Thesis.code

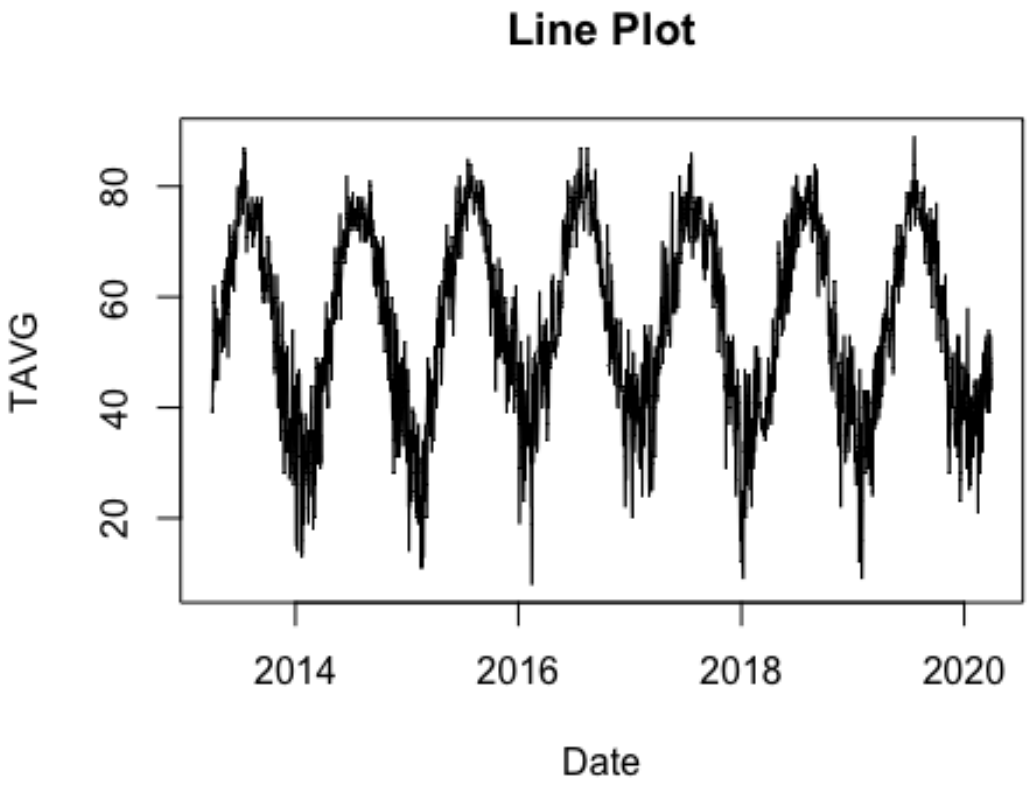
2023-06-29

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:

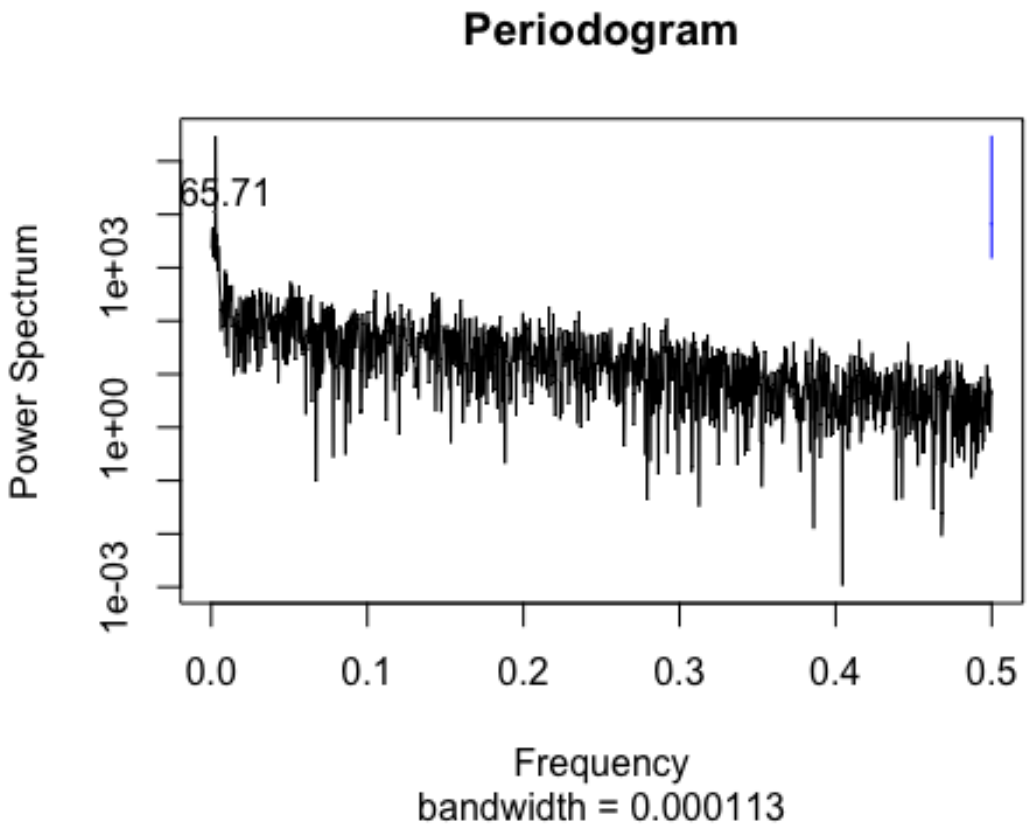
```
knitr::opts_chunk$set(warning = FALSE)
library(lmtest)
library(ggplot2)
library(forecast)
library(readxl)
library(rnoaa)
library(dplyr)
library(stats)
library(lubridate)
library(PerformanceAnalytics)
library(tseries)
library(rugarch)
library(quantmod)
library(TTR)
library(tidyverse)
library(xts)

#set environment
getwd()

## [1] "/Users/nicholasalgie/Downloads"

#read New york daily temperature CSV file from NOAA
T <- read.csv("~/Downloads/3367577.csv", header= TRUE, sep = , )
T <- as.data.frame(T)
#TAVG available only for 2013, therefore we are forced to use a smaller sampl
e
TAVG2013 <- T[T$DATE > "2013-04-01" & T$DATE < "2020-04-01"
              , c("DATE", "TAVG", "TMIN", "TMAX")]
TAVG2013<-na.omit(TAVG2013)
TAVG2013$DATE <- as.Date(TAVG2013$DATE)
#after making sure that there a
plot(TAVG2013$DATE, TAVG2013$TAVG, type = "l", xlab = "Date", ylab = "TAVG",
     main = "Line Plot")
```




```r
#We need to make a smaller data frame as well, as we will need it later
Temp<-TAVG2013[c("DATE","TAVG")]
Temp<-as.data.frame(Temp)
Temp$DATE<-as.Date(Temp$DATE)
# Determine Seasonality in Data
  # Remove linear trend
  TAVG2013$Detrended <- residuals(lm(TAVG2013$TAVG ~ TAVG2013$DATE + 0))

  # Compute the periodogram
  spectrum <- spec.pgram(TAVG2013$Detrended, pad = 0, log = "no",
                         detrend = TRUE, plot = TRUE)
```

```r
  # Identify the top two component frequencies and periods
  topFrequencies <- spectrum$freq[order(spectrum$spec, decreasing = TRUE)][1:
2]
  topPeriods <- 1 / topFrequencies

  # Plot the power spectrum
  plot(spectrum, type = "l", xlab = "Frequency", ylab = "Power Spectrum",
       main = "Periodogram")
  abline(v = topPeriods, lty = 2, col = "red")
```

```
  text(topFrequencies, spectrum$spec[which(spectrum$freq %in% topFrequencies)
],
       round(topPeriods, 2), pos = 3)
```



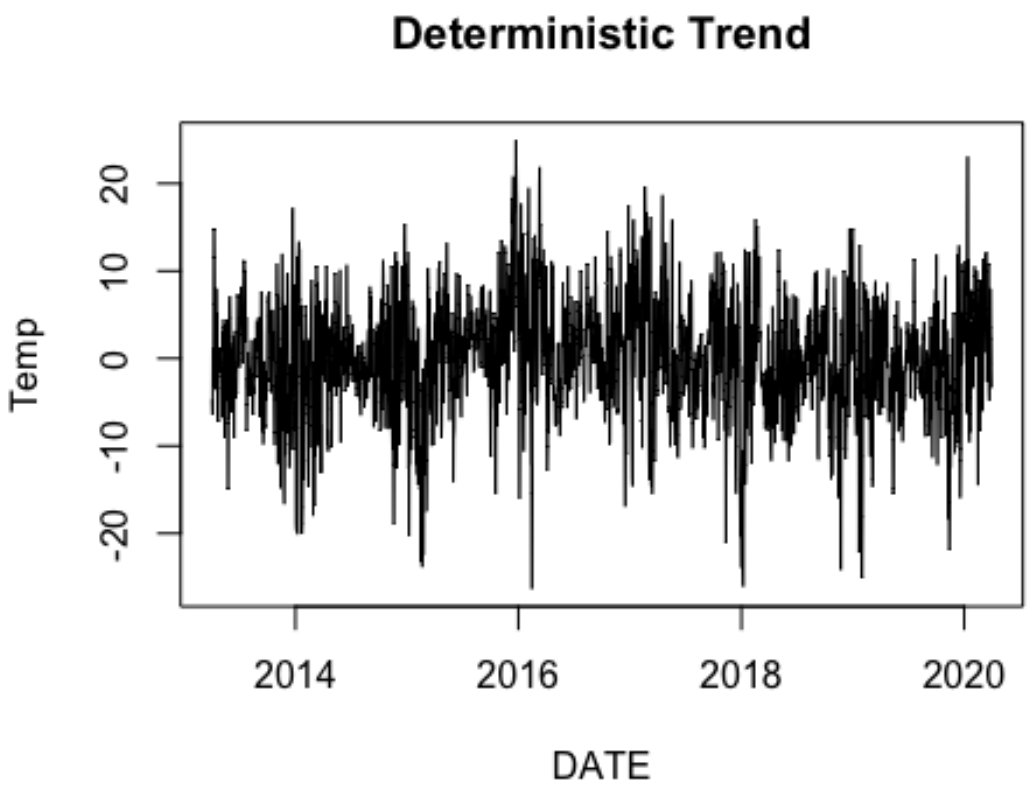

```
print (topFrequencies)

## [1] 0.002734375 0.001953125

print (topPeriods)

## [1] 365.7143 512.0000

#Fit Deterministic Trend
#first we apply the elapsed time by using the difftime function
elapsedTime <- as.numeric(difftime(TAVG2013$DATE, TAVG2013$DATE[1],
                                   units = "days"))/365
#second we create the parameters functions (#) in the R environment
designMatrix <- function(t) {
  matrix(c(t, cos(2 * pi * t), sin(2 * pi * t), cos(4 * pi * t)), ncol = 4)
}
trendPreds <- c("t", "cos(2*pi*t)", "sin(2*pi*t)", "cos(4*pi*t)")
trendModel <- lm(TAVG2013$TAVG ~ designMatrix(elapsedTime))
print (trendModel)
```

```
##
## Call:
## lm(formula = TAVG2013$TAVG ~ designMatrix(elapsedTime))
##
## Coefficients:
##                (Intercept)  designMatrix(elapsedTime)1
##                   54.45701                     0.23548
## designMatrix(elapsedTime)2  designMatrix(elapsedTime)3
##                   -9.72608                    19.50138
## designMatrix(elapsedTime)4
##                   -0.02627
```

```
summary (trendModel)
```

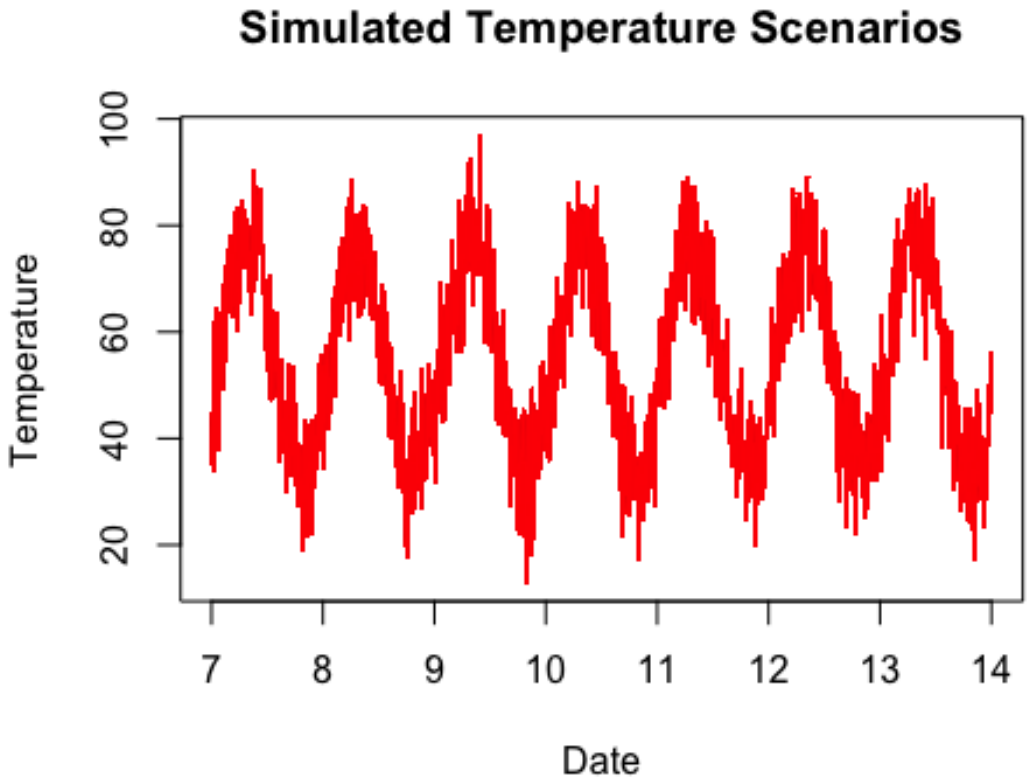
```
##
## Call:
## lm(formula = TAVG2013$TAVG ~ designMatrix(elapsedTime))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -26.3096  -3.9461  -0.0279   4.1423  24.9095
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 54.45701    0.26162 208.153  < 2e-16 ***
## designMatrix(elapsedTime)1  0.23548    0.06483   3.632 0.000286 ***
## designMatrix(elapsedTime)2 -9.72608    0.18415 -52.817  < 2e-16 ***
## designMatrix(elapsedTime)3 19.50138    0.18537 105.202  < 2e-16 ***
## designMatrix(elapsedTime)4 -0.02627    0.18415  -0.143 0.886571
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.584 on 2551 degrees of freedom
## Multiple R-squared:  0.8452, Adjusted R-squared:  0.8449
## F-statistic:  3481 on 4 and 2551 DF,  p-value: < 2.2e-16
```

```
#the coefficients are all signifcant but the last, which is consistent
trendModel$coefficients <- coef(summary(trendModel))[, "Estimate"]
trendModel$residuals <- residuals(trendModel)
trendModelplot<- plot(Temp$DATE , trendModel$residuals, type = "l",
                      xlab = "DATE", ylab = "Temp",
                      main = "Deterministic Trend")
```



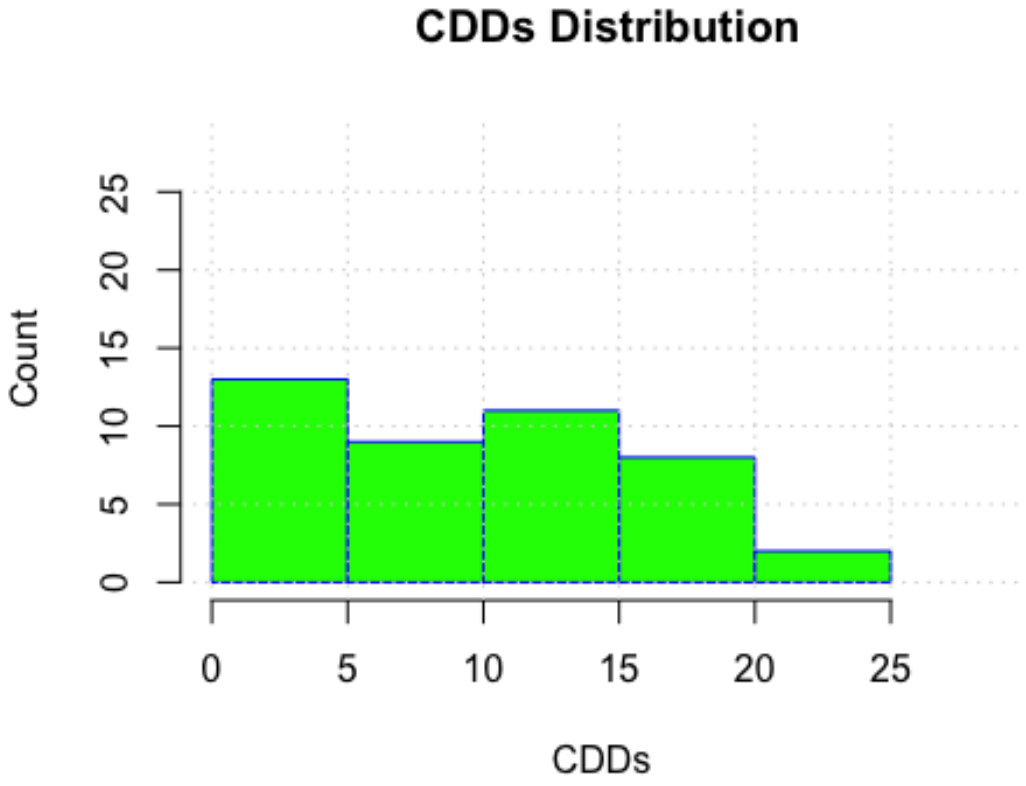

```
#we can notice by eye that the plot shows that the temp is slightly increasin
g
#we can also notice the increase of rare events

# Analyze and Fit Model Residuals
trendRes <- trendModel$residuals
#Fit the ARMA model
esModel <- arima(trendRes, order = c(1, 0, 1))

# Prepare the simulation time
StartDate<- "2020-04-01"
StartDate<- as.Date(StartDate)
sim_dates <- seq.Date(StartDate, by = "day", length.out = 2556)


#simulate time
simTime <- as.numeric(difftime(sim_dates,Temp$DATE[1], units = "days") / 365)
# Predict the deterministic trend
trendPred <- predict(trendModel, newdata = data.frame(Time = simTime))
SimModel<-cbind(simTime,trendPred)
# Simulate using the ARIMA model
simRes <- simulate(esModel, nsim = 2556, startMethod = "user",
                   startPos = as.vector(trendRes), n.ahead = nDays)
```

```r
# Add the deterministic trend to the simulated residuals
simTemp <- simRes + trendPred

# Visualize the temperature scenarios
plot(simTime, simTemp, type = "l", xlab = "Date", ylab = "Temperature",
     main = "Simulated Temperature Scenarios")
for (i in 2:1000) {
  lines(simTime, simTemp, col = "red", alpha = 0.3)
}
```




```r
#Now we can do the Montecarlo sim. with sim. temperature
#But first we must subset the temp. for the dates we want
simTemp<- cbind.data.frame(simTemp)
NotSimDate<-"2020-04-01"
NotSimDate<-as.Date(NotSimDate)
future_date <- NotSimDate + days(2555)
DatesCHD <- seq(as.Date("2020-04-01"), as.Date("2027-03-31"), by = "day")
DatesCHD<- data_frame(DatesCHD)
Temp.Dates<- cbind.data.frame(simTemp,DatesCHD)
Sim.Temp.NYCHD.Days<- Temp.Dates[ 448:490,]
#create histogram of CDD from simulation
```

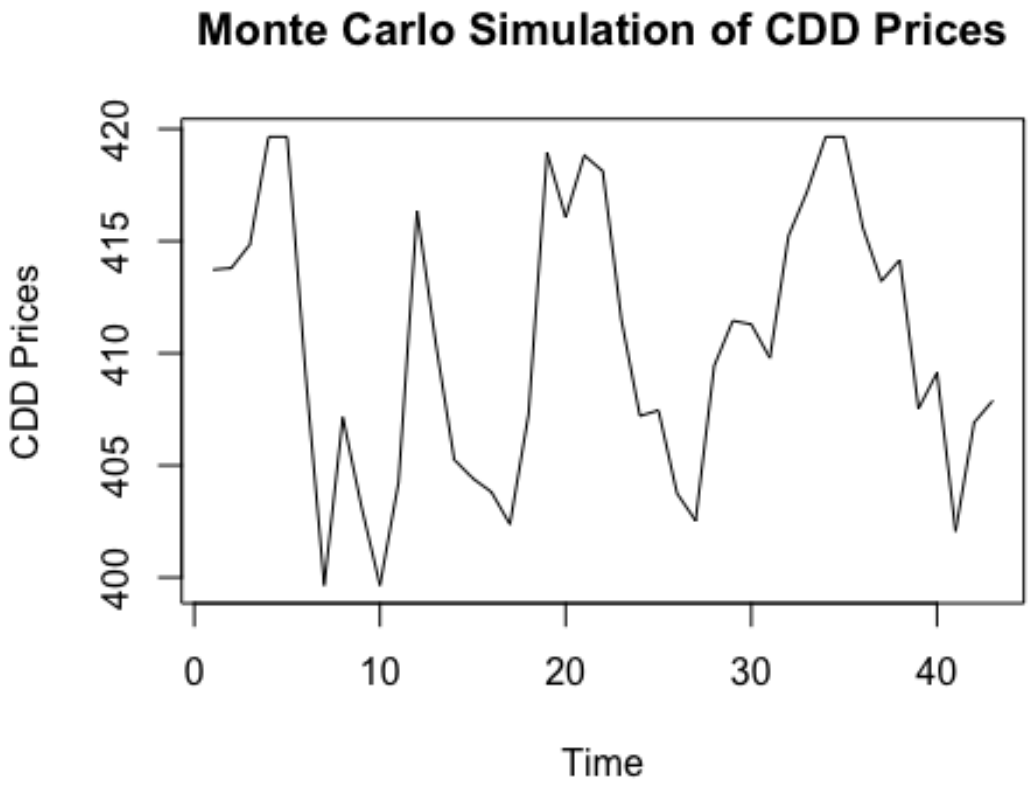
```r
C <- pmax(Sim.Temp.NYCHD.Days$simTemp-65, 0)
hist(C, xlab = "CDDs", ylab = "Count", main = " CDDs Distribution",
     col = "green", border = "blue", xlim = c(0, max(C) + 5), ylim = c(0, max
(C) + 5))
grid()
```




```r
#with the C we
muC<-mean(C)
sigmaC<- StdDev(C)
CDDs<-sum(pmax(Sim.Temp.NYCHD.Days$simTemp-65, 0))

# Set the random seed for reproducibility
set.seed(40)

# Define the number of simulations
num_simulations <- 1000


# Calculate the CDD price using a formula and tick/strike price
library(MASS)

K <- 20 # Strike value
a s
```

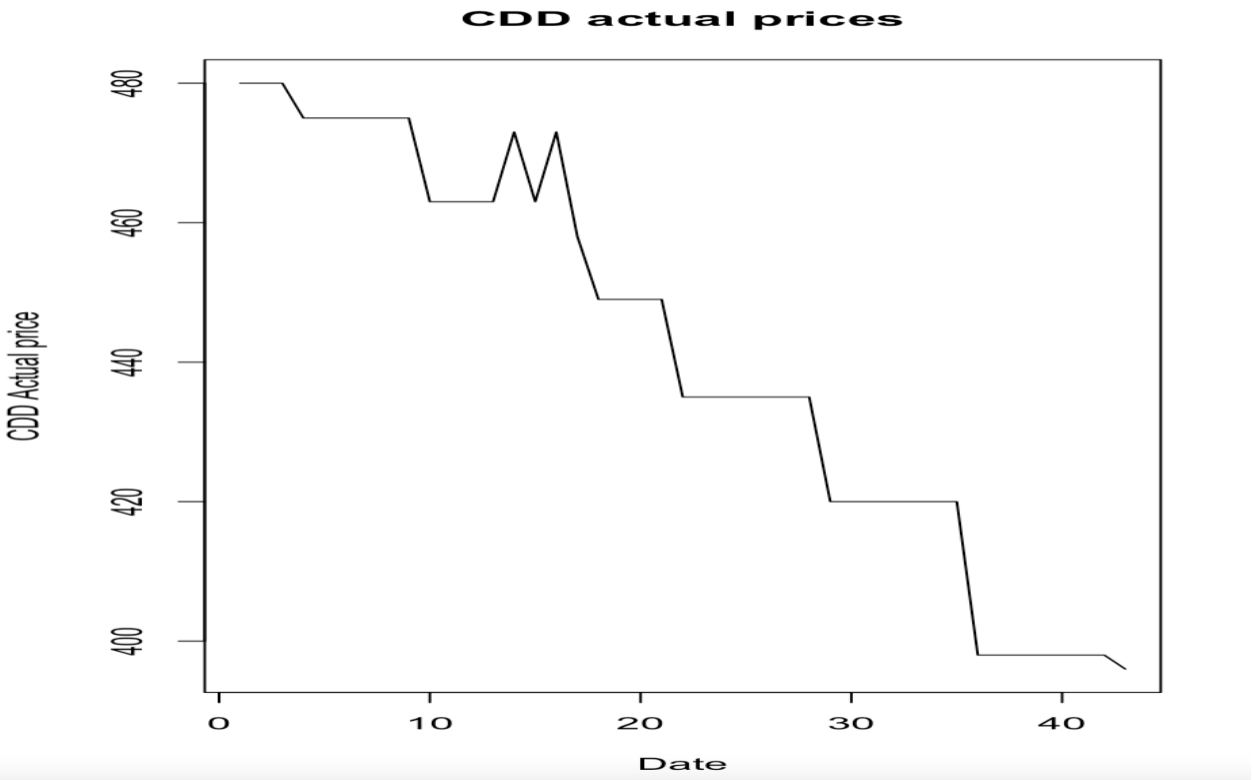
```r
# Compute the price of the put option for 1$ of investment
cdd_payoff <- pmax(K-C,0)
#now to estimate the price we sum the payoff to the base simulated CDDs
CDD_price <-CDDs+cdd_payoff
#now we have the simulated prices for CDD, now we want to repeat this 1000 ti
mes
#find the averages of CDDs and do the least squared error with the actual pri
ces


# Set the number of iterations and time horizon
nIterations <- 1000
nDays <- 43

# Create an empty matrix to store the simulated CDD prices
CDD_prices.m <- matrix(NA, nrow = nDays, ncol = nIterations)

# Simulate time using the trend model
startTime<-"2021-06-22"
startTime<-as.Date(startTime)
EndTime<- "2021-08-03"
EndTime<-as.Date(EndTime)
trendPred <- predict(trendModel, newdata = data.frame(Time = seq.Date(startTi
me,
                                                      by = "day", length.out = 43))
)

# Perform the Monte Carlo simulation
for (i in 1:nIterations) {

  simRes.m <- simulate(esModel, nsim = nDays, startMethod = "user",
                       startPos = as.vector(trendRes), n.ahead = nDays)
  simTemp.m <- simRes + trendPred

  C.m <- pmax(simTemp.m-65, 0)
  cdd_payoff.m <- pmax(K - C.m, 0)
  CDD.m<- sum(C)
  # Compute the CDD prices by summing the payoff to the base simulated CDDs
  CDD_prices.m[, i] <- CDD.m + cdd_payoff

}

# Plot the simulated CDD prices
time <- seq(1, nDays)
plot(time, rowMeans(CDD_prices.m), type = "l", xlab = "Time",
     ylab = "CDD Prices",
     main = "Monte Carlo Simulation of CDD Prices", ylim = c(min(CDD_prices.m
),
```

```
                                                                max(CDD_prices.m
)))
```




```
#find the average of the CDD prices simulated
average_CDD <- rowMeans(CDD_prices.m)

# Create a time series object
time <- seq(as.Date("2021-06-22"), by = "day", length.out = nDays)
average_CDDs.time<-cbind.data.frame(time,average_CDD)
average_CDDs.time$average_CDD<-as.numeric(average_CDDs.time$average_CDD)
#import the CDD actual data for july 2021

NY_CDD_july_2021_options <- read_excel("NY.CDD.july.2021. options.xlsx")

## New names:
## • `` -> `...2`
## • `` -> `...3`

View(NY_CDD_july_2021_options)
CDD_NY<-cbind.data.frame(NY_CDD_july_2021_options$...3)
CDD_NY<-CDD_NY[-1,]
CDD_NY<-as.data.frame(CDD_NY)
#do the LSD between the sim and the actual prices
average_CDDs.time$average_CDD<-as.numeric(average_CDDs.time$average_CDD)
```

```
CDD_NY<-unlist(CDD_NY)
CDD_NY<-as.numeric(CDD_NY)

plot(time, CDD_NY, type = "l", xlab = "Date", ylab = "CDD Actual price
", main = "CDD actual prices")
```




```
Squared_diff <- function(average_CDD) {
  diff <-  average_CDD - CDD_NY
  (sum(diff^2))/43
}

parameter.alfa <- optimise(Squared_diff, interval=c(-100, 100) )
print(parameter.alfa$minimum)

## [1] 99.99993
```