**ĐẠI HỌC HUẾ**



KHOA KỸ THUẬT VÀ CÔNG NGHỆ

🙠🙟🕮🙝🙢

**Icon

Description automatically generated**

**BÁO CÁO**

**ĐỒ ÁN PHÂN TÍCH DỮ LIỆU R**

**NĂM HỌC 2021-2022**

**Giáo viên hướng dẫn: HỒ QUỐC DŨNG**

**Lớp: Khoa học Dữ liệu và Trí tuệ Nhân tạo**

|  |
| --- |
| Số phách  *(Do hội đồng chấm thi ghi)* |

**Thừa Thiên Huế, ngày 30 tháng 12 năm 2021**



**ĐẠI HỌC HUẾ**

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

🙠🙟🕮🙝🙢

****

(MẪU BÌA PHỤ)

**BÁO CÁO**

**ĐỒ ÁN PHÂN TÍCH DỮ LIỆU R**

**NĂM HỌC 2021-2022**

**Giảng viên hướng dẫn: Hồ Quốc Dũng**

**Lớp: Khoa học Dữ liệu và Trí tuệ Nhân tạo K1**

**Sinh viên thực hiện: Lê Thanh Hùng**

|  |
| --- |
| Số phách  *(Do hội đồng chấm thi ghi)* |

**Thừa Thiên Huế, ngày 30 tháng 12 năm 2021**

**MỤC LỤC**

[**Chương I: Web Crawler** 2](#_Toc92790079)

[1. Mục tiêu 2](#_Toc92790080)

[2.Tiến hành Craw data 2](#_Toc92790081)

[2.1 Crawler nghĩa là gì ? 2](#_Toc92790082)

[2.2 Giới thiệu về ngôn ngữ lập trình Pyhton và thư viện Selenium 2](#_Toc92790083)

[2.3 Tiến hành Craw Dữ liệu 3](#_Toc92790084)

[**Chương II: Phân tích dữ liệu bằng R** 15](#_Toc92790085)

[1.Mục tiêu 15](#_Toc92790086)

[2. Phân tích dữ liệu 15](#_Toc92790087)

[2.1 Đọc dữ liệu, kiểm tra thông số dữ liệu (Data Information). 15](#_Toc92790088)

[2.2 Các phép tính hướng tâm (The Center Tendency) 20](#_Toc92790089)

[2.3 Trực quan hóa dữ liệu(Data visualization) 23](#_Toc92790090)

[2.3 Các phép kiểm định 28](#_Toc92790091)

[**Chương III: Kết quả kiểm tra đạo văn** 40](#_Toc92790092)

[**Chương IV: Tài liệu tham khảo** 41](#_Toc92790093)

# **Chương I: Web Crawler**

## 1. Mục tiêu

* Lấy được dữ liệu đủ để có thể tiến hành phân tích dữ liệu.
* Lưu dữ liệu craw được thành file csv để có thể đọc dữ liệu.



## 2.Tiến hành Craw data

### 2.1 Crawler nghĩa là gì ?

Crawl (Spider hay là Bot) là một thuật ngữ mô tả quá trình thu thập dữ liệu trên web được gọi chung là Web Crawler. Phần mềm này được thiết kế để có thể duyệt website trên mạng World Wide Web, chúng tiến hành phân tích mã nguồn HTML, đọc dữ liệu và lọc ra theo yêu cầu của người sử dụng. Quá trình này sẽ dừng lại cho đến khi con bot duyệt hết tất cả các đường liên kết trang đầu cũng như các trang có liên quan.

Ở đây, tôi sẽ sử dụng python(phiên bản 3.10) và thư viện selenium để craw dữ liệu.

### 2.2 Giới thiệu về ngôn ngữ lập trình Pyhton và thư viện Selenium

Giới thiệu về ngôn ngữ lập trình Python.

Ngôn ngữ lập trình python là ngôn ngữ bậc cao hỗ trợ nhiều thư viện cho người dùng, python khá dễ để sử dụng và thích hợp cho người mới bắt đầu làm quen với code.

Ngôn ngữ python có cú pháp rõ ràng, đáp ứng cho hầu hết tất cả yếu cầu của lập trình.

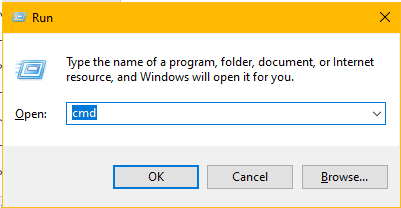
Giới thiệu về Seleium

Seleium là một thư viện hỗ trợ cho việc tự đồng hóa các trình duyệt mạnh mẽ và dễ dàng thông qua Webdriver.

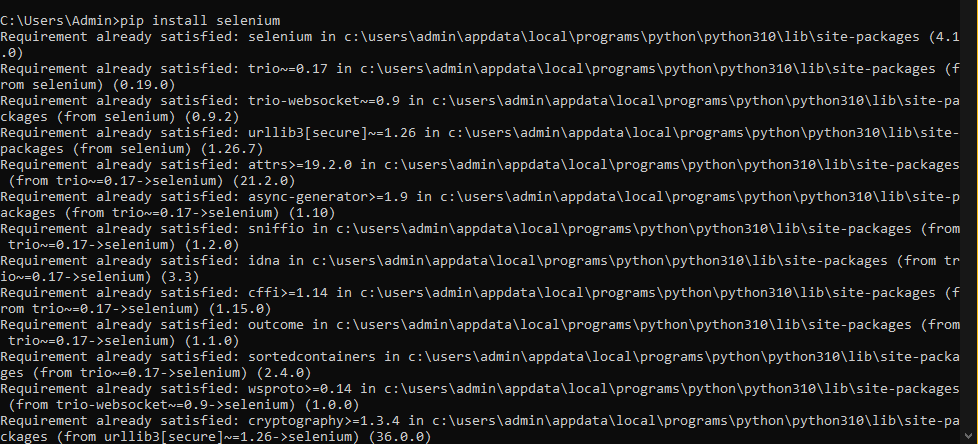
Cách cài đặt thư viện selenium.

Sau khi cài python, và gói pip (nếu nó không có trong thư mục khi bạn tải python) việc tiếp theo bạn cần làm chính là tải thư viện Seleium từ **CMD**.

Từ màn hình desktop chọn tổ hợp phím **Windown + R**. Bảng Run sẽ hiện lên, tiếp theo chỉ cần gõ Cmd và nhấn **ENTER**.



Từ của sổ của CMD gõ lệnh : **pip install selenium**



Bước chuẩn bị cuối cùng, cần tải thêm file ChromeDriver hoặc FileforeDriver để có thể giả lập trình duyệt. Ở đây tôi sẽ dùng ChromeDriver.

Lưu ý: Cần đặt file ChromeDriver cạnh file code để có thể chạy.

### 2.3 Tiến hành Craw Dữ liệu

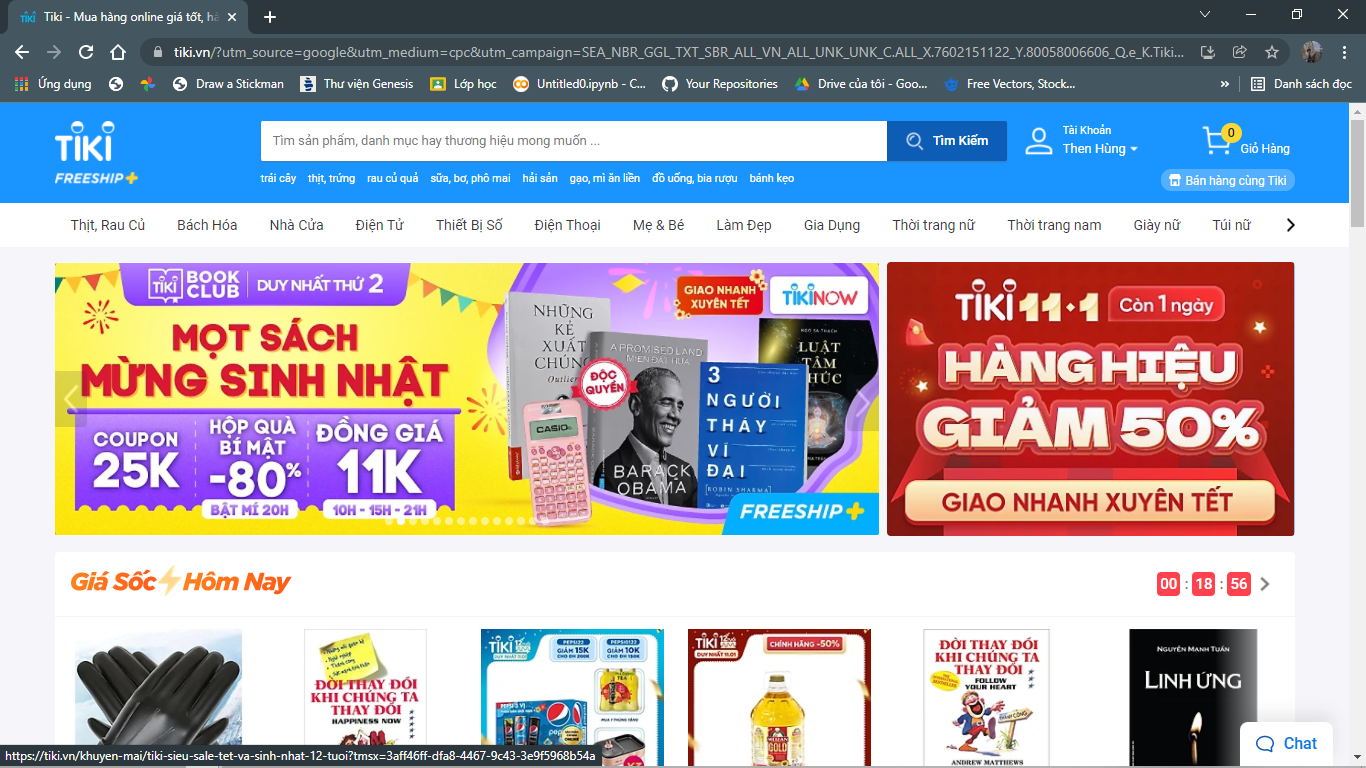
**a.import thư viện.**

****

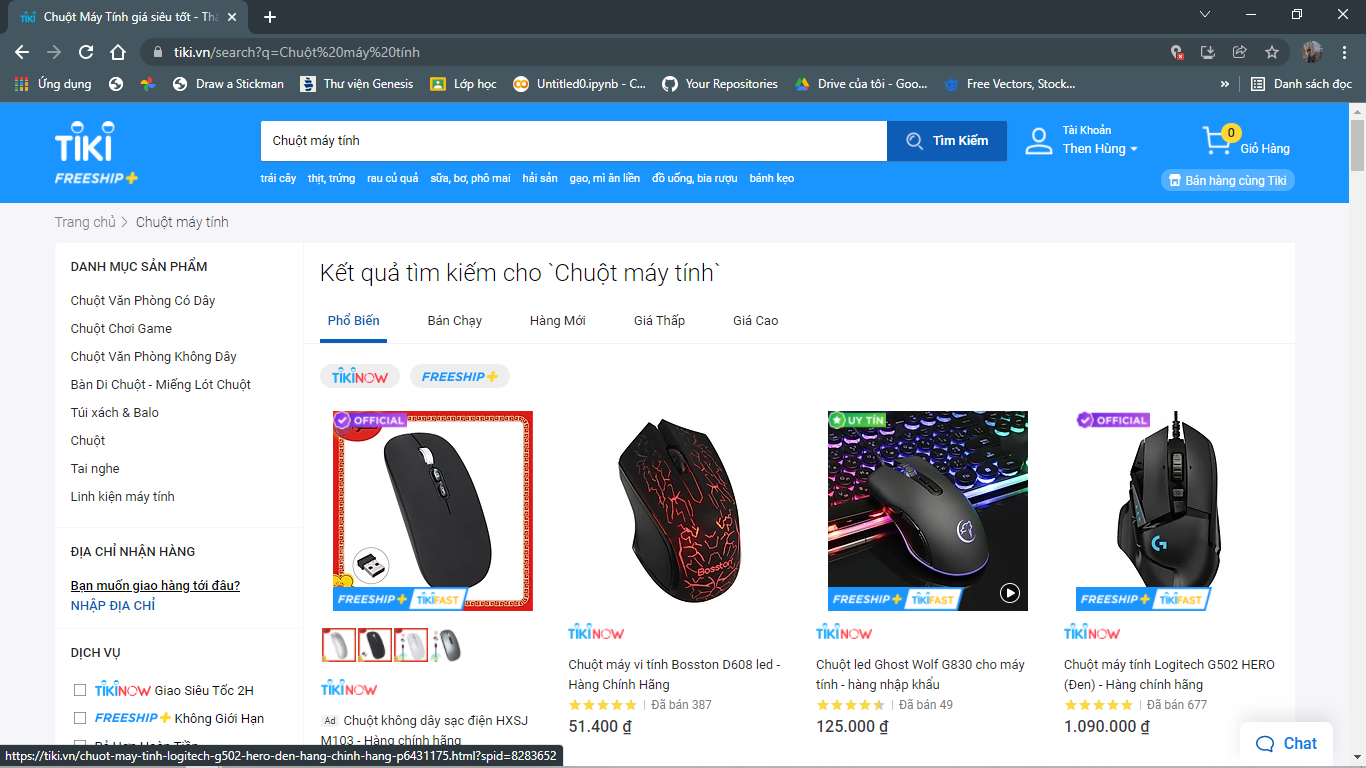
Đây là một số thư viện cần cài đặt và một số thiết lập của trình duyệt.

**b. Lấy link sản phẩm.**

Khi truy cập vào Tiki ta sẽ thấy trang chủ như sau:



Gõ vào ô Tìm kiếm về thông tin sản phẩm đang muốn craw rồi nhấn ENTER.



Xác định những sản phẩm sẽ Craw ở đây.

Khi chuyển sang trang thứ 2.

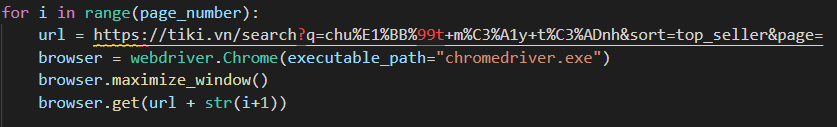
url trang 2: <https://tiki.vn/search?q=Chu%E1%BB%99t+m%C3%A1y+t%C3%ADnh&page>=2

url trang 3: <https://tiki.vn/search?q=Chu%E1%BB%99t+m%C3%A1y+t%C3%ADnh&page=3>

Tương tự trang thứ n: <https://tiki.vn/search?q=Chu%E1%BB%99t+m%C3%A1y+t%C3%ADnh&page>= + n

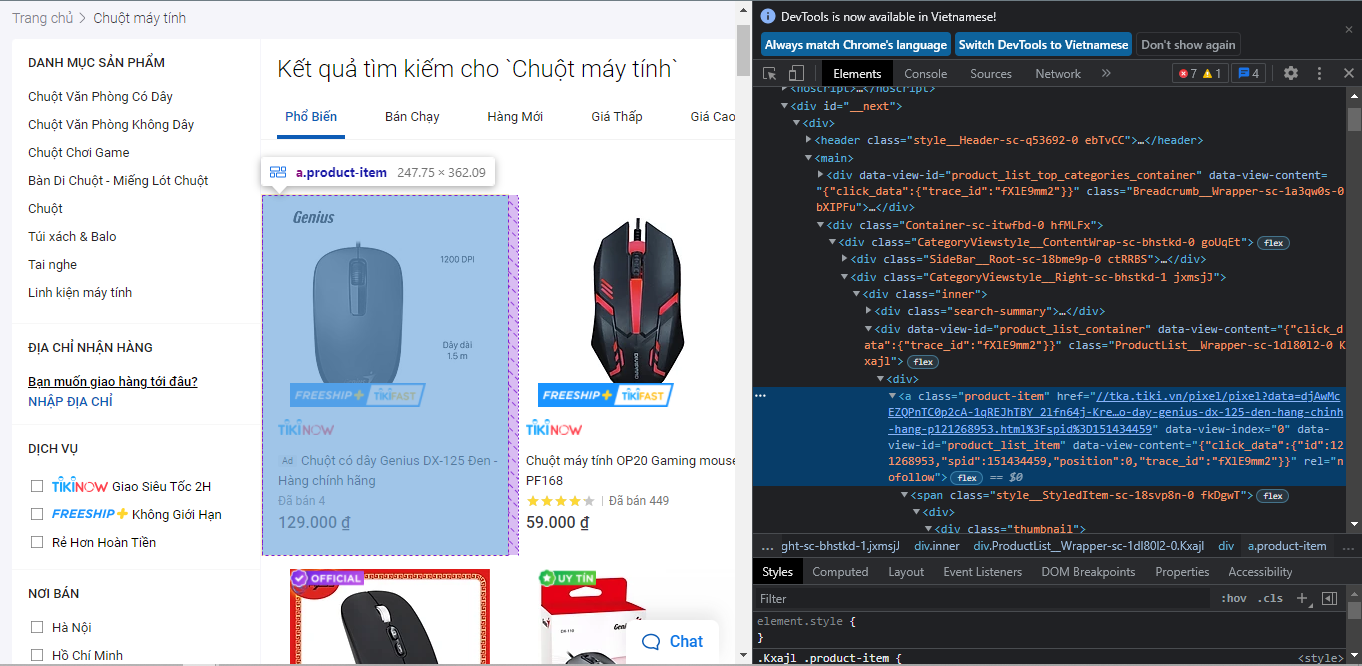
Vậy ý tưởng để có thể vào n trang 🡪 url + n

Kết quả sẽ là:



Đã có thể truy cập vào các trang, tiếp theo sẽ lấy link toàn bộ sản phẩm có trong trang. Tiếp theo sẽ lấy link của tất cả sản phẩm có trong trang.

Nhấp chuột phải vào sản phẩm sau đó chọn kiểm tra. Ta có thể thấy được link sản phẩm nằm ở đây.



Phân tích một chút, link sản phẩm nằm trong thẻ **a** thuộc class **product\_item.** Đường link nằm trong thẻ **href**.

Vậy trước hết ta cần chờ cho toàn bộ sản phẩm ở trên Web được tải xong, để làm việc nằm ta sẽ dùng WebDriverWait để chờ.



Như vậy chương trình sẽ tạm thời dừng lại nhiều nhất 10 giây đến khi toàn bộ sản phẩm trong đường link tải xong.

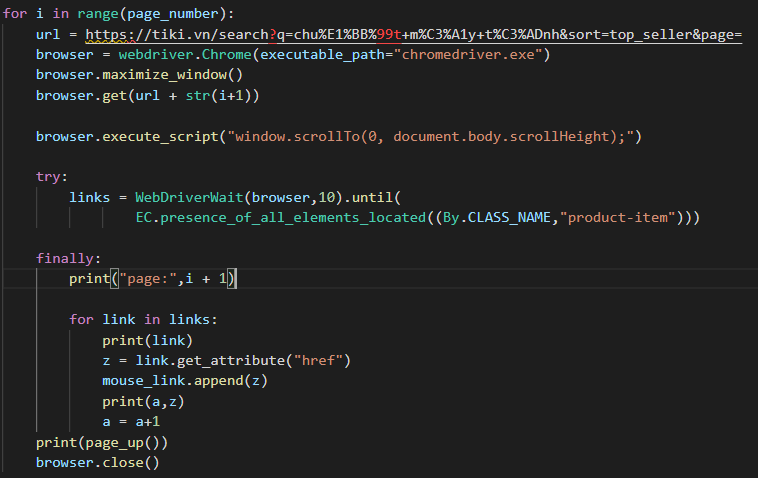
Như đã phân tích sau khi truy cập vào tất cả các thẻ a có class product\_item, ta còn cần phải trỏ vào thẻ href để lấy đường link sản phẩm. Ở đây sẽ dùng hàm get\_attribute() để lấy thuộc tính href.



Kết hợp tất cả lại, thêm vào các thuộc tính như

* maximize\_windown() : Full màn hình
* execute\_script() : Cuộn trang.

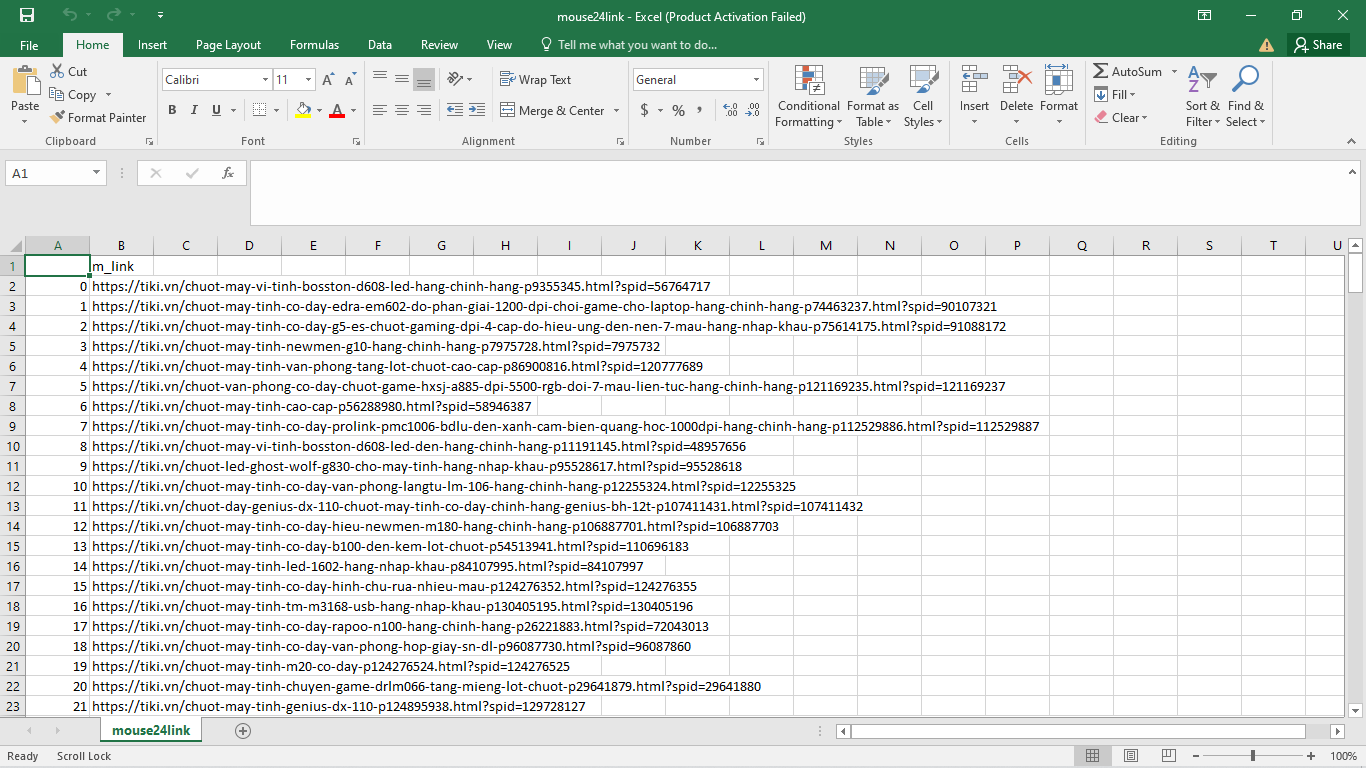
Và sau khi lấy được thẻ href thì thêm vào 1 list, Cuối cùng là đồng trình duyệt khi kết thúc.



Sau khi lấy được toàn bộ đường link, ta viết hàm lưu nó vào thành 1 file csv



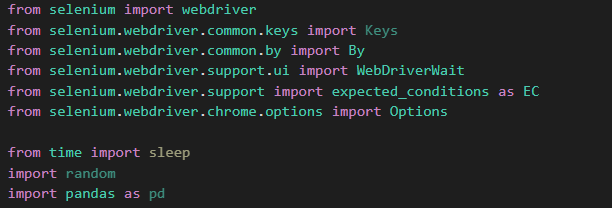
Kết quả:



**c. Lấy element.**

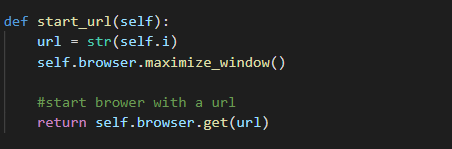
Sau khi lấy toàn bộ link sản phẩm, việc tiếp theo của chúng ta là truy cập vào từng link đó và lấy dữ liệu của từng sản phẩm.

Bước đầu vẫn là import những thư viện cần cho thao tác craw.

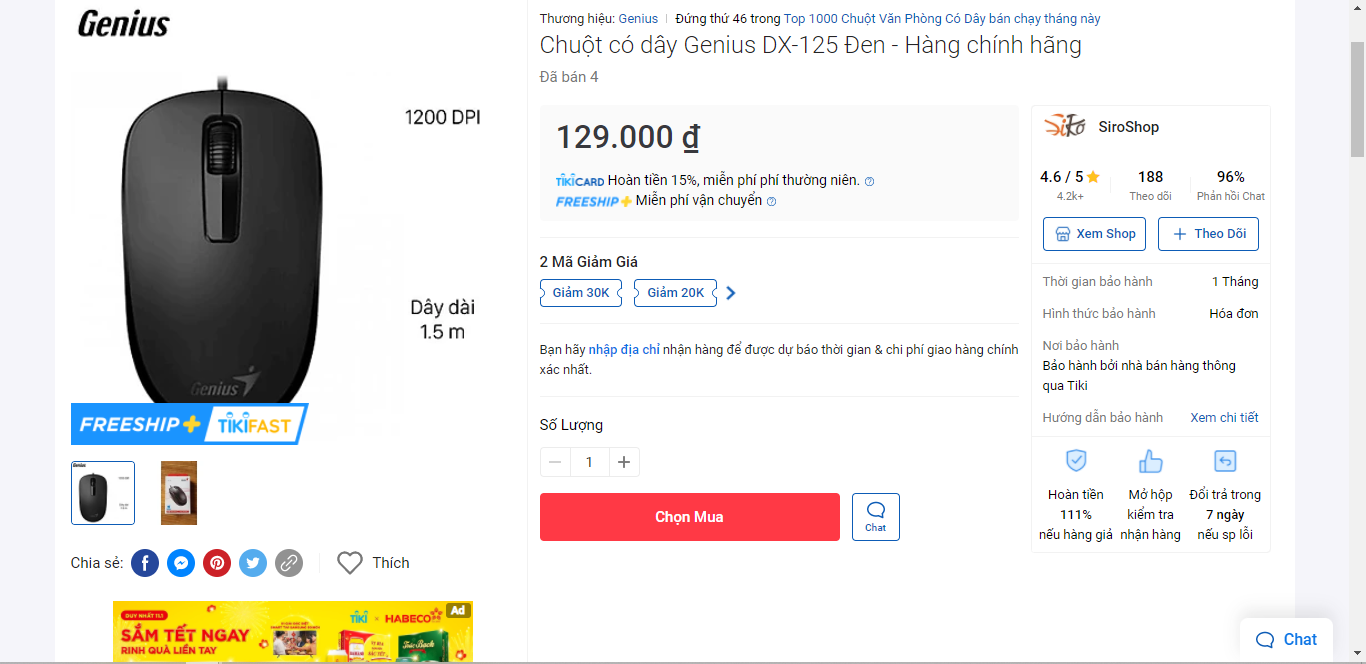


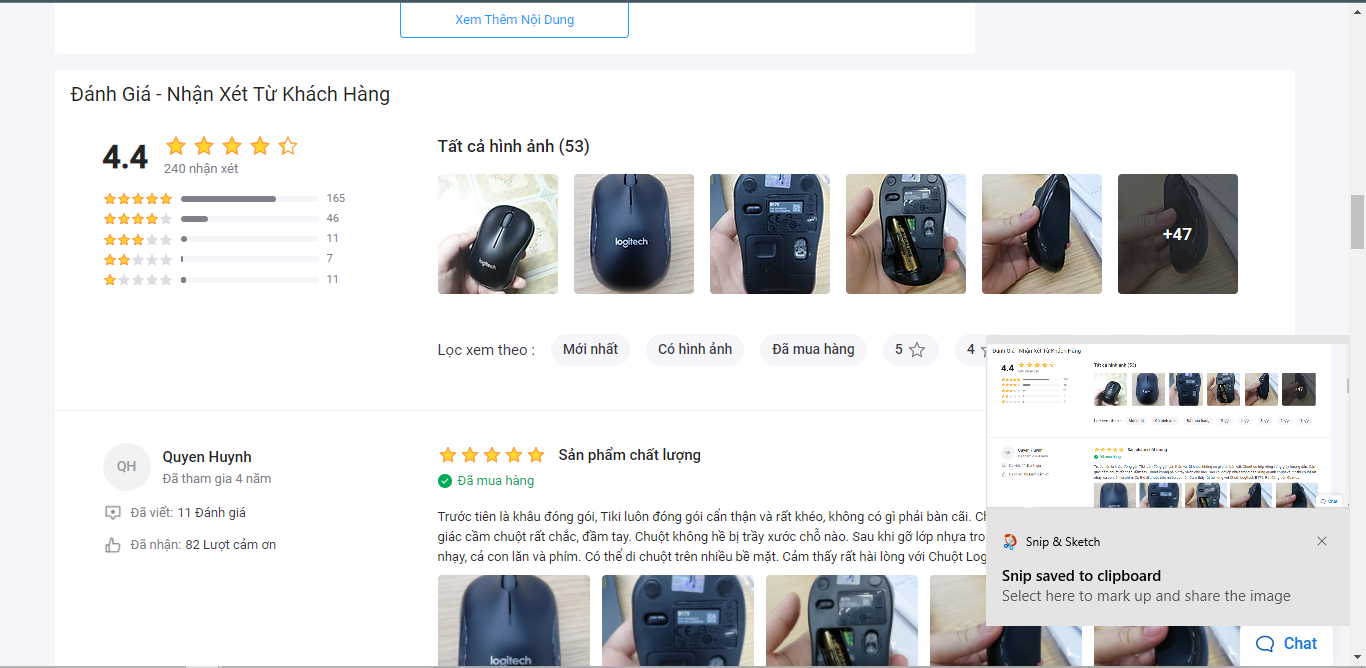
Lần này, chúng ta sẽ xây dựng 1 Class để mỗi hàm sẽ có chức năng lấy mỗi element khác nhau.

Hàm strart\_url để có thể bắt đầu trình duyệt với một đường link.



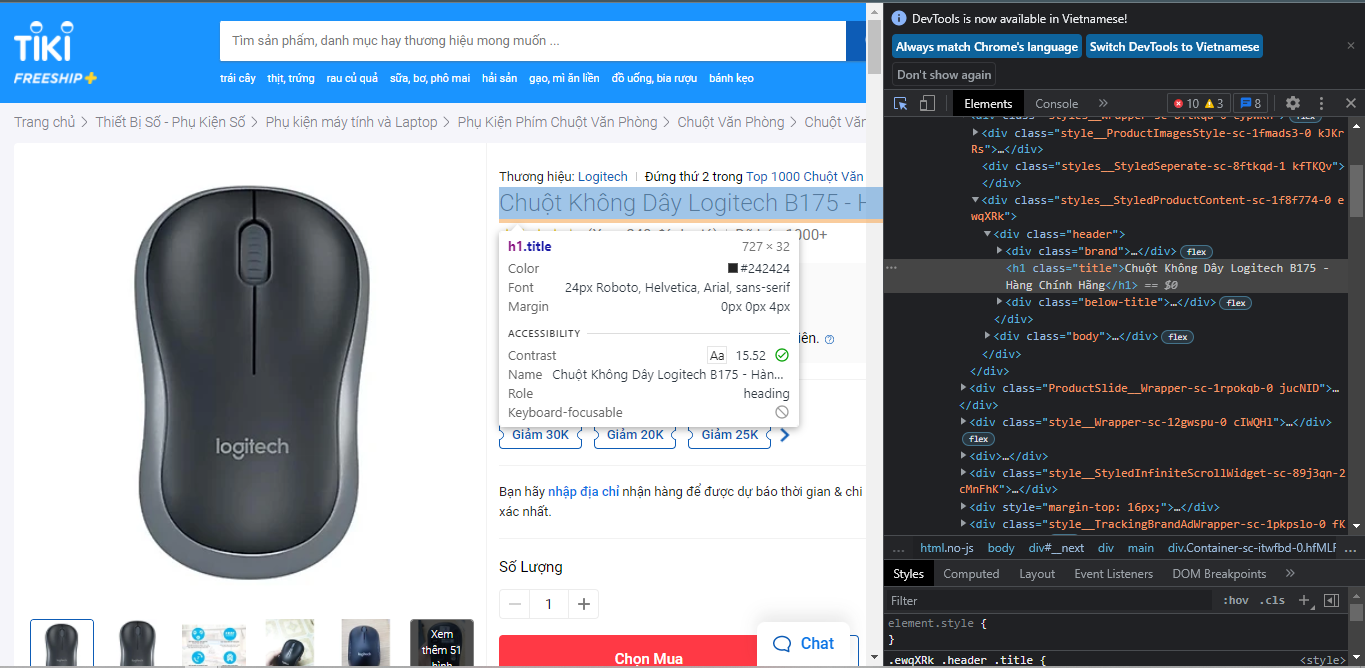
Xác định những element cần lấy.





Để có thể lấy được những element này cũng khá đơn giản, tương như khi kiểm tra link: Chuột phải 🡪 Kiểm tra.

Ví dụ:



Ta thấy tên sản phẩm nằm trong **class title**.

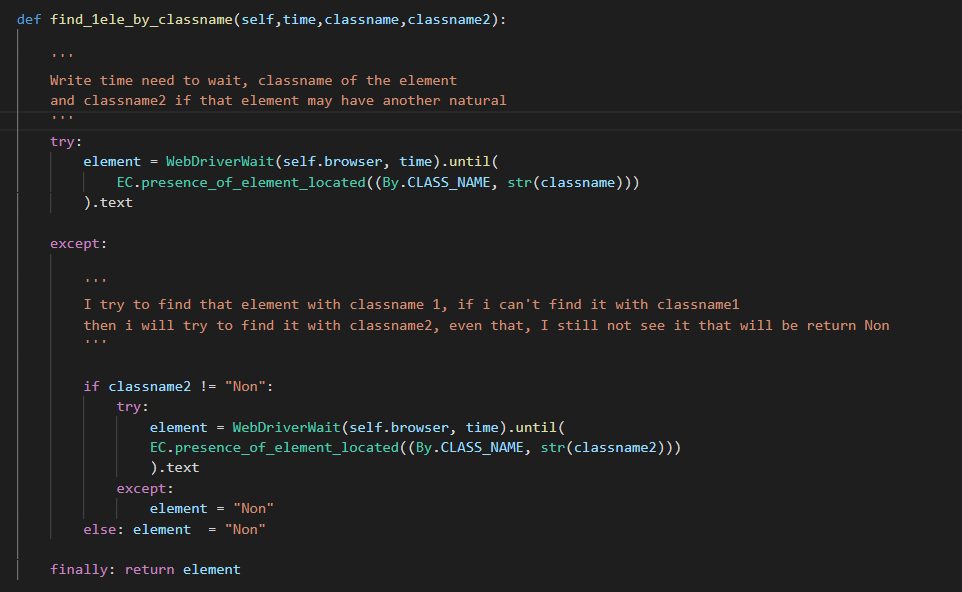
Ở đây tôi sẽ xây dựng hàm để lấy các element có thuộc tính class.

Đầu tiên, với selenium thì chương trình phải tìm thấy element ở trong trình duyệt, nếu không thể tìm thấy element thì chường trình sẽ kết thực bởi vậy tôi sẽ sử dụng **try.** Như vậy tôi sẽ thư tìm kiếm element đó trên trình duyệt, nếu thông thể tìm thấy nó tôi sẽ tìm nó với một class name khác (nếu element đó có thể xuất hiện ở 2 trường hợp với 2 classname khác nhau) hoặc tôi sẽ gán nó với giá trị Non và ngầm hiểu rằng ở sản phẩm đó, biến đó không có.

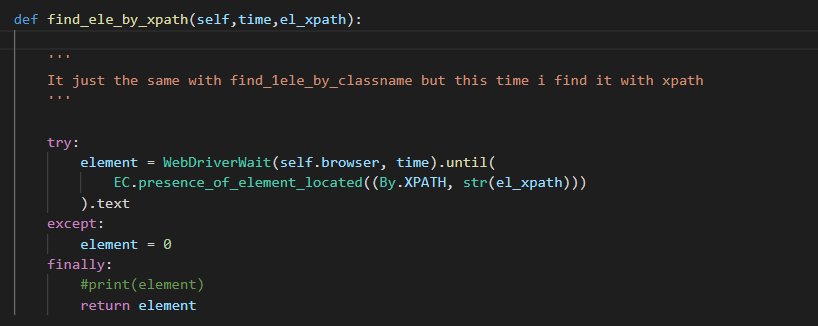
Có một trường hợp khác khiến cho chương trình không thể tìm thấy được element của chúng ta nữa đó là do web vẫn chưa kịp load thì chương trình đã tiến hành tìm kiếm rồi dẫn đến không tìm ra được element.

Khắc phục vấn đề này thì WebDriveWait là một biện pháp tốt đảm bảo tiết kiệm thời gian hơn.

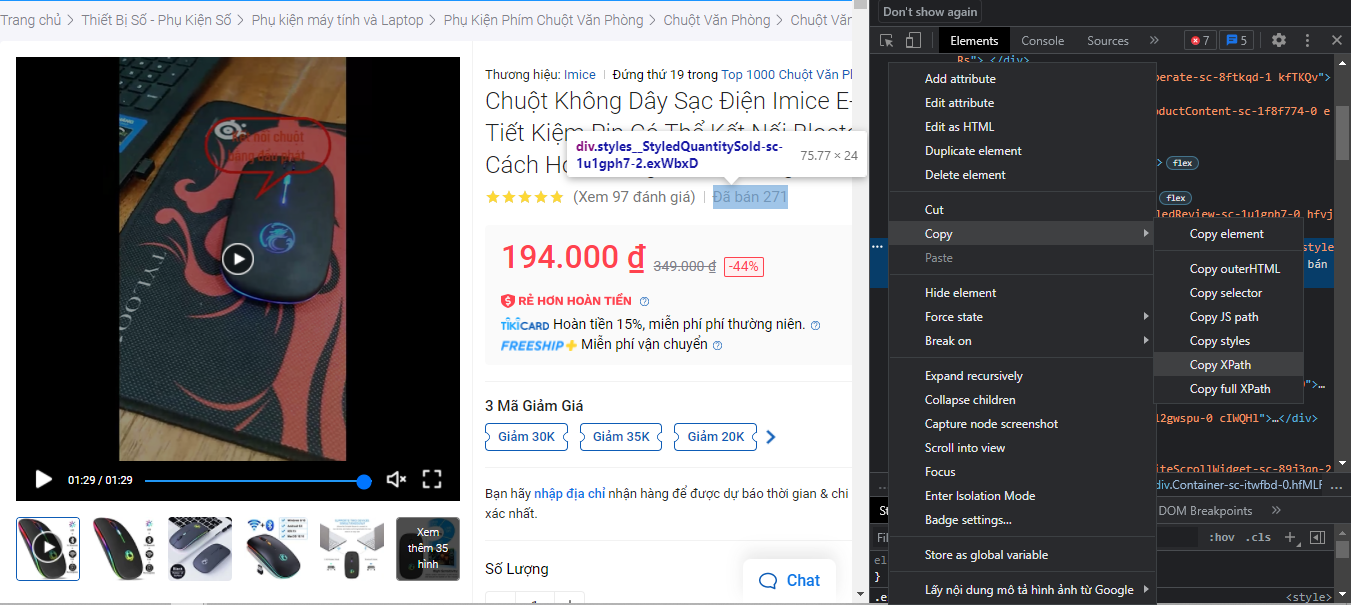
WebDriverWait sẽ chờ trng 1 thời gian nhất định nếu như element đó xuất hiện thì chương trinh sẽ tiếp tục, nếu hết thời gian mà element đó vẫn chưa xuất hiện nghĩa là element đó không tồn tại.



Đối với một số element không có các thuộc tính như classname, id thì chúng ta chỉ có thể tìm chúng thông qua xpath. Nó cũng tương tự như tìm element bằng classname nhưng thay classname thành xpath



Để thể lấy được xpath của 1 element. Nhấp chuột phải vào element rồi tiếp tục nhấp chuột phải vào dòng code thể hiện element đó rồi chọn copy xpath hoặc copy full xpath.



Riêng để lấy được Rate và các đánh giá từ 5 đến 1 sao và Bảo hành khá đặc biệt.

def get\_rate\_star(self,time,string):

        '''

        I create a list(star\_list1) to check what i get after try to find all element with that classname

        If I can't find anything that mean nobody had been rated is yet or that shop don't give anywway to warranty so it will be [0,0,0,0,0].

        if i find 4 element or lower with that classname , it must be warranty.

        '''

        star\_list1 = []

        self.browser.execute\_script("window.scrollTo(0, 2350)")

        try:

            star = WebDriverWait(self.browser, time).until(

                EC. presence\_of\_all\_elements\_located((By.CLASS\_NAME, str(string)))

            )

            # Add all a found to list

            def rate\_star\_list(star):

                if len(star) > 0:

                    for i in star:

                        star\_list1.append(i.text)

                return star\_list1

            #print("done Try")

            rate\_star\_list(star)

            #print(star\_list1)

            #star = self.browser.find\_elements\_by\_class\_name(str(string)).text

        except:

            star\_list1 = [0,0,0,0,0]

        finally:

            def last\_star\_list(star\_list = []):

                #print("len", len(star\_list1))

                if len(star\_list1) == 5:

                    # this mean that i was found rate star

                    star\_list = star\_list1

                elif (len(star\_list1) == 4) :

                    # this mean that i was found warranty

                    star\_list = star\_list1

                elif len(star\_list1) < 4 :

                    '''

                    This mean that shop given warranty information less than i want

                    But it will alwway have warranty time so i push it right the fist

                    '''

                    for i in star\_list1:

                        star\_list.append(i)

                    for i in range(4 - len(star\_list1)):

                        star\_list.append(0)

                return star\_list

            #last\_star\_list()

            def show\_rate\_star(star\_list):

                #print("list ")

                for i in star\_list:

                    print(i)

                return

            #show\_rate\_star(last\_star\_list())

        return last\_star\_list()

Bởi vì đánh giá và bảo hành điều được viết liên tiếp tạo thành 1 list nên cần lấy tất cả các element cùng xpath (khi lấy về dữ liệu sẽ ở dạng list) và tách dữ liệu ra thành từng biến để đưa vào bảng dữ liệu.

Ta thấy nếu lấy được list gồm 5 element thì đó phải là các lượt đánh giá

Lấy được từ 3 element trở xuống (trong các chính sách bảo hành đôi khi chỉ có 2 trường hoặc chỉ có thời gian bảo hành) thì đó chính là các chính sách bảo hành.

Đặc biệt với các chính sách bảo hành thì dù có bao nhiêu trường thì thời gian bảo hành luôn nằm trên cùng nên ta có thể thấy nếu chỉ lấy được 1 element thì đó chắc chắn là thời gian bảo hành.

Tôi sẽ thêm các giá trị 0 vào những vị trí còn lại và khi đọc dữ liệu tôi sẽ hiểu những biến có giá trị 0 chính là những biến không có khi lấy dữ liệu.

Trường hợp list không có element nào thì có thể sản phẩm đó không có đánh giá hoặc không có chính sachs bảo hành nào. Vậy tôi sẽ gán nó là 1 list gồm 5 giá trị 0, nếu đó là lượt đánh giá thì nó sẽ nhận 5 giá trị không vì chưa có ai đánh giá, nếu nó là chính sách bảo hành thì cũng không sao vì chính sách bảo hành sẽ lấy 3 biến đầu tiên của list vậy vẫn là 3 biến 0 và cũng được hiểu như không có chính sách bảo hành.

Tiếp theo ta tạo thêm 1 file code để lấy text từ những element.

Tất nhiên là ta vẫn cần import vào các thư viện.

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

from selenium.webdriver.chrome.options import Options

from get\_items import Mouse\_Tiki\_Crawler

from time import sleep

import random

import pandas as pd

df = pd.read\_csv("mouse\_crawler\\mouse24link.csv")

Sử dụng thư viện pandas để đọc fil link sản phẩm đã lấy được.

class start\_craw:

    def \_\_init\_\_(self,df):

        self.df = df

    def get\_link(self):

        self.df[["Name", "Price","Product\_price","Discount", "Shop", "Brand", "Sold", "Warranty\_time",

                 "Warranty\_way", "Warranty\_place", "Rate", "Comment", "5", "4", "3", "2", "1","More\_inf"]] = None

        print(self.df)

        return self.df

Tạo sẵn các trường dữ liệu để có thể điền vào sau khi craw element.

Cuối cùng là viết hàm đưa các Element vào

    def craw(self):

        n = 0

        self.df = SC.get\_link()

        cls2 = "Non"

        for i in df["m\_link"]:

            MTC = Mouse\_Tiki\_Crawler(df, i, n)

            MTC.start\_url()

            self.df["Name"][n] = MTC.find\_1ele\_by\_classname(10,"title",cls2)

            self.df["Price"][n] = MTC.find\_1ele\_by\_classname(10,"product-price\_\_current-price","flash-sale-price")

            pro\_p = MTC.find\_1ele\_by\_classname(0,"product-price\_\_list-price",cls2)

            if pro\_p == 0:

                self.df["Product\_price"][n] = self.df["Price"][n]

            else: self.df["Product\_price"][n] = pro\_p

            self.df["Discount"][n] = MTC.find\_1ele\_by\_classname(0,"product-price\_\_discount-rate",cls2)

            self.df["Shop"][n] = MTC.find\_1ele\_by\_classname(10,"seller-name",cls2)

            self.df["Brand"][n] =  MTC.find\_1ele\_by\_classname(0,"brand-and-author",cls2)

            warranty = MTC.get\_rate\_star(2,"itemRight")

            self.df["Warranty\_time"][n] = warranty[0]

            self.df["Warranty\_way"][n] = warranty[1]

            self.df["Warranty\_place"][n] = warranty[2]

            self.df["Sold"][n] = MTC.find\_ele\_by\_xpath(2,"/html/body/div[1]/div[1]/main/div[3]/div[1]/div[3]/div[1]/div[2]/div[2]")

            MTC.script\_page()

            self.df["More\_inf"][n] = MTC.find\_ele\_by\_xpath(2,"content has-table")

            self.df["Rate"][n] = MTC.find\_1ele\_by\_classname(2,"review-rating\_\_point",cls2)

            self.df["Comment"][n] = MTC.find\_1ele\_by\_classname(2,"review-rating\_\_total",cls2)

            star = MTC.get\_rate\_star(2,"review-rating\_\_number")

            self.df["5"][n] = star[0]

            self.df["4"][n] = star[1]

            self.df["3"][n] = star[2]

            self.df["2"][n] = star[3]

            self.df["1"][n] = star[4]

            MTC.browser.close()

            n += 1

Các element dược đưa vào kèm theo đó là những classname, xpath chứa element đó, thời gian chờ element xuất hiện.

Và công việc cuối cùng chính là đưa các element vào fie và lưu thành csv.

        def Create\_file(data):

            f\_name = input("file name:")

            path = "C:\\Users\\Admin\\Desktop\\PTDL\_R\\mouse\_crawler\\" + str(f\_name) + ".csv"

            data.to\_csv(path)

            return print("Your file has been saved in", path)

        Create\_file(self.df)

        Return

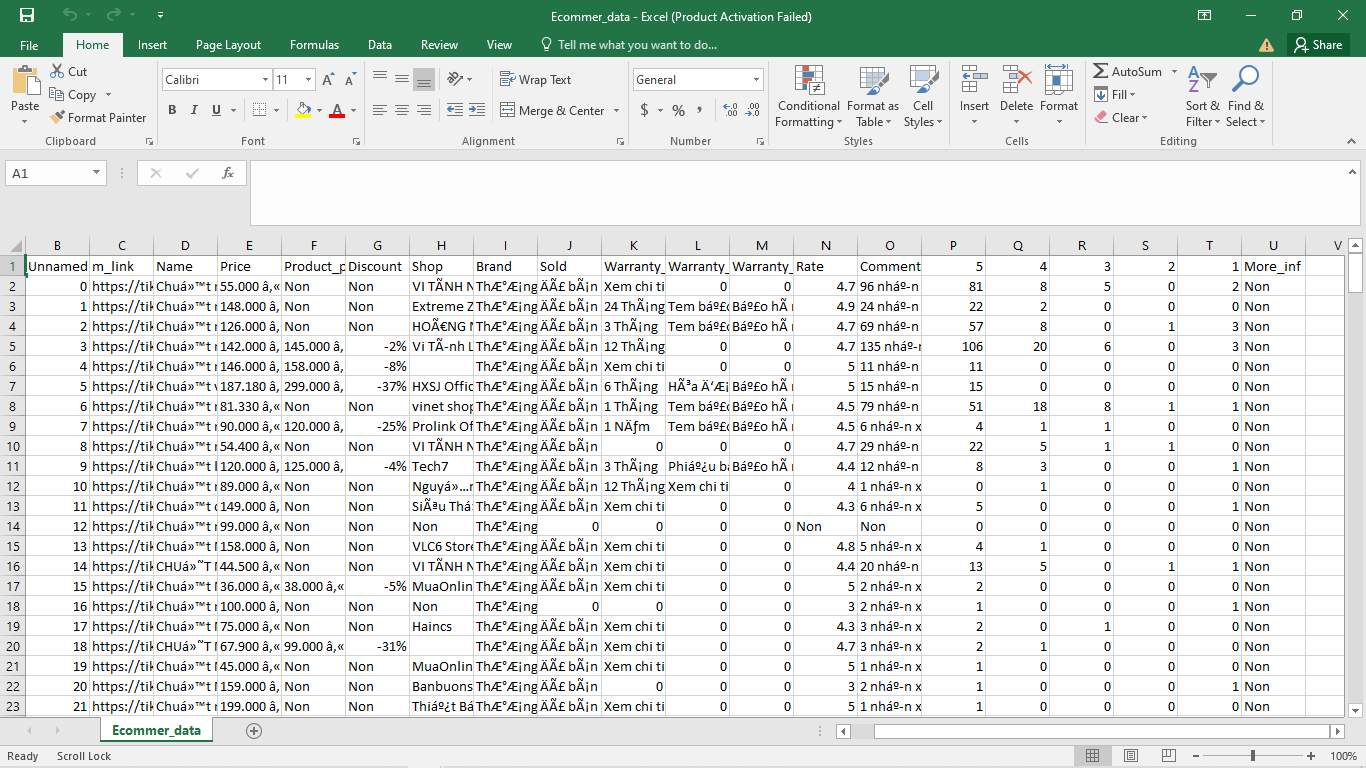
SC = start\_craw(df)

SC.craw()

Những element tôi đã lấy được:

* Name: Tên sản phẩm
* Brand: Thương hiệu
* Discount: Giảm giá
* Price: Giá bán
* Shop: tên shop
* Sold: Số sản phẩm đã bán
* Warranty: Thời gian bảo hành
* Warranty\_way: Phương thức bảo hành
* Warranty\_place: Nơi bảo hành
* Rate: Đánh giá sản phẩm
* Comment: Lượt đánh giá sản phẩm
* X5: Lượt đánh giá 5 sao
* X4: Lượt đánh giá 4 sao
* X3: Lượt đánh giá 3 sao
* X2: Lượt đánh giá 2 sao
* X1: Lượt đánh giá 1 sao

Kết quả thu được:

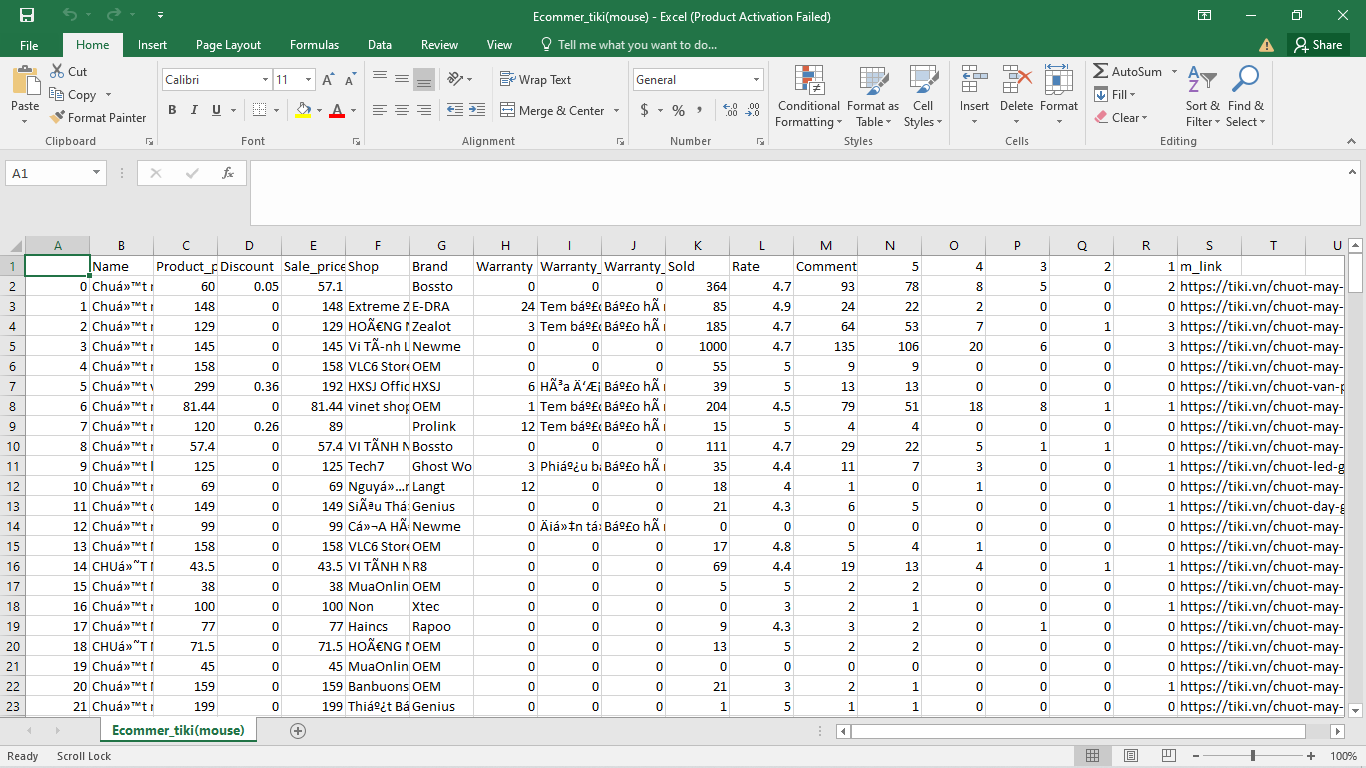


File đọc với nodepad:



Dữ liệu khi craw về vẫn còn cần xử lý tiếp, sau khi clearn data thì mới có thể tiến hành phân tích dữ liệu.

Data sau khi xử lý sẽ được bảng như sau:



# **Chương II: Phân tích dữ liệu bằng R**

## 1.Mục tiêu

* Sử dụng R để tiến hành phân tích dữ liệu
* Hiểu về dữ liệu, các thông số, biến định tính, định lượng
* Đưa ra được các biểu đồ trực quan
* Thực hiện các phép kiểm định và đưa ra nhận định về dữ liệu

## 2. Phân tích dữ liệu

### 2.1 ĐỌc Dữ liệu, kiểm tra thông số dữ liệu (Data information).

Trước hết ta sẻ tải và gọi các thư viện cần dùng:

# packages

#install.packages("ggcharts")

#install.packages("rhandsontable")

#install.packages("gridExtra")

#install.packages("Hmisc")

#install.packages("tidyverse")

#install.packages("plyr")

#install.packages("corrplot")

#install.packages("ggpubr")

#install.packages("ggstatsplot")

#install.packages("palmerpenguins")

library(ggstatsplot)

library(palmerpenguins)

library(ggpubr)

library(tidyverse)

library(Hmisc)

library(rhandsontable)

library(readxl)

library(plyr)

library(corrplot)

# turn off warning

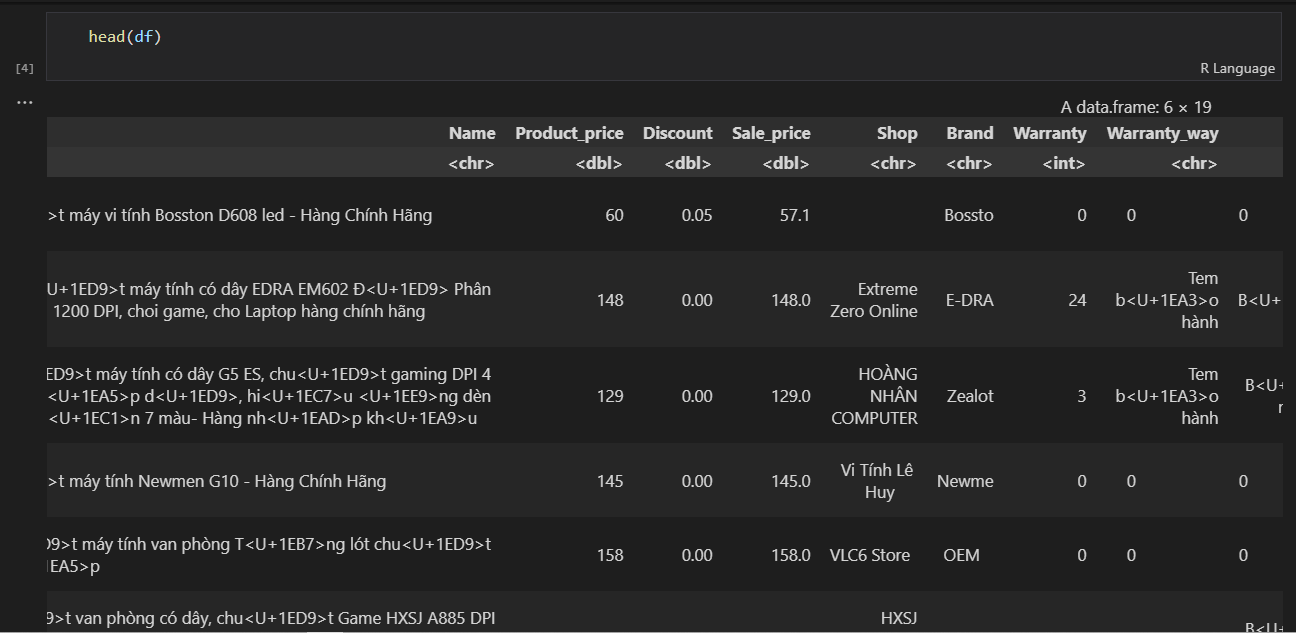
options(warn=0)

Tôi đã đưa file dữ liệu này lên github nên ở đây tôi sẽ đọc trực tiếp nó từ github, như vậy thì bất cứ ai khi tải source code này về đều có thể đọc ngay được mà không cần tải file dữ liệu.

url <-"https://raw.githubusercontent.com/ThanhHung2112/PTDL\_R/main/mouse\_crawler/Ecommer\_tiki(mouse).csv"

df <- read.csv(url, header = TRUE, quote = "\"",  fill = TRUE, comment.char = "", encoding = "UTF-8")

Bảng dữ liệu với 19 cột và hiện tại chỉ hiện 6 dòng(6x19).



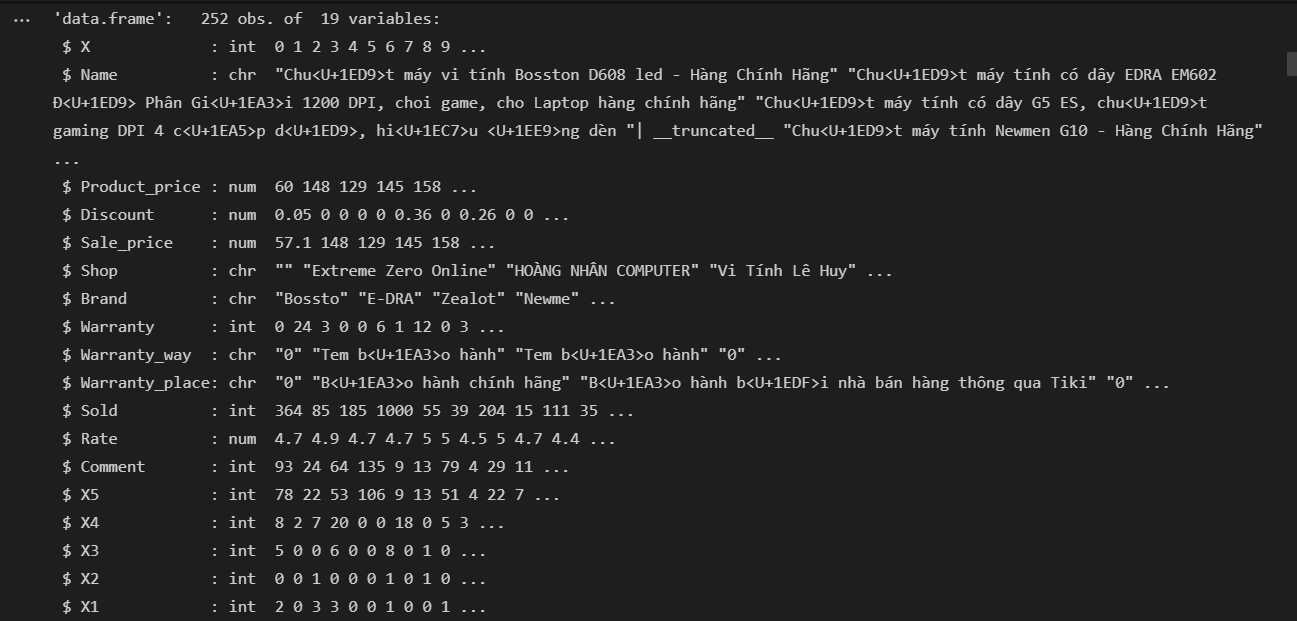
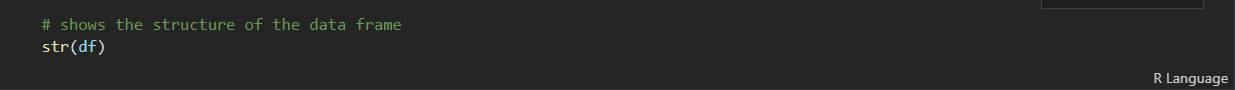
Data Information(Thông số dữ liệu)

Sau khi mở file dữ liệu chúng ta sử dụng một số hàm để kiểm tra dữ liệu và xác định tính chất của dữ liệu

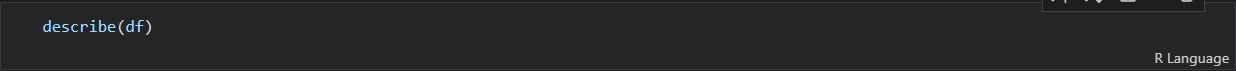
Các hàm sau đây sẽ cho chúng ta thông tin về việc:

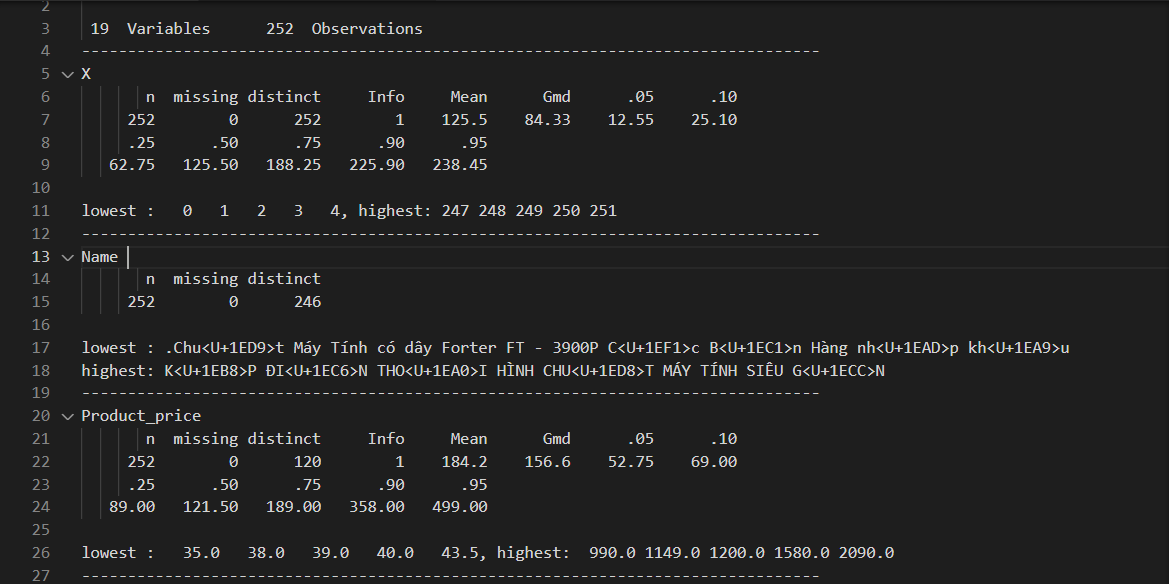
* Missing data (thiếu dữ liệu)
* Số cột, dòng
* Thuộc tính dữ liệu( số, ký tự)

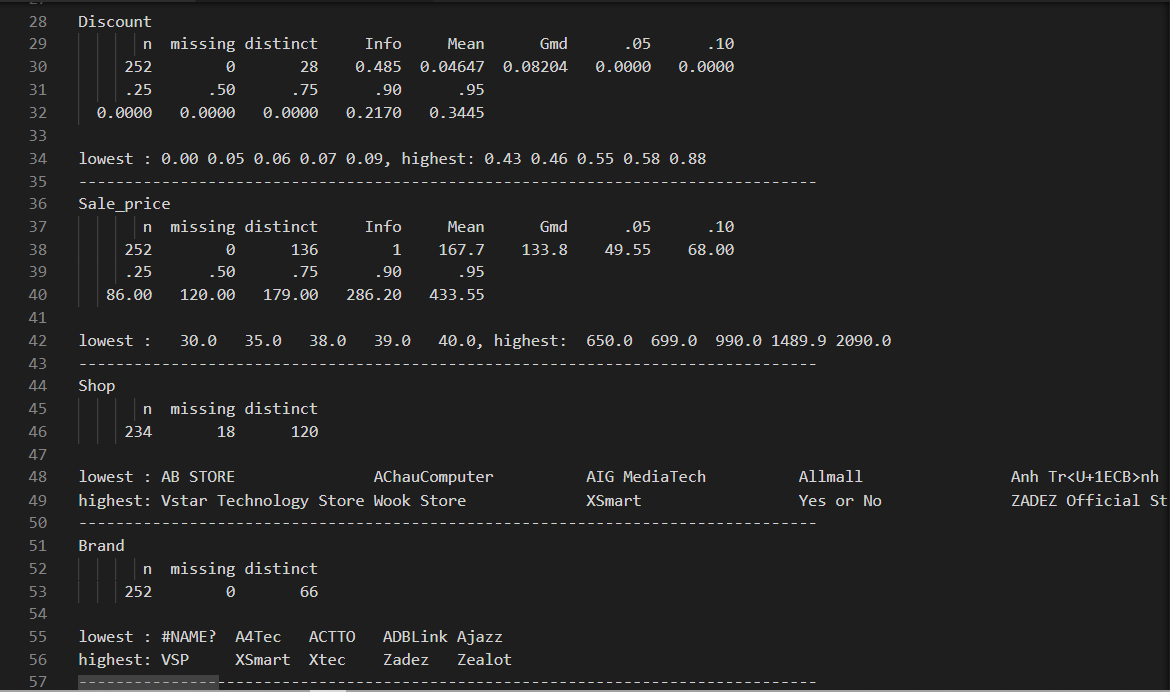
Từ đó, ta có thể dễ dàng xác định các biến định tính và định lượng.



Hàm str() Cho ta biết số lượng hàng(252) và cột(19), cùng loại dữ liệu của các trường dữ liệu.







Hàm decriable() cho ta thông tin về nhiều vấn đề của toàn tập dữ liệu dựa theo kiểu dữ liệu của mỗi trường dữ liệu.

Sau khi kiếm tra thông số dữ liệu ta thấy răng không có dữ liệu bị thiếu hụt trong data frame. Bây giờ, tôi sẽ tạo thêm 3 cột thể hiện tính chất của dữ liệu để ta có thể hiểu hơn

* Valuation (Định giá) : gồm 3 phân loại: low price, good price, hight price( Giá cao, Giá tốt, Giá thấp) thể hiện tính chất của giá bán
* Liquidity (Thanh khoản) : gồm 3 phân loại: Fast, Good, Slow (Nhanh, tốt, chậm) thể hiện xu thế của chuột máy tính trên thị trường
* Quality (Chất lượng) : gồm 3 phân loại: Highquality, Goodquality, Lowquality, Unknow (Chất lượng cao, Chất lượng tốt, Chất lượng thấp, Chưa xác định)thể hiện chất lượng của sản phẩm dựa trên lượt đánh giá.

Với Valuation(Định giá) dựa trên thông tin ở mục Data information ta chia ra làm 3 mức.

* Dưới 100k: Low price( Giá thấp)
* Từ 100 --> 400k : Good price (Giá tốt)
* Trên 400k : High price( Giá cao)

Valuation <- list()

for (i in df[,c('Sale\_price')]) {

    if (i < 100){

        Valuation <- c(Valuation,'low price')

    } else if (i < 400) {

        Valuation <- c(Valuation,'good price')

    } else {

        Valuation <- c(Valuation,'high price')

    }

}

Tương tự với Liquidity (Thanh khoản) ta cũng dựa vào Data Information và chia làm 3 mức:

* Slow(chậm) : Số lượng sản phẩm bán ra < 30
* Good(tốt) : Số lượng sản phẩm bán ra từ 30 -> 300
* Fast(nhanh) : Số lượng sản phẩm bán ra từ 300

Liquidity <- list()

for (i in df[,c('Sold')]) {

    if (i < 30){

        Liquidity <- c(Liquidity,'Slow')

    } else if (i < 300) {

        Liquidity <- c(Liquidity,'Good')

    } else {

        Liquidity <- c(Liquidity,'Fast')

    }

}

Cuối cùng là cột Quality(Chất lượng) dựa trên đánh giá của người tiêu dùng.

Với những sản phẩm chưa có đánh giá thì ta chưa thể xác định rõ chất lượng nên sẽ thuộc loại "Unknow" (Chưa xác định)

Với những sản phẩm có lượt đânhs giá từ dười 15 thì:

* Điểm đánh giá trên 4.7 : Good quality (Chất lượng tốt)
* Điểm đánh giá dưới 4.7 : Low qualiy (Chất lượng thấp)

Sở  dĩ như vậy là bởi với 20 sản phẩm thì ta không thể cho rẳng sản phẩm có chất lượng cao được. Ngoài ra chỉ với 20 sản phẩm mà điểm đánh giá nhỏ hơn 4.7 tức là có những lượt đánh giá 1,2,3 khá nhiều. Với 20 lượt bán mà có lượt đánh giá 1,2,3 khá nhiều thì đây không thể là sản phẩm tốt được.

Cuối cùng, khi lượt đánh giá nhiều hơn thì cũng dễ phân loại hơn hẵn, ở đây tôi cũng sẽ phân làm 3 loại:

* Điểm đánh giá trên 4.7 : High quality (Chất lượng cao)
* Điểm đánh giá từ 4.4 - 4.6 : Good quality (Chất lượng tốt)
* Điểm đánh giá dưới 4.4 : Low qualiy (Chất lượng thấp)

Quality <- list()

n = 1

for (i in df[,c('Comment')]) {

    rate <- df[,c('Rate')][n]

    if (i == 0){

        Quality <- c(Quality,'Unknown')

    } else if (i < 20){

        if (rate > 4.7){

            Quality <- c(Quality,'Good quality')

        } else {

            Quality <- c(Quality,'Low quality')

        }

    } else {

        if (rate > 4.7){

            Quality <- c(Quality,'High quality')

        } else if (rate >= 4.4){

            Quality <- c(Quality,'Good quality')

        } else {

            Quality <- c(Quality,'Low quality')

        }

    }

    n <- n+1

}

Cuối cùng, thêm các cột trên vào dataframe với vị trí tương ứng.

df <- df %>%

  add\_column(Valuation = as.character(Valuation),

             .after = "Sale\_price")

df <- df %>%

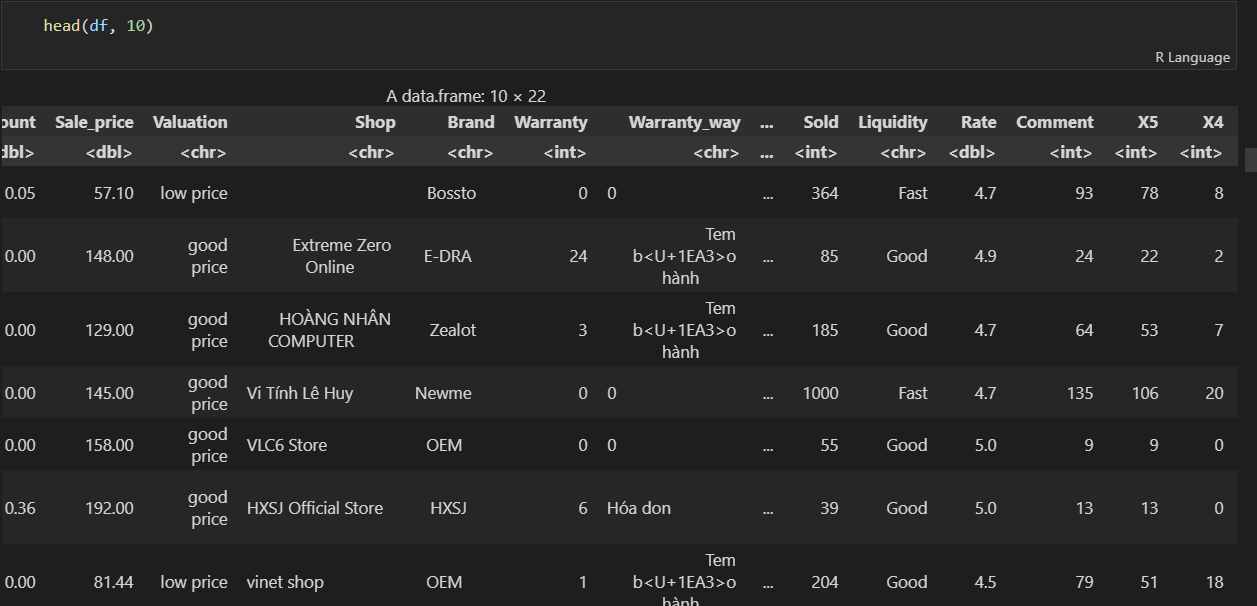
  add\_column(Liquidity = as.character(Liquidity),

             .after = "Sold")

df <- df %>%

  add\_column(Quality = as.character(Quality),

             .after  = 'Warranty\_place')

Xem lại file ta thấy đã xuất hiện thêm các trường dữ liệu mới.

### 2.2 Các phép tính hướng tâm (The center Tendency)

Ta thấy cột "Link" trong dataframe không cung cấp thông tin cho việc phân tích dữ liệu nên ta sẽ tiến hành loại bỏ cột này khỏi dataframe để tiến hành phân tích dữ liệu một cách tiện và gọn hơn

df <- subset( df, select = - m\_link)

df <- subset( df, select = - X)

Nhờ việc xem thông số dữ liệu, ta có thể dễ dàng tìm ra được các biến định tính và định lượng:

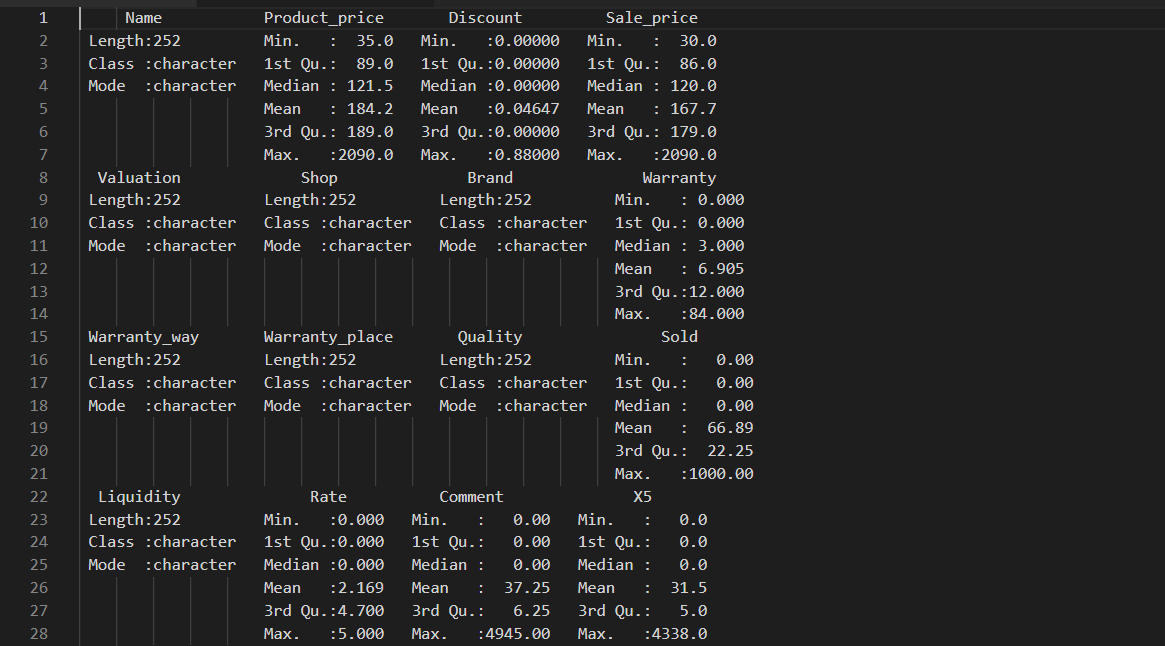
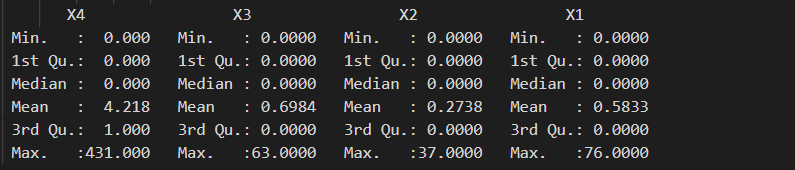
* Quantitative variable (Biến định lượng): Product\_price, Sale\_price, Discount, Warranty, Sold, Rate, Comment, 5, 4, 3, 2, 1
* Categorical variable (Biến định tính): Valuation, Brand , Warranty\_way, Warranty\_place, Liquidity

quantitative <- c('Product\_price','Sale\_price', 'Discount', 'Warranty', 'Sold', 'Rate', 'Comment', 'X5', 'X4', 'X3', 'X2', 'X1')

categorical <- c('Valuation', 'Brand' ,'Warranty\_way', 'Warranty\_place', 'Liquidity')

Sử dụng hàm summary() để xem các kết quả vầ một số phép kiểm tra dữ liệu.

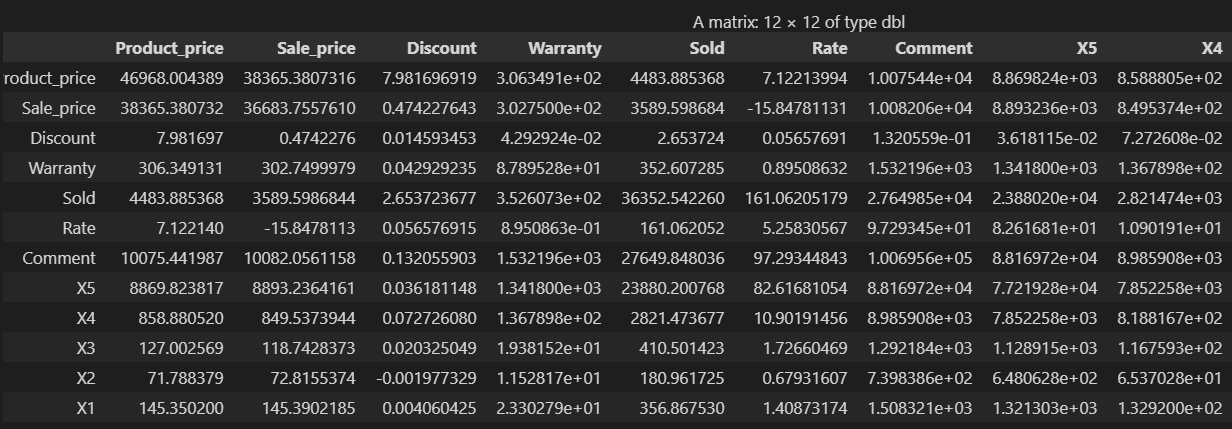
summary(df)

Bảng tính phương sai các biến định lượng.

#Varrian value(Phương sai)

var(df[,c(quantitative)])



Tính giá trị yếu vị của tập dữ liệu.

Mode <- function(dataframe){

    mode <- c()

    name <- c()

    getmode <- function(v) {

        uniqv <- unique(v)

        uniqv[which.max(tabulate(match(v, uniqv)))]

    }

    for (i in names(dataframe)){

        name <- c(name,i)

    }

    for (i in 1:length(dataframe)){

        mode <- c(mode, getmode(df[,i]))

    }

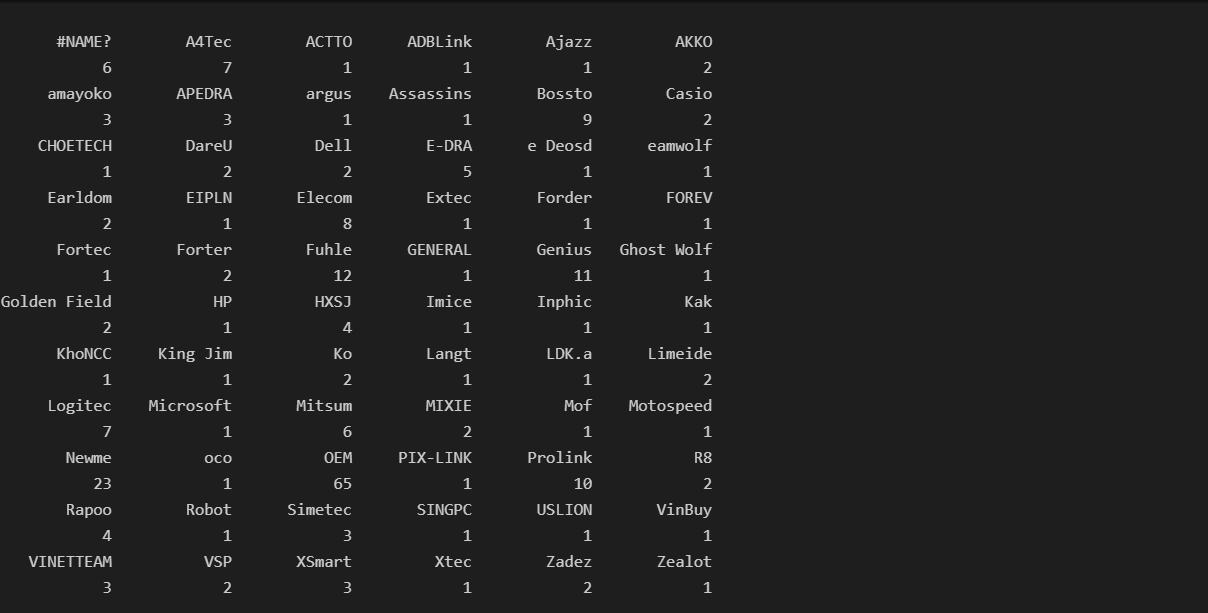
    data.frame(name,mode)

}

Mode(df)

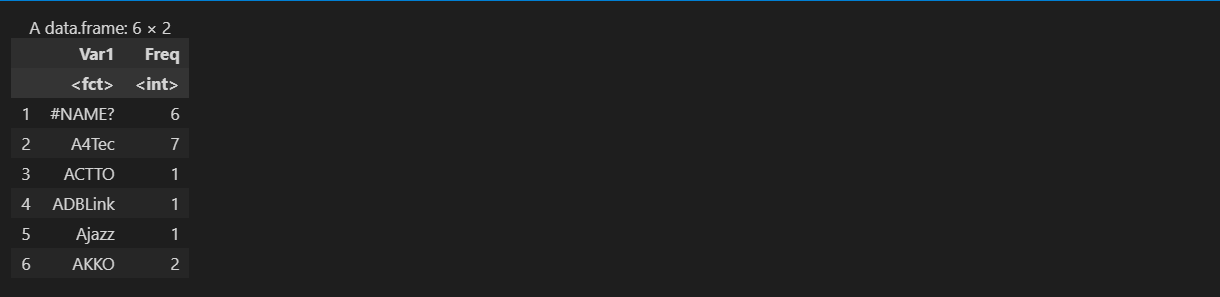
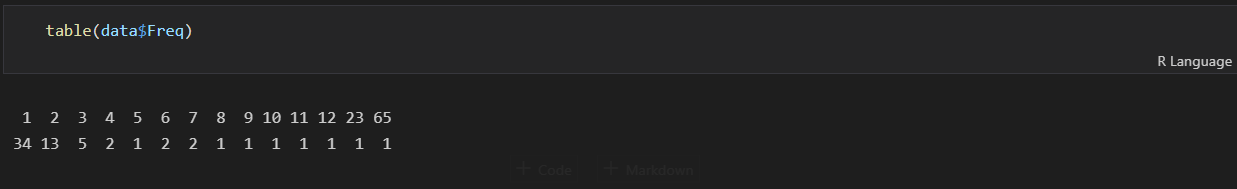


Số lượng sản phẩm của các Brand.



data = data.frame(table(df$Brand))

head(data)

Bảng giá trị Correlation

# dependencies between values(sư phụ thuộc của dữ liệu)

correlation = cor(df[,c(quantitative)])

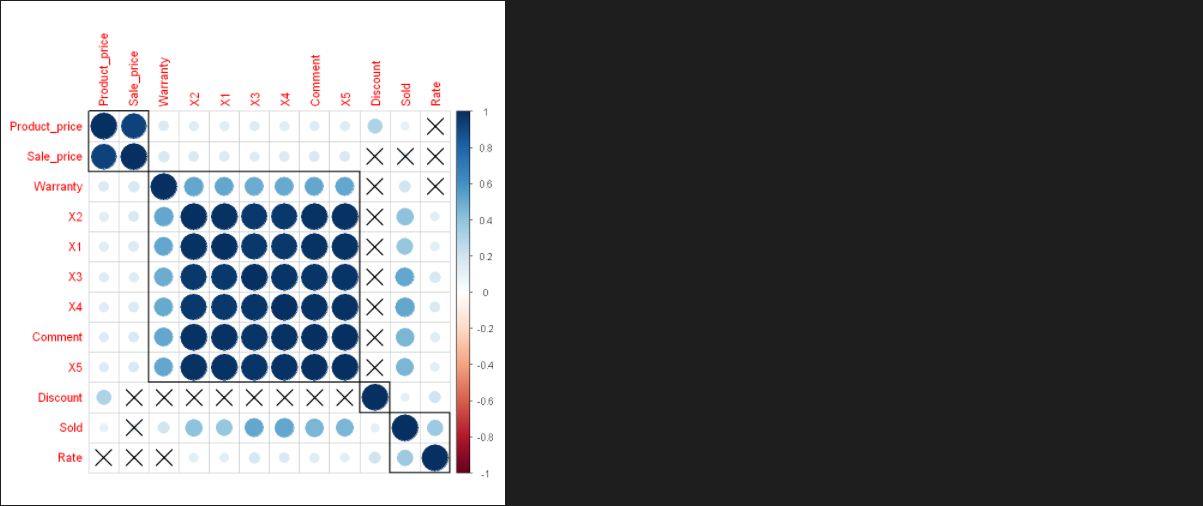
correlation



# plot of dependencies value (biểu đồ thể hiện sự phụ thuộc của dữ liệu)

testRes = cor.mtest(df[,c(quantitative)], conf.level = 0.95)

corrplot(correlation, order = 'hclust', addrect = 4,p.mat = testRes$p, sig.level = 0.10)

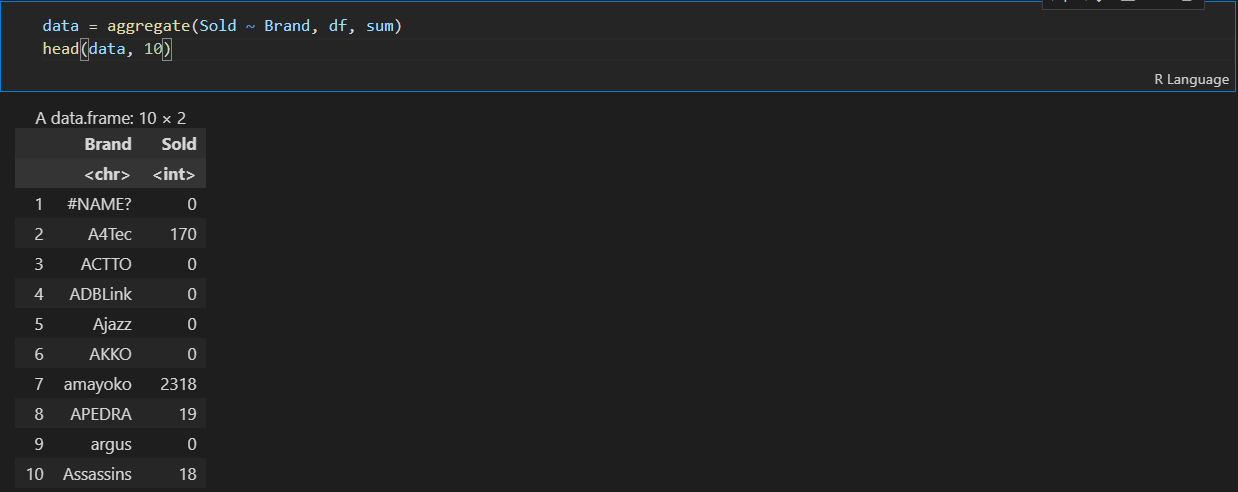


### 2.3 Trực quan hóa dữ liệu(DATa visualization)

Trực quan hóa dữ liệu là việc sử dụng các biểu đồ, hình ảnh để thể hiện dữ liệu.

Việc trực quan hóa dữ liệu giúp ta có một cái nhìn rõ ràng và thêm tổng quát hơn về dữ liệu, giúp nhìn nhận rõ ràng những vấn đề của dữ liệu và so sánh một cách dễ dàng.

**Biểu đồ thể hiện số lượng sản phẩm bán ra thị trưởng ở Tiki của các Brand**

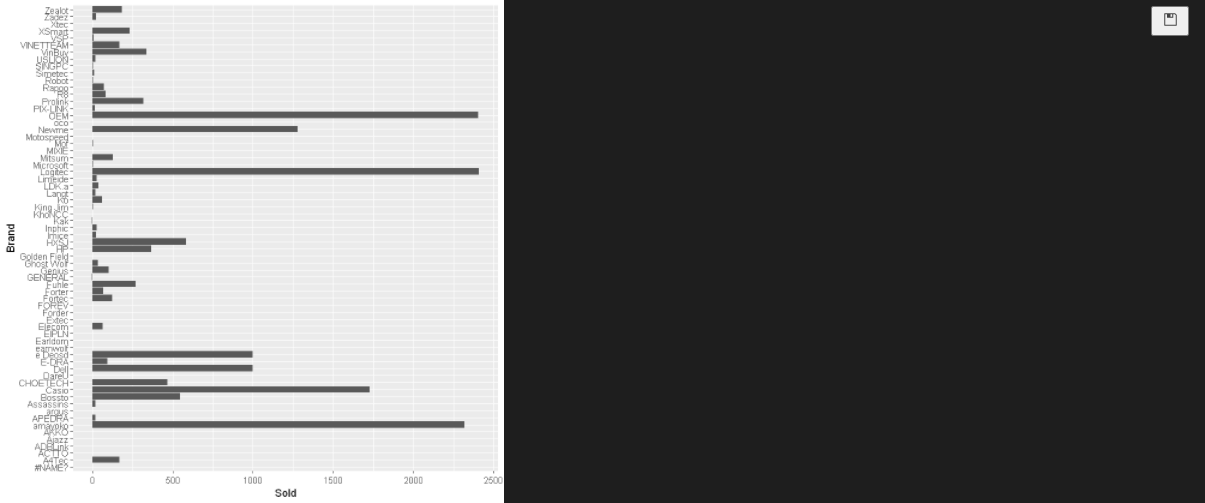
****

# Biểu đồ thể hiện số lượng sản phẩm bán được giữa các brand

ggplot(data, aes(x=Brand, y=Sold)) +

  geom\_bar(stat = "identity") +

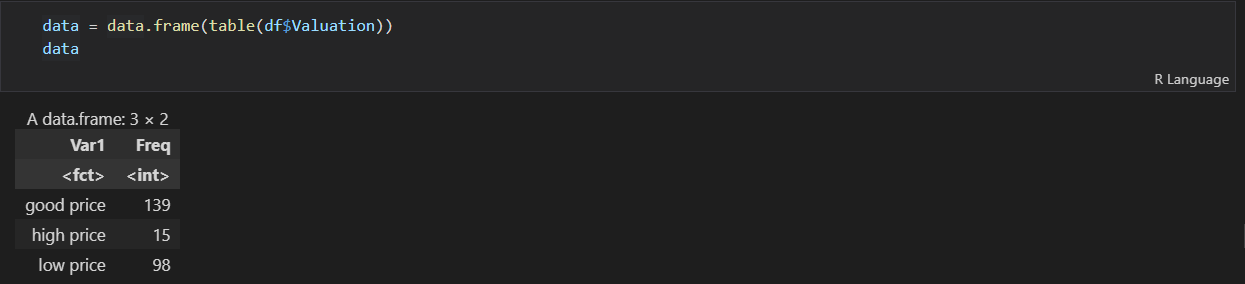
  coord\_flip()

****

**Nhận xét:**

* Từ biểu đồ ta có thể thấy được có rất nhiều Brand ở đây.Tuy nhiên giữa các Brand có sự chênh lệch rất lớn về số lượng sản phẩm bán ra.
* Dẫn đầu là các hãng rất nổi tiếng như Logitech, OEM, Amayoko, Newme... Đây đều là những thương hiệu rất nổi tiếng và có được sự tin cậy từ người dùng nên việc số lượng chuột được bán ra tập trung vào cấc hãng này cũng là điều dễ hiểu.
* Đa số các Brand khác bán được rất ít hoặc không bán được sản phẩm của mình ở Tiki.
* Ngoài ra ở tiki dữ liệu hiển thị số lượt bán ra tối da là 1000 nên dữ liệu thật tế có thể còn chênh lệch nhiều hơn nữa.

**Biểu đồ thể hiện giá bán ở Tiki**

****

barplot(height=data$Freq, names=data$Var1,

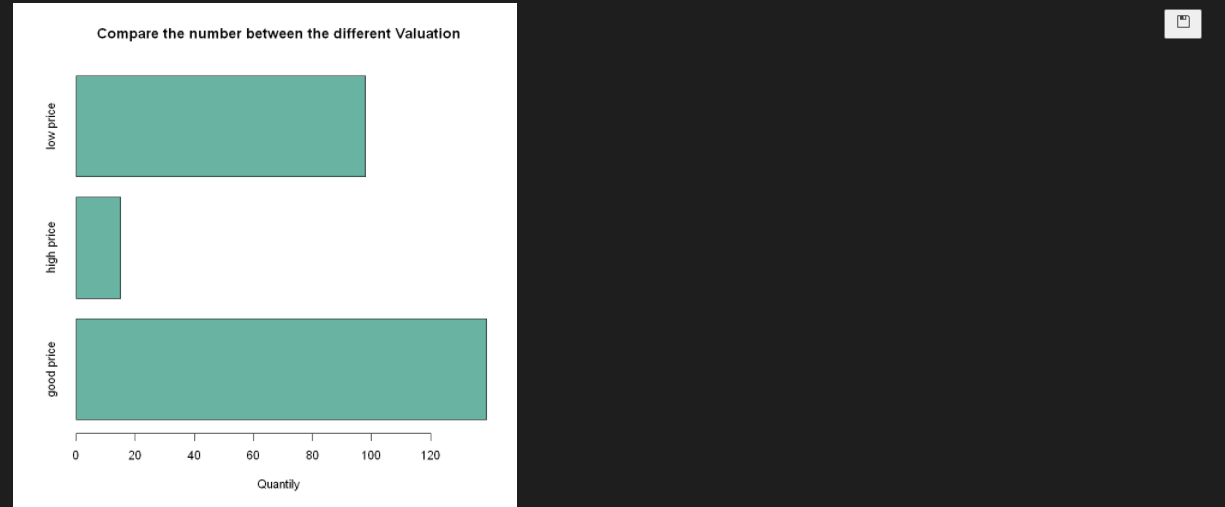
        col="#69b3a2",

        horiz=T, las=0,

        xlab="Quantily",

        main="Compare the number between the different Valuation",

        )



**Nhận xét:**

* Đa số sản phẩm đều có mức giá thấp(lowprice) hoặc giá tốt (Goodprice)
* Sản phẩm có giá cao (Highprice) có số lượng ít hơn các mức giá còn lại rất nhiều (khoảng 20 sản phẩm)
* Từ đồ thị ta thấy các mặt hàng chuột máy tính trên Tiki được bán với giá Giá mềm và chủ yếu dưới 400k

**Biểu đồ thể hiện tốc độ bán của các mặt hàng trên Tiki**

data = data.frame(table(df$Liquidity))

data

barplot(height=data$Freq, names=data$Var1,

        col="#69b3a2",

        horiz=T, las=0,

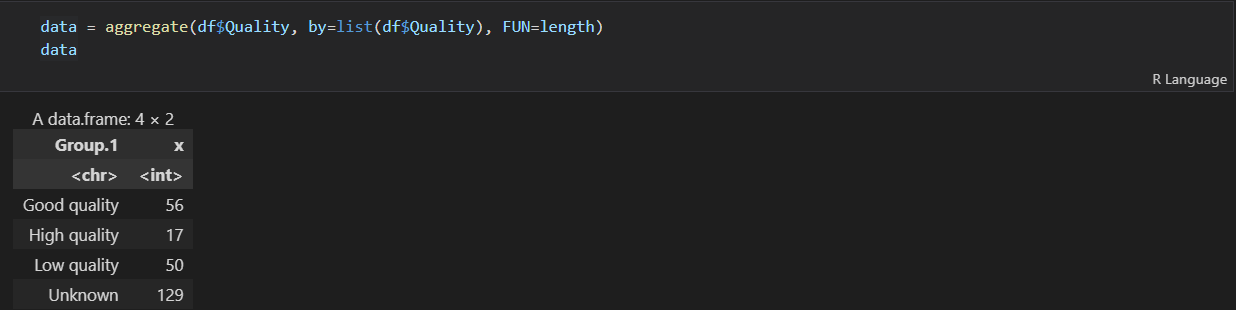
        xlab="Quantily",

        main="Compare the number between the different Liquidity",

        )

**Nhận xét:**

* Tốc độ bán càng nhanh thì số lượng càng ít
* Ở mức slow có sự chênh lệch lớn chiếm đến tận hơn 50% tổng số sản phẩm được bán ra. Xem ra khách hàng vẫn chưa thật sự tin tưởng khi mua hàng trên Tiki
* Các hãng vẫn chưa thể bán tốt sản phẩm của mình ở thị trưởng của Tiki

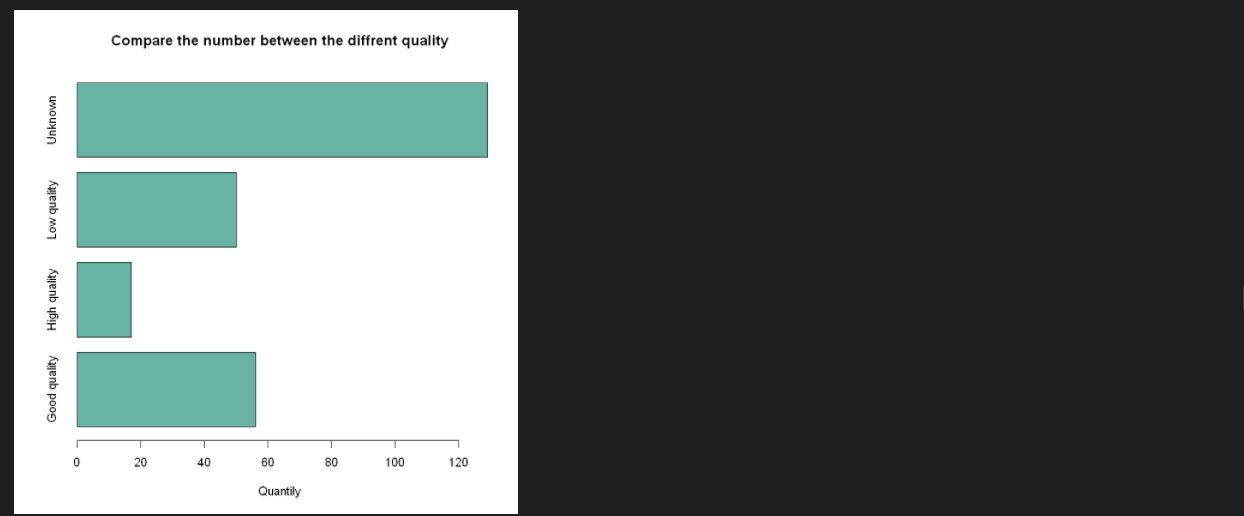
**Biểu đồ thể hiện chất lượng sản phẩm dựa trên đánh giá người dùng ở Tiki**barplot(height=data$x, names=data$Group.1,

        col="#69b3a2",

        horiz=T, las=0,

        xlab="Quantily",

        main="Compare the number between the diffrent quality",

        )****

Nhận xét:

* Mục Unknow có số lượng sản phẩm lớn hơn hẳn so với tất cả các mục chất lượng còn lại
* Số lượng sản phẩm đạt chất lượng cao ơ trên Tiki rất ít dưới 10% tổng số lượng
* Nhìn chung những sản phẩm được bán trên Tiki khi đã được đánh giá thì có tỉ lệ đạt chất lượng tốt trở lên khá cao( điểm đánh giá trên 4.4)

**Biểu đồ thể hiện thời gian bảo hành các sản phẩm ở Tiki**

counts <- data.frame(table(df$Warranty))

names(counts) <- c('Time','Number')

counts****

barplot(height=counts$Number, names=counts$Time,

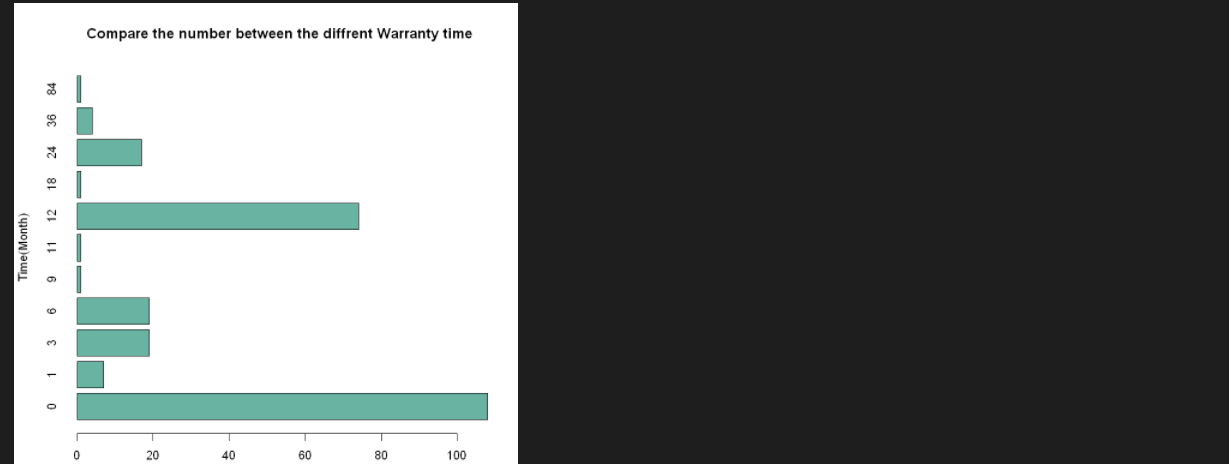
        col="#69b3a2",

        horiz=T, las=0,

        ylab="Time(Month)",

        main="Compare the number between the diffrent Warranty time",

        )

****

**Nhận xét:**

* Có rất nhiều sản phẩm được bán mà không có chính sách bảo hành kèm theo.
* Trong các sản phẩm nếu đươc bảo hành thì đa số đều bảo hành 1 năm.
* Sản phảm hưởng chính sách bảo hành cao nhất lên đến 84 tháng(7năm).

**Biểu đồ thể hiện các cách bảo hành ở Tiki**

data = data.frame(table(df$Warranty\_way))

data****

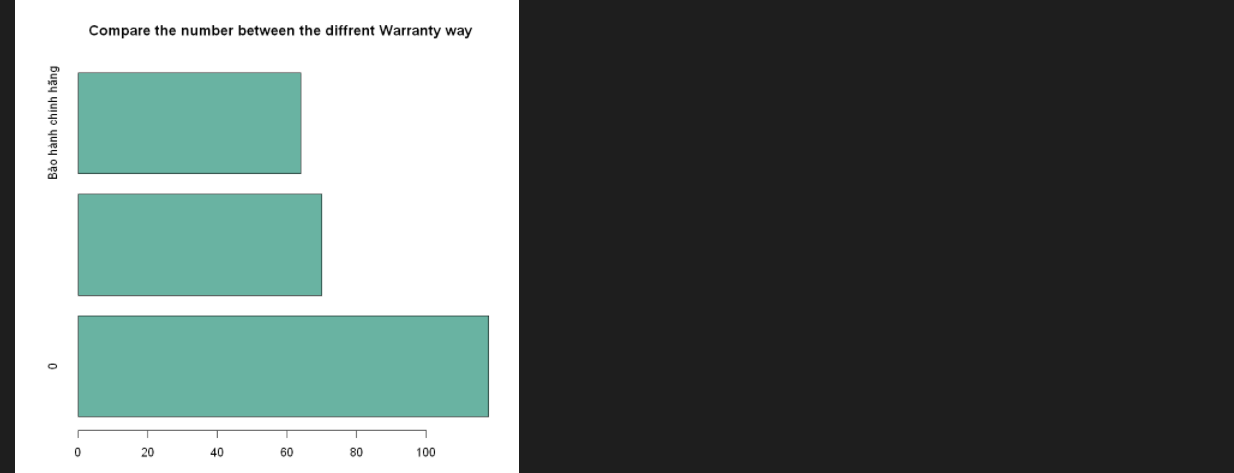
barplot(height=data$Freq, names=data$Var1,

        col="#69b3a2",

        horiz=T, las=0,

        main="Compare the number between the diffrent Warranty way",

        )

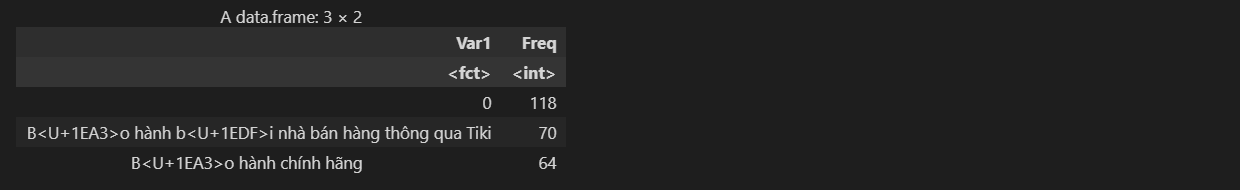


**Nhận xét:**

* Có rất nhiều sản phẩm không có hình thức bảo hành rõ ràng trên thị trường.
* Rất ít sản phẩm được bảo hành theo phiếu bảo hành, trong khi đó hình thức bảo hành phổ biến nhất là team bảo hành.
* Bảo hành dựa trên hóa đơn và team bảo hành dường như không được phổ biến do không tiện lợi

**Biểu đồ cho thấy các nơi bảo hành của Tiki**

data = data.frame(table(df$Warranty\_place))

data

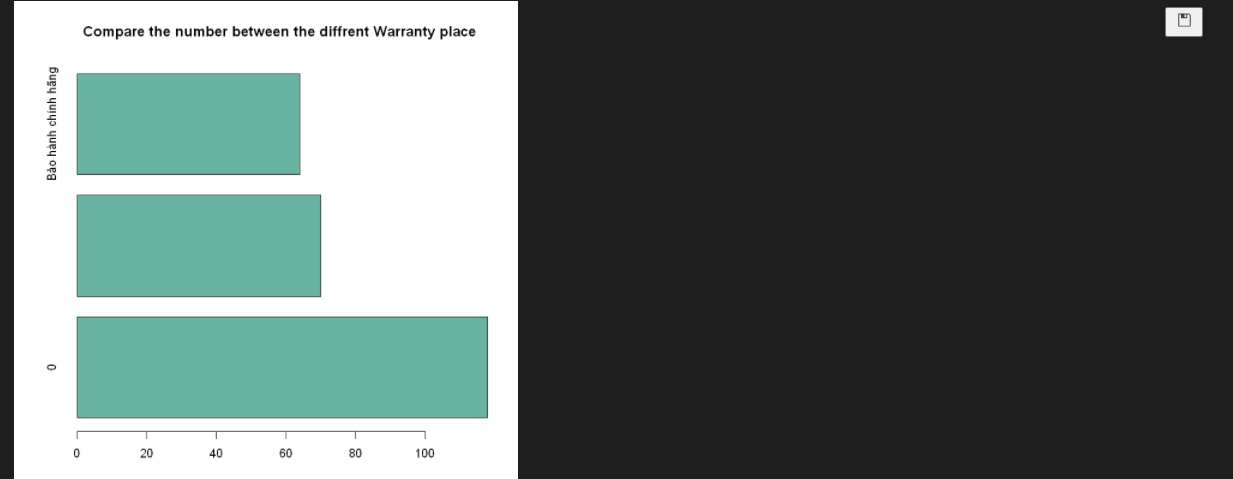
barplot(height=data$Freq, names=data$Var1,

        col="#69b3a2",

        horiz=T, las=0,

        main="Compare the number between the diffrent Warranty place",

        )



### 2.3 Các phép kiểm định

#### 2.3.1 Kiểm tra phân phối chuẩn(Normal ditribution)

Trước khi tiến hành các phép kiểm định, ta cần kiểm tra phân phối chuẩn của dữ liệu. Nếu dữ liệu không chuẩn, thì ta phải chọn những phép kiểm định phi tham số để kiểm định.

Ở đây, tôi sẽ xây dựng hàm kiểm định phân phối chuẩn trước, như vậy trc khi tiến hành kiểm định tôi có thể dễ dàng kiểm tra phân phối chuẩn và chọn một phép kiểm định khác.

#install.packages("nortest")

library(nortest)

normal\_ditribution <- function(data, col){

    # shapiro - wilk

    print(shapiro.test(col))

    # Anderson-Darling

    print(ad.test(col))

    # histplot

    histplot = ggplot(data = data,aes(col))+

    geom\_histogram(aes(y=..density..),color="black",fill="cyan")+

    stat\_function(fun = dnorm,

                    args = list(mean = mean(col,na.rm = TRUE),

                                sd = sd(col,na.rm = TRUE)),

                    color ='red',size = 1)

    print(histplot)

    # qqplot

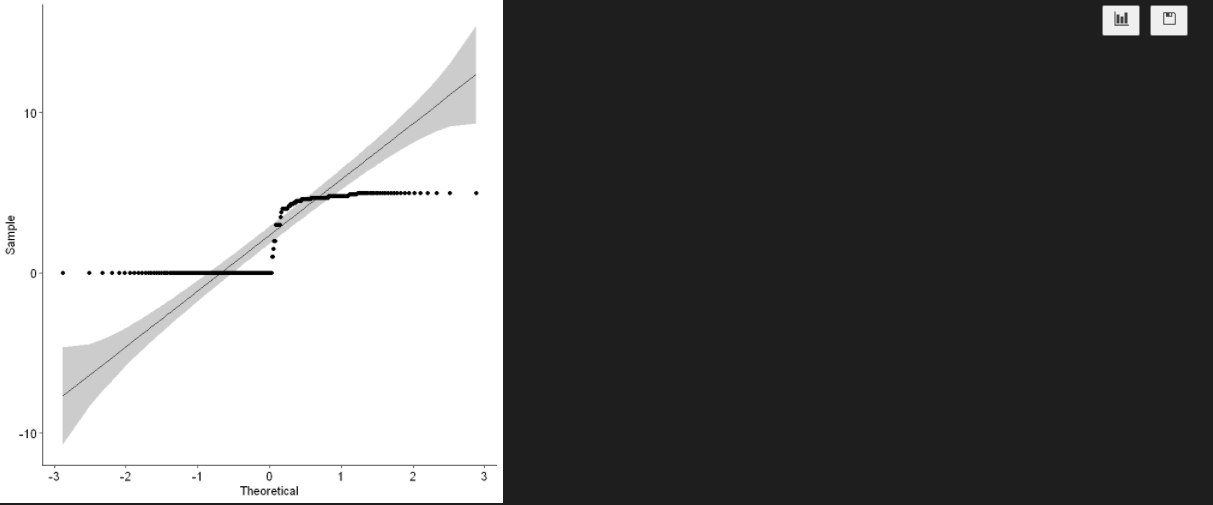
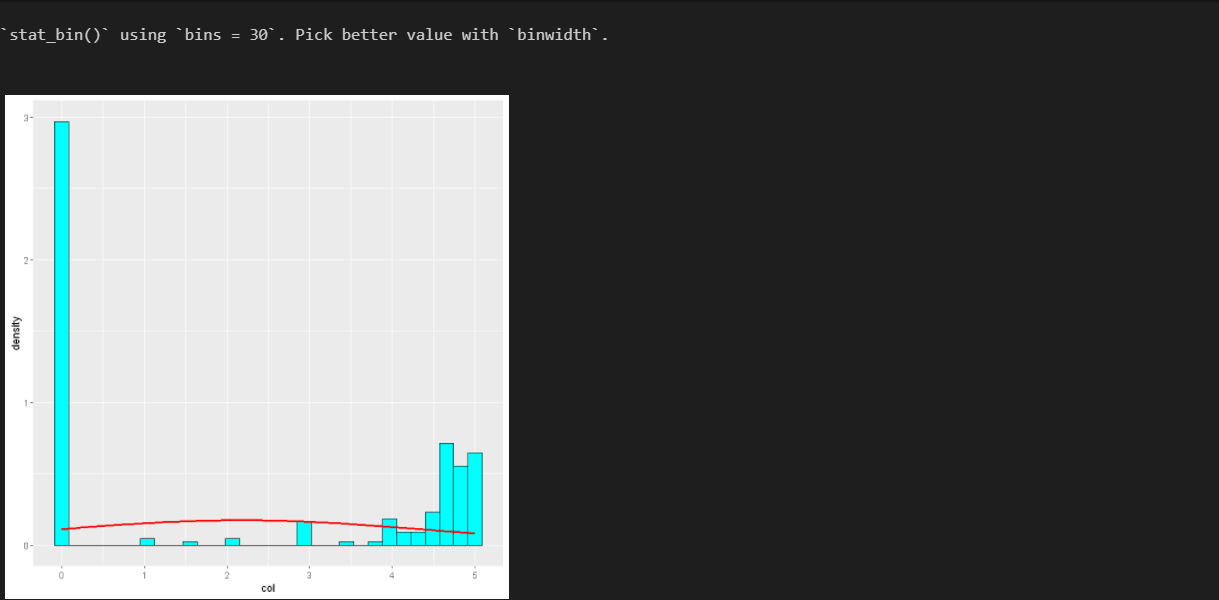
    qqplot = ggqqplot(col)

    print(qqplot)

}

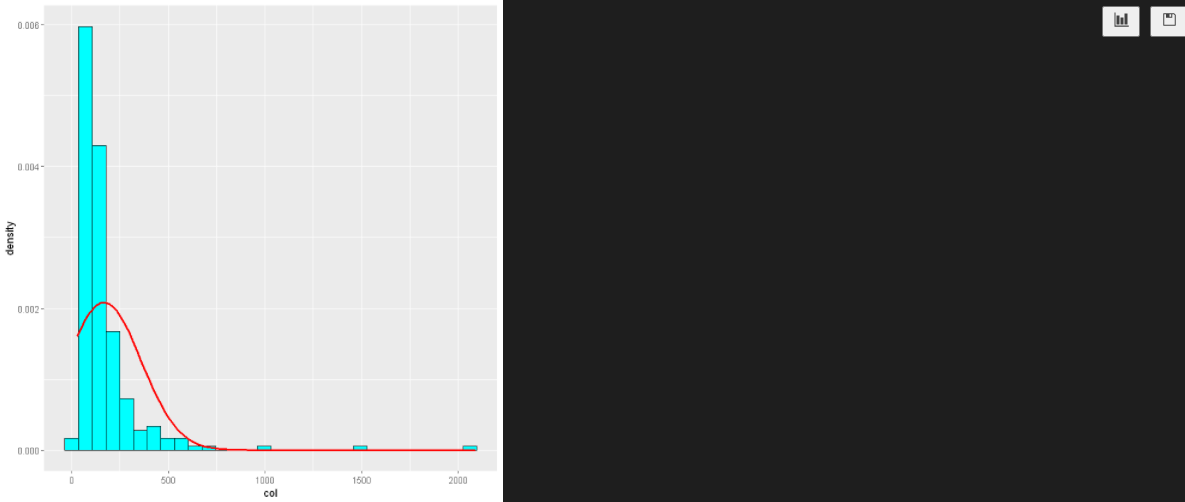
#normal\_ditribution(df,df$Rate)

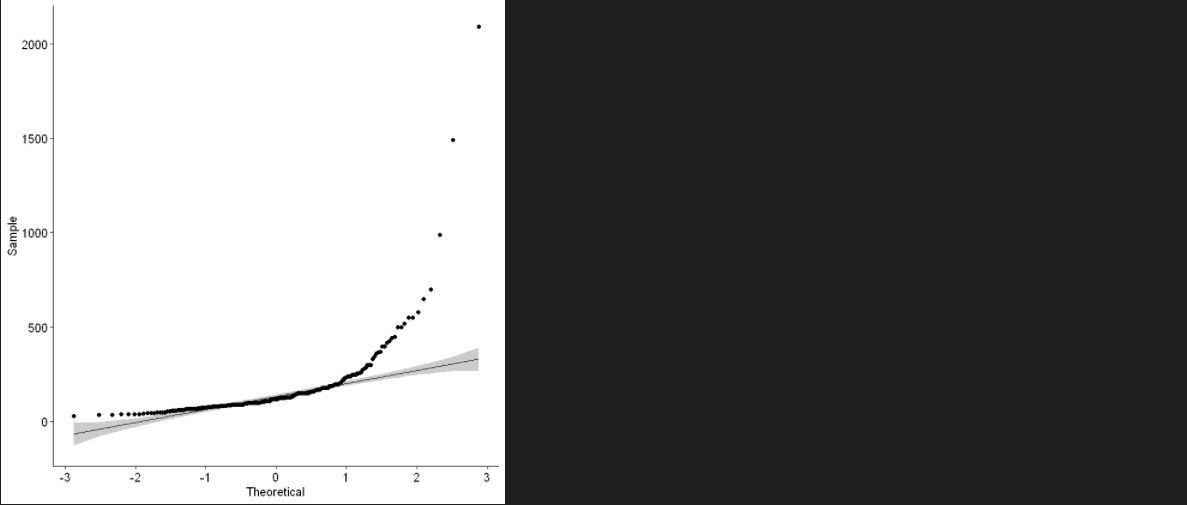
normal\_ditribution(df, df$Rate)



normal\_ditribution(df, df$Sale\_price)

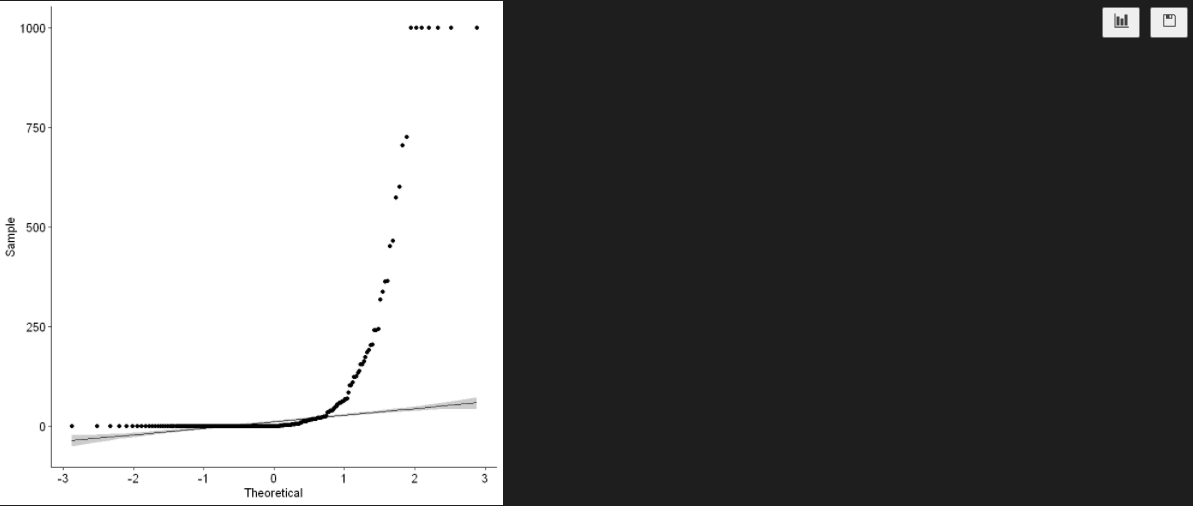
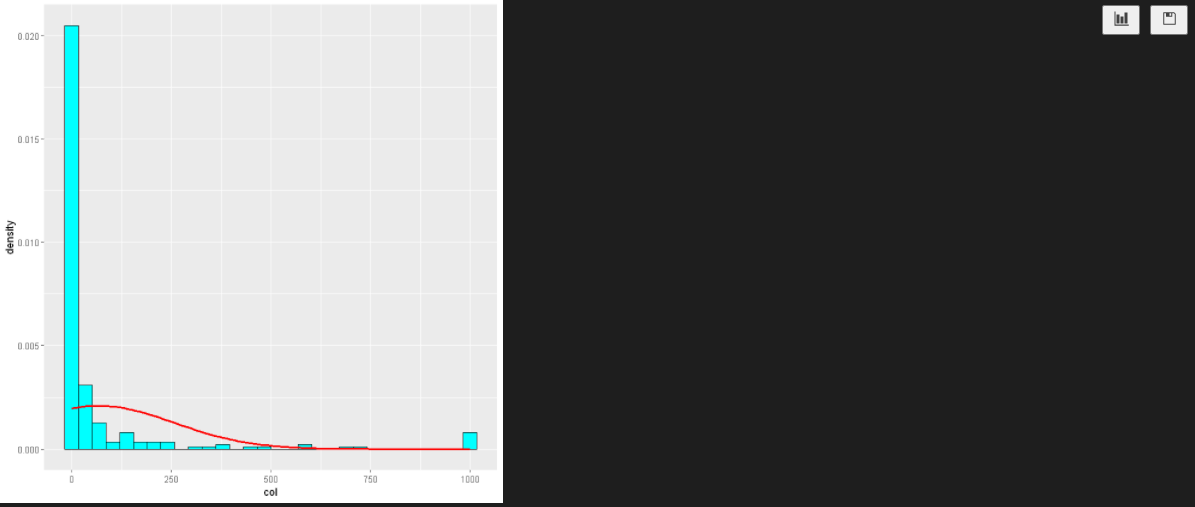




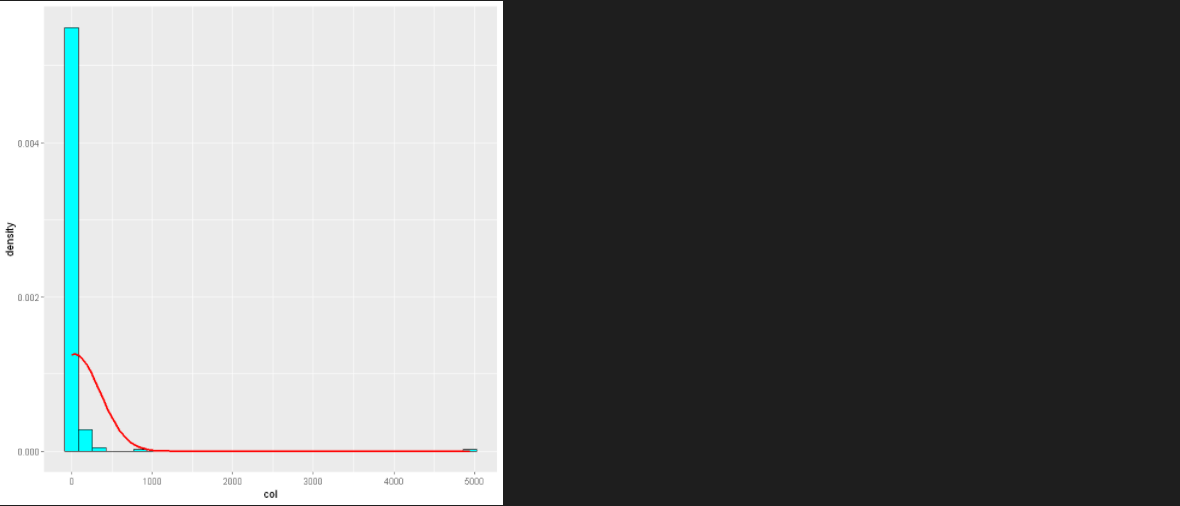


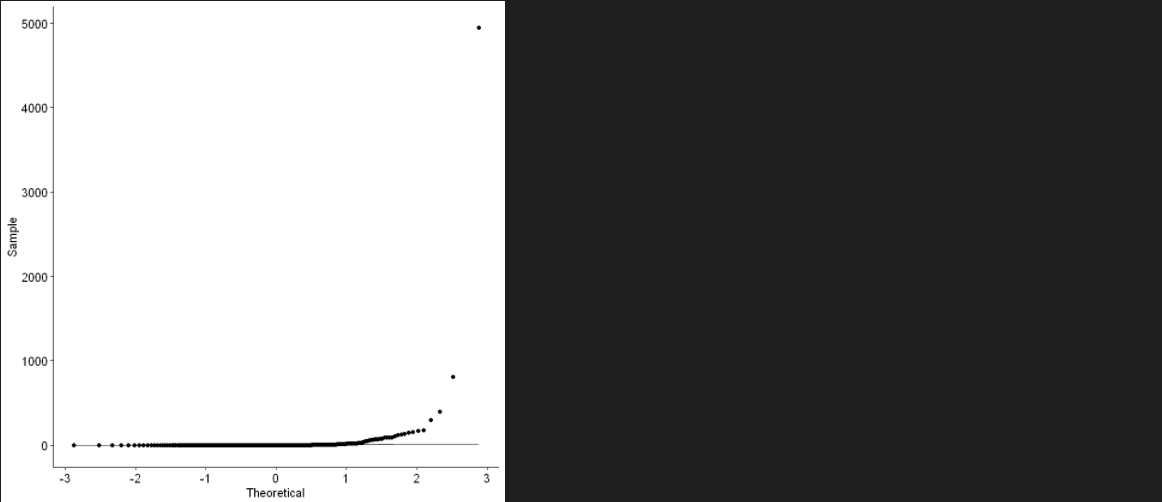
normal\_ditribution(df, df$Sold)





normal\_ditribution(df, df$Comment)





Sau khi kiểm tra các các trường dữ liệu thì kết quả cho thấy dữ liệu không có phân phối chuẩn nên ta sẽ sử dụng các loại kiểm định phi tham số để kiểm tra các giả thuyết

#### 2.3.2 Kiểm định Chisquare.

Chisquare là kiểm định kiểm tra sự độc lập giữa hai biến mà không cần kiểm tra phân phối chuẩn.

**Phát biểu giả thuyết:**

Kiểm định xem Giá cả có ảnh hưởng đến chất lượng của sản phẩm với sự đánh giá của khách hàng không.

* H0 : Giá cả Không ảnh hưởng đến chất lượng sản phẩm (Hai giá trị này độc lập với nhau)
* H1 : Giá cả ảnh hưởng đến chất lượng sản phẩm (Hai giá trị này phụ thuộc nhau)

table(df$Valuation,df$Quality)

chisq.test(df$Valuation, df$Quality)



pvalue < 0.05

Chưa giả đủ điều kiện chấp nhận giả thuyết H0. Điều này có nghĩa là Giá cả ảnh hưởng đến chất lượng sản phẩm

Chú ý: Chất lượng ở đây nói về sự phù hợp của giá cả so với chất lượng sản phẩm.

ggbetweenstats(

  data = df,

  x = Quality,

  y = Sale\_price

) + labs(

    title = "Distribution of Price across Quality"

  ) +

    theme(

    # This is the new default font in the plot

    text = element\_text(size = 12, color = "black"),

    plot.title = element\_text(

      #family = "Tahoma",

      size = 20,

      face = "bold",

      color = "#2a475e"

    ))

Nhận xét:

* Từ biểu đồ ta có thể thấy chất lượng sản phẩm càng tốt thì khoảng tập trung giá cả của dữ liệu càng cao hơn
* Mục Unknow chiếm tỉ lệ rất cao trong tổ số lượng sản phẩm khiến việc phân loại chất lượng và đánh giá gặp khó khăn
* Các sản phẩm Highquality có sự phân bố đồng đều hơn về giá. Trong khi đó Lowquality và Goodquality só sự tập trung dữ liệu về 1 điểm khá giống nhau

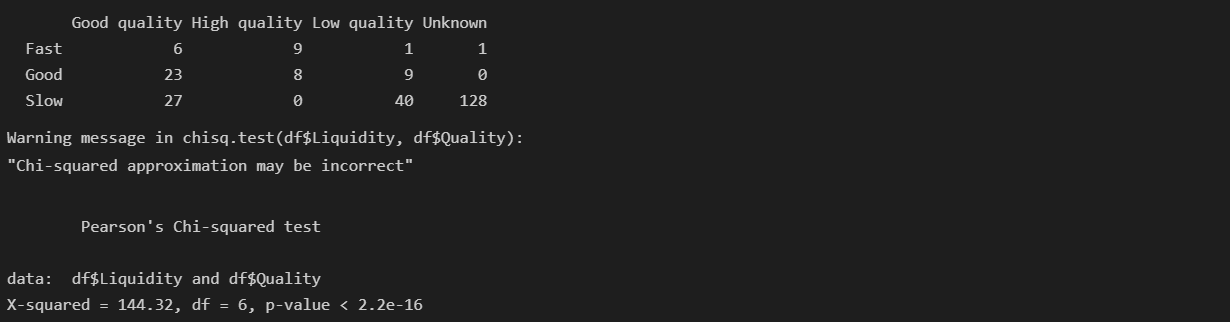
**Phát biểu giả thuyết:**

Kiểm định xem chất lượng sản phẩm có ảnh hưởng đến tốc độ bán của sản phẩm không ?

* H0 : Chất lượng sản phẩm(Quality) không ảnh hưởng đến tốc độ bán của sản phẩm(Liquidity) (Hai Biến này độc lập với nhau)
* H1 : Chất lượng sản phẩm(Quality) ảnh hưởng đến tốc độ bán của sản phẩm( Liquidity)( Hai biến này phụ thuộc nhau)

table(df$Liquidity,df$Quality)

chisq.test(df$Liquidity,df$Quality)



pvalue < 0.05

Chấp nhận giả thuyết H1. Điều này có nghĩa là chất lưởng sản phẩm ảnh hưởng đến tốc độ bán của sản phẩm.

ggbetweenstats(

  data = df,

  x = Quality,

  y = Sold

) + labs(

    title = "Distribution of Sold across Quality"

  ) +

    theme(

    # This is the new default font in the plot

    text = element\_text(size = 15, color = "black"),

    plot.title = element\_text(

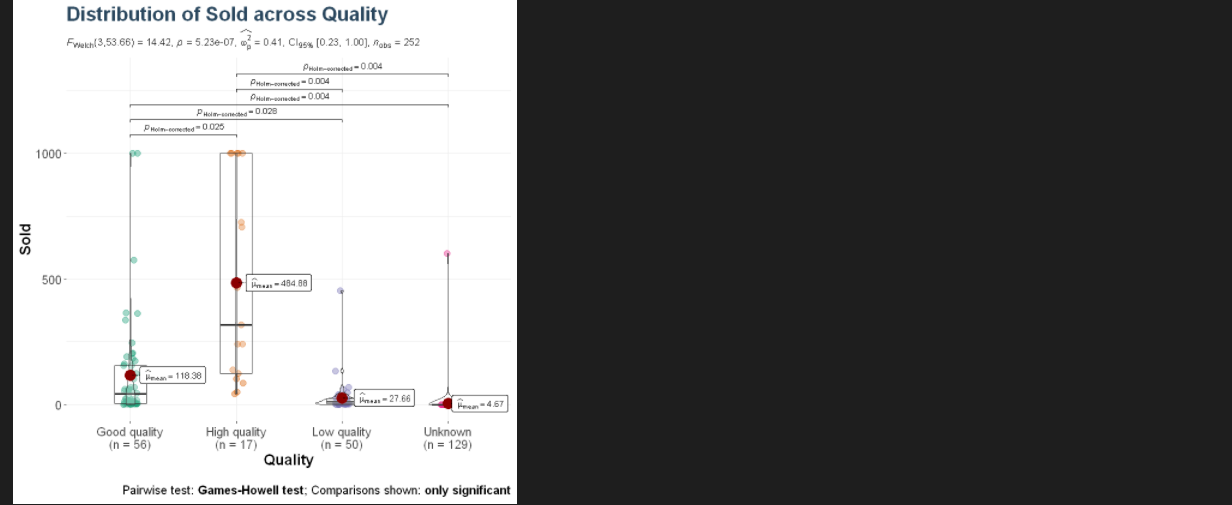
      #family = "Tahoma",

      size = 20,

      face = "bold",

      color = "#2a475e"

    ))



**Nhận xét:**

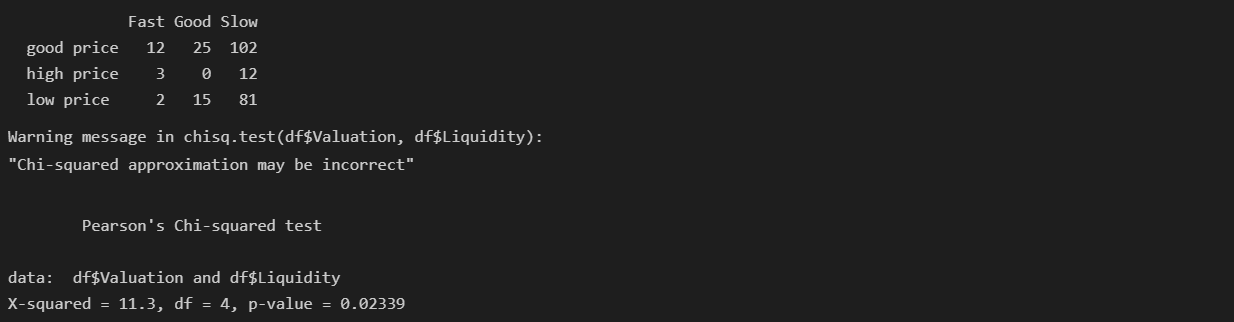
* Nhìn vào biểu đồ lập tức có thể thấy rằng có sự khác biệt rõ ràng giữa Highquality - Goodquality  và Lowquality - Uknow. Đăc biệt sản phẩm chất lượng cao có lượt bán ra chệnh lệch rất lớn.
* Ta thấy riêng nhóm High quality có điểm xuất phát cao hơn hẵn so với các nhóm khác do quy định cần ít nhất 20 lượt bán mới có thể coi đạt chất lượng cao.
* Ta thấy rằng sản phẩm có chất lượng tốt(lượt đánh giá tốt) thì có số lượng bán ra cao hơn rõ rệt. Điều nãy cũng dễ hiểu bởi ngay tại thời điểm mà sản phẩm đó có lượt đánh giá thấp, người tiêu dùng sẽ lập tức không để tâm hay phí tiền để mua nó nữa. Ngược lại sẽ có xu hướng tập trung vào những sản phẩm có lượt đánh giá tốt nên các sản phẩm đó lại có lượt bán tăng nhanh.
* Ta nhận thấy có hơn 50% số lượng cho phân loại Unknow. Điều này cho thấy thị trường Tiki vẫn còn nghèo nàn và khó cho các doanh nghiệp mới có thể phát triển khi chưa có được sự tin tưởng từ khách hàng. Bởi khi chưa có lượt đánh giá để hiểu hơn về sản phẩm thì người tiêu dùng thường sẽ chọn 1 sản phẩm đáng tin hơn cho bản thân khiến các sản phẩm khó để bán ra trong thời gian đầu.
* Không có nhiều sản phẩm đạt Highquality bởi khi càng có nhiều đánh giá thì chất lượng sản phẩm sẽ càng rõ ràng, những nhược điểm của sản phẩm sẽ lộ ra nhiều hơn và đưa sản phẩm về đúng chất lượng của nó.

Phát biểu giả thuyết:

Kiểm tra xem có sự ảnh hưởng giữa giá tiền đến số lượng sản phẩm bán ra không ?

* H0: Giá bán không có sự ảnh hưởng  đến tốc độ bán sản phẩm (Hai biến này là độc lập)
* H1: Giá bán ảnh hưởng đến tốc độ bán sản phẩm (Hai biến này có phụ tuộc vào nhau)

table(df$Valuation,df$Liquidity)

chisq.test(df$Valuation,df$Liquidity) pvalue < 0.05

Chấp nhận giả thuyết H1. Điều này có nghĩa là Giá bán có ảnh hưởng đến tốc độ bán của sản phẩm

ggbetweenstats(

  data = df,

  x = Liquidity,

  y = Sale\_price

) + labs(

    title = "Distribution of Sold across Quality"

  ) +

    theme(

    # This is the new default font in the plot

    text = element\_text(size = 15, color = "black"),

    plot.title = element\_text(

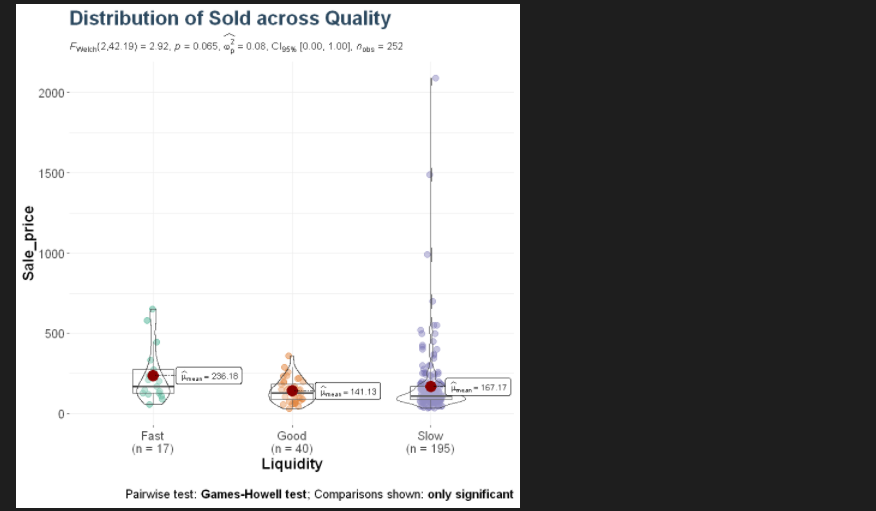
      #family = "Tahoma",

      size = 20,

      face = "bold",

      color = "#2a475e"

    ))



**Nhận xét:**

* Ở mục Slow có nhiều sản phẩm chưa bán được nên có sự phân bố giá tương đối rộng hơn so với cái thanh khoản(Liquidity) khác.
* Sản phẩm ở thanh khoản Good và Fast có sự phân bố rõ rằng tập trung nằm trong một khoảng nhất định.
* Các sản phẩm có tốc độ bán càng cao thì có điểm xuất phát càng cao hơn.
* Có khoảng 70% sản phẩm trên Tiki được bán với tốc độ chậm, Tiki nên xem xét lại chính sách của mình và quảng cáo, thu hút người dùng nhiều hơn.
* Có rất ít sản phẩm có thể phát triển tốt ở Tiki khi mục Fast chỉ chiếm chưa đến 10% trên tông sản phẩm.

#### 2.3.2 Kiểm định kruskal wallis.

Kiểm định Kruskal - Wallis giúp kiểm tra sự phân phối của hai mẫu(khá giống kiểm định ANOVA) mà không cần phải giả định phân phối chuẩn

**Phát biểu giả thuyết:**

Kiểm tra xem liệu các sản phẩm có bảo hành có bán sản phẩm với giá cao hơn các sản phẩm khác ?

* H0 : Không có sự khác biệt về giá giữa các sẩn phẩm có bảo hành và không bảo hành
* H1 : Có sự khác biệt về giá giữa các sản phẩm có bảo hành và không có bảo hành

check\_w = c()

for (i in df$Warranty) {

    if (i == 0){

        check\_w <- c(check\_w,'Non')

    } else {

        check\_w <- c(check\_w,'Warranty')

    }

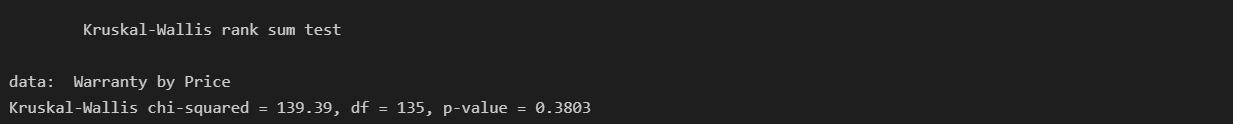
}

War\_tab = data.frame(df$Sale\_price,check\_w)

data.frame(table(check\_w))

names(War\_tab) <- c('Price','Warranty')

head(War\_tab)

kruskal.test(Warranty ~ Price, data = War\_tab) 

pvalue > 0.05

Với mức ý nghĩa 0.05, có thể chấp nhận giả thuyết H0 rằng không có sự khác biệt giữa nhóm có bảo hành và nhóm không có bảo hành.

ggbetweenstats(

  data = War\_tab,

  x = Warranty,

  y = Price

) + labs(

    title = "The distribution of data between two groups"

  ) +

    theme(

    # This is the new default font in the plot

    text = element\_text(size = 15, color = "black"),

    plot.title = element\_text(

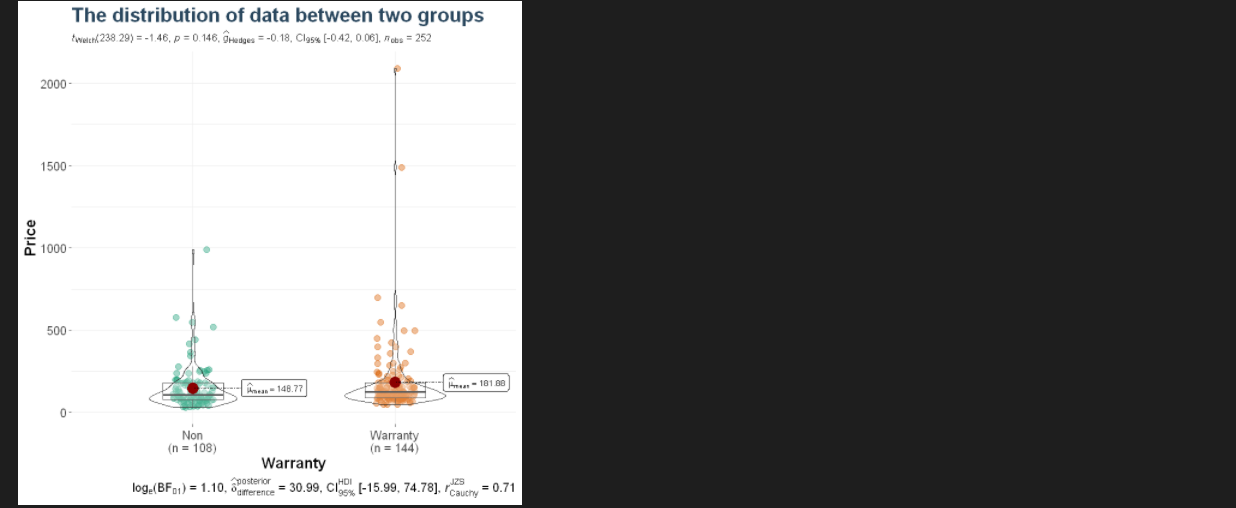
      #family = "Tahoma",

      size = 20,

      face = "bold",

      color = "#2a475e"

    ))



**Phát biểu giả thuyết:**

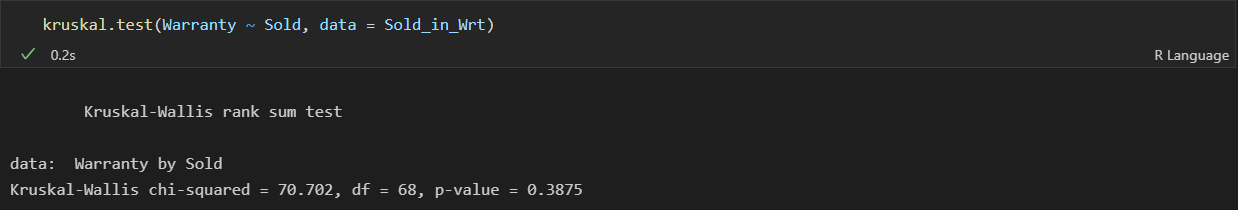
Kiểm tra xem có sự khác biệt về lượng sản phẩm bán ra giữa các sản phẩm có chính sách bảo hành và không có chinh sách bảo hành

* H0: Không có sự khác biệt về lượng sản phẩm bán ra giữa nhóm sản phẩm có chính sách bảo hành và nhóm sản phẩm không có chính sách bảo hành.
* H1:  Có sự khác biệt về lượng sản phẩm bán ra giữa nhóm sản phẩm có chính sách bảo hành và nhóm sản phẩm không có chính sách bảo hành.

Sold\_in\_Wrt <- data.frame(df$Sold,check\_w)

names(Sold\_in\_Wrt) <- c('Sold','Warranty')

head(Sold\_in\_Wrt)

  ggbetweenstats(

  data = Sold\_in\_Wrt,

  x = Warranty,

  y = Sold

) + labs(

    title = "The distribution of data between two groups"

  ) +

    theme(

    # This is the new default font in the plot

    text = element\_text(size = 15, color = "black"),

    plot.title = element\_text(

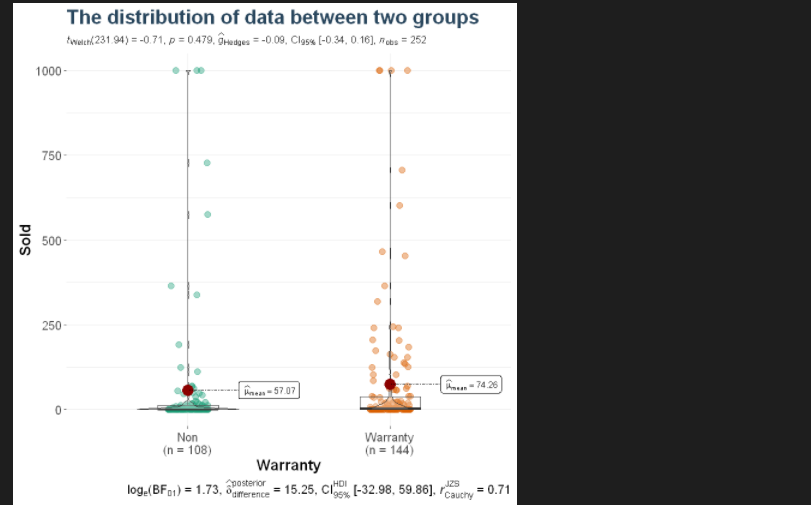
      #family = "Tahoma",

      size = 20,

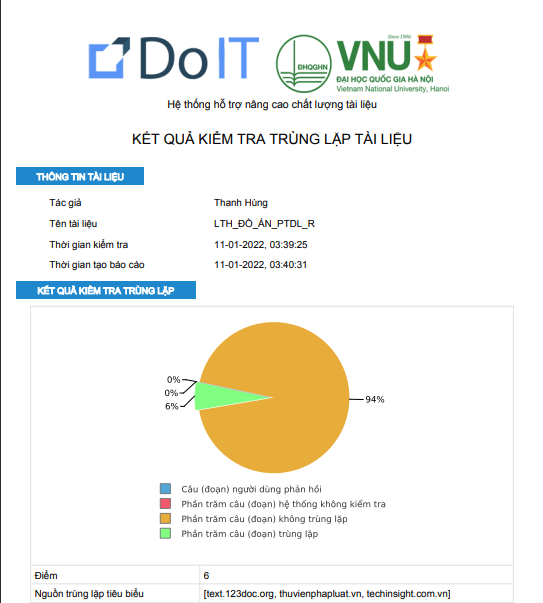
      face = "bold",

      color = "#2a475e"

    ))



# **Chương III: Kết quả kiểm tra đạo văn**



# **Chương IV: Tài liệu tham khảo**

**BOOK**

[1] Website Scraping with Python: Using BeautifulSoup and Scrapy - Gábor László Hajba

[2] R-for-Data-Science\_-Import-Tidy-Transform-Visualize-and-Model-Data-OReilly-Media-2017

[3] Web Scraping with Python: Collecting More Data from the Modern Web - Ryan Mitchell

**INTERNET**

[1] <https://daotaotester.com/huong-dan-su-dung-selenium/>

[2] <https://www.kaggle.com/michau96/what-determines-price-of-a-laptop/notebook>

[3] <https://brightdata.com/blog/how-tos/web-scraping-with-python?kw=&cpn=14745430544&cam=aw_blog_dynamic__547760284460&utm_term=&utm_campaign=blog&utm_source=adwords&utm_medium=ppc&utm_content=dynamic&hsa_acc=1393175403&hsa_cam=14745430544&hsa_grp=131242020607&hsa_ad=547760284460&hsa_src=g&hsa_tgt=dsa-39587879683&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gclid=CjwKCAiAz--OBhBIEiwAG1rIOp54HgwV0wez_PzhdSKJlZxAlXXaDuhxDDrWclWbBxgVcKxu7EnYqxoCSYwQAvD_BwE>

[4] <https://www.edureka.co/blog/web-scraping-with-python/>