ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KĨ THUẬT MÁY TÍNH

BÁO CÁO BÀI TẬP LỚN

MÔN HỌC XÁC SUẤT THỐNG KÊ HỌC KÌ 231

ĐỀ TÀI 02

LỚP: LO9 – NHÓM: MT14

GIẢNG VIÊN HƯỚNG DẪN: TS. NGUYỄN BÁ THI

DANH SÁCH NHÓM:

STT	HỌ VÀ TÊN	MSSV	Kí tên
1	HUỲNH GIA BẢO	2210209	
2	NGUYỄN QUANG HUY	2211237	
3	VÕ HOÀNG HUY	2211298	
4	PHAN TUẤN KIỆT	2211771	
5	NGUYỄN TRƯỜNG THỊNH	2213298	

MỤC LỤC

I. CƠ SỞ LÝ THUYẾT

1. Tìm hiểu về các mô hình và các kiến thức có trong bài:

Các định nghĩa cơ bản:

- **Trung bình cộng:** trung bình cộng trong thống kê là một đại lượng mô tả thống kê, được tính ra bằng cách lấy tổng giá trị của toàn bộ các quan sát trong tập chia cho số lượng các quan sát trong tập.
- **Trung vị:** trong xác suất và thống kê, số trung vị là một số tách giữa nửa lớn hơn và nửa bé hơn của một mẫu, một quần thể, hay một phân bố xác suất.
- **Độ lệch chuẩn:** độ lệch chuẩn, hay độ lệch tiêu chuẩn là một đại lượng thống kê mô tả dùng để đo mức độ phân tán của một tập dữ liệu đã được lập thành bảng tần số. Có thể tính ra độ lệch chuẩn bằng cách lấy căn bậc hai của phương sai.
- **Giá trị lớn nhất:** là giá trị lớn nhất trong toàn bộ các giá trị của một tập mẫu.
- **Giá trị nhỏ nhất:** là giá trị nhỏ nhất trong toàn bộ các giá trị của một tập mẫu.

- **Hồi quy tuyến tính đa biến:**
  - Mô hình hồi quy tuyến tính đa biến: là phương trình mô tả mối quan hệ giữa biến phụ thuộc y với các biến độc lập $(x_1, x_2, \dots, x_n)$ và sai số ε.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \varepsilon$$

    Với $\beta_0, \beta_1, \dots, \beta_n$ là các tham số.

  ⇨ Mục tiêu của hồi quy đa biến là đi ước lượng giá trị tối ưu của các hệ số $\beta_0, \beta_1, \dots, \beta_n$ sao cho mô hình có mỗi quản ánh quan hệ giữa các biến là tốt nhất.

  - Các phương pháp giải phương trình hồi quy đa biến:
    + Phương pháp bình phương cực tiểu:

$$min \sum (y_i - \hat{y_i})^2$$

    + Phương pháp sử dụng các công cụ tính toán Excel, R,…

  - Kiểm định hồi quy tuyến tính đa biến:

    + R-squared (R^2): R-squared là một thước đo quan trọng về mức độ biến động của biến phụ thuộc được giải thích bởi mô hình. Giá trị R^2 càng gần 1, mô hình càng tốt. Nó chỉ ra phần trăm biến động của biến phụ thuộc được giải thích bởi các biến độc lập.

+ Giá trị p (p-value): Giá trị p của từng hệ số hồi quy đo độ tin cậy của ước lượng. Giá trị p thấp (thường dưới 0.05) thường được coi là có ý nghĩa thống kê. Nếu giá trị p lớn, có thể đặt ra nghi ngờ về ý nghĩa thống kê của hệ số đó.

+ F-test: F-test kiểm tra ý nghĩa toàn bộ mô hình. Nếu giá trị p của F-test là nhỏ, có thể bác bỏ giả thuyết không có ảnh hưởng nào của các biến độc lập lên biến phụ thuộc.

+ Kiểm định t (t-test): Kiểm định t được sử dụng để kiểm tra ý nghĩa thống kê của từng hệ số hồi quy độc lập. Giá trị t-test có thể giúp xác định xem mỗi biến có ảnh hưởng đáng kể đến biến phụ thuộc hay không.

2. Các lệnh R sử dụng trong bài

1. `library(stringr), library(tidyr), library(dplyr), library(mice):` Nhúng các gói thư viện R để xử lý dữ liệu.
2. `read.csv(path):` Đọc dữ liệu từ tệp CSV tại đường dẫn path vào một dataframe.
3. `na.omit(dataframe):` Loại bỏ các hàng chứa giá trị NA từ dataframe.
4. `separate(dataframe, col, into, sep):` Tách cột col thành các cột con dựa trên ký tự phân tách sep và lưu vào các cột into.
5. `as.numeric(vector):` Chuyển đổi một vector sang dạng số.
6. `select(dataframe, -col):` Loại bỏ cột col từ dataframe.
7. `apply(dataframe, 2, function):` Áp dụng một hàm function lên các cột (axis 2) của dataframe và trả về kết quả.
8. `mean(vector):` Tính trung bình mẫu của vector.
9. `sd(vector):` Tính độ lệch chuẩn của vector.
10. `quantile(vector, probs):` Tính phân vị cho vector tại các tỷ lệ probs.
11. `median(vector):` Tính giá trị trung vị của vector.
12. `max(vector):` Tìm giá trị lớn nhất trong vector.
13. `min(vector):` Tìm giá trị nhỏ nhất trong vector.
14. `table(vector):` Tạo bảng tần số cho vector.
15. `hist(vector):` Vẽ biểu đồ histogram cho vector.
16. `plot(x, y):` Vẽ biểu đồ phân tán giữa hai biến x và y.

17. `lm(formula, data=dataframe):` Xây dựng mô hình hồi quy tuyến tính với formula và dữ liệu từ dataframe.
18. `summary(model):` Hiển thị thông tin tóm tắt về mô hình hồi quy hoặc dataframe.
19. `predict(model, newdata):` Dự đoán giá trị dựa trên mô hình hồi quy và dữ liệu mới newdata
20. `mice(data, m, method, seed):` Tạo bản sao dữ liệu với giá trị bị thiếu được điền. (m là số lần lặp, method là phương pháp sử dụng để điền giá trị, và seed là giá trị khởi tạo cho số ngẫu nhiên)
21. `complete(imputed_data):` Lấy dữ liệu đã được điền từ mô hình mice.

II. NỘI DUNG BÁO CÁO

1. NGHIÊN CỨU NHỮNG YẾU TỐ ẢNH HƯỞNG ĐẾN CPU:

Dữ liệu 02:
https://www.kaggle.com/datasets/iliassekkaf/computerparts?resource=downlo&select=Intel_CPUs.csv

Tập tin **"Intel_CPUs.csv"** chứa thông tin kĩ thuật chi tiết về ngày phát hành, thông số kĩ thuật về các cấu hình Intel CPU theo thời gian. Tập tin dữ liệu gồm có 45 cột, chúng ta chọn ra 7 biến chính sau:

- **Launch_Date**: ngày phát hành
- **TDP:** công suất điện tiêu thụ
- **Nb_of_cores:** số lượng nhân trong chip
- **Nb_of_threads**: số lượng luồng trong chip
- **Processor_Base_Frequency**: xung nhịp của chip
- **Lithography:** kỹ thuật in thạch bản (kích thước bóng bán dẫn trong chip).
- **Max_Memory_Size:** bộ nhớ tối đa của CPU

Các bước thực hiện:

1. Đọc dữ liệu (Import data): ***"Intel_CPUs.csv"***
2. Tiền xử lý dữ liệu:
   a) Xử lý các biến và đơn vị
   b) Xử lý dữ liệu khuyết
3. Làm rõ dữ liệu (Data visualization)
4. Xây dựng mô hình hồi quy tuyến tính để đánh giá các nhân tố có thể ảnh hướng đến các thông số CPU
5. Thực hiện sự báo thông số cho CPU trong tương lai

2. THỰC HIỆN

2.1 Đọc dữ liệu (Import Data)

Đọc dữ liệu từ file ***"Intel_CPUs.csv"***

Cú pháp code R:

```
# đọc dữ liệu
path<-"Intel_CPUs.csv" #gán địa chỉ file cần đọc vào 1 biến path
CPU<-read.csv(path) #đọc dữ liệu vào
head(CPU) #xem 6 hàng đầu của file dữ liệu đọc vào
```

Kết quả thu được (xem 6 dòng đầu tiên của dữ liệu)

	Product_Collection <chr>	Vertical_Segment <chr>	Processor_Number <chr>	Status <chr>
1	7th Generation Intel® Core™ i7 Processors	Mobile	i7-7Y75	Launched
2	8th Generation Intel® Core™ i5 Processors	Mobile	i5-8250U	Launched
3	8th Generation Intel® Core™ i7 Processors	Mobile	i7-8550U	Launched
4	Intel® Core™ X-series Processors	Desktop	i7-3820	End of Life
5	7th Generation Intel® Core™ i5 Processors	Mobile	i5-7Y57	Launched
6	Intel® Celeron® Processor 3000 Series	Mobile	3205U	Launched

6 rows | 1-5 of 46 columns

2.2 Tiền xử lý dữ liệu:

a) Xử lý các biến và đơn vị:

Tạo một dữ liệu liệu mới chỉ bao gồm các biến chính mà ta quan tâm, lưu với tên **New_CPU**

Cú pháp code R:

```
# chọn ra những biến cần dùng trong file CPU sang một dataframe mới
new_CPU<-CPU[, c("Launch_Date", "nb_of_Cores", "nb_of_Threads",
            "Processor_Base_Frequency", "Max_Memory_Size","Lithography", "TDP")]
head(new_CPU)
```

Kết quả thu được (xem 6 dòng đầu tiên của dữ liệu)

	Launch_Date <chr>	nb_of_Cores <int>	nb_of_Threads <int>	Processor_Base_Frequency <chr>	Max_Memory_Size <chr>	Lithography <chr>	TDP <chr>
1	Q3'16	2	4	1.30 GHz	16 GB	14 nm	4.5 W
2	Q3'17	4	8	1.60 GHz	32 GB	14 nm	15 W
3	Q3'17	4	8	1.80 GHz	32 GB	14 nm	15 W
4	Q1'12	4	8	3.60 GHz	64.23 GB	32 nm	130 W
5	Q1'17	2	4	1.20 GHz	16 GB	14 nm	4.5 W
6	Q1'15	2	2	1.50 GHz	16 GB	14 nm	15 W

6 rows

Vì dữ liệu hiện tại của mỗi cột không đồng nhất về đơn vị và còn chưa hoàn toàn ở dạng số nên ta cần sử lý các biến dữ liệu để đạt được dữ liệu chuẩn hơn.

Xử lý biến Launch_Date:

Xử lý cột dữ liệu Launch_Date các khoảng trống thành NA. Sau đó tách thành Launch_Quarter và Launch_Year để dễ dàng xử lý.

```
# xử lý khoảng trống cho thành giá trị NA
new_CPU$Launch_Date<-ifelse(new_CPU$Launch_Date == "", NA, new_CPU$Launch_Date)
# tách cột launch_date thành 2 phần đó là cột launch_quarter và launch_year
new_CPU<-separate(new_CPU, Launch_Date, c("Launch_Quarter", "Launch_Year"), sep = "'" )
```

Chuyển đổi cột Launch_Year thành dạng số và đưa về dạng chuẩn để dễ dàng thao tác dữ liệu.

```
# đổi cột launch_year thành số
new_CPU$Launch_Year<-as.numeric(new_CPU$Launch_Year)
new_CPU$Launch_Year<-ifelse(new_CPU$Launch_Year == "99",
                                        new_CPU$Launch_Year+1900, # cộng vào 1900 để đưa năm về dạng chuẩn
                                      new_CPU$Launch_Year+2000) # cộng vào 2000 để đưa năm về dạng chuẩn
table(new_CPU$Launch_Year)
```

Kết quả thu được sau thao tác này:

```
##
## 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014
##    3   10    4   18    6   50   28   58   88  110   79  148  154  169  239  218
## 2015 2016 2017
##  186  125  178
```

Xóa bỏ đi cột Launch_Quarter và giữ lại cột Launch_Year.

```
# xóa bỏ đi cột Launch_Quarter
new_CPU<-select(new_CPU, -Launch_Quarter)
```

Xử lý biến Processor_Base_Frequency:

Thay thế những cột trống thành NA để sau đó lọc những cột đó. Tách cột Processor_Base_Frequency thành Processor_Base_Frequency_val và Processor_Base_Frequency_unit, chuyển đổi cột frequency_val thày dạng số.

```
# Xử lý những ô trống thành NA
new_CPU$Processor_Base_Frequency<-ifelse(new_CPU$Processor_Base_Frequency == "", NA, new_CPU$Processor_Base_Frequency)
# tách cột Frequency thành 2 phần là val và unit
new_CPU<-separate(new_CPU, Processor_Base_Frequency, c("Processor_Base_Frequency_val", "Processor_Base_Frequency_unit"), sep = " " )
# chuyển đổi cột frequency_val thành dạng số
new_CPU$Processor_Base_Frequency_val<-as.numeric(new_CPU$Processor_Base_Frequency_val)
table(new_CPU$Processor_Base_Frequency_unit)
```

Nhận thấy đơn vị của một số Processor có sự khác nhau, cụ thể là GHz và MHz:

```
##
##  GHz  MHz
## 2098  167
```

Nhận thấy số đơn vị là GHz nhiều hơn nên sẽ quy chung về GHz để xử lý.

```
# đổi những ô có đơn vị là MHz thành GHz
new_CPU$Processor_Base_Frequency<-ifelse(new_CPU$Processor_Base_Frequency_unit == "MHz",
                                         new_CPU$Processor_Base_Frequency_val/1000,
                                       new_CPU$Processor_Base_Frequency_val)
```

Sau khi đưa tất cả về chung đơn vị thì xóa đi những cột không dùng.

```
# xóa bỏ đi 2 cột là frequency_val và frequency_unit
new_CPU<-select(new_CPU, -Processor_Base_Frequency_val)
new_CPU<-select(new_CPU, -Processor_Base_Frequency_unit)
table(new_CPU$Processor_Base_Frequency)
```

Kết quả đạt được sau thao tác này:

```
##
##  0.032  0.033  0.075   0.09    0.1   0.12  0.133   0.15  0.166   0.18    0.2  0.233  0.266
##      2      1      1      1      1      1      1      3      2      1      4      3      6
##    0.3  0.333   0.35  0.366    0.4  0.433   0.45  0.466    0.5  0.533   0.55  0.566    0.6
##      7      3      1      3     16      2      7      2      8      3      6      1     13
##  0.633   0.65  0.667    0.7  0.733   0.75  0.766    0.8   0.85  0.866    0.9   0.93  0.933
##      1      5      3      7      5      4      1     15      4      4     12      1      5
##   0.95      1   1.04   1.05   1.06   1.07    1.1   1.13    1.2   1.24   1.25   1.26    1.3
```

Xử lý biến Max_Memory_Size:

Thay thế những cột trống thành NA để dễ dàng lọc dữ liệu. Sau đó tách cột thành 2 cột Max_Memory_Size_val và Max_Memory_Size_unit.

```
# xử lý những ô trống thành giá trị NA
new_CPU$Max_Memory_Size<-ifelse(new_CPU$Max_Memory_Size == "", NA, new_CPU$Max_Memory_Size)
# tách cột memory_size thành 2 phần là val và unit
new_CPU<-separate(new_CPU, Max_Memory_Size, c("Max_Memory_Size_val", "Max_Memory_Size_unit"), sep = " ")
# đổi cột max_memory_size_val thành dạng số
new_CPU$Max_Memory_Size_val<-as.numeric(new_CPU$Max_Memory_Size_val)
# xem đơn vị nào là chủ yếu trong cột memory_size_unit
table(new_CPU$Max_Memory_Size_unit)
```

Nhận thấy tỉ lệ dữ liệu có đơn vị là GB nhiều hơn TB.

```
##
##   GB   TB
## 1291  112
```

Quyết định chuyển đổi những cột có đơn vị là TB thành GB.

```
# số hàng có đơn vị là GB nhiều hơn là TB nên đưa  tất cả về GB
new_CPU$Max_Memory_Size<-ifelse(new_CPU$Max_Memory_Size_unit == "TB",
                                        new_CPU$Max_Memory_Size_val*1000,
                                      new_CPU$Max_Memory_Size_val)
```

Sau khi đưa tất cả về chung đơn vị thì xóa đi những cột không dùng

```
# xóa bỏ đi cột memory_size_val và memory_size_unit
new_CPU<-select(new_CPU, -Max_Memory_Size_val)
new_CPU<-select(new_CPU, -Max_Memory_Size_unit)
table(new_CPU$Max_Memory_Size)
```

Kết quả thu được sau thao tác này:

```
##
##     1     2   2.4  2.44  2.93     4  4.88     6     8  8.01   8.3  8.79    16
##     6    47     1     1     6    24     2     2   104    10     1     7   166
## 16.11 16.38  16.6    24    32 32.23 32.77    64 64.23 64.45   128   144   256
##     2    14    15    21   404     3    10   144     2     1    40    18    20
##   288   375   384   512   768  1020  1500  1540  2050  3070  4100
##    22     2    65     7   124     6     7    75     6    12     6
```

Xử lý biến Lithography:

Xử lý các dữ liệu trống của cột Lithography và chia cột thành 2 phần Value, Unit.

```
#xử lý những ô trống thành NA
new_CPU$Lithography<-ifelse(new_CPU$Lithography == "", NA, new_CPU$Lithography)
#tách cột Lithography thành 2 phần là lithography_val và lithography_unit
new_CPU<-separate(new_CPU, Lithography, c("Lithography_val", "Lithography_unit"), sep = " ")
```

Đổi cột Lithography_val thành dạng số:

```
#đổi cột Lithography_val thành dạng số
new_CPU$Lithography_val<-as.numeric(new_CPU$Lithography_val)
table(new_CPU$Lithography_unit)
```

Tiếp theo xóa bỏ những cột dữ liệu NA

```
# xóa bỏ đi cột lithography_unit
new_CPU<-select(new_CPU, -Lithography_unit)
#gán lại cột lithography_val cho lithography
new_CPU$Lithography<-new_CPU$Lithography_val
new_CPU<-select(new_CPU, -Lithography_val)
table(new_CPU$Lithography)
```

Xử lý biến TDP:

Xử lý cột TDP tách thành TDP_val, TDP_unit. Chuyển đổi TDP_val thành dạng số.

```
# Xử lý khoảng trống thành NA
new_CPU$TDP<-ifelse(new_CPU$TDP == "", NA, new_CPU$TDP)
# tách cộng TDP thành val và unit
new_CPU<-separate(new_CPU, TDP, c("TDP_val", "TDP_unit"), sep = " ")
# chuyển đổi cột TDP thành dạng số
new_CPU$TDP_val<-as.numeric(new_CPU$TDP_val)
# xem đơn vị nào là chủ yếu trong TDP
table(new_CPU$TDP_unit)
```

Nhận thấy các dữ liệu đều đã cùng kiểu dữ liệu

```
##
##    W
## 2216
```

Xóa đi những cột không dùng

```
new_CPU$TDP<-new_CPU$TDP_val
new_CPU<-select(new_CPU, -TDP_unit)
new_CPU<-select(new_CPU, -TDP_val)
table(new_CPU$TDP)
```

Kết quả thu được sau thao tác này

```
##
## 0.025  0.65   1.3   1.4     2   2.2   2.4   2.5     3   3.3   3.5   3.6     4
##     1     1     1     1     3    14     2     3     3     2     1     4     3
##   4.3   4.5     5   5.5   5.9     6   6.1   6.5     7   7.5     8   8.1   8.3
##     2    20     5    15     1    20     1     7    16    22     2     1     1
##   8.5     9   9.5   9.7   9.8    10  10.1  10.3 10.63  11.1  11.5  11.6 11.61
##     3     5     3     1     2    34     1     2     1     1    15     1     1
##  11.7  11.8    12  12.1    13  13.1  13.2  13.8    14  14.1  14.5    15  15.5
##     1     2     5     1     9     5     1     1     3     1     1    84     2
##  15.7  15.8    16  16.5 16.59  16.8    17  17.5  17.6  17.8    18  18.3 18.48
##     1     3     1     1     1     2    61     2     1     1    19     2     1
##  18.6  18.8    19  19.2  19.4  19.5  19.7    20  20.1  20.5  20.6  20.7  20.8
##     1     1     3     2     1     3     1     8     1     1     3     1     5
##    21  21.3  21.5  21.7  21.8    22  22.5  22.8    23  23.2  23.3  23.6  23.7
##     9     1     4     1     1     6     1     1     2     2     2     1     1
##  23.8    24  24.1  24.3  24.4  24.5  24.6    25  25.3  25.4  25.7  25.8  25.9
##     1     2     1     1     1     8     2    51     1     1     1     1     1
##  26.1  26.7  26.9    27  27.1  27.3  27.4  27.5  27.8    28  28.3  28.7  28.9
##     1     1     1    20     1     1     1     1     1    23     1     1     1
##    29  29.1  29.6  29.7  29.9    30  30.4  30.7  30.8    31  31.2  31.7    32
##     3     1     3     1     2    13     1     1     3    24     1     2     4
##  32.1  32.2    33  33.4    34  34.5  34.8    35    36  36.2    37  37.9    38
##     1     1     2     1    10     4     3   238     1     1    25     1     2
##  39.5    40    43    44    45  46.7    47    48    50    51  51.6    52  52.8
##     1    15     1     6    88     2    30     2    13    14     1     1     1
```

Sau khi xử lý các các cột và kiểu dữ kiệu mong muốn thì ta thu được dataframe như sau (xử dụng lệnh head để xem 6 dòng đầu tiên trong dataframe)

Cú pháp code R:

```
head(new_CPU)
```

Kết quả thu được:

	Launch_Year <dbl>	nb_of_Cores <int>	nb_of_Threads <int>	Processor_Base_Frequency <dbl>	Max_Memory_Size <dbl>	Lithography <dbl>	TDP <dbl>
1	2016	2	4	1.3	16.00	14	4.5
2	2017	4	8	1.6	32.00	14	15.0
3	2017	4	8	1.8	32.00	14	15.0
4	2012	4	8	3.6	64.23	32	130.0
5	2017	2	4	1.2	16.00	14	4.5
6	2015	2	2	1.5	16.00	14	15.0

6 rows

b) Xử lý dữ liệu khuyết

Tính số lượng dữ liệu khuyết từng cột.

Cú pháp code R:

```
# Tính số lượng dữ liệu khuyết của từng cột
apply(is.na(new_CPU), 2, sum)
```

Kết quả thu được:

```
##              Launch_Year             nb_of_Cores           nb_of_Threads
##                      412                       0                     856
## Processor_Base_Frequency         Max_Memory_Size             Lithography
##                       18                     880                      71
##                      TDP
##                       67
```

Tính toán tỉ lệ khuyết dữ liệu

Cú pháp code R:

```
# Tính tỉ lệ dữ liệu khuyết
apply(is.na(new_CPU), 2, mean)
```

Kết quả thu được:

```
##              Launch_Year             nb_of_Cores           nb_of_Threads
##              0.180464301             0.000000000             0.374945247
## Processor_Base_Frequency         Max_Memory_Size             Lithography
##              0.007884363             0.385457731             0.031099431
##                      TDP
##              0.029347350
```

Nhận xét: Từ kết quả trên nhận thấy ở hai cột Max_Memory_Size và nb_of_Threads có lần lượt 880 và 856 dữ liệu bị khuyết. Vì số lượng bị khuyết khá lớn (38,55% và 37,49%) nên không thể loại bỏ ngay lập tức mà nên dùng một phương pháp tối ưu để thay thế những dữ liệu đấy mà không ảnh hưởng đến chất lượng báo cáo.

Tóm tắt lý thuyết phương pháp MICE

Xử lý giá trị NA: Như ở phần mở đầu do giá phần trăm NA khá lớn nên không vội để loại bỏ. Sau khi đã đưa các số liệu về dạng chuẩn thì nhóm đề xuất cách xử lý giá trị NA như sau: MICE giả định rằng dữ liệu bị mất là *Missing at Random* (MAR), có nghĩa là xác suất mà một giá trị bị thiếu chỉ phụ thuộc vào giá trị quan sát được và có thể dự đoán chúng bằng các giá trị cùng chủng loại. Nó tính dữ liệu missing dựa vào qui luật biến thiên của các biến số trong dataset.

Ví dụ: Giả sử chúng ta có các biến $X_1, X_2, \ldots, X_k$. Nếu X_1 có các giá trị còn thiếu, thì nó sẽ được hồi quy trên các biến khác từ X_2 đến X_k. Các giá trị còn thiếu trong X_1 sẽ được thay thế bằng các giá trị mà nó dự đoán từ các biến số còn lại. Tương tự, nếu X_2 có các giá trị còn thiếu, thì các biến X_1, X_3 đến X_k sẽ được sử dụng trong mô hình dự đoán như các biến độc lập. Sau đó, giá trị còn thiếu sẽ được thay thế bằng giá trị dự đoán. Theo mặc định, hồi quy tuyến tính được sử dụng để dự đoán giá trị missing của biến liên tục. Hồi quy logistic được sử dụng cho giá trị thiếu trong biến rời rạc. Khi chu kỳ này hoàn tất, nhiều bộ dữ liệu được tạo ra. Những bộ dữ liệu này chỉ khác nhau trong các giá trị bị thiếu. Nói chung, đây được coi là một phương pháp hay để xây dựng các mô hình trên các bộ dữ liệu riêng rẽ và có thể kết hợp các kết quả của chúng.

Thực hiện phương pháp MICE

Sử dụng lệnh mice để tạo 5 bộ data set mới, mỗi bộ chứa dữ liệu khác nhau ở những chỗ bị khuyết.

```
#sử dụng lệnh mice để tạo 5 bộ data set mới,mỗi bộ chứa dữ liệu khác nhau ở những chỗ bị khuyết
imputed_Data <- mice(new_CPU, m=5, maxit = 50, method = 'pmm', seed = 500, print=FALSE)
```

Mỗi một biến số có NA, MICE sẽ cho ra 5 subsets có thể thay thế được. Trong 5 lần imputations trên thì nhóm quyết định chọn subset lần thứ 1 (vì thường là xác xuất cao nhất) để thay thế chính thức cho những NA.

```
new_CPU <- complete(imputed_Data,1)
```

Thống kê số lượng dữ liệu khuyết sau khi sử dụng phương pháp MICE

```
apply(is.na(new_CPU), 2, sum)
```

Nhận thấy không còn dữ liệu khuyết.

```
##             Launch_Year             nb_of_Cores          nb_of_Threads
##                       0                       0                      0
## Processor_Base_Frequency         Max_Memory_Size             Lithography
##                       0                       0                      0
##                     TDP
##                       0
```

2.3 Làm rõ dữ liệu (Data visualization)

a) Tính toán các giá trị thống kê mô tả

Trong tập dữ liệu lúc này có các biến định lượng liên tục gồm**:** *"nb_of_Cores", "nb_of_Threads", "Processor_Base_Frequency", "Max_Memory_Size", "TDP", "Launch _Year", "Lithography"* . Ta tính các giá trị thống kê mô tả gồm: trung bình, trung vị. độ lệch chuẩn, giá trị lớn nhất và giá trị nhỏ nhất của các biến này và xuất kết quả dưới dạng bản.

Cú pháp code R tính trung bình mẫu:

```
mean<-apply(new_CPU[,c("Launch_Year","Lithography","nb_of_Cores", "nb_of_Threads", "Processor_Base_Frequency", "M
ax_Memory_Size", "TDP")], 2, mean)
```

Cú pháp code R tính độ lệch chuẩn:

```
sd<-apply(new_CPU[,c("Launch_Year","Lithography","nb_of_Cores", "nb_of_Threads", "Processor_Base_Frequency", "Max
_Memory_Size", "TDP")], 2, sd)
```

Cú pháp code R tính phân vị 1:

```
Q1<-apply(new_CPU[,c("Launch_Year","Lithography","nb_of_Cores", "nb_of_Threads", "Processor_Base_Frequency", "Max
_Memory_Size", "TDP")], 2, quantile, probs = 0.25)
```

Cú pháp code R tính trung vị:

```
med<-apply(new_CPU[,c("Launch_Year","Lithography","nb_of_Cores", "nb_of_Threads", "Processor_Base_Frequency", "Ma
x_Memory_Size", "TDP")], 2, median)
```

Cú pháp code R tính phân vị 3:

```
Q3<-apply(new_CPU[,c("Launch_Year","Lithography","nb_of_Cores", "nb_of_Threads", "Processor_Base_Frequency", "Max
_Memory_Size", "TDP")], 2, quantile, probs = 0.75)
```

Cú pháp code R tính giá trị lớn nhất:

```
Max<-apply(new_CPU[,c("Launch_Year","Lithography","nb_of_Cores", "nb_of_Threads", "Processor_Base_Frequency", "Ma
x_Memory_Size", "TDP")], 2, max)
```

Cú pháp code R tính giá trị nhỏ nhất:

```
Min<-apply(new_CPU[,c("Launch_Year","Lithography","nb_of_Cores", "nb_of_Threads", "Processor_Base_Frequency", "Ma
x_Memory_Size", "TDP")], 2, min)
```

Cú pháp code R thống kê các kết quả đã tính được

```
data.frame(mean, sd, Q1, med, Q3, Max, Min)
```

Kết quả thu được:

	mean <dbl>	sd <dbl>	Q1 <dbl>	med <dbl>	Q3 <dbl>	Max <dbl>	Min <dbl>
Launch_Year	2010.177836	5.0073477	2007.00	2011.0	2014.0	2017.0	1999.000
Lithography	53.180464	52.7087796	22.00	32.0	65.0	250.0	14.000
nb_of_Cores	4.066579	6.3298840	1.00	2.0	4.0	72.0	1.000
nb_of_Threads	6.858519	8.8762873	2.00	4.0	8.0	56.0	1.000
Processor_Base_Frequency	2.220607	0.8244135	1.66	2.2	2.8	4.3	0.032
Max_Memory_Size	192.148879	462.1207930	8.00	32.0	64.0	4100.0	1.000
TDP	59.080633	44.8533043	25.00	45.0	84.0	300.0	0.025

7 rows

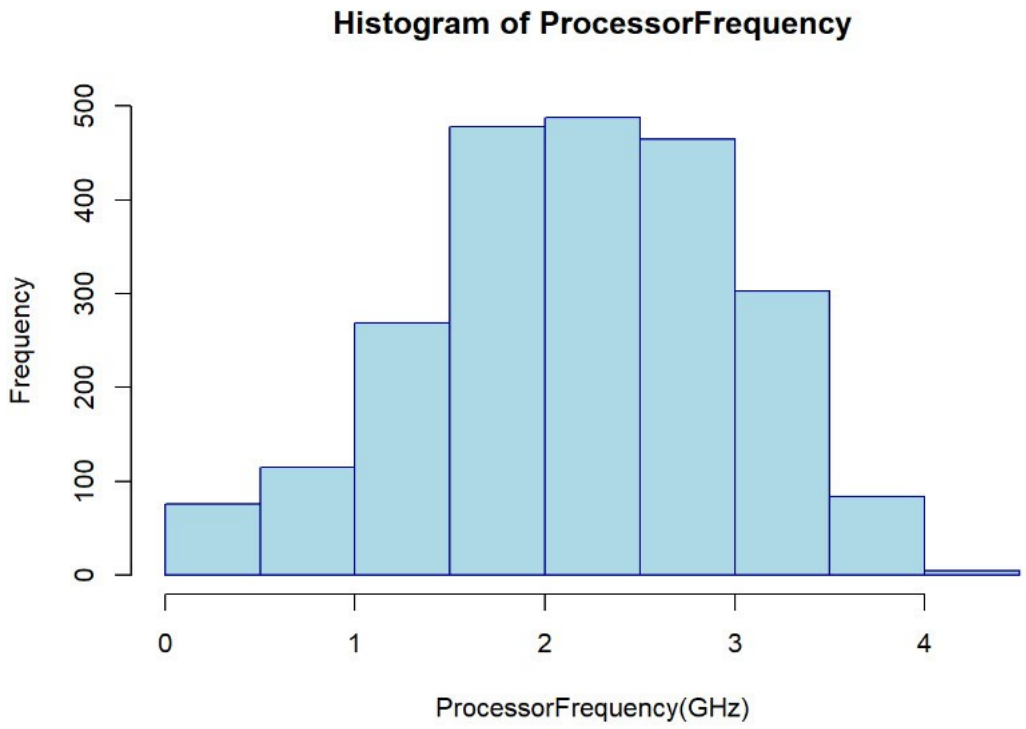
b) Vẽ đồ thị của các biến dữ liệu:

Vẽ đồ thị phân phối của biến Processor_Frequency

Cú pháp code R sử dụng hist để biển đồ phân phối Processor_Frequency:

```
hist(new_CPU$Processor_Base_Frequency,
main = "Histogram of ProcessorFrequency",
xlab = "ProcessorFrequency(GHz)",
ylim = c(0,300),
col = "lightblue",
border = "darkblue",
freq = TRUE
)
```

Kết quả đồ thị thu được:



Hình 3.1: Biểu đồ phân phối của biến Processor Frequency (GHz)

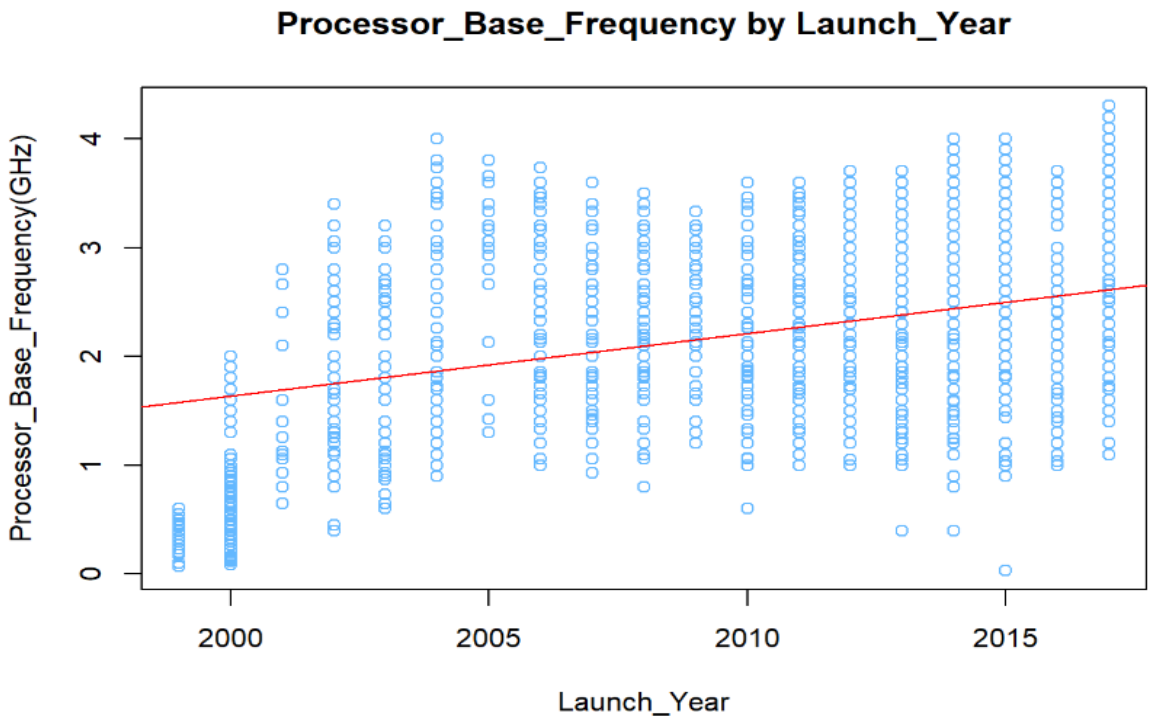
⇨ **Nhận xét:** Đây là biểu đồ Histogram đối xứng và không có giá trị ngoại lai. Dựa vào biểu đồ Histogram, ta thấy biến Processor Frequency phân bố chủ yếu từ 1.5 GHz tới 3 GHz.

Vẽ đồ thị phân phối của biến Processor_Frequency theo biến Launch_Year.

Cú pháp code R sử dụng plot để vẽ đồ thị phân phối Processor_Base_Frequency theo biến Launch_Year:

```
plot(new_CPU$Processor_Base_Frequency~new_CPU$Launch_Year,
main = "Processor_Base_Frequency by Launch_Year",
xlab = "Launch_Year",
ylab = "Processor_Base_Frequency(GHz)",
col = "steelblue1"
)
abline(lm(Processor_Base_Frequency~Launch_Year,data=new_CPU),col='red')
```

Kết quả thu được:



Hình 3.2: Đồ thị phân phối Processor_Base_Frequency theo biến Launch_Year

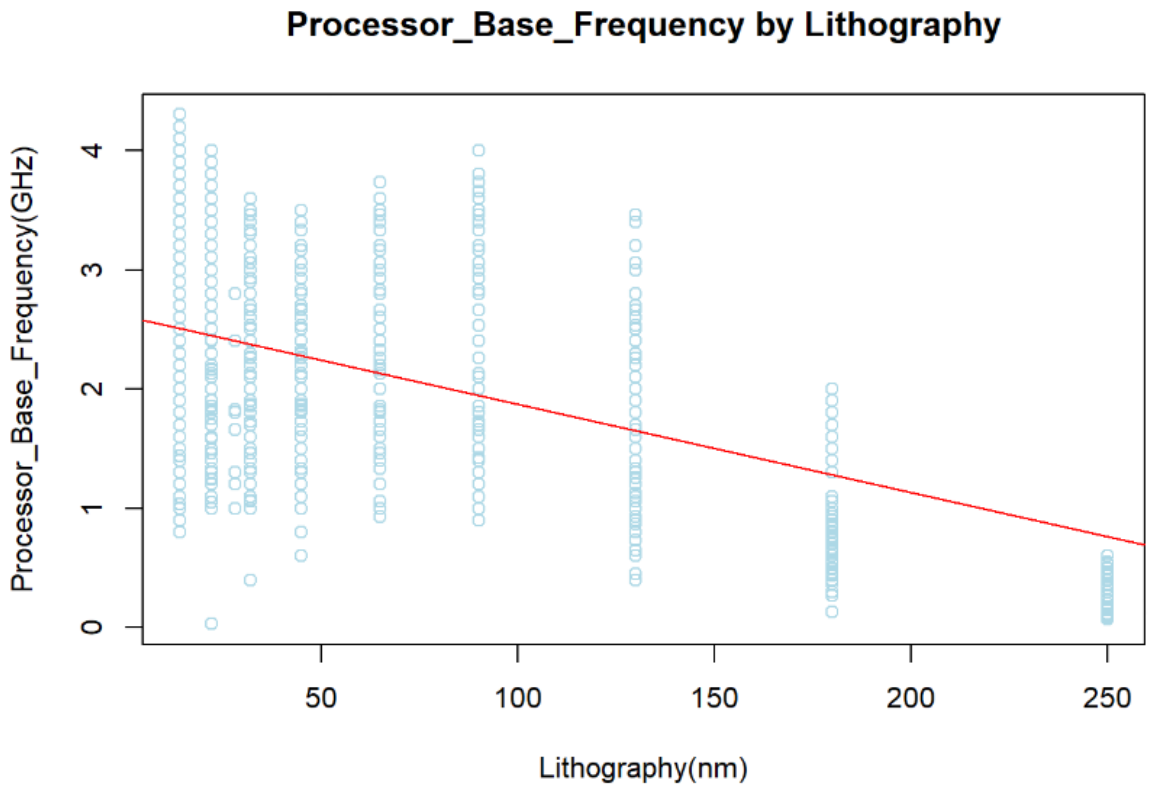
⇨ **Nhận xét:** biến Processor_Base_Frequency và biến Launch_Year không có mối liên hệ tuyến tính.

Vẽ đồ thị phân phối của biến Processor_Frequency theo biến Lithography

Code R sử dụng plot để vẽ đồ thị phân phối Processor_Base_Frequency theo biến Lithography**:**

```
plot(new_CPU$Processor_Base_Frequency~new_CPU$Lithography,
     main = "lithography by ProcessorFrequency",
     xlab = "Lithography",
     ylab = "ProcessorFrequency",
     col = "lightblue"
)
abline(lm(new_CPU$Processor_Base_Frequency~new_CPU$Lithography),col="red")
```

Kết quả thu được:



Hình 3.3: Đồ thị phân phối Processor_Base_Frequency theo biến Lithography

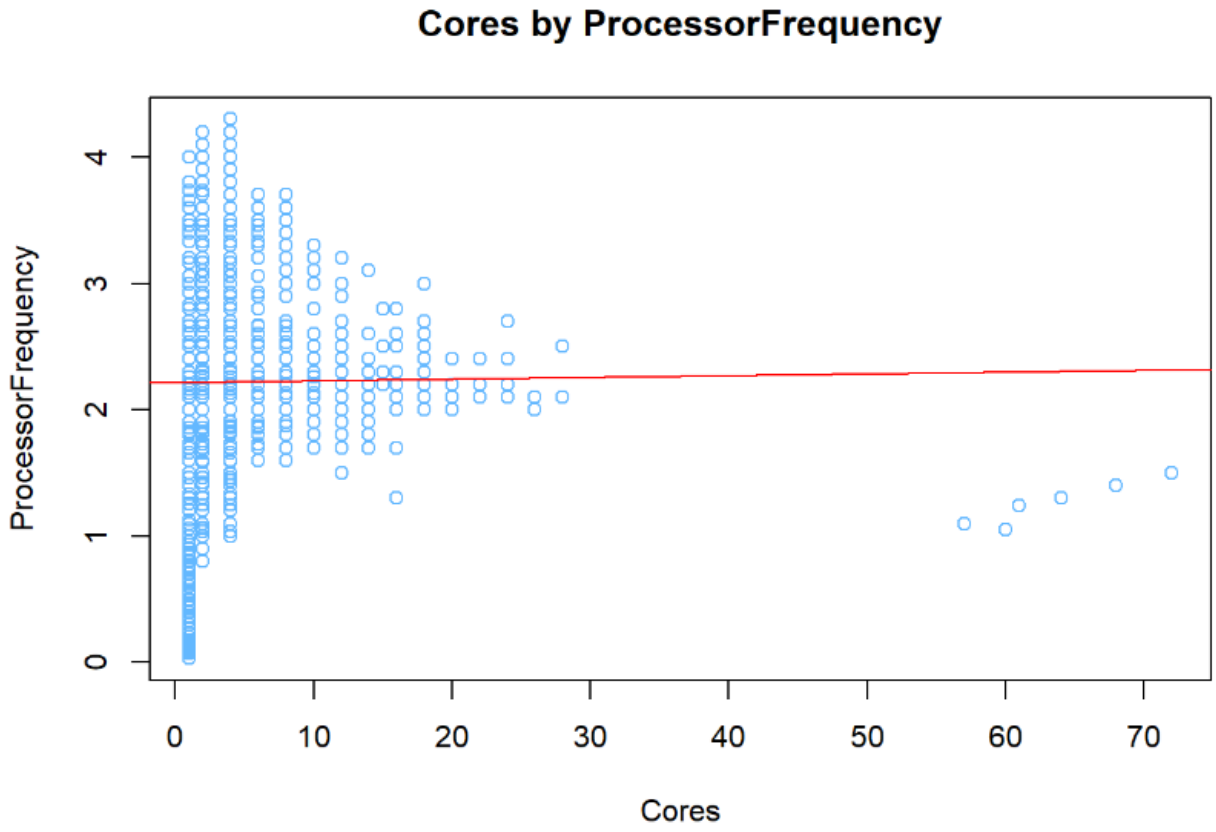
⇨ **Nhận xét**: biến Processor_Base_Frequency và biến Lithography có mối liên hệ tuyến tính yếu.

Vẽ đồ thị phân phối của biến Processor_Frequency theo biến np_of_Cores

Cú pháp code R sử dụng plot để vẽ đồ thị phân phối Processor_Base_Frequency theo biến np_of_Cores

```
plot(new_CPU$Processor_Base_Frequency~new_CPU$nb_of_Cores,
     main = "Cores by ProcessorFrequency",
     xlab = "Cores",
     ylab = "ProcessorFrequency",
     col = "steelblue1"
)
abline(lm(new_CPU$Processor_Base_Frequency~new_CPU$nb_of_Cores),col="red")
```

Kết quả thu được:



Hình 3.4: Đồ thị phân phối Processor_Base_Frequency theo biến np_of_Cores

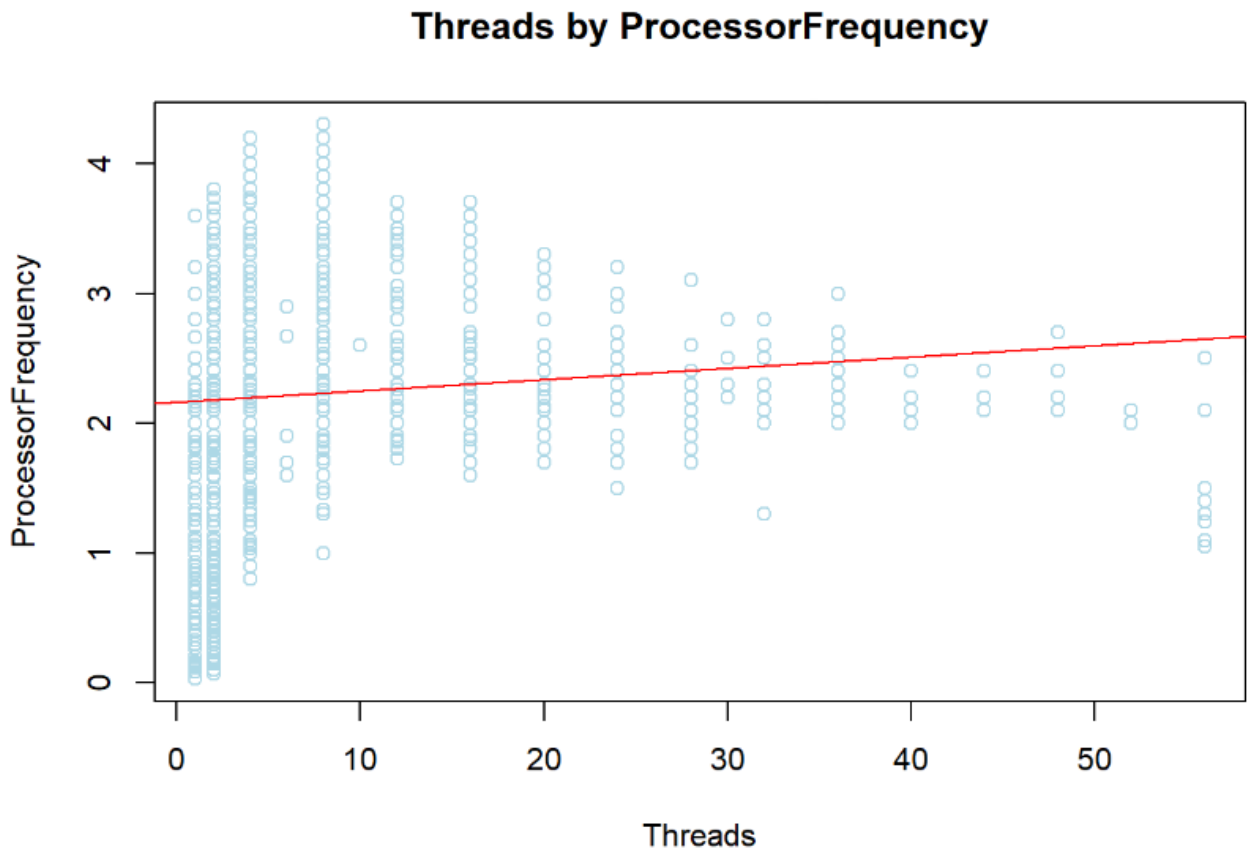
⇨ **Nhận xét**: biến Processor_Base_Frequency và biến np_of_Cores có mối liên hệ tuyến tính trung bình.

Vẽ đồ thị phân phối của biến Processor_Frequency theo biến np_of_Threads

Cú pháp code R sử dụng lệnh plot để vẽ đồ thị phân phối Processor_Base_Frequency theo biến np_of_Threads

```
plot(new_CPU$Processor_Base_Frequency~new_CPU$nb_of_Threads,
     main = "Threads by ProcessorFrequency",
     xlab = "Threads",
     ylab = "ProcessorFrequency",
     col = "lightblue"
)
abline(lm(new_CPU$Processor_Base_Frequency~new_CPU$nb_of_Threads),col="red")
```

Kết quả thu được:



Hình 3.5: Đồ thị phân phối Processor_Base_Frequency theo biến np_of_Threads

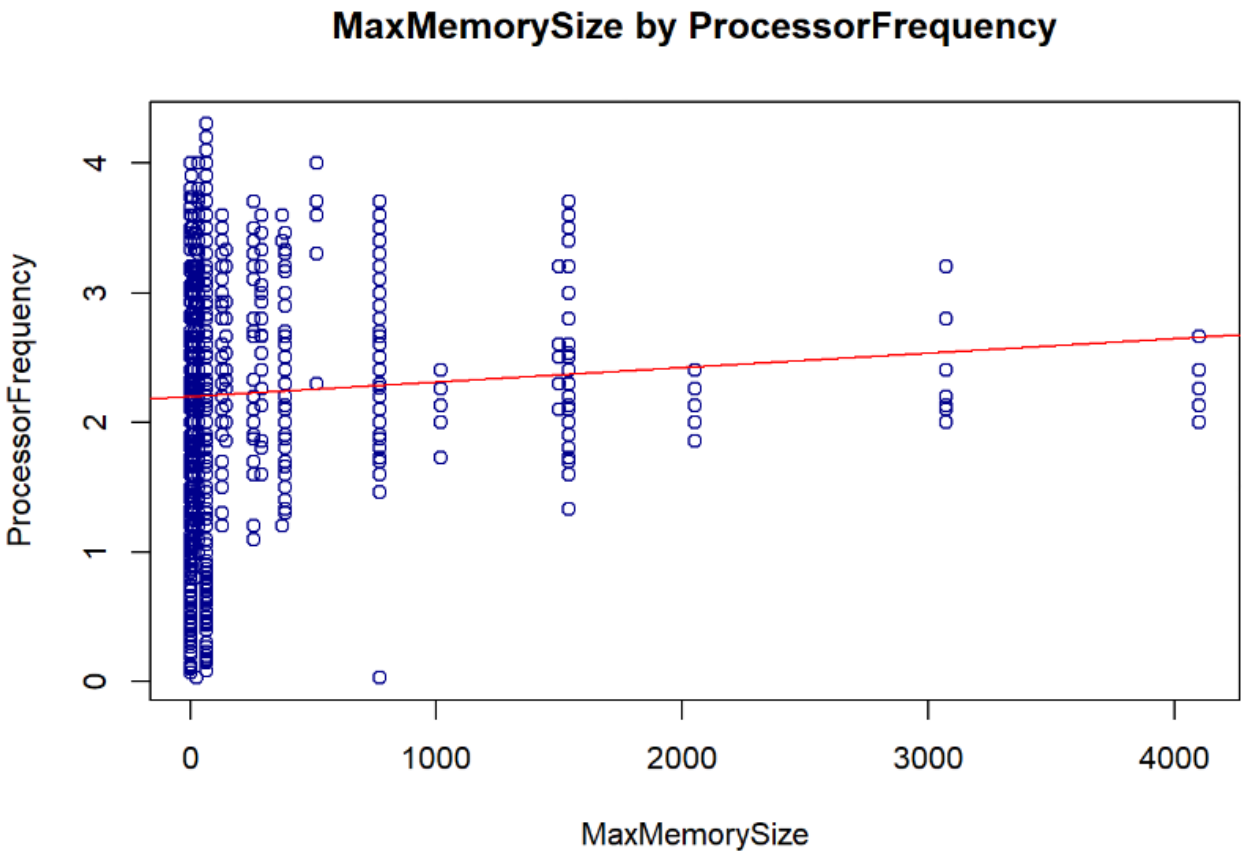
⇨ **Nhận xét**: biến Processor_Base_Frequency và biến np_of_Threads không có mối liên hệ tuyến tính.

Vẽ đồ thị phân phối của biến Processor_Frequency theo biến Max_memory_size

Cú pháp code R sử dụng lệnh plot để vẽ đồ thị phân phối Processor_Base_Frequency theo biến Max_memory_size

```
plot(new_CPU$Processor_Base_Frequency~new_CPU$Max_Memory_Size,
     main = "MaxMemorySize by ProcessorFrequency",
     xlab = "MaxMemorySize",
     ylab = "ProcessorFrequency",
     col = "darkblue"
)
abline(lm(new_CPU$Processor_Base_Frequency~new_CPU$Max_Memory_Size),col="red")
```

Kết quả thu được:



Hình 3.6: Đồ thị phân phối Processor_Base_Frequency theo biến Max_memory_size

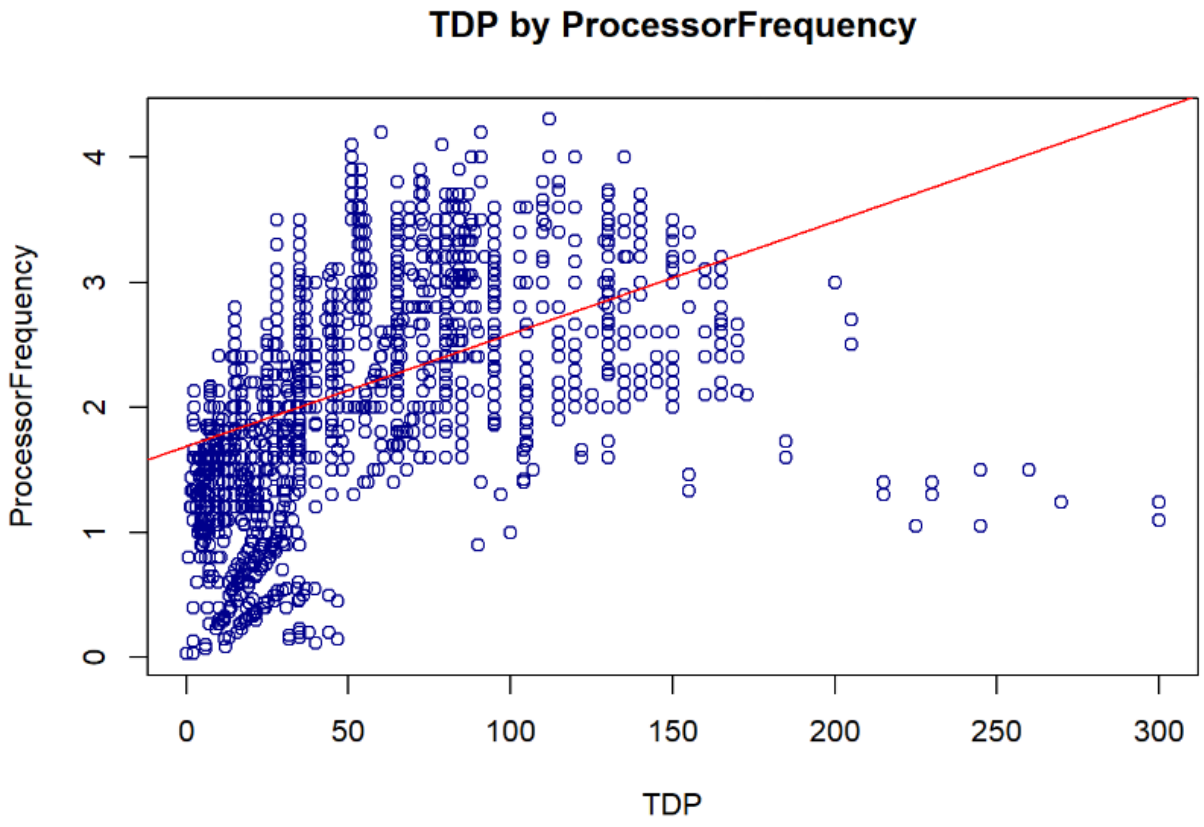
⇨ **Nhận xét**: biến Processor_Base_Frequency và biến Max_memory_size có mối liên hệ tuyến tính yếu.

Vẽ đồ thị phân phối của biến Processor_Frequency theo biến TDP

Cú pháp code R sử dụng lệnh plot để vẽ đồ thị phân phối Processor_Base_Frequency theo biến TDP

```
plot(new_CPU$Processor_Base_Frequency~new_CPU$TDP,
     main = "TDP by ProcessorFrequency",
     xlab = "TDP",
     ylab = "ProcessorFrequency",
     col = "darkblue"
)
abline(lm(new_CPU$Processor_Base_Frequency~new_CPU$TDP),col="red")
```

Kết quả thu được:



Hình 3.7: Đồ thị phân phối Processor_Base_Frequency theo biến TDP

⇨ **Nhận xét**: biến Processor_Base_Frequency và biến TDP có mối liên hệ tuyến tính yếu.

2.4 Xây dựng mô hình hồi quy tuyến tính đa biến để đánh giá các nhân tố ảnh hưởng đến *Processor_Base_Frequency* của CPU

Thiết lập mô hình hồi quy mà trong đó,

- Biến phụ thuộc là: *Processor_Base_Frequency*

- Các biến độc lập là:
nb_of_Cores","nb_of_Threads",“Lithography”"Max_Memory_Size", "TDP", “Launch _Year”,

Mô hình đc thiết lập như sau:

$$Processor\_Base\_Frequency = \beta_0 + \beta_1\, nb\_of\_cores + \beta_2\, nb\_of\_Threads + \beta_3\, Lithography + \beta_4\, Max\_Memory\_Size + \beta_5\, TDP + \beta_6\, Launch\_Year.$$

Ước lượng các hệ số $\beta_i, i = \overline{1,2,\dots 6}$ dựa trên model (1)

Cú pháp code R sử dụng lệnh lm để thiết lập mô hình hồi quy, sử dụng summary để thể hiện kết quả.

```
model1<-lm(Processor_Base_Frequency~nb_of_Cores+nb_of_Threads+Lithography+Max_Memory_Size+TDP+Launch_Year,data=new
_CPU)
summary(model1)
```

Kết quả thu được:

```
##
## Call:
## lm(formula = Processor_Base_Frequency ~ nb_of_Cores + nb_of_Threads +
##     Lithography + Max_Memory_Size + TDP + Launch_Year, data = new_CPU)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.26197 -0.28148  0.00154  0.30954  1.51695
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -3.100e+01  1.043e+01  -2.972  0.00299 **
## nb_of_Cores     -6.847e-02  3.812e-03 -17.964  < 2e-16 ***
## nb_of_Threads   -9.870e-03  3.265e-03  -3.023  0.00253 **
## Lithography     -7.261e-03  4.595e-04 -15.802  < 2e-16 ***
## Max_Memory_Size -3.945e-04  2.940e-05 -13.417  < 2e-16 ***
## TDP              1.663e-02  3.242e-04  51.291  < 2e-16 ***
## Launch_Year      1.644e-02  5.179e-03   3.175  0.00152 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4763 on 2276 degrees of freedom
## Multiple R-squared:  0.6671, Adjusted R-squared:  0.6662
## F-statistic: 760.1 on 6 and 2276 DF,  p-value: < 2.2e-16
```

⇨ **Nhận xét:** từ kết quả trên ta thu được các giá trị:

$\beta_0 = -3{,}100\text{e}+01,\ \beta_1 = -6{,}847\text{e}-02, \beta_2 = -9{,}870\text{e}-03, \beta_3 = -7{,}261\text{e}-03,$

$\beta_4 = -3{,}945\text{e}-04, \beta_5 = 1{,}663\text{e}-02, \beta_6 = 1{,}644\text{e}-02$

Từ đây đc mô hình hồi quy với phương trình như sau:

$Processor\_Base\_Frequency = (-3{,}100e+01) + (-6{,}847e-02) \times nb\_of\_Cores$
$(-9.870e-02) \times nb\_of\_Threads + (-7.261e-03) \times Lithography +$

$(-3.945e-04) \times Max\_Memory\_Size + (1.663e-02) \times TDP + (1.644e-02) \times Launch\ \_Year.$

Kiểm định các hệ số hồi quy:

- *Giả thiết H_0: Hệ số hồi quy không có ý nghĩa thống kê (B_i =0)*
- *Giả thiết H_1 : Hệ số hồi quy có ý nghĩa thống kê ($B_i \neq 0$)*

Do hệ số Pr(>|t|) của các hệ số tương ứng với các biến đều bé hơn mức ý nghĩa α=0,05 nên ta bác bỏ giả thuyết H_0 do đó hệ số ứng với các biến này đều có ý nghĩa với mô hình hồi quy ta xây dựng

Phân tích sự tác động lên Processor_Base_Frequency

$Processor\_Base\_Frequency = (-3{,}100e+01) + (-6{,}847e-02) \times nb\_of\_Cores$
$(-9.870e-02) \times nb\_of\_Threads + (-7.261e-03) \times Lithography + (-3.945e-04) \times Max\_Memory\_Size + (1.663e-02) \times TDP + (1.644e-02) \times Launch\ \_Year.$

Trước hết, ta thấy rằng p − value tương ứng với thống kê F bé hơn 2e - 16, có ý nghĩa rất cao. Điều này chỉ ra rằng, ít nhất một biến dự báo trong mô hình có ý nghĩa giải thích rất cao đến giá nhà. Để xét ảnh hưởng cụ thể của từng biến độc lập, ta xét trọng số (hệ số β_i) và p – value tương ứng. Ta thấy rằng p − value tương ứng với các biến đều bé hơn 2e – 16 (ngoài trừ biến *nb_of_Threads* có p – value = 0.00253 và biến *Launch _Year* có p – value = 0.00152), điều này nói lên rằng ảnh hưởng của các biến này có ý nghĩa rất cao lên *Processor_Base_Frequency*.

Mặt khác, hệ số hồi quy β_i của một biến dự báo cũng có thể được xem như ảnh hưởng trung bình lên biến phụ thuộc *Processor_Base_Frequency* khi tăng một đơn vị của biến dự báo, giả sử rằng các biến dự báo khác không đổi. Cụ thể, $\beta_1 = -6{,}847e-02$ thì khi biến *nb_of_Cores* tăng 1 đơn vị ta có thể kỳ vọng *Processor_Base_Frequency* giảm $6{,}847e-02$ đơn vị về mặt trung bình (giả sử rằng các biến dự báo khác không đổi). Với $\beta_2 = -9{,}870\text{e}-03$ thì khi *nb_of_Threads* tăng thêm 1 đơn vị, ta có thể kỳ vọng *Processor_Base_Frequency* giảm $9{,}870\text{e}-03$ đơn vị về mặt trung bình (giả sử rằng các biến dự báo khác không đổi). Hoặc với $\beta_3 = -7{,}261\text{e}-03$ thì khi *Lithography* tăng 1 đơn vị, ta có thể kỳ vọng *Processor_Base_Frequency* giảm $7{,}261\text{e}-03$ đơn vị về mặt trung bình (giả sử rằng các biến dự báo khác không đổi). Với $\beta_4 = -3{,}945\text{e}-04$ thì khi *Max_Memory_Size* tăng 1 đơn vị, ta có thể kỳ vọng *Processor_Base_Frequency* giảm

$3{,}945e-04$ đơn vị về mặt trung bình (giả sử rằng các biến dự báo khác không đổi). Với $\beta_5 = 1{,}663e-02$ thì khi *TDP* tăng 1 đơn vị, ta có thể kỳ vọng *Processor_Base_Frequency* tăng $1{,}663e-02$ đơn vị về mặt trung bình (giả sử rằng các biến dự báo khác không đổi).Với $\beta_6 = 1{,}644e-02$ thì khi *Launch _Year* tăng 1 đơn vị, ta có thể kỳ vọng *Processor_Base_Frequency* tăng $1{,}644e-02$ đơn vị về mặt trung bình (giả sử rằng các biến dự báo khác không đổi).

Hệ số R hiệu chỉnh bằng 0.6662 nghĩa là 66,62% sự biến thiên trong *Processor_Base_Frequency* đến được giải thích bởi các biến các biến độc lập.

Kiểm tra các giả định của mô hình

Nhắc lại các giả định của mô hình hồi quy: $Y_i = \beta_0 + \beta_1 X_1 + \beta_i X_i + \in_i, i = 1,\dots,n$

+ Tính tuyến tính của dữ liệu: mối quan hệ giữa biến dự báo X và biến phụ thuộc Y được giả sử là tuyến tính.

+ Sai số có phân phối chuẩn

+ Phương sai của các sai số là hằng số: $\epsilon_i \sim N(0,\sigma^2)$ *và sai số có kỳ vọng bằng 0*

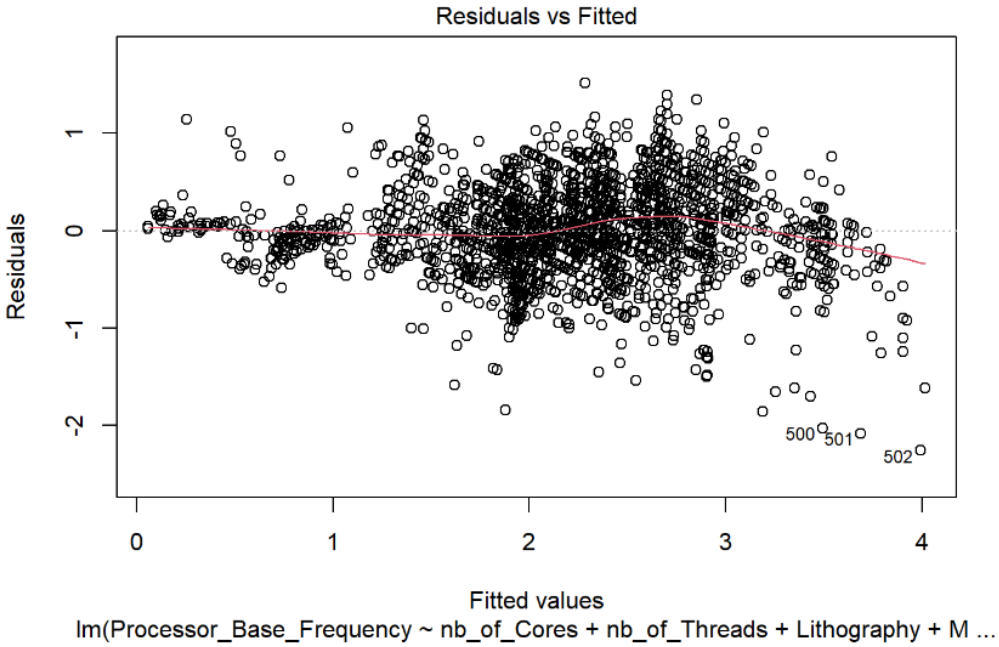
+ Các sai số $\epsilon_i, \epsilon_1, \epsilon_2, \dots, \epsilon_i$ thì độc lập với nhau.

Ta thực hiện phân tích thặng dư để kiểm tra các giả định của mô hình:

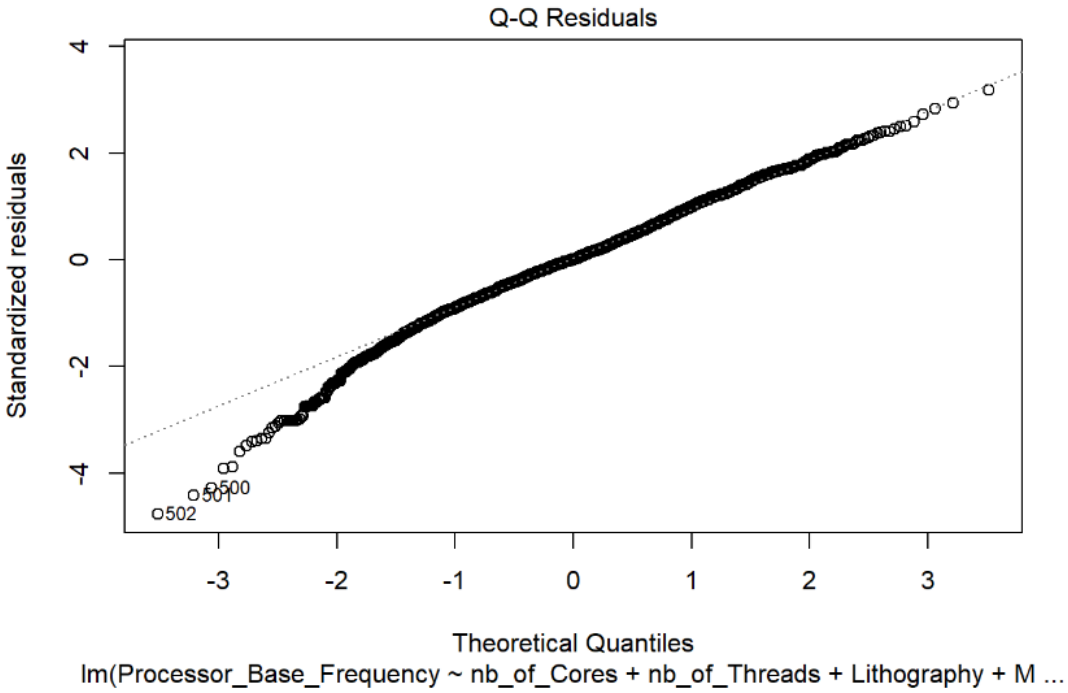
Code R thể hiện kết quả vẽ đồ thị dựa theo model (1)

```
plot(model1)
```

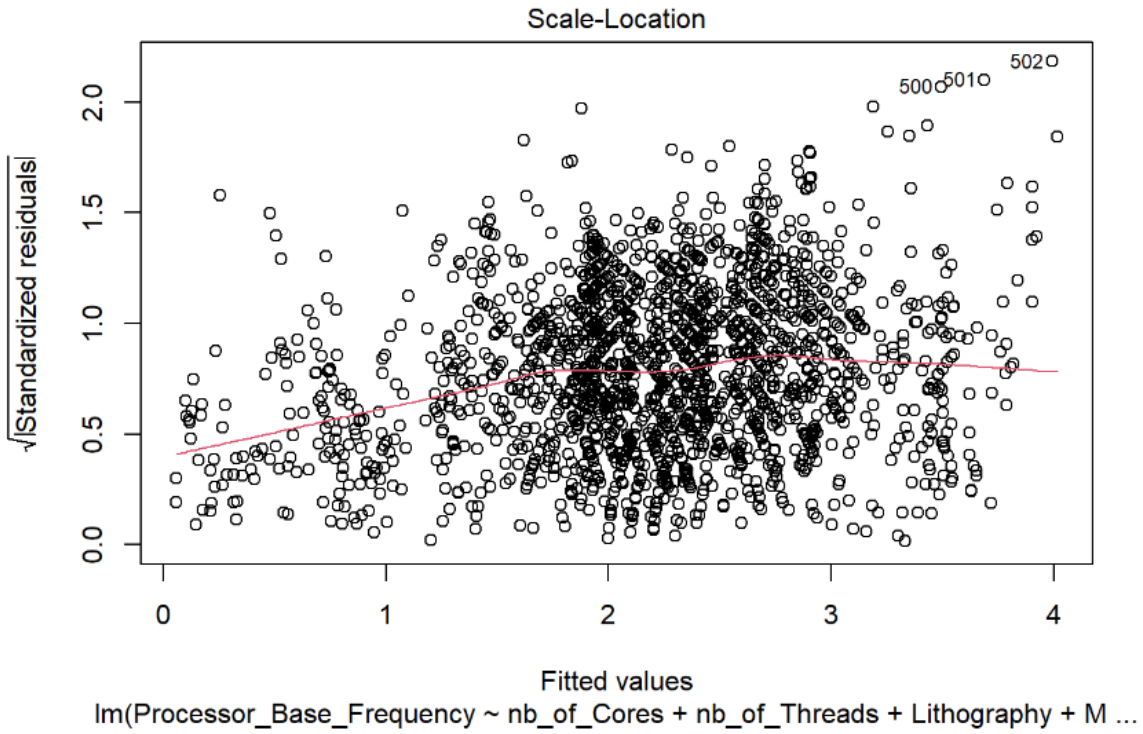
Đồ thị 1:



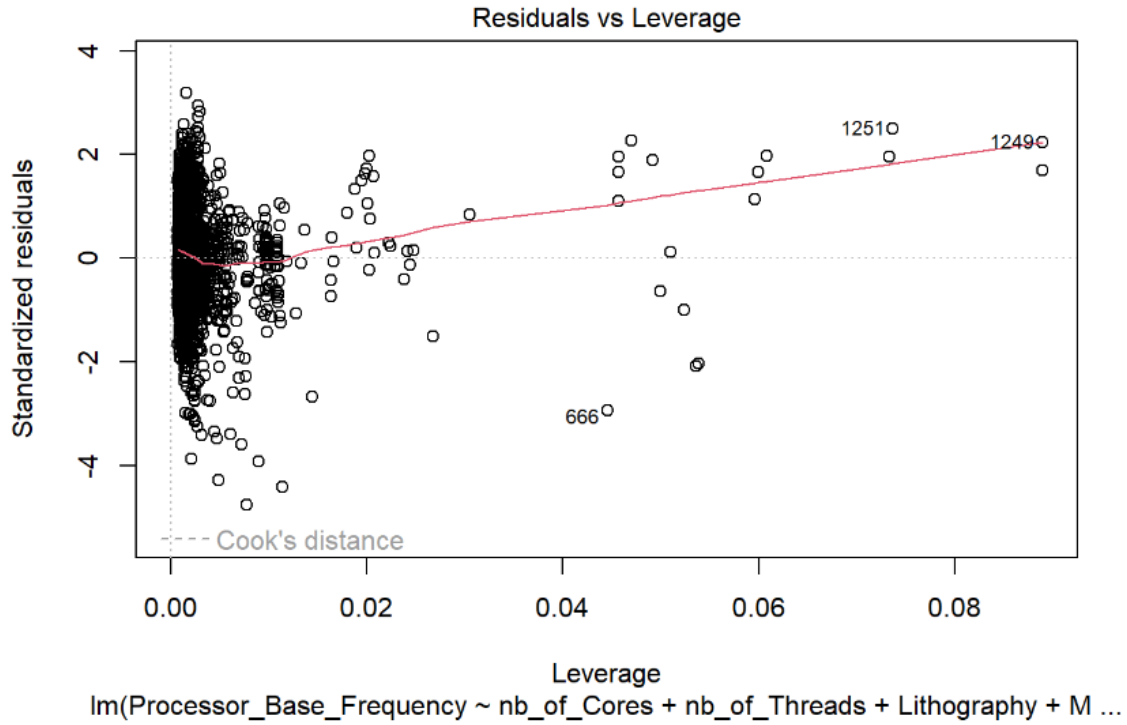
Đồ thị 2:



Đồ thị 3:



Đồ thị 4:



Ý nghĩa 4 đồ thị trên:

Đồ thị thứ 1 (Residuals vs Fitted) vẽ các giá trị dự báo với các giá trị thặng dư (sai số) tương ứng, dùng để kiểm tra tính tuyến tính của dữ liệu (giả định 1) và tính đồng nhất của các phương sai sai số (giả định 3). Nếu như giả định về tính tuyến tính của dữ liệu **KHÔNG** thỏa, ta sẽ quan sát thấy rằng các điểm thặng dư (residuals) trên đồ thị sẽ phân bố theo một hình mẫu (pattern) đặc trưng nào đó (ví dụ parabol). Nếu đường màu đỏ trên đồ thị phân tán là đường thẳng nằm ngang mà không phải là đường cong, thì giả định tính tuyến tính của dữ liệu được thỏa mãn. Để kiểm tra giả định thứ 3 (phương sai đồng nhất) thì các điểm thặng dự phải phân tán đều nhau xung quanh đường thẳng y = 0.

Đồ thị thứ 2 (Normal Q-Q) cho phép kiểm tra giả định về phân phối chuẩn của các sai số. Nếu các điểm thặng dư nằm trên cùng 1 đường thẳng thì điều kiện về phân phối chuẩn được thỏa.

Đồ thị thứ 3 (Scale - Location) vẽ căn bậc hai của các giá trị thặng dư được chuẩn hóa với các giá trị dự báo, được dùng để kiểm tra giả định thứ 3 (phương sai của các sai số là hằng số). Nếu như đường màu đỏ trên đồ thị là đường thẳng nằm ngang và các điểm thặng dư phân tán đều xung quanh đường thẳng này thì giả định thứ 3 được thỏa. Nếu như đường màu đỏ có độ dốc (hoặc cong) hoặc các điểm thặng dư phân tán không đều xung quanh đường thẳng này, thì giả định thứ 3 bị vi phạm.

Đồ thị thứ 4 (Residuals vs Leverage) cho phép xác định những điểm có ảnh hưởng cao (influen- tial observations), nếu chúng có hiện diện trong bộ dữ liệu. Những điểm có ảnh hưởng cao này có thể là các điểm outliers, là những điểm có thể gây nhiều ảnh hưởng nhất khi phân tích dữ liệu. Nếu như ta quan sát thấy một đường thẳng màu đỏ đứt nét (Cook’s distance), và có một số điểm vượt qua đường thẳng khoảng cách này, nghĩa là các điểm đó là các điểm có ảnh hưởng cao. Nếu như ta chỉ quan sát thấy đường thẳng khoảng cách Cook ở góc của đồ thị và không có điểm nào vượt qua nó, nghĩa không có điểm nào thực sự có ảnh hưởng cao.

Nhận xét

- Đồ thị 1 (Residuals vs Fiited) cho thấy giả định về tính tuyến tính của dữ liệu chưa thực sự được thỏa mản và kì vọng sai số bằng 0 chưa thật sự thỏa mãn
- Đồ thị Normal Q-Q cho thấy giả định sai số có phân phối chuẩn chưa thật sự thỏa mãn
- Đồ thị thứ 1 và thứ 3(Scale – Location) cho thấy giả định về tính chất đồng nhất của phương sai tương đối thỏa mãn
- Đồ thị thứ 4 chỉ ra có các quan trắc 1251,666 và 1249 có thể có ảnh hưởng cao tới bộ dữ liệu

2.5 Dự đoán

Thực hiện dự đoán *Processor_Base_Frequency* của CPU vào năm 2025 với các giá trị như sau:

nb_of_Cores là 28, *nb_of_Threads* là 56, *Lithography* là 14, *Max_Memory_Size* là 1,54, *TDP* là 165 và *Launch_Year* là 2025.

Cú pháp code R sử dụng lệnh predict để dự báo:

```
#thực hiện dự đoán cho Processor_Base_Frequency của CPU năm 2025
M=data.frame("nb_of_Cores"=28,"nb_of_Threads"=56,"Lithography"=14,
             "Max_Memory_Size"=1,54,"TDP"=165,
             "Launch_Year"=2025
               )
#xem kết quả
predict(model1,M,interval="confidence")
```

Kết quả thu được:

```
##        fit      lwr      upr
## 1 2.461782 2.274456 2.649109
```

⇨ **Nhận xét:** *Processor_Base_Frequency* của một CPU vào năm 2030 dự báo được là 12.461782, khoảng tin cậy cho giá trị dự báo là (2.274456, 2.649109).

III. TƯ LIỆU THAM KHẢO

[1] Nguyễn Đình Huy (chủ biên), Đậu Thế Cấp, Lê Xuân Đại - *Giáo trình xác suất và thống kê.*

[2] Science of Economics – *Hướng dẫn học R:* https://www.youtube.com/watch?v=_xHf8eEn2zI&list=PLMIaO-u3S5-jO2rMt8r8HD5ifiZv_Sd9O

[3] Learn to do SCIENCE - *Hồi quy tuyến tính đơn biến:* https://www.youtube.com/watch?v=_Qq03tM90DA

[4] Learn to do SCIENCE - *Hồi quy tuyến tính đa biến:* https://www.youtube.com/watch?v=La8HC_KxBxY&t=332s

[5] Learn to do SCIENCE - *Đánh giá mô hình hồi quy tuyến tính:* https://www.youtube.com/watch?v=wAP8z2fmE_c

[6] Xử Lí Missing Data Với MICE và VIM: https://rstudio-pubs-static.s3.amazonaws.com/301064_24da4725e1384388b47a0b08c9054779.html