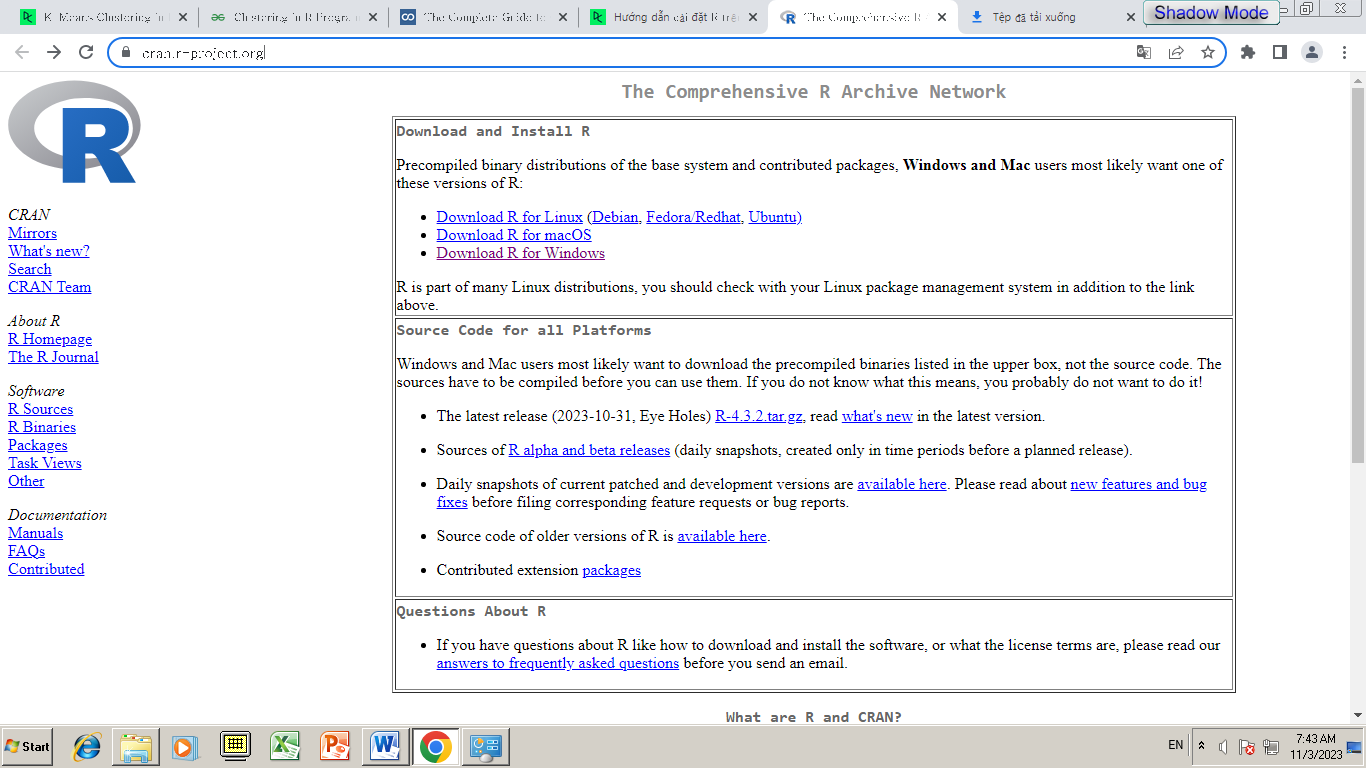
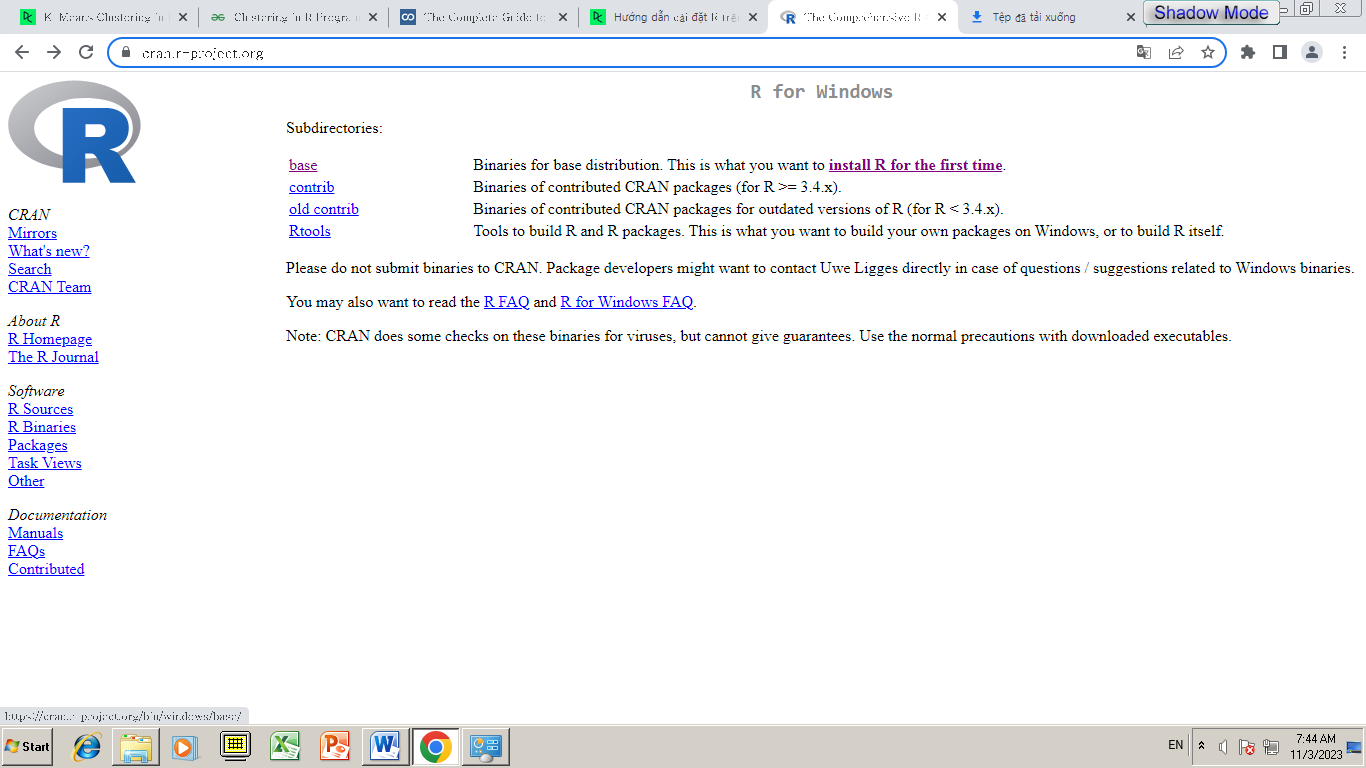
Đầu tiên là cài đặt R

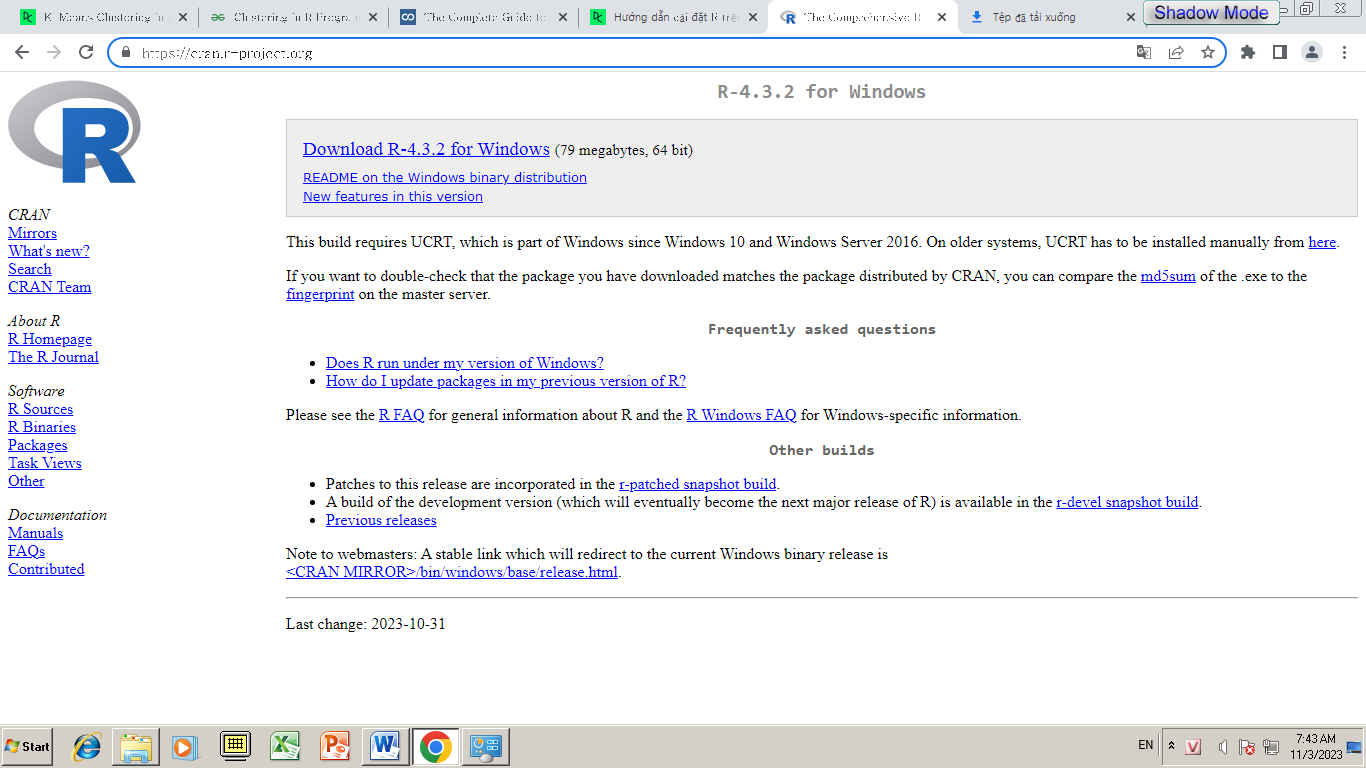
Vào trang <https://cran.r-project.org/>



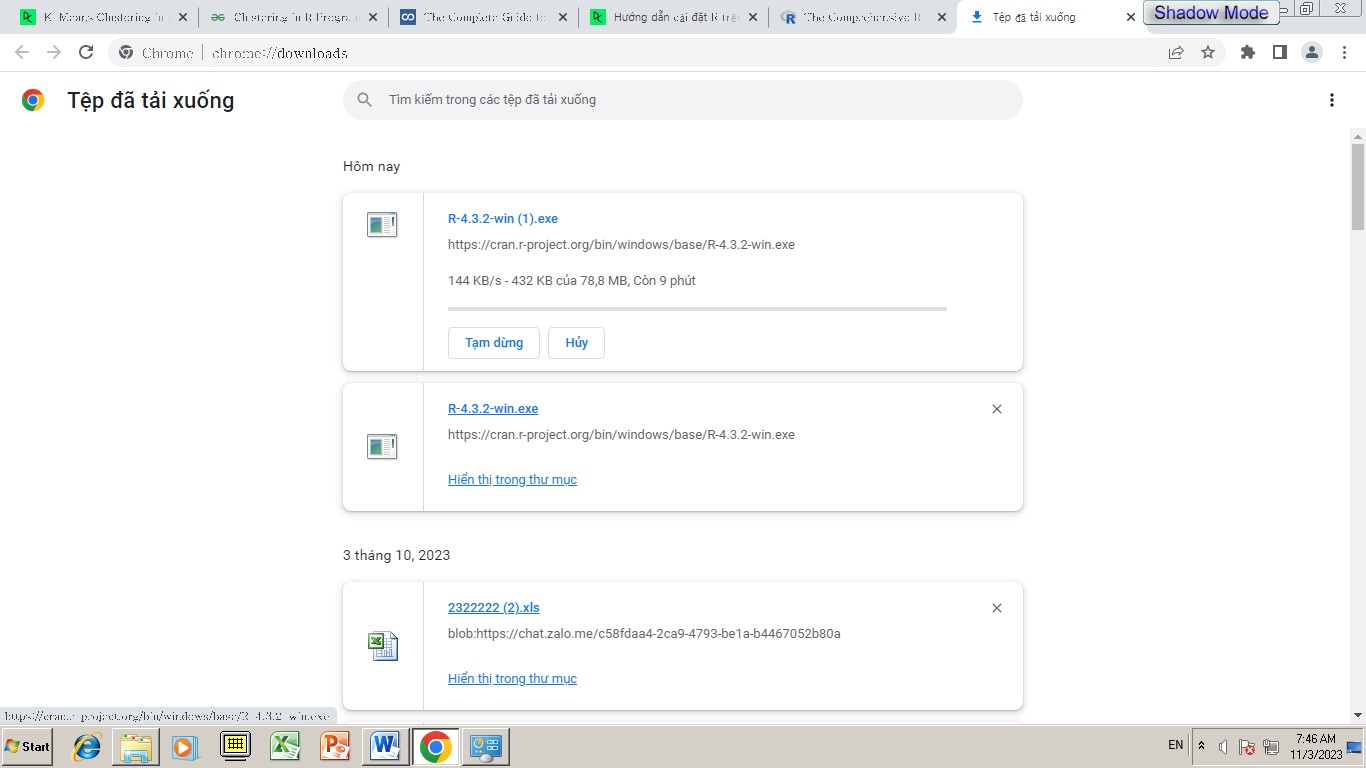
Chọn Download R for Windows



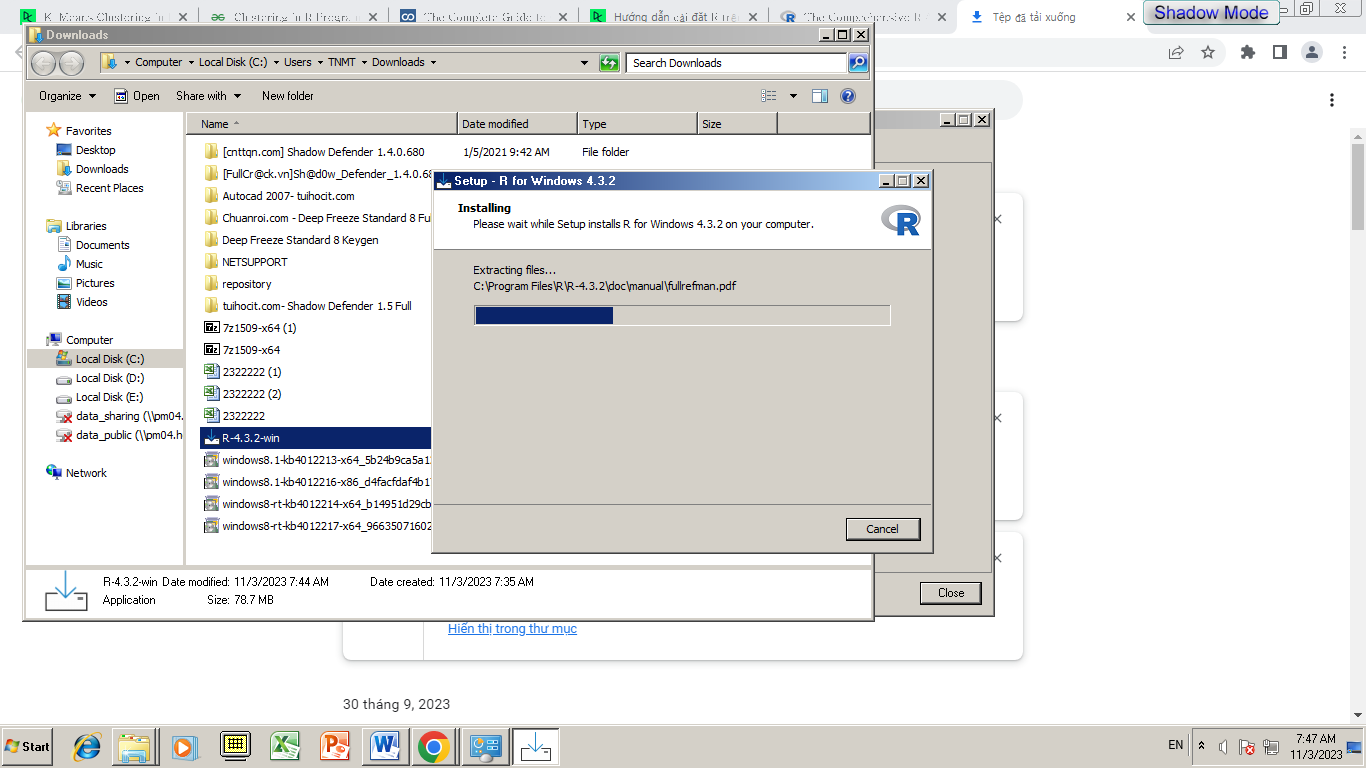
Chọn Install R for the first time



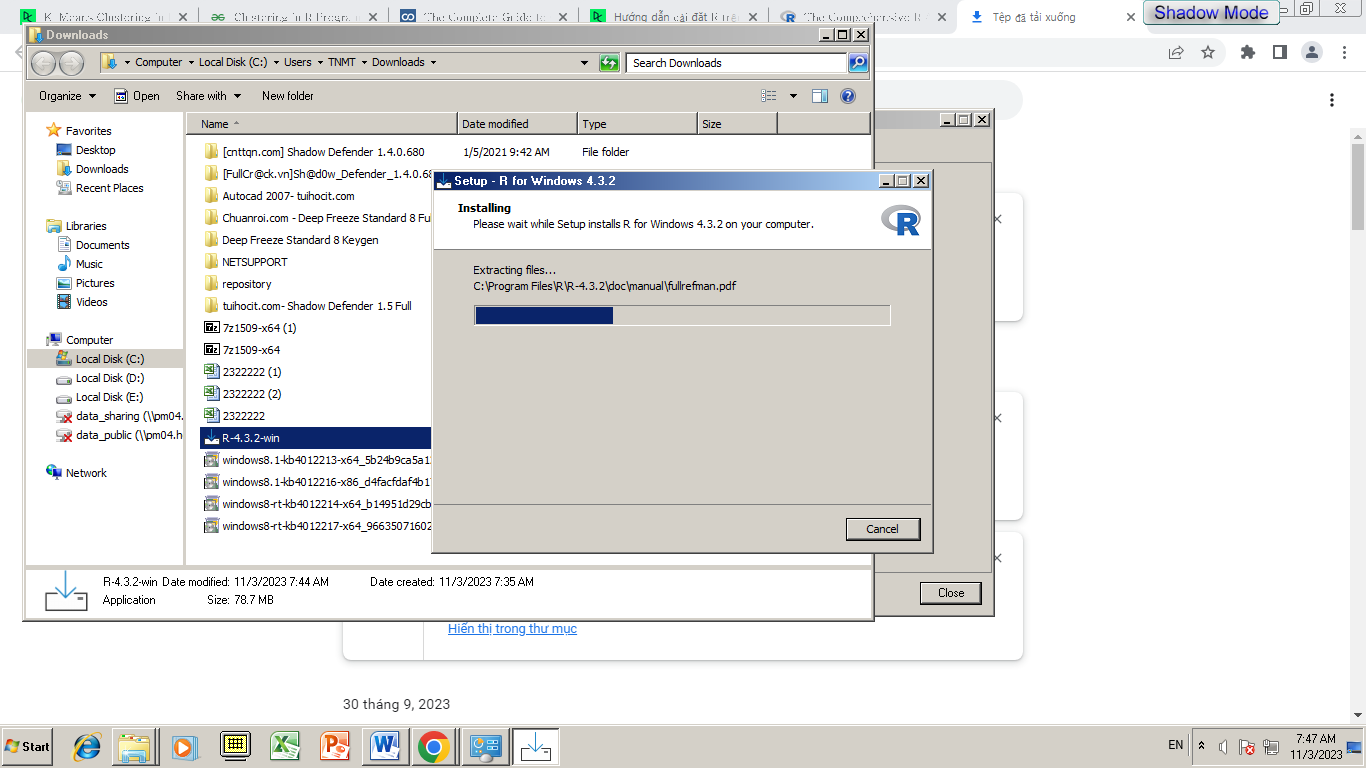
Bấm download R-4.3.2 for Windows

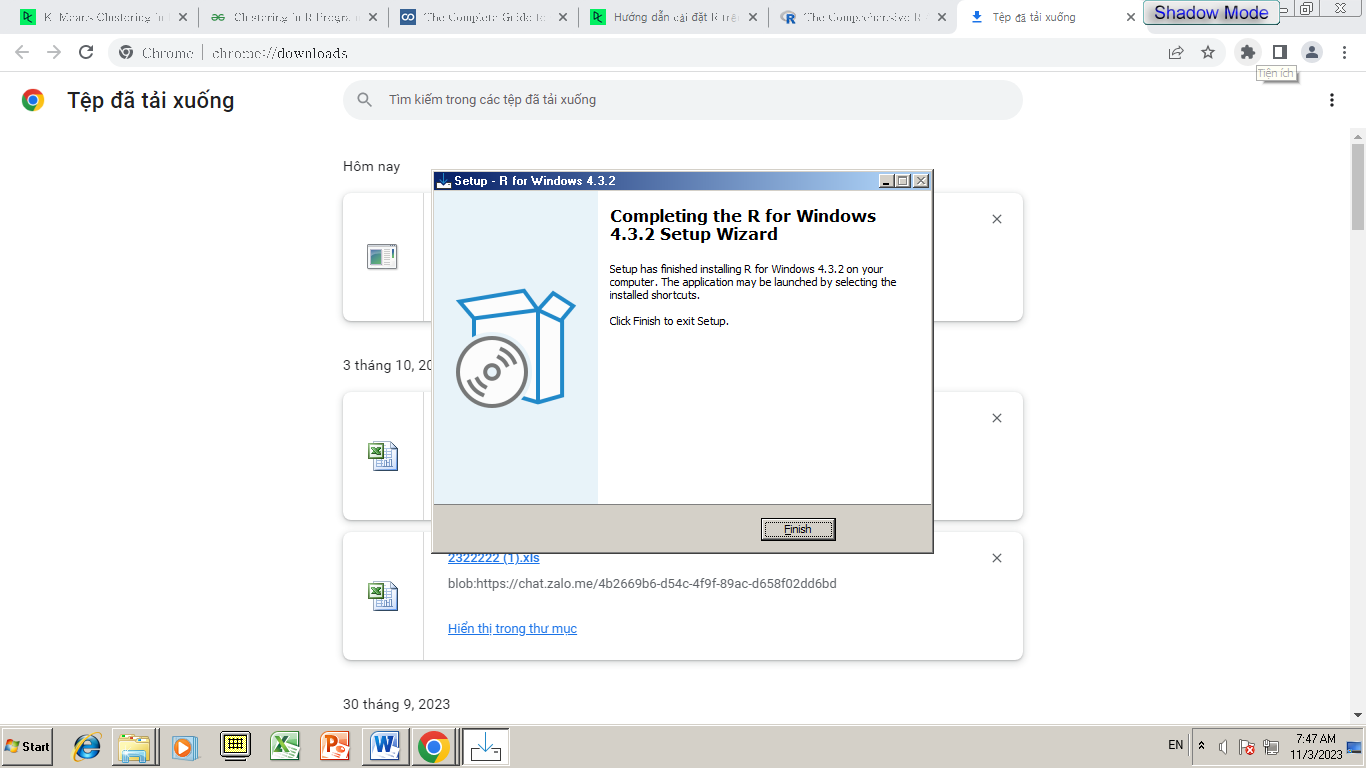


Và chờ tải về



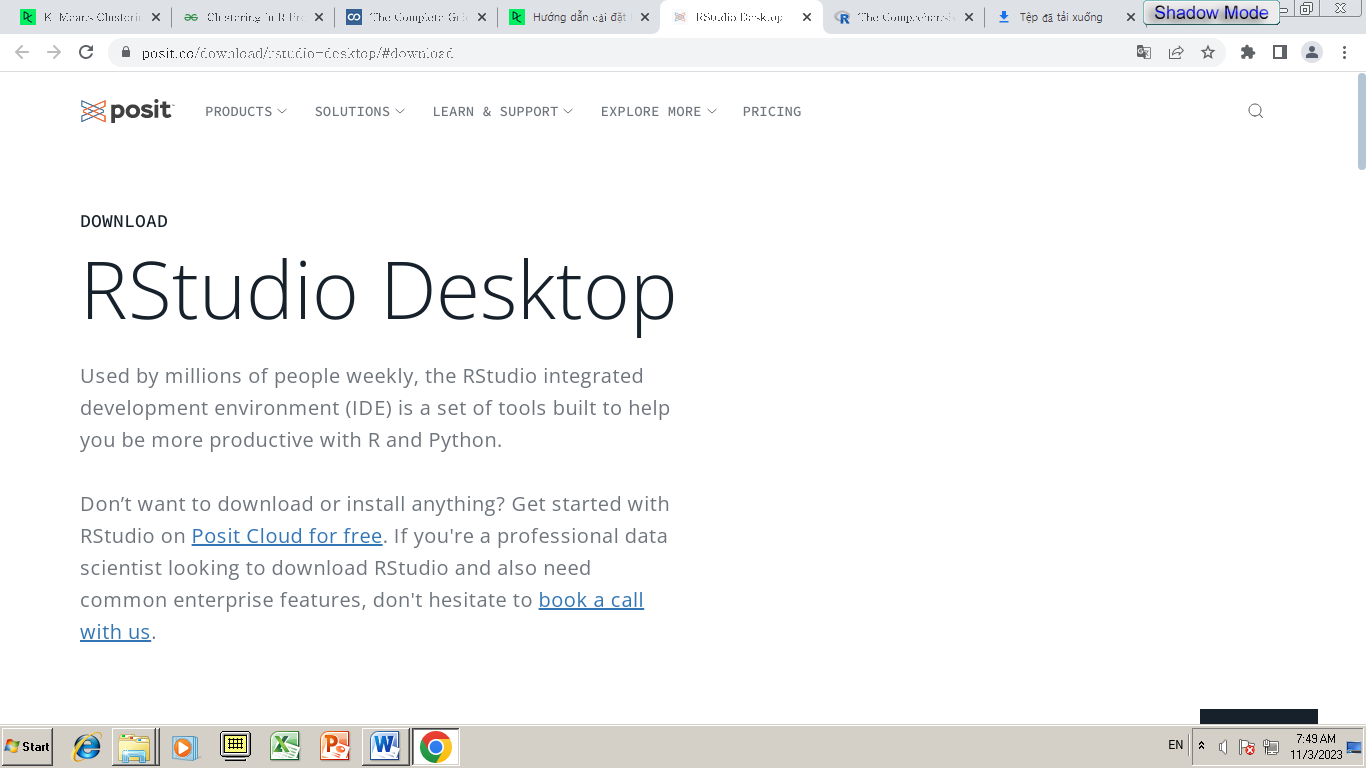
Cài đặt R vào máy



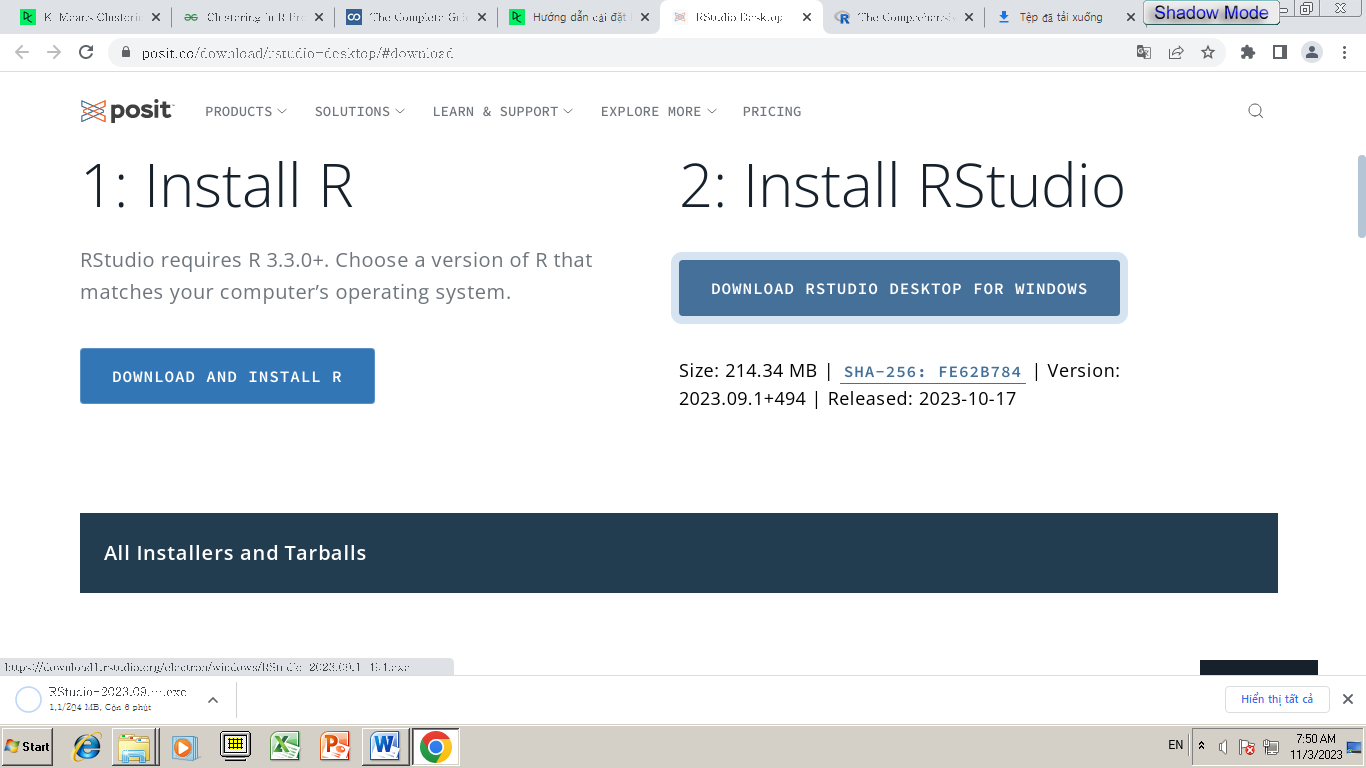


R đã cài đặt thành công

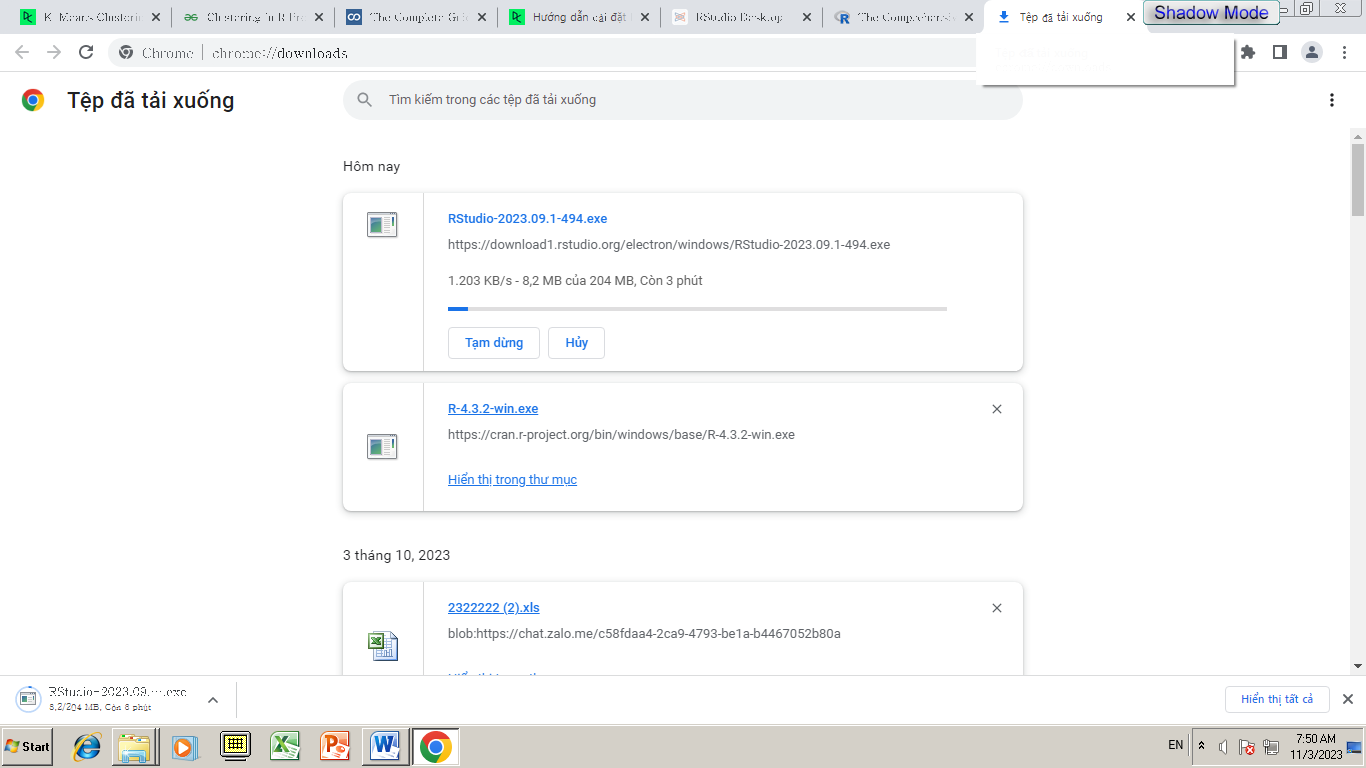
Tiếp sau đó là phải cài đặt **RStudio**



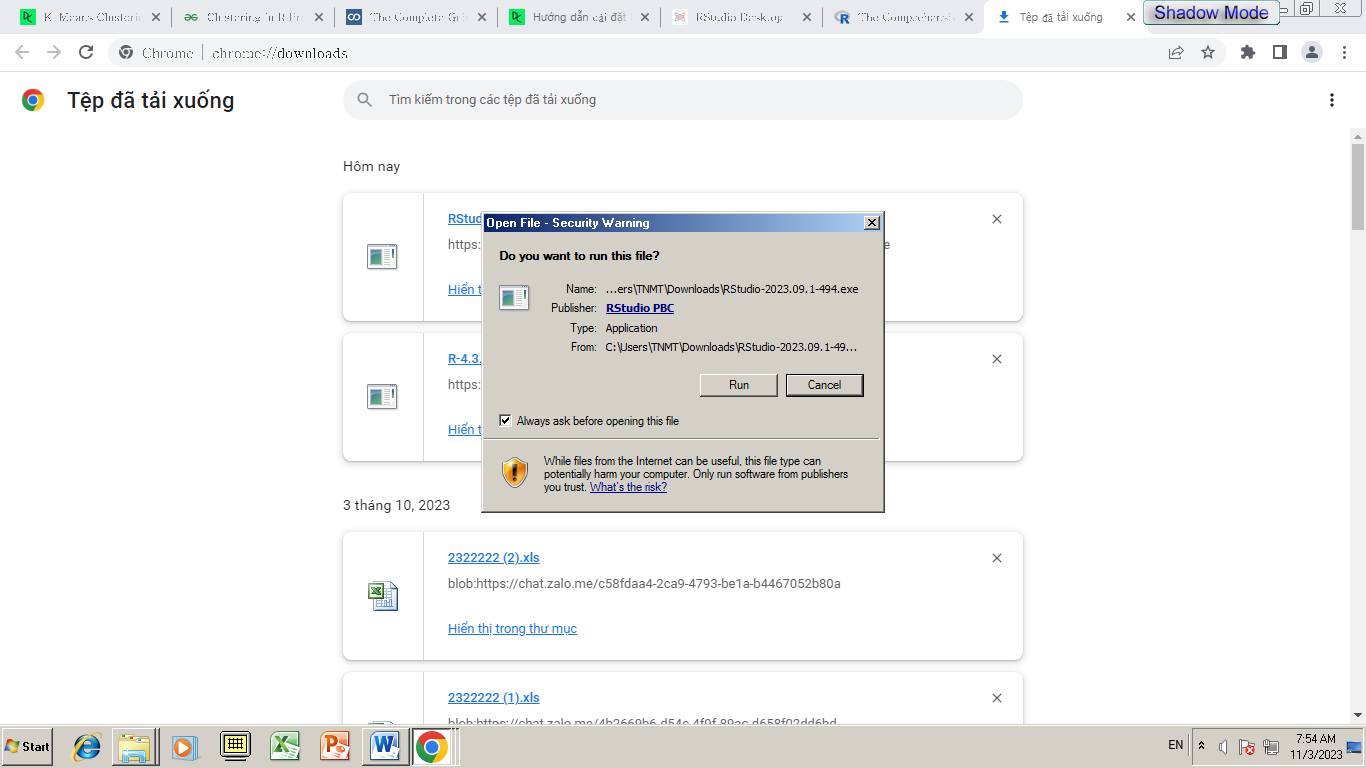
Vào trang <https://posit.co/download/rstudio-desktop/#download> để download



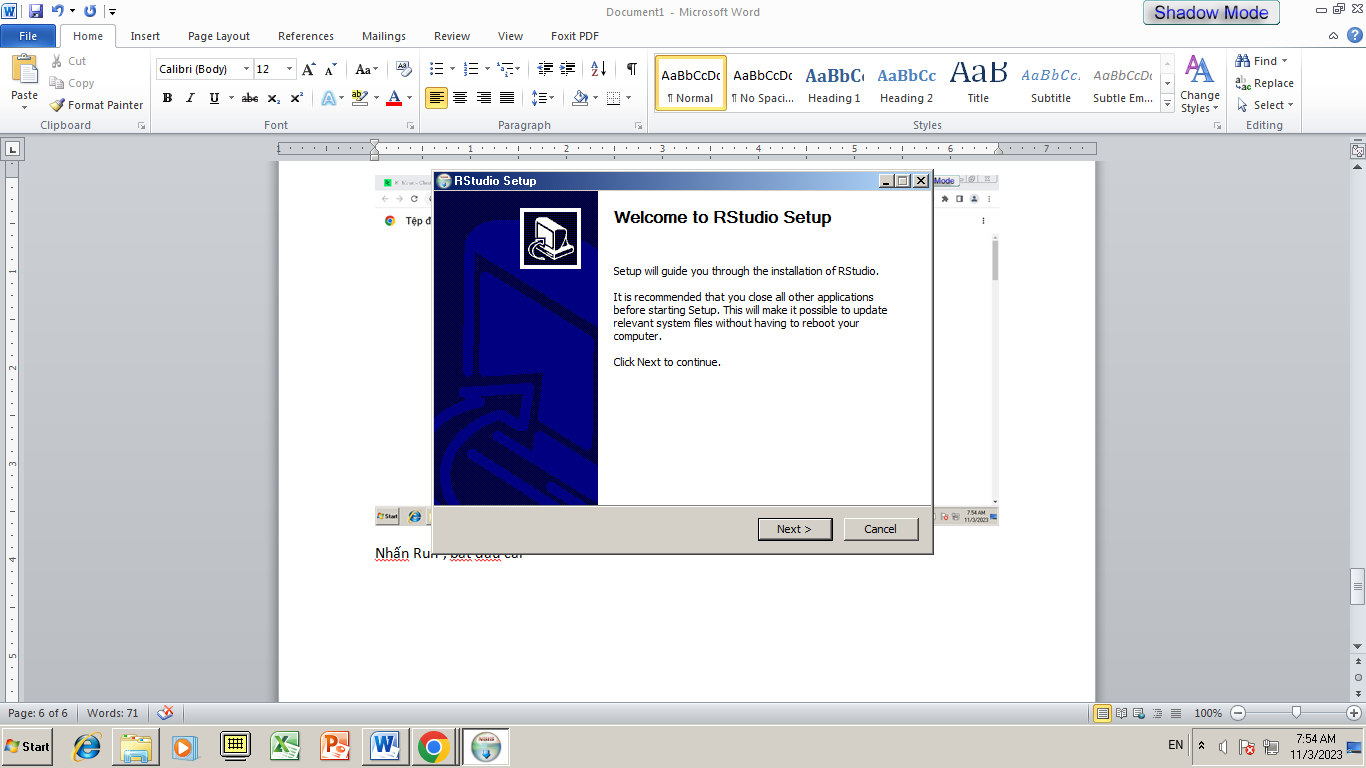
Nhấn Download RStudio Desktop for Winodows



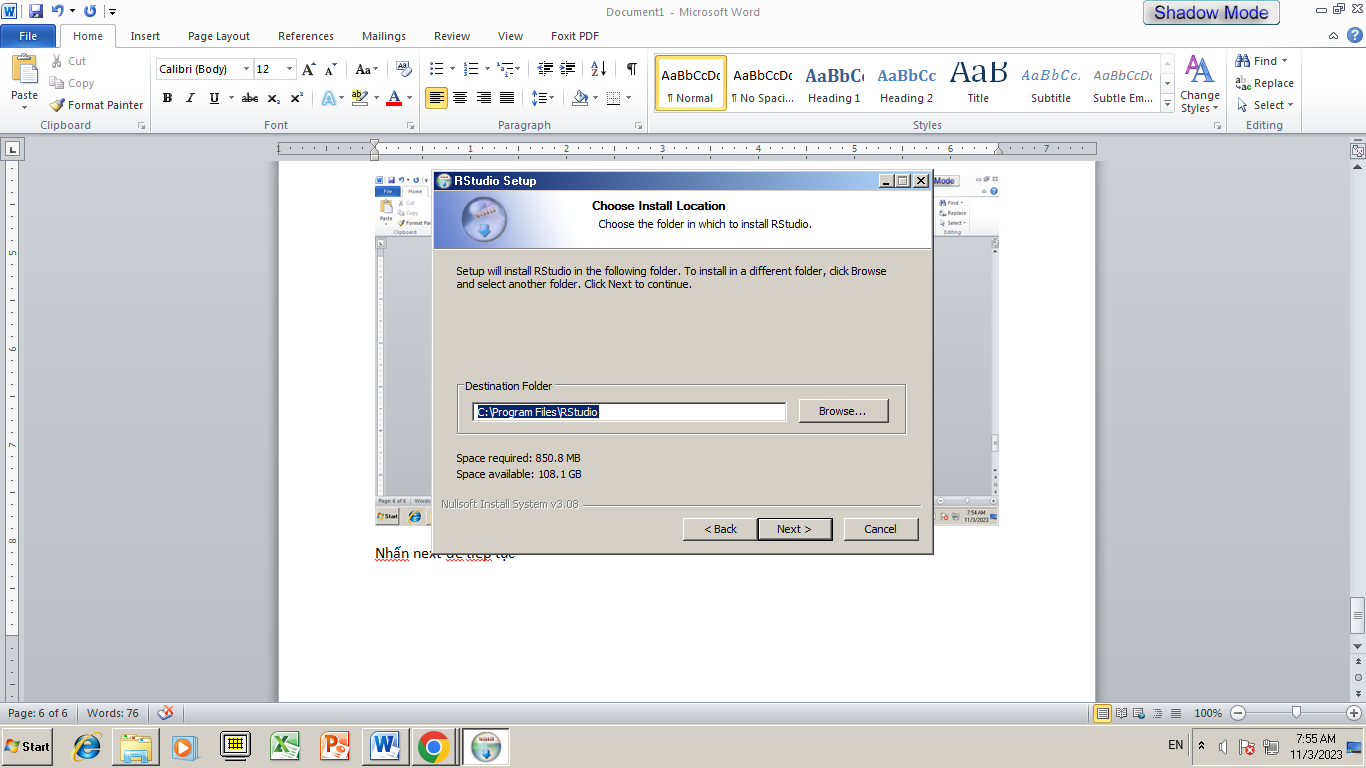
Chờ tải xuống hoàn tất



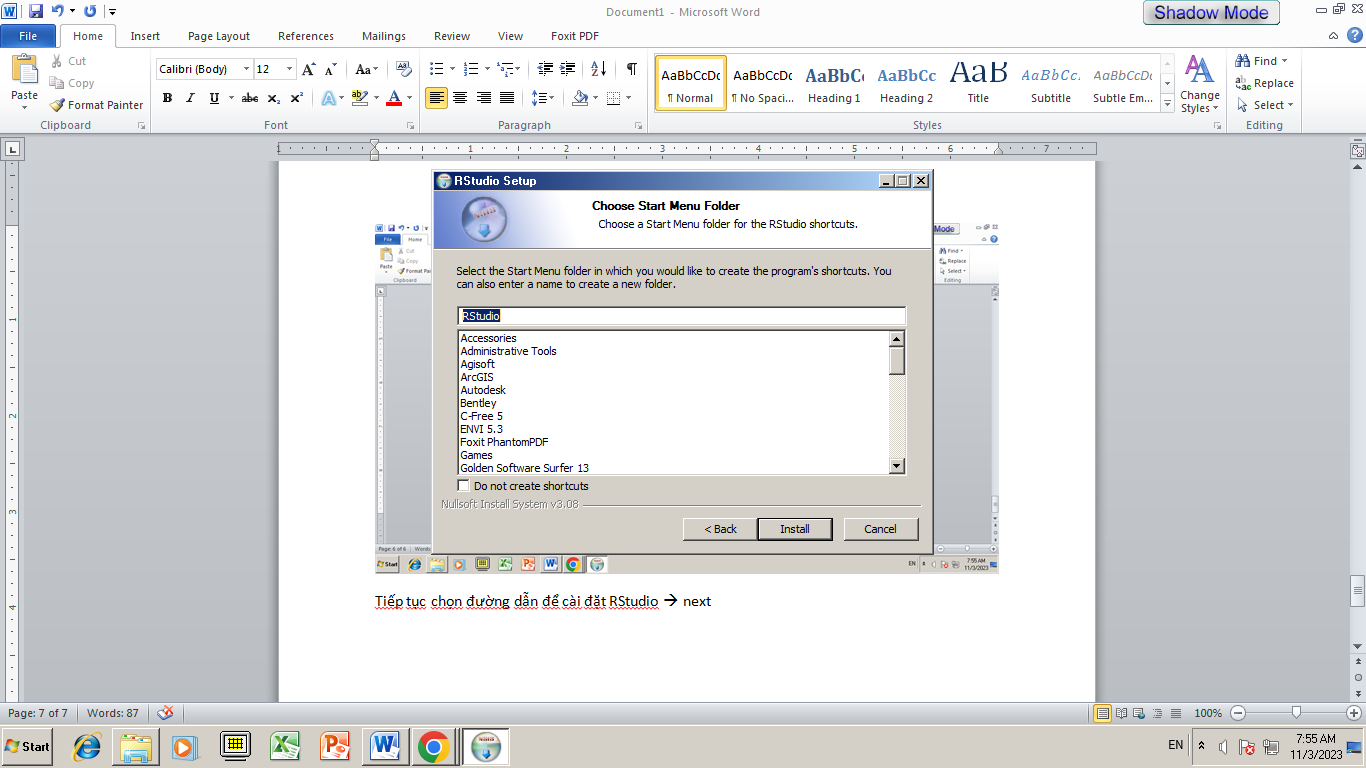
Nhấn Run , bắt đầu cài



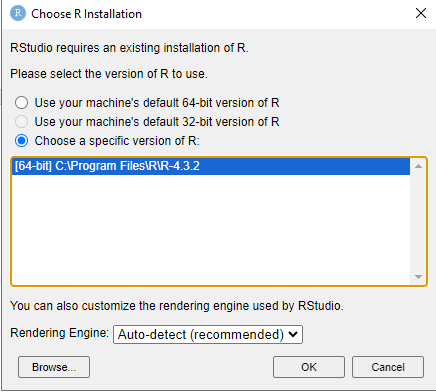
Nhấn next để tiếp tục



Tiếp tục chọn đường dẫn để cài đặt RStudio 🡪 next



Tiếp tục nhấn install 🡪 chờ cài đặt xong 🡪 Đã cài xong RStudio



Mở RStudio , chọn phiên bản R muốn chạy 🡪 Ok

Viết code vào RStudio :

require(ggplot2)

require(MASS)

set.seed(42)

set1 <- MASS::mvrnorm(n = 150, c(-10,10), matrix(c(1.5,1,1,1.5),2))

set2 <- MASS::mvrnorm(n = 150, c(-5,10), matrix(c(1,2,2,6),2))

set3 <- MASS::mvrnorm(n = 150, c(-1,1), matrix(c(4,0,0,4),2))

set4 <- MASS::mvrnorm(n = 150, c(10,-10), matrix(c(4,0,0,4),2))

set5 <- MASS::mvrnorm(n = 150, c(3,-3), matrix(c(4,0,0,4),2))

DF = data.frame(rbind(set1,set2,set3,set4,set5),cluster=as.factor(c(rep(1:5,each=150))))

ggplot(DF,aes(x=X1,y=X2,color=cluster))+geom\_point()

kmeans = function(data,K=4,stop\_crit=10e-3)

{

#Initialisation of clusters

centroids = data[sample.int(nrow(data),K),]

current\_stop\_crit = 1000

cluster = rep(0,nrow(data))

converged = F

it = 1

while(current\_stop\_crit>=stop\_crit & converged==F)

{

it=it+1

if (current\_stop\_crit<=stop\_crit)

{

converged=T

}

old\_centroids=centroids

##Assigning each point to a centroid

for (i in 1:nrow(data))

{

min\_dist=10e10

for (centroid in 1:nrow(centroids))

{

distance\_to\_centroid=sum((centroids[centroid,]-data[i,])^2)

if (distance\_to\_centroid<=min\_dist)

{

cluster[i]=centroid

min\_dist=distance\_to\_centroid

}

}

}

#Assigning each point to a centroid

for (i in 1:nrow(centroids))

{

centroids[i,]=apply(data[cluster==i,],2,mean)

}

current\_stop\_crit=mean((old\_centroids-centroids)^2)

}

return(list(data=data.frame(data,cluster),centroids=centroids))

}

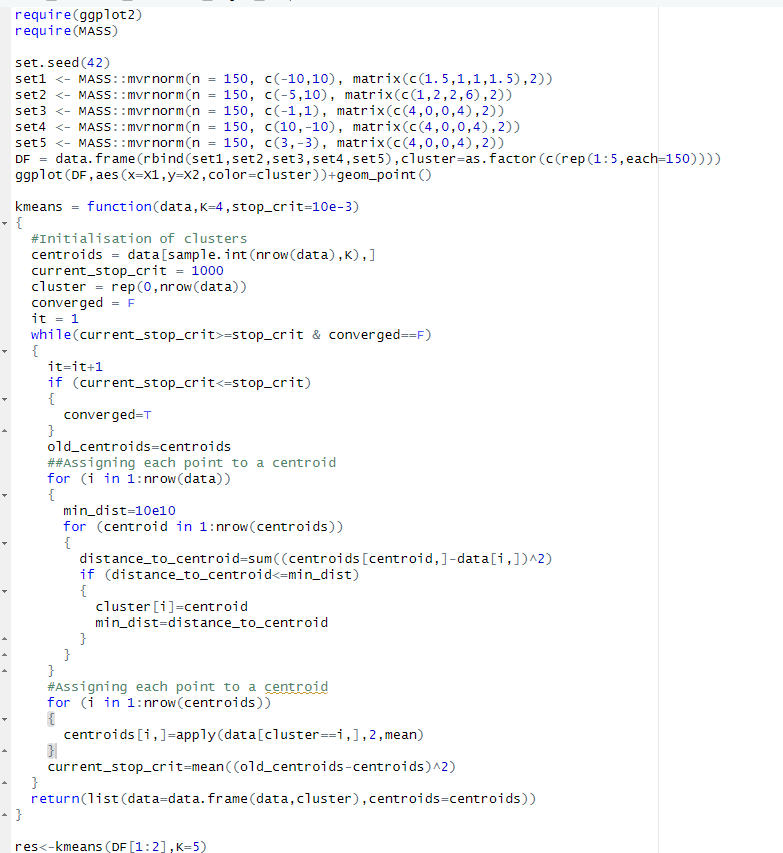
res<-kmeans(DF[1:2],K=5)

res$centroids$cluster=1:5

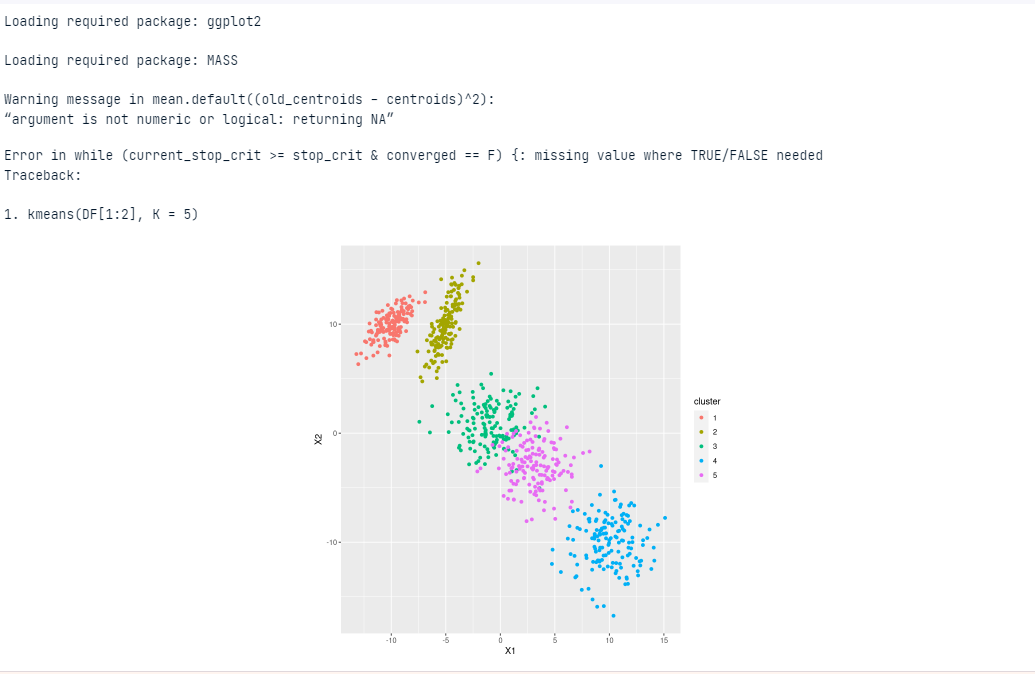
res$data$isCentroid=F

res$centroids$isCentroid=T

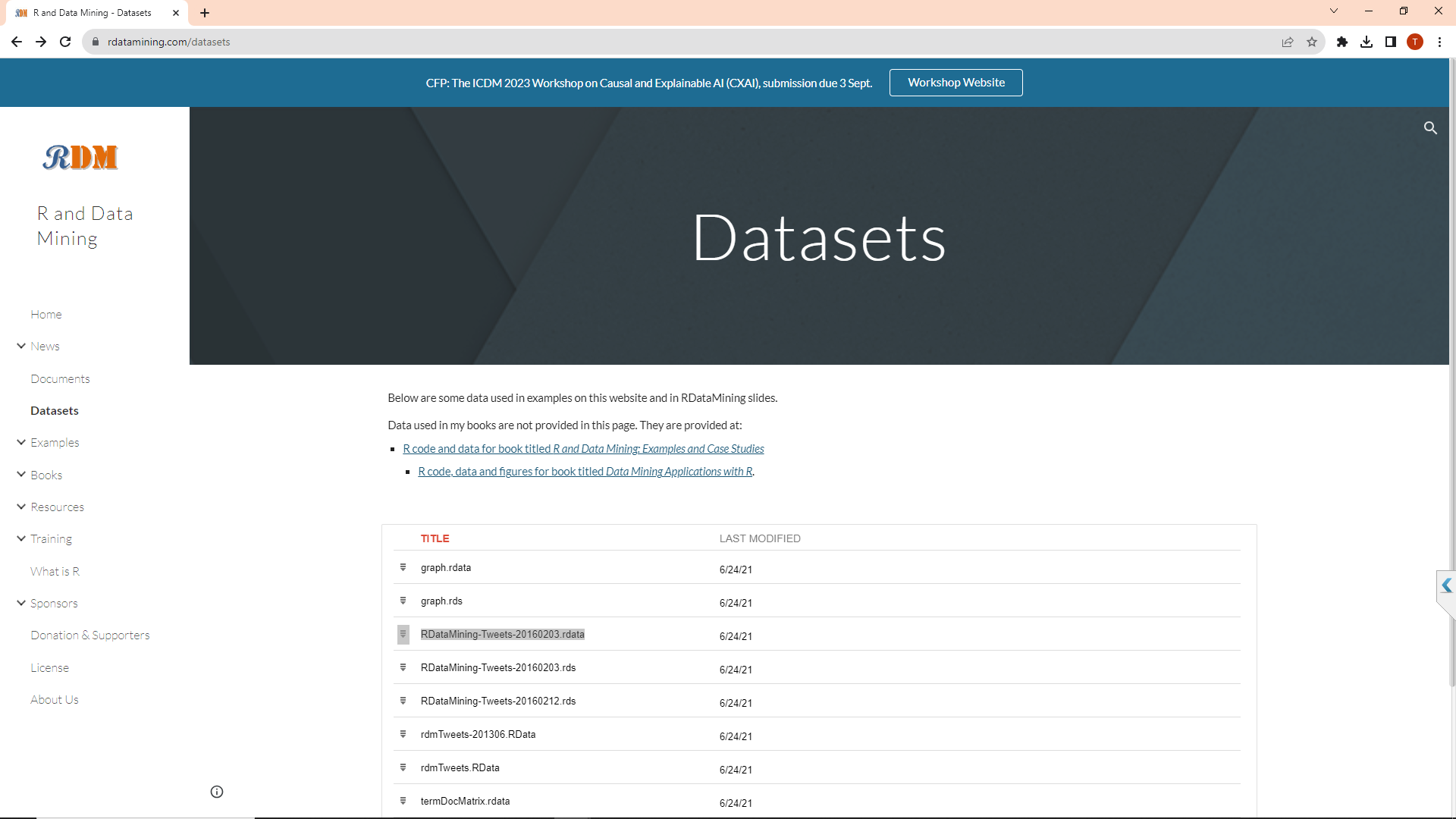
data\_plot=rbind(res$centroids,res$data)



Ra kết quả :



**Program – 10**



Do phần URL / download : link đã quá cổ , không còn tồn tại nữa , nên thay vì dùng URL để download . Ta sẽ download trực tiếp datasets từ trang và copy đường dẫn vào load file

Code :

install.packages("dplyr")

install.packages("twitteR")

install.packages("tidytext")

install.packages("tm")

install.packages("SnowballC")

library(twitteR)

library(tidytext)

library(dplyr)

library(tm)

library(SnowballC)

# workingDir <- "~/Documents/R programs"

# setwd(workingDir)

#Tên file cũ : rdmTweets-201306.RData

#load file trực tiếp – copy path bỏ vào

load(file = "./RDataMining-Tweets-20160203.rdata")

tweets <- twListToDF(tweets)

load(file = "./RDataMining-Tweets-20160203.rdata")

tweets <- twListToDF(tweets)

tweets <- tweets %>%

mutate(text=gsub("(http|https).+$|\\n|&amp|[[:punct:]]","",text),

rowIndex=as.numeric(row.names(.))) %>%

select(text,retweetCount,rowIndex)

docList <- as.list(tweets$text)

N.docs <- length(docList)

QrySearch <- function(queryTerm) {

# Record starting time to measure your search engine performance

start.time <- Sys.time()

# store docs in Corpus class which is a fundamental data structure in text mining

my.docs <- VectorSource(c(docList, queryTerm))

# Transform/standaridze docs to get ready for analysis

my.corpus <- VCorpus(my.docs) %>%

tm\_map(stemDocument) %>%

tm\_map(removeNumbers) %>%

tm\_map(content\_transformer(tolower)) %>%

tm\_map(removeWords,stopwords("en")) %>%

tm\_map(stripWhitespace)

# Store docs into a term document matrix where rows=terms and cols=docs

# Normalize term counts by applying TDiDF weightings

term.doc.matrix.stm <- TermDocumentMatrix(my.corpus,

control=list(

weighting=function(x)

weightSMART(x,spec="ltc"),

wordLengths=c(1,Inf)))

# Transform term document matrix into a dataframe

term.doc.matrix <- tidy(term.doc.matrix.stm) %>%

group\_by(document) %>%

mutate(vtrLen=sqrt(sum(count^2))) %>%

mutate(count=count/vtrLen) %>%

ungroup() %>%

select(term:count)

docMatrix <- term.doc.matrix %>%

mutate(document=as.numeric(document)) %>%

filter(document<N.docs+1)

qryMatrix <- term.doc.matrix %>%

mutate(document=as.numeric(document)) %>%

filter(document>=N.docs+1)

# Calculate top ten results by cosine similarity

searchRes <- docMatrix %>%

inner\_join(qryMatrix,by=c("term"="term"),

suffix=c(".doc",".query")) %>%

mutate(termScore=round(count.doc\*count.query,4)) %>%

group\_by(document.query,document.doc) %>%

summarise(Score=sum(termScore)) %>%

filter(row\_number(desc(Score))<=10) %>%

arrange(desc(Score)) %>%

left\_join(tweets,by=c("document.doc"="rowIndex")) %>%

ungroup() %>%

rename(Result=text) %>%

select(Result,Score,retweetCount) %>%

data.frame()

# Record when it stops and take the difference

end.time <- Sys.time()

time.taken <- round(end.time - start.time,4)

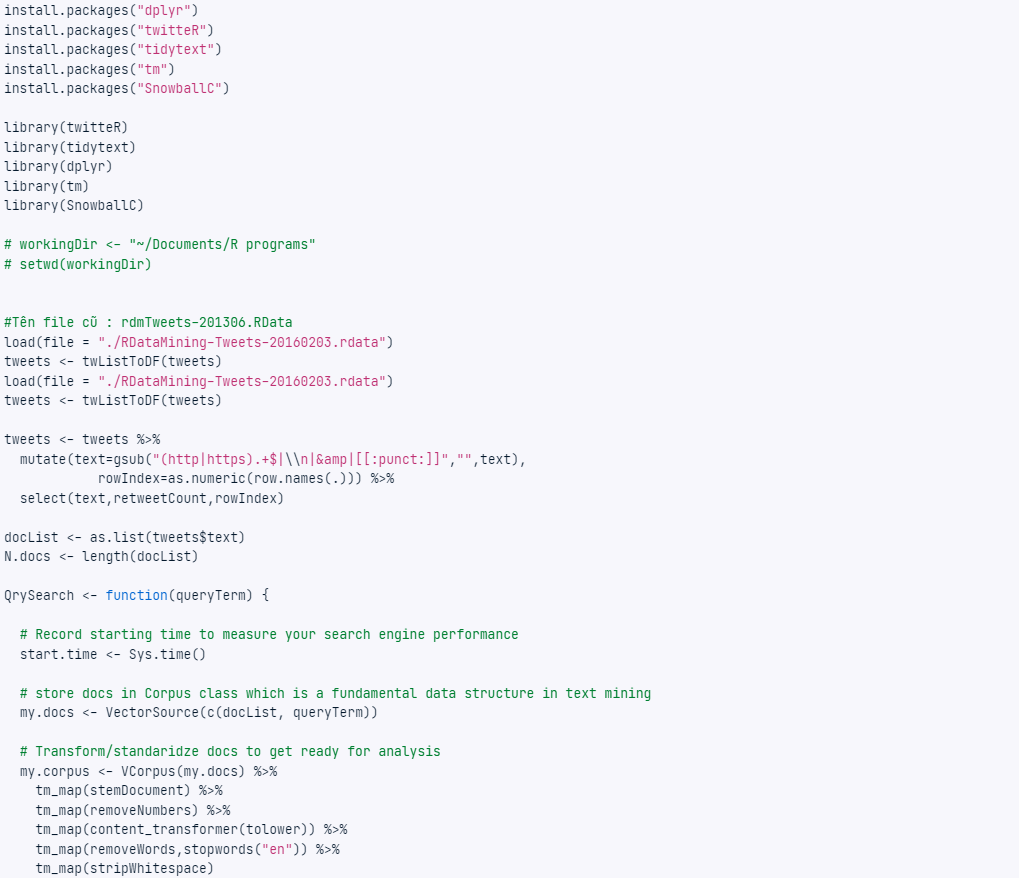
print(paste("Used",time.taken,"seconds"))

return(searchRes)

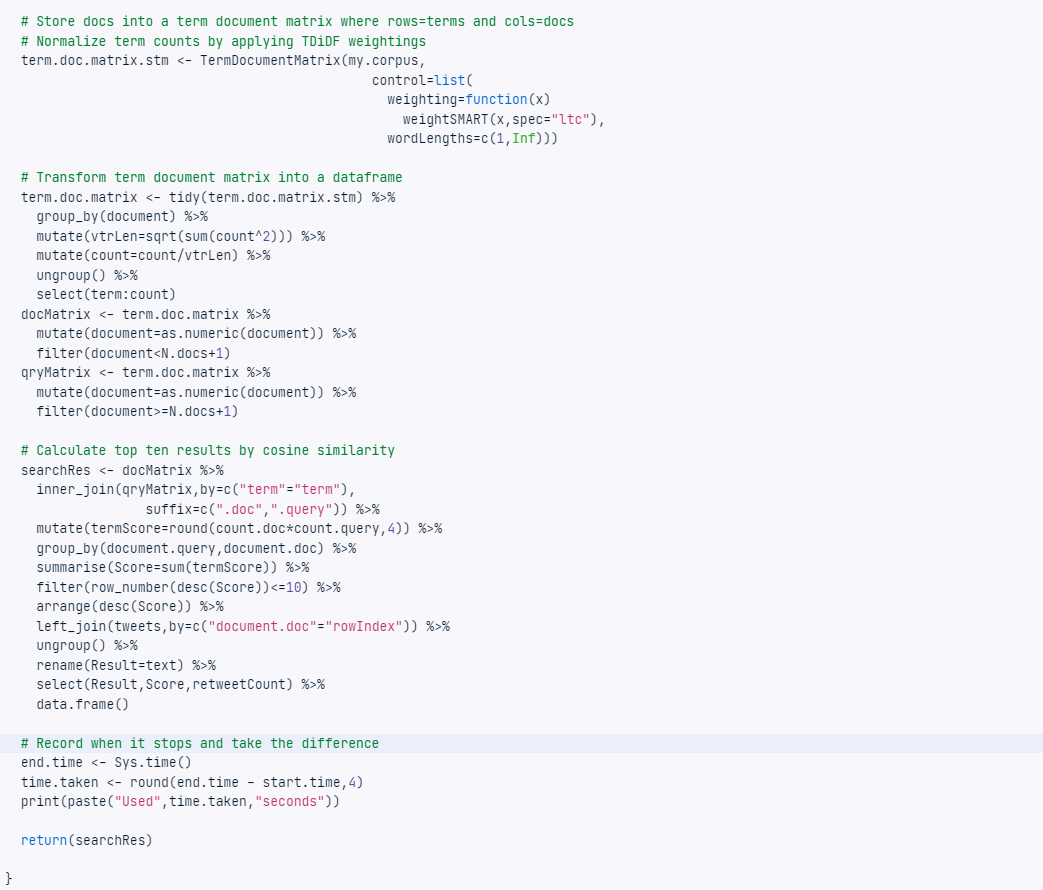
}

QrySearch("data science")

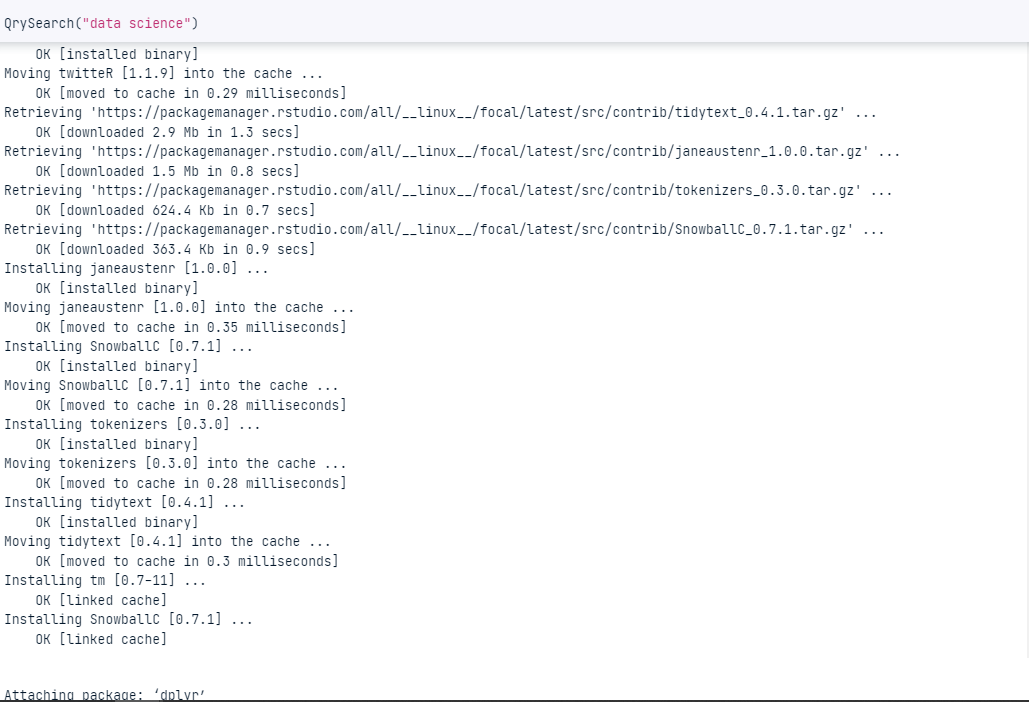
**Code part 1 :**



**Code part 2 :**



**Chạy thành công :**



**Kết quả :**

