, unlist))
mainDf1 <- slide(mainDf1,"x_t", "t", NewVar="xtLag1", slideBy = -1)

##
## Lagging x_t by 1 time units.
```

```r
for (i in 1:23){
  if (!is.na(mainDf1$x_t[i]) & !is.na(mainDf1$xtLag1[i])){
    mainDf1$y_t[i] <- mainDf1$x_t[i] - mainDf1$xtLag1[i]
  }
  else{
    mainDf1$y_t[i] <- NA
  }
}

# The y_t function is the difference operator of the US population
# It calculates the detrending of the population by using the one step lag
# It takes the different between two adjacent population data and see what
type
# of trend it creates. In this case, y_t creates a linear trend.

# plot yt vs t
y_t <- mainDf1$y_t
t <- mainDf1$year
plot(t, y_t, type = "l", col= colors()[100])
```

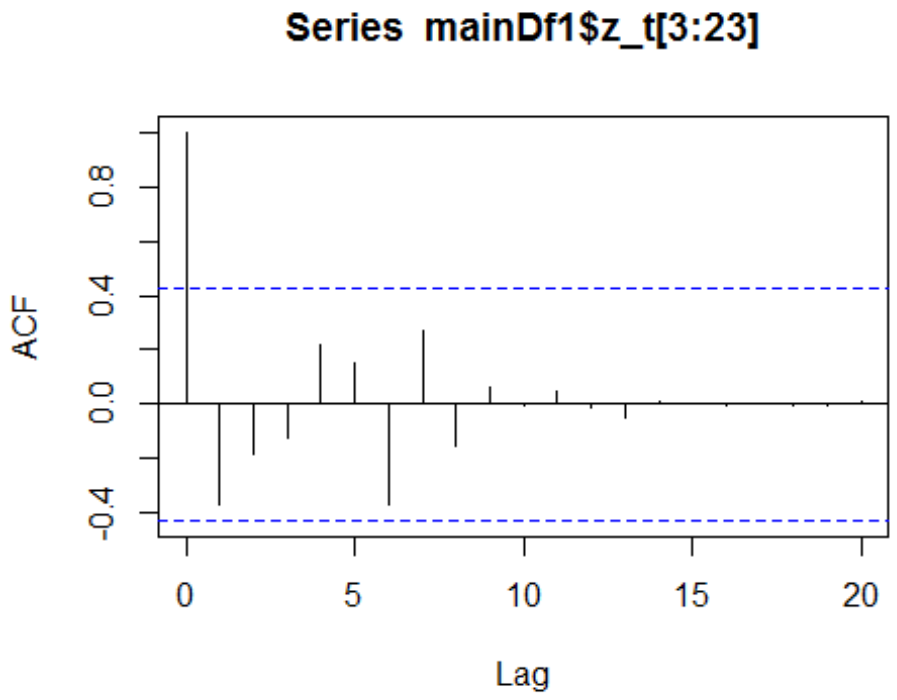



```r
mainDf1 <- slide(mainDf1,"y_t", "t", NewVar="ytLag1", slideBy = -1)

##
## Lagging y_t by 1 time units.

for (i in 1:23){
  mainDf1$z_t[i] <- mainDf1$y_t[i] - mainDf1$ytLag1[i]
```

```
}

# The z_t function is removing the linear trend of y_t.
# It calculates the detrending of y_t by using the one step lag.
# It takes the different between two adjacent y_t and see what type
# of trend it creates. In this case, z_t close to eliminate the linear trend.
# As you can see from the z_t vs t plot, it detrend y_t and fluctuates around
zero.

# plot zt vs t
z_t <- mainDf1$z_t
t <- mainDf1$year
plot(t, z_t, type = "l", col= colors()[30])
```

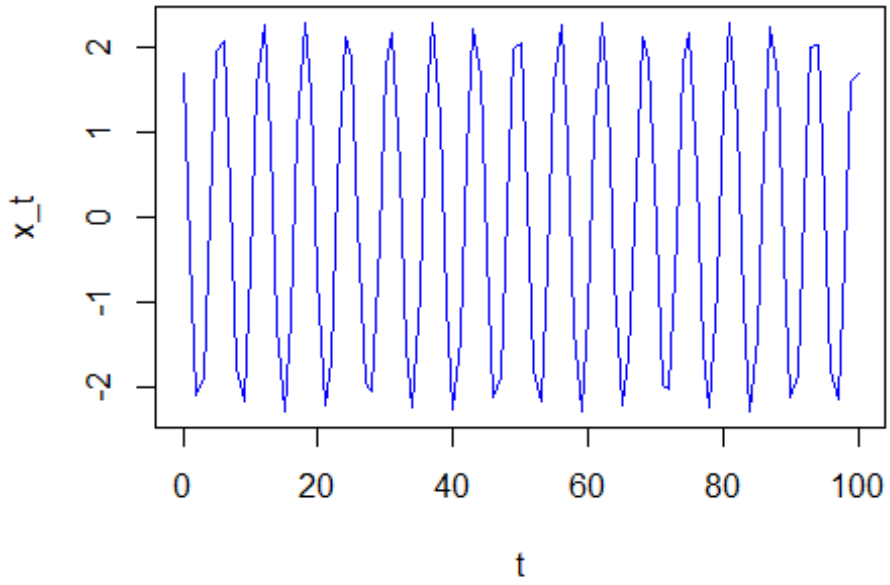



```
#mean, variance, and autocorrelation
mean(mainDf1$z_t,na.rm=TRUE) #na.rm remove NA value

## [1] 1235446

var(mainDf1$z_t,na.rm=TRUE)

## [1] 2.190531e+13

acf (mainDf1$z_t[3:23], lag.max = 23, type=c("correlation"),plot=TRUE)
```



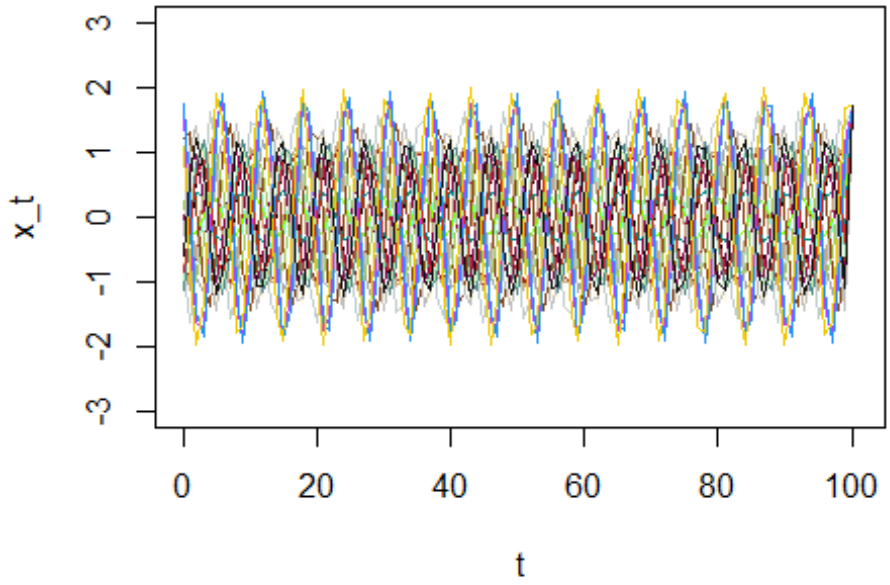

```
####################################
# Problem 2
####################################

# var for 100 samples
sampleSize = 100

# Create a main data frame for problem 2 with i = 0 to 100 and a, b ~ iid
N(0,1) values
mainDf2 <- data.frame(t = seq(0, sampleSize, by = 1))
mainDf2$a <- rnorm(sampleSize + 1, mean = 0, sd = 1)
mainDf2$b <- rnorm(sampleSize + 1, mean = 0, sd = 1)

# Obtain 100 estimates of x_t
for(k in mainDf2$t){
  # Calculate the x_t
  #mainDf2$x_t[k] <-
(mainDf2$a[k]*cos(mainDf2$t[k]))+(mainDf2$b[k]*sin(mainDf2$t[k]))
  #pick one constant for a and b to plot it over t
  mainDf2$x_t[k] <-
(mainDf2$a[2]*cos(mainDf2$t[k]))+(mainDf2$b[2]*sin(mainDf2$t[k]))
}

# plot for x_t vs t for 0 <= t <= 100
t <- mainDf2$t
```

```r
x_t <- mainDf2$x_t
plot(t, x_t, type = "l", col = "blue")
```




```r
# plot 20 new x_t in the same plot
for (s in 1:20){
  # Create 100 new samples of a and b ~ iid N(0,1)
  mainDf2$a <- rnorm(sampleSize + 1, mean = 0, sd = 1)
  mainDf2$b <- rnorm(sampleSize + 1, mean = 0, sd = 1)

  # Calculate the x_t
  for(k in mainDf2$t){
    #mainDf2$x_t[k] <-
(mainDf2$a[k]*cos(mainDf2$t[k]))+(mainDf2$b[k]*sin(mainDf2$t[k]))
    mainDf2$x_t[k] <-
(mainDf2$a[s]*cos(mainDf2$t[k]))+(mainDf2$b[s]*sin(mainDf2$t[k]))
  }

  # plot all in the same plot with different colors
  t <- mainDf2$t
  x_t <- mainDf2$x_t
  if (s == 1){
    plot(t, x_t, type = "l", col= colors()[8*s], xlim=c(0,length(t)),
ylim=c(-3,3))
  }
  else{
    lines(t, x_t, type = "l", col= colors()[8*s], xlim=c(0,length(t)),
ylim=c(-3,3))
```

```
  }
}
```


x_t
3
2
1
0
-1
-2
-3
0
20
40
60
80
100
t