

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC ỨNG DỤNG
BỘ MÔN TOÁN ỨNG DỤNG



BÀI TẬP LỚN
XÁC SUẤT VÀ THỐNG KÊ (MT2013)

Computer Parts - CPUs

Nhóm MT01 - HK232

Giảng viên hướng dẫn: ThS. Nguyễn Kiều Dung

STT	Họ và tên	MSSV	Lớp	Ngành học
1	Lâm Nữ Uyển Nhi	2212429	DL03	Kỹ thuật máy tính
2	Hồ Đăng Khoa	2211588	DL03	Kỹ thuật máy tính
3	Hoàng Văn Huy	2211173	DL03	Kỹ thuật máy tính
4	Hoàng Sỹ Xuân Sơn	2212937	DL03	Kỹ thuật máy tính
5	Trần Minh Tâm	2213039	DL01	Kỹ thuật máy tính
6	Lê Phan Bảo Như	2212466	DL03	Kỹ thuật máy tính

Thành phố Hồ Chí Minh, tháng 8 năm 2024

Danh sách thành viên

STT	Họ và tên	MSSV	Nhiệm vụ	Đóng góp
1	Lâm Nữ Uyên Nhi	2212429	Thực hiện: Tiền xử lý số liệu, Thống kê mô tả, Thống kê suy diễn, Thảo luận và mở rộng	100%
2	Hồ Đăng Khoa	2211588	Thực hiện: Tiền xử lý số liệu, Thống kê mô tả, Thống kê suy diễn, Thảo luận và mở rộng	100%
3	Hoàng Văn Huy	2211173	Thực hiện: Tiền xử lý số liệu, Thống kê mô tả, Thống kê suy diễn	100%
4	Hoàng Sỹ Xuân Sơn	2212937	- Thực hiện: Tiền xử lý số liệu, Thống kê mô tả, Thống kê suy diễn - Viết báo cáo	100%
5	Trần Minh Tâm	2213039	Thực hiện: Tiền xử lý số liệu, Thống kê mô tả, Thống kê suy diễn	100%
6	Lê Phan Bảo Như	2212466	- Thực hiện: Kiến thức nền, Tiền xử lý số liệu, Thống kê mô tả, Thống kê suy diễn - Viết báo cáo	100%

Mục lục

1	Tổng quan dữ liệu	4
1.1	Mô tả tập tin	4
1.2	Mô tả biến	4
2	Kiến thức nền	6
2.1	Khái niệm về một số đại lượng cơ bản của thống kê	6
2.2	Kiểm định giả thiết thống kê	7
2.2.1	Một số khái niệm	7
2.2.2	Bài toán ước lượng cho trung bình một mẫu	8
2.2.3	Bài toán kiểm định tỷ lệ hai mẫu	8
2.3	Phân tích phương sai ANOVA	9
2.3.1	Khái niệm	9
2.3.2	Phân loại	10
2.3.3	Giả thiết của bài toán ANOVA	10
2.4	Hồi quy tuyến tính	12
2.4.1	Khái niệm	12
2.4.2	Mô hình hồi quy tuyến tính đơn	12
2.4.3	Hồi quy tuyến tính bội	13
2.4.4	Mô hình hồi quy tuyến tính bội	13
2.4.5	Sự khác biệt giữa hồi quy tuyến tính bội và đơn	15
2.5	Các loại biểu đồ được sử dụng	16
2.5.1	Biểu đồ Histogram	16
2.5.2	Biểu đồ Boxplot	16
2.6	Kiến thức R	17
2.6.1	Tìm khoảng tin cậy một mẫu và kiểm định hai mẫu	17
2.6.2	Phân tích phương sai một yếu tố	17
2.6.3	Phân tích hồi quy tuyến tính	18

3	Tiền xử lý số liệu	19
3.1	Đọc dữ liệu	19
3.2	Trích xuất dữ liệu	19
3.3	Làm sạch dữ liệu	20
3.3.1	Kiểm tra các dữ liệu bị khuyết	20
3.3.2	Xử lý dữ liệu	21
3.3.3	Xử lý các biến có dữ liệu khuyết	24
3.4	Xác định ngoại lai	28
4	Thống kê mô tả	30
4.1	Các giá trị đặc trưng của mẫu đối với biến định lượng	30
4.2	Thống kê số lượng đối với các biến định tính	32
4.3	Đồ thị mô tả dữ liệu	33
4.3.1	Biểu đồ phân phối tần số của các biến định lượng	33
4.4	Phân phối chuẩn	39
4.5	Mối liên hệ giữa các biến	41
5	Thống kê suy diễn	45
5.1	Bài toán Kiểm định 1 mẫu	45
5.1.1	Mục tiêu	45
5.1.2	Bài toán	45
5.1.3	Tiến hành	45
5.1.4	Kết luận	47
5.2	Bài toán Kiểm định 2 mẫu	47
5.2.1	Mục tiêu	47
5.2.2	Bài toán	48
5.2.3	Kiểm định giả thiết	48
5.2.4	Tiến hành	48
5.2.5	Giải thích kết quả	50
5.2.6	Kết luận	50
5.3	Phân tích phương sai	50



5.3.1	Mục tiêu	50
5.3.2	Bài toán	51
5.3.3	Kiểm tra giả thuyết	51
5.3.4	Tiến hành	51
5.3.5	Kết luận	55
5.4	Phân tích hồi quy tuyến tính	55
5.4.1	Mục tiêu	55
5.4.2	Bài toán	55
5.4.3	Tiến hành	55
5.4.4	Ước lượng khoảng tin cậy các hệ số	58
5.4.5	Kiểm tra giả thiết	59
5.4.6	Dự đoán	66
5.4.7	Kết luận	67
6	Thảo luận và mở rộng	69
7	Nguồn dữ liệu và nguồn code	72
	Tài liệu tham khảo	73

1 Tổng quan dữ liệu

1.1 Mô tả tập tin

CPU (Central Progressing Unit) là một bộ phận vi mạch của máy tính có chức năng chuyên dụng để thực hiện các câu lệnh của chương trình máy tính.

Tập `Intel_CPUs.csv` là một tập chứa thông tin về các bộ vi xử lý (CPUs) của Intel. Tập chứa 45 thông số (hay *biến* - tương ứng với 45 cột trong tập) của 2283 bộ xử lý trung tâm (hay *quan trắc* - tương ứng với 2283 hàng trong tập).

1.2 Mô tả biến

Tập tin `Intel_CPUs.csv` chứa một nhiều thông số quan trọng giúp người dùng xác định được độ mạnh mẽ và hiệu quả của CPU. Có thể kể những thông số nổi bật như sau:

- `Product_Collection` : Tên dòng sản phẩm
- `Vertical_Segment` : Phân loại sản phẩm
- `Launch_Date` : Ngày ra mắt sản phẩm
- `Bus_Speed` : Tốc độ giao tiếp giữa CPU và mainboard
- `Cache` : Bộ nhớ đệm, cụ thể hơn là dung lượng của bộ nhớ đệm
- `Lithography` : Khoảng cách nhỏ nhất giữa các thành phần trên chip
- `Max_Memory_Bandwith` : Tốc độ băng thông tối đa của bộ nhớ, hay khả năng truyền tải dữ liệu của bộ nhớ
- `Max_nb_of_Memory_Channels` : Số lượng kênh bộ nhớ độc lập mà CPU có thể sử dụng để kết nối với các module RAM
- `Max_Memory_Size` : Dung lượng RAM lớn nhất mà một CPU có thể hỗ trợ
- `nb_of_Cores` : Số nhân - đơn vị xử lý trung tâm độc lập
- `Processor_Base_Frequency` : Tốc độ xung nhịp của CPU
- `Recommended_Customer_Price` : Giá bán lẻ được đề xuất

- **TDP**: Lượng nhiệt tối đa mà một bộ vi xử lý (hoặc các thành phần khác như GPU) được thiết kế để tỏa ra khi hoạt động dưới tải trọng tối đa trong điều kiện bình thường
- **DirectX_Support**: Phiên bản hỗ trợ của DirectX (một tập hợp các API do Microsoft phát triển, được sử dụng chủ yếu để xử lý các tác vụ đa phương tiện trên nền tảng Windows)
- **PCI_Express_Revision**: Phiên bản cụ thể của giao diện PCI Express (PCIe) mà một thiết bị hoặc bo mạch chủ hỗ trợ

2 Kiến thức nền

2.1 Khái niệm về một số đại lượng cơ bản của thống kê

- **Kì vọng toán (Expectation/Mean):** Là giá trị trung bình theo xác suất của biến ngẫu nhiên X . Kí hiệu là $E(X)$.
- **Phương sai (Variance):** Được định nghĩa bằng trung bình phương sai lệch giữa biến ngẫu nhiên với kì vọng của biến ngẫu nhiên X . Kí hiệu là $V(X)$.
- **Độ lệch chuẩn (Standard deviation):** 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. Kí hiệu là $\sqrt{V(X)}$.
- **Cực tiểu (Min):** Là giá trị nhỏ nhất trong toàn bộ các giá trị của một tập mẫu.
- **Cực đại (Max):** Là giá trị lớn nhất trong toàn bộ các giá trị của một tập mẫu.
- **Trung vị (Median):** Là giá trị nằm giữa, chia tập giá trị X thành hai phần bằng nhau.
- **Tứ phân vị:** Là đại lượng mô tả sự phân bố và sự phân tán của tập dữ liệu. Tứ phân vị có 3 giá trị, đó là tứ phân vị thứ nhất, thứ nhì và thứ ba. Ba giá trị này chia một tập hợp dữ liệu (đã sắp xếp dữ liệu theo trật từ từ bé đến lớn) thành 4 phần có số lượng quan sát đều nhau.
 - **Giá trị tứ phân vị thứ hai Q_2** chính bằng giá trị trung vị.
 - **Giá trị tứ phân vị thứ nhất Q_1** bằng trung vị phần dưới.
 - **Giá trị tứ phân vị thứ ba Q_3** bằng trung vị phần trên.
 - **Khoảng tứ phân vị IQR:** Là khoảng chênh lệch giữa hai giá trị Q_3 và Q_1 .
 - **Giá trị ngoại lai** là phần dữ liệu bất thường, là những quan sát trong tập dữ liệu có giá trị khác biệt rõ rệt so với các quan sát khác. **Những điểm này thường nằm xa so với phần lớn dữ liệu** và có thể là kết quả của sai sót đo lường, nhập liệu, hoặc là những trường hợp cực đoan hợp lý.

2.2 Kiểm định giả thiết thống kê

Kiểm định giả thiết thống kê là dùng các thống kê từ một mẫu để khẳng định hay bác bỏ một giả thiết nào đó nói về tổng.

Giả sử cần kiểm định một giả thiết H . Khi kiểm định có thể xảy ra một trong hai loại sai lầm sau:

- Loại 1: Bác bỏ H trong khi H đúng.
- Loại 2: Chấp nhận H trong khi H sai.

Phương pháp chung để kiểm định là cho phép xác suất xảy ra sai lầm loại 1 không quá α , số α gọi là mức ý nghĩa kiểm định. Với mức ý nghĩa đã cho, ta chấp nhận H nếu xác suất xảy ra sai lầm loại 2 nhỏ nhất.

2.2.1 Một số khái niệm

- **Giả thiết không H_0** : Là giả thiết về yếu tố cần kiểm định của tổng thể ở trạng thái bình thường, không chịu tác động của các hiện tượng liên quan. Yếu tố trong H_0 phải được xác định cụ thể.
- **Giả thiết đối H_1** : Là một mệnh đề mâu thuẫn với H_0 , thể hiện xu hướng cần kiểm định.
- **Tiêu chuẩn kiểm định**: Là hàm thống kê $G = G(X_1, X_2, \dots, X_n, \theta_0)$ được xây dựng trên mẫu ngẫu nhiên $W = W(X_1, X_2, \dots, X_n)$ và tham số θ_0 liên quan đến H_0 . Điều kiện đặt ra với thống kê G là nếu H_0 đúng thì quy luật phân phối xác suất của G phải hoàn toàn xác định.
- **Miền bác bỏ giả thuyết RR**: Là miền số thực thoả xác suất G thuộc vào đó với điều kiện H_0 đúng là α . Tức là

$$P(G \in \mathbb{R} | H_0 \text{ đúng}) = \alpha$$

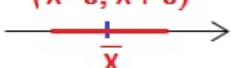
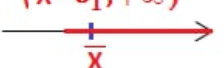
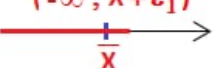
- **Quy tắc kiểm định**: từ mẫu thực nghiệm, ta tính được một giá trị cụ thể của tiêu chuẩn kiểm định, gọi là **giá trị kiểm định thống kê**

$$g_{qs} = G(x_1, x_2, \dots, x_n, \theta_0)$$

Theo nguyên lý xác suất bé, biến cố $G \in \mathbb{R}$ có xác suất nhỏ nên với một mẫu thực nghiệm ngẫu nhiên, nó không thể xảy ra. Do đó

- Nếu $g_{qs} \in \mathbb{R}$: bác bỏ giả thuyết H_0 , thừa nhận giả thuyết H_1 .
- Nếu $g_{qs} \notin \mathbb{R}$: chưa đủ dữ liệu khẳng định H_0 sai. Vì vậy ta chưa thể chứng minh được H_1 đúng.

2.2.2 Bài toán ước lượng cho trung bình một mẫu

Tham số cần ước lượng	Phân bố của tổng thể	Khoảng tin cậy đối xứng	Khoảng tin cậy bên trái	Khoảng tin cậy bên phải
Trung bình tổng thể μ (2)		$(\bar{X} - \varepsilon; \bar{X} + \varepsilon)$ 	$(\bar{X} - \varepsilon_1; +\infty)$ 	$(-\infty; \bar{X} + \varepsilon_1)$ 
	<ul style="list-style-type: none"> • Phân phối chuẩn • Đã biết σ^2 (2a)	$\varepsilon = \frac{z_{\alpha/2} \times \sigma}{\sqrt{n}}$	$\varepsilon_1 = \frac{z_{\alpha} \times \sigma}{\sqrt{n}}$	
	<ul style="list-style-type: none"> • Phân phối chuẩn • Chưa biết σ^2 (2b)	$\varepsilon = t_{\frac{\alpha}{2}; (n-1)} \times \frac{s}{\sqrt{n}}$	$\varepsilon_1 = t_{\alpha; (n-1)} \times \frac{s}{\sqrt{n}}$	
	<ul style="list-style-type: none"> • Phân phối tùy ý • Mẫu lớn ($n \geq 30$) (2c) <i>Sử dụng định lý giới hạn</i>	$\varepsilon = \frac{z_{\alpha/2} \times \sigma}{\sqrt{n}}$	$\varepsilon_1 = \frac{z_{\alpha} \times \sigma}{\sqrt{n}}$ <i>Trường hợp chưa biết σ thì thay thế bởi s</i>	

2.2.3 Bài toán kiểm định tỷ lệ hai mẫu

Bài toán: Cho hai tổng thể độc lập X và Y , trong đó tỷ lệ phần tử mang dấu hiệu A nào đó trong hai tổng thể lần lượt là p_1 và p_2 (p_1 và p_2 chưa biết). Dùng thống kê từ hai mẫu thu được từ hai tổng thể, thực hiện kiểm định để so sánh p_1 và p_2 , xét với mức ý nghĩa α .

Giả định:

Hai mẫu độc lập, cỡ mẫu lớn và $n_1 \cdot f_1 > 5$ và $n_1(1 - f_1) > 5$ và $n_2 \cdot f_2 > 5$ và $n_2(1 - f_2) > 5$ (với n_1, n_2 lần lượt là hai mẫu được chọn ra từ hai tổng thể và $f_1 = \frac{m_1}{n_1}, f_2 = \frac{m_2}{n_2}$ lần lượt là tỷ lệ phần tử mang dấu hiệu A trong mẫu)

Dàn ý tóm tắt:

1. Đặt giả thuyết và xác định miền bác bỏ tương ứng (sử dụng một trong hai cách viết miền bác bỏ):

Giả thuyết	Miền bác bỏ (1)	Miền bác bỏ (2)
$H_0 : p_1 = p_2 \quad H_1 : p_1 \neq p_2$	$RR = (-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; +\infty)$	$ z_0 > z_{\alpha/2}$
$H_0 : p_1 = p_2 \quad H_1 : p_1 < p_2$	$RR = (-\infty; -z_{\alpha})$	$z_0 < -z_{\alpha}$
$H_0 : p_1 = p_2 \quad H_1 : p_1 > p_2$	$RR = (z_{\alpha}; +\infty)$	$z_0 > z_{\alpha}$

2. Tính giá trị thống kê kiểm định

$$z_0 = \frac{f_1 - f_2}{\sqrt{\frac{\bar{f}(1 - \bar{f})}{\bar{n}}}} \text{ với } \bar{n} = \frac{n_1 \cdot n_2}{n_1 + n_2}; \bar{f} = \frac{m_1 + m_2}{n_1 + n_2}$$

3. Đưa ra kết luận

• Trường hợp đặt $H_1 : p_1 \neq p_2$:

- Nếu $|z_0| > z_{\alpha/2} \Leftrightarrow z_0 \in RR \Rightarrow$ Bác bỏ H_0 , chấp nhận H_1
- Nếu $|z_0| < z_{\alpha/2} \Leftrightarrow z_0 \notin RR \Rightarrow$ không bác bỏ H_0 (chưa bác bỏ được H_0 , chấp nhận H_1)

• Trường hợp đặt $H_1 : p_1 < p_2$:

- Nếu $z_0 < -z_{\alpha} \Leftrightarrow z_0 \in RR \Rightarrow$ Bác bỏ H_0 , chấp nhận H_1
- Nếu $z_0 > -z_{\alpha} \Leftrightarrow z_0 \notin RR \Rightarrow$ không bác bỏ H_0 (chưa bác bỏ được H_0 , chấp nhận H_0)

• Trường hợp đặt $H_1 : p_1 > p_2$:

- Nếu $z_0 > z_{\alpha} \Leftrightarrow z_0 \in RR \Rightarrow$ Bác bỏ H_0 , chấp nhận H_1
- Nếu $z_0 < z_{\alpha} \Leftrightarrow z_0 \notin RR \Rightarrow$ không bác bỏ H_0 (chưa bác bỏ được H_0 , chấp nhận H_0)

2.3 Phân tích phương sai ANOVA

2.3.1 Khái niệm

Phân tích phương sai (Analysis of Variance – ANOVA) là một mở rộng của phương pháp kiểm định t cho các mẫu độc lập khi so sánh trung bình của các nhóm gồm các quan sát độc lập. ANOVA có thể so sánh nhiều hơn hai nhóm.

2.3.2 Phân loại

Có hai loại phân tích phương sai thông dụng nhất:

- Phân tích phương sai một yếu tố (one-way ANOVA): Kiểm định giả thiết trung bình bằng nhau của các nhóm mẫu với khả năng phạm sai lầm chỉ là 5%, trong một mẫu chỉ xem xét một yếu tố hoặc một biến độc lập.
- Phân tích phương sai hai yếu tố (two-way ANOVA): Là phần mở rộng của phân tích phương sai một yếu tố. Với two-way ANOVA, sẽ có hai yếu tố độc lập được dùng để phân tích.

2.3.3 Giả thiết của bài toán ANOVA

- Các tổng thể có phân phối chuẩn $\mathcal{N}(\mu_i, \sigma_i^2)$; $i = 1; 2; \dots; k$. Với k là số lượng tổng thể (thông thường $k \geq 3$).
- Phương sai các tổng thể bằng nhau ($\sigma_1^2 = \sigma_2^2 = \dots = \sigma_n^2$).
- Các mẫu quan sát (từ các tổng thể) được lấy độc lập.

Giả thiết kiểm định $H_0: \mu_1 = \mu_2 = \dots = \mu_i$.

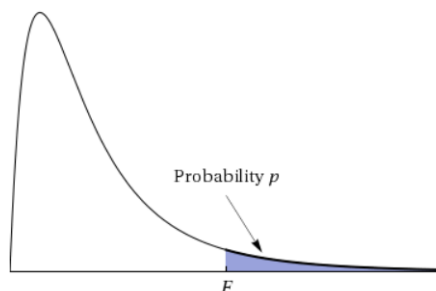
Giả thiết đối $H_1: \exists \mu_i \neq \mu_j$ với $i \neq j$.

- Quy trình tính toán ANOVA được tổng hợp từ bảng sau:

Nguồn biến thiên	Tổng bình phương chênh lệch	Bậc tự do	Phương sai	Tiêu chuẩn kiểm định
Giữa các nhóm	SSB	$k - 1$	$MSB = \frac{SSB}{k - 1}$	$F = \frac{MSB}{MSW}$
Trong nội bộ nhóm	SSW	$N - k$	$MSW = \frac{SSW}{N - k}$	
Tổng cộng	SST	$N - 1$		

- Miền bác bỏ RR: Là miền bác bỏ nằm ở bên phải của phân phối F, được xác định dựa trên mức ý nghĩa thống kê α cùng với bậc tự do của biến đổi giữa các nhóm (dfB) và biến đổi bên trong từng nhóm (dfW). Tức là

$$RR = f_{\alpha}(df_B, df_W); +\infty = f_{\alpha}(k - 1, N - k); +\infty)$$



Hình 1: Phân phối Fisher

- Kết luận

- Nếu $F \in RR$: ta bác bỏ H_0 , thừa nhận giả thiết H_1 .
- Nếu $F \notin RR$: ta chưa đủ dữ liệu khẳng định H_0 sai. Vì vậy ta chưa thể chứng minh được H_1 đúng.

- Các kiểm định hậu ANOVA: LSD (Least Significant Difference).

Dùng LSD test: Kiểm định so sánh lần lượt tất cả các cặp trung bình của hai nhóm khác nhau với các giả thiết tương ứng.

$$H_0 : \mu_i = \mu_j$$

$$H_1 : \mu_i \neq \mu_j \text{ với } i \neq j$$

Tính giá trị thống kê kiểm định: $LSD_{j;j} = t_{\alpha/2}(N - k) \sqrt{MSW(\frac{1}{n_i} + \frac{1}{n_j})}$

Bác bỏ H_0 khi $|\bar{x}_i - \bar{x}_j| > LSD_{i;j}$

Dùng các khoảng tin cậy (LSD confidence intervals) để ước lượng chênh lệch trung bình 2 nhóm bất kì.

Khoảng ước lượng LSD với độ tin cậy $1 - \alpha$ cho độ lệch $(\mu_i - \mu_j)$ là:

$$((\bar{x}_i - \bar{x}_j) - \epsilon; (\bar{x}_i - \bar{x}_j) + \epsilon)$$

Trong đó: $\epsilon = LSD_{i;j}$ và được tính như công thức nêu trên.

- Nếu khoảng tin cậy không chứa 0: Có sự khác biệt giữa μ_i và μ_j .
- Nếu khoảng tin cậy chứa 0: Không kết luận được có sự khác biệt giữa μ_i và μ_j .

2.4 Hồi quy tuyến tính

2.4.1 Khái niệm

Phân tích hồi quy tuyến tính là nghiên cứu mối liên hệ phụ thuộc của một biến (gọi là biến phụ thuộc hay biến được giải thích) với một hay nhiều biến khác (được gọi là biến độc lập hay biến giải thích).

2.4.2 Mô hình hồi quy tuyến tính đơn

Một mô hình thống kê tuyến tính đơn (*Simple linear regression model*) liên quan đến một biến ngẫu nhiên Y và một biến giải thích X là Phương trình 1 có dạng

$$Y_i = f(x) + \epsilon_i = \beta_0 + \beta_1 X_i + \epsilon_i \quad (1)$$

Trong đó:

- β_0, β_1 là các tham số chưa biết, được gọi là các hệ số hồi quy
- X_i là biến độc lập, giải thích cho Y
- ϵ_i là thành phần sai số, ϵ_i được giả sử có phân phối chuẩn với $E(\epsilon_i) = 0$ và $Var(\epsilon_i) = \sigma^2$

Trong mô hình (1), sự thay đổi của Y được giả sử ảnh hưởng bởi hai yếu tố:

- Mối liên hệ tuyến tính của X và Y :

$$E[Y_i|X_i] = \beta_0 + \beta_1 X_i,$$

Trong đó, β_0 được gọi là hệ số chặn (*intercept*) và β_1 gọi là hệ số góc (*slope*).

- Tác động của các yếu tố khác (không phải X): thành phần sai số ϵ .

Một mô hình hồi quy tuyến tính đơn cần các giả định:

- Các thành phần sai số ϵ_i là độc lập với nhau.
- $\epsilon_i \sim N(0, \sigma^2)$ hoặc $Y \sim N(\beta_0 + \beta_1 x, \sigma^2)$

2.4.3 Hồi quy tuyến tính bội

Hồi quy tuyến tính bội là một phương pháp được sử dụng khi ta muốn dự đoán giá trị của một biến phản hồi dựa trên giá trị của hai hoặc nhiều biến giải thích khác. Biến mà chúng ta muốn dự đoán được gọi là biến phản hồi (biến phụ thuộc). Các biến mà ta đang sử dụng để dự đoán giá trị của biến phản hồi được gọi là các biến giải thích (biến dự báo, biến phụ thuộc).

Hồi quy bội cũng cho phép chúng ta xác định sự phù hợp tổng thể của mô hình và đóng góp tương đối của từng yếu tố dự báo và tổng phương sai được giải thích.

2.4.4 Mô hình hồi quy tuyến tính bội

Mô hình hồi quy tuyến tính bội liên quan đến biến Y và $(p - 1)$ biến dự báo X có dạng:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \dots + \beta_{p-1} X_{i(p-1)} + \epsilon_i$$

hay còn được viết dưới dạng:

$$Y_i = \beta_0 + \sum_{k=1}^{p-1} \beta_k X_{ik} + \epsilon_i$$

Trong đó:

- Y_i là biến phản hồi hay biến phụ thuộc (*response variable/dependent variable*).
- X_{ik} là các biến giải thích hay biến độc lập (*explanatory variables/independent variables*).
- β_k là tham số của các biến độc lập trong đó β_0 là hệ số hồi quy (hệ số chặn, *intercept*).
- $\epsilon_i \sim N(0, \sigma^2)$ là số hạng nhiễu hay sai số ngẫu nhiên.
- Y_i , với $i = 1, 2, \dots, n$.

Khi $p - 1 = 1$, mô hình hồi quy $Y_i = \beta_0 + \beta_1 X_{i1} + \epsilon_i$ là mô hình hồi quy tuyến tính đơn. Bên cạnh đó, ta có các giả định cho mô hình hồi quy như sau:

- ϵ_i có phân phối chuẩn và phương sai không đổi.
- Các biến X độc lập với nhau.

Kiểm định từng tham số hồi quy tổng thể:

- Trường hợp $\beta_k = 0$ thì X_i và Y không có mối quan hệ nào.
- Trường hợp $\beta_k > 0$ thì X_i và Y có mối quan hệ thuận.
- Trường hợp $\beta_k < 0$ thì X_i và Y có mối quan hệ nghịch.
- Ở mức ý nghĩa α , giả thiết vô hiệu H_0 được kiểm định ở các trường hợp sau:

$$(1) H_0 : \beta_k \leq 0 \text{ và } H_1 : \beta_k > 0$$

$$(2) H_0 : \beta_k \geq 0 \text{ và } H_1 : \beta_k < 0$$

- Giá trị kiểm định: $t = \frac{\beta_k}{SE(\beta_k)}$.
- Quyết định bác bỏ giả thiết H_0 :
 - Bác bỏ giả thiết (1) khi $t > t_{n-p,\alpha}$
 - Bác bỏ giả thiết (1) khi $t < -t_{n-p,\alpha}$

Ý nghĩa hồi quy:

- Trường hợp: $H_0 : \beta_0 = \beta_1 = \dots = \beta_{p-1} = 0 \rightarrow Y_i = \beta_0 + \epsilon_i$, kí hiệu các dữ liệu thỏa là y_{0i} . Khi đó: $\bar{y} = y_{0i}$
- Trường hợp: $H_1 : \text{có ít nhất một } \beta_j \neq 0, j = 1, 2, \dots, (p-1)$

Biến thiên	Tổng độ lệch bình phương	Bậc tự do	Phương sai	Giá trị kiểm định
Hồi quy	$SSR = \sum_{i=1}^n (y_{1i} - \bar{y})^2$	$p - 1$	$MSR = \frac{SSR}{p - 1}$	$F = \frac{MSR}{MSE}$
Sai số	$SSE = \sum_{i=1}^n (y_i - y_{1i})^2$	$n - p$	$MSE = \frac{SSE}{n - p}$	
Tổng	$SST = \sum_{i=1}^n (y_i - \bar{y})^2$	$n - 1$		

Bảng 1: Bảng ANOVA cho mô hình hồi quy tuyến tính tổng quát

Nếu $F \leq F_{p-1, n-p, \alpha}$, ta chấp nhận giả thiết H_0 . Ngược lại, nếu $F > F_{p-1, n-p, \alpha}$, ta bác bỏ giả thiết H_0 .

Ta có hệ số xác định $R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$

Đại diện cho tỷ lệ biến thiên theo biến Y_i với các biến dự báo X_{1i}, X_{2i}, \dots . Hệ số R^2 nói lên tính chặt chẽ giữa biến phản hồi Y_i và các biến dự báo X_{ki} .

Ta có $0 \leq R^2 \leq 1$. Khi $R^2 = 0$ thì chấp nhận H_0 . Khi $R^2 = 1$ thì tất cả các quan sát đều nằm trên mặt đáp ứng.

Nếu chúng ta bắt đầu với một mô hình hồi quy tuyến tính đơn giản với biến dự báo X_{1i} sau đó thêm biến dự báo X_{2i} , SSE sẽ giảm (hoặc giữ nguyên) trong khi SST không đổi và do đó R^2 sẽ tăng (hoặc giữ nguyên). Nói cách khác, R^2 luôn tăng (hoặc giữ nguyên) khi có nhiều yếu tố dự báo vào mô hình hồi quy tuyến tính bội, ngay cả khi các yếu tố dự báo được thêm vào không liên quan đến biến phản hồi. Do đó bản thân R^2 không thể được sử dụng để giúp chúng ta xác định những yếu tố dự báo nào nên được đưa vào một mô hình và yếu tố nào nên được loại bỏ.

Hệ số xác định đã điều chỉnh:

$$R^2_{\alpha} = \frac{\frac{SSR}{n-p}}{\frac{SST}{n-1}} = 1 - \left(\frac{n-1}{1-p} \right) (1 - R^2)$$

Hệ số lạm phát phương sai (Variance Inflation Factor - VIF): Hệ số đo lường mối tương quan và cường độ tương quan giữa các biến dự đoán trong mô hình hồi quy. Hệ số lạm phát phương sai của biến ngẫu nhiên X_j được tính bởi công thức

$$VIF_j = \frac{1}{1 - R_j^2}$$

Trong đó, R_j^2 là hệ số xác định từ mô hình hồi quy tuyến tính của X_j theo các biến độc lập còn lại trong mô hình.

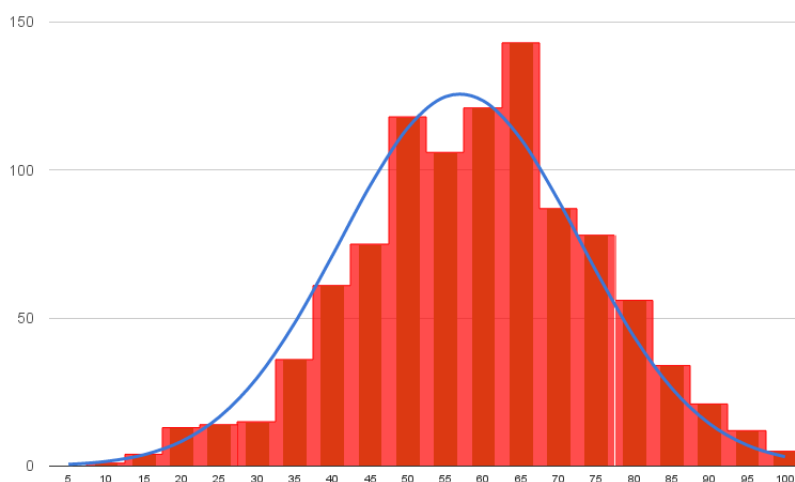
2.4.5 Sự khác biệt giữa hồi quy tuyến tính bội và đơn

Hồi quy tuyến tính bội và đơn là hai phương pháp khác nhau trong mô hình hồi quy, một phần của thống kê và máy học được sử dụng để dự đoán giá trị của một biến phụ thuộc dựa trên một hoặc nhiều biến độc lập.

2.5 Các loại biểu đồ được sử dụng

2.5.1 Biểu đồ Histogram

Biểu đồ histogram là một dạng biểu đồ dùng để thể hiện sự phân bố của một tập hợp dữ liệu liên tục bằng cách chia dữ liệu thành các khoảng giá trị và hiển thị số lượng hoặc tần suất các giá trị trong mỗi khoảng. Trục ngang của biểu đồ đại diện cho các khoảng giá trị của dữ liệu, còn trục dọc thể hiện số lượng hoặc tần suất các giá trị rơi vào mỗi khoảng đó.

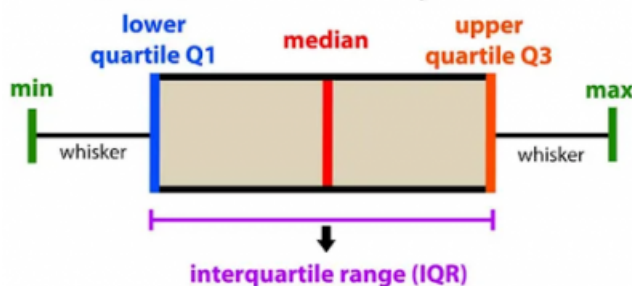


Hình 2: Hình ảnh minh họa biểu đồ Histogram

2.5.2 Biểu đồ Boxplot

Biểu đồ hộp (hay còn gọi là biểu đồ boxplot hoặc box-and-whisker plot) là một công cụ trực quan hóa dữ liệu dùng để tóm tắt và hiển thị sự phân bố của một tập hợp dữ liệu thông qua các số liệu thống kê như giá trị trung vị (median), tứ phân vị (quartiles), và các giá trị ngoại lai (outliers).

introduction to data analysis: Box Plot



Hình 3: Hình ảnh minh họa biểu đồ Boxplot

2.6 Kiến thức R

2.6.1 Tìm khoảng tin cậy một mẫu và kiểm định hai mẫu

Trong ngôn ngữ R, các hàm tìm khoảng tin cậy cho trung bình (cũng là các hàm kiểm định cho trung bình) cùng các tham số trong từng hàm được xây dựng dựa trên các trường hợp đã phân ra trong phần lý thuyết. Cụ thể như sau:

- Khi tổng thể tuân theo phân phối chuẩn, đã biết phương sai, ta dùng hàm `z.test()`.
- Khi tổng thể tuân theo phân phối chuẩn, không biết phương sai, ta dùng hàm `t.test()`.
- Khi tổng thể không tuân theo phân phối chuẩn nhưng cỡ mẫu lớn ta dùng hàm `z.test()`. Tuy nhiên, khi cỡ mẫu lớn thì phân phối chuẩn và phân phối Student xấp xỉ nhau nên trong trường hợp này, ta cũng có thể dùng phân phối Student để thay thế, tức dùng hàm `t.test()`.

2.6.2 Phân tích phương sai một yếu tố

2.6.2.1 Kiểm định phân phối chuẩn

Để kiểm định một mẫu có tuân theo phân phối chuẩn hay không, ta sử dụng hàm `shapiro.test()` để thực hiện kiểm định Shapiro.

2.6.2.2 Kiểm định sự tương đồng về phương sai giữa các mẫu

Để kiểm tra sự tương đồng về phương sai giữa các nhóm quan sát, ta sử dụng hàm `leveneTest()` để thực hiện kiểm định Levene.

2.6.2.3 Mô hình ANOVA một yếu tố

Trong ngôn ngữ R, hàm `aov()` dùng để thực hiện phân tích phương sai cho một mẫu liên tục được chia ra theo từng nhóm dựa trên mẫu phân loại.

Lưu ý, ta cần dùng thêm hàm `summary()` cho kết quả của hàm `aov()` để thu được thống kê chi tiết.

2.6.2.4 Phân tích hậu định

Sau khi tiến hành xây dựng mô hình ANOVA, ta cần kiểm tra xem giữa các nhóm có sự khác nhau như thế nào. Đồng thời kiểm tra sự khác nhau giữa các nhóm có ý nghĩa thống kê hay không. Ta có thể sử dụng hàm `TukeyHSD()`, tức là sử dụng phương pháp phân tích hậu định Tukey's Honest Significant Difference (TukeyHSD).

2.6.3 Phân tích hồi quy tuyến tính

Để xây dựng mô hình hồi quy tuyến tính, ta bổ sung thêm gói thư viện `car`, chuyên dùng để cung cấp các chức năng trong việc phân tích hồi quy.

Bên cạnh đó, ngôn ngữ R có hỗ trợ hàm `lm()` để tiến hành xây dựng mô hình hồi quy tuyến tính với các tham số phù hợp.

Để xác định khoảng ước lượng cho các hệ số của mô hình hồi quy tuyến tính tổng thể với mức ý nghĩa α cho trước, ta có thể sử dụng hàm `confint()` sau khi tìm được mô hình hồi quy tuyến tính mẫu.

Sau khi xây dựng mô hình thành công, ta cần phải kiểm định các giả thiết của mô hình. Ngôn ngữ R cũng hỗ trợ thêm các hàm sau để có thể tiến hành kiểm định:

- `fitted()` : Trích các giá trị trung bình từ mô hình đã xây dựng.
- `resid()` : Trích các phần dư từ mô hình đã xây dựng.
- `shapiro.test()` : Kiểm định Shapiro.
- `vif()` : Kiểm định tính đa cộng tuyến của các biến độc lập trong mô hình.

3 Tiền xử lý số liệu

3.1 Đọc dữ liệu

Bước 1: Đặt đường dẫn đến tệp `Intel_CPUs.csv` vào biến `file_path`

Bước 2: Dùng hàm `read.csv` để đọc dữ liệu từ file vào biến `data`

Bước 3: Dùng lệnh `View()` để tổng quan bảng dữ liệu đã nhập

Bên cạnh đó, có thể dùng hàm `str()` để nắm được cấu trúc của dữ liệu nhập bao gồm loại dữ liệu, kích thước, các thành phần cụ thể của đối tượng.

```
> str(data)
'data.frame': 2283 obs. of 45 variables:
 $ Product_Collection : chr "7th Generation Intel® Core™ i7 Processors" "8th Generation Intel® Core™ i5 Processors" "8th Gen
eration Intel® Core™ i7 Processors" "Intel® Core™ X-series Processors" ...
 $ Vertical_Segment : chr "Mobile" "Mobile" "Desktop" ...
 $ Processor_Number : chr "i7-7Y75" "i5-8250U" "i7-8550U" "i7-3820" ...
 $ Status : chr "Launched" "Launched" "Launched" "End of Life" ...
 $ Launch_Date : chr "Q3'16" "Q3'17" "Q3'17" "Q1'12" ...
 $ Lithography : chr "14 nm" "14 nm" "14 nm" "32 nm" ...
 $ Recommended_Customer_Price : chr "$393.00 " "$297.00 " "$409.00 " "$305.00 " ...
 $ nb_of_Cores : int 2 4 4 4 2 2 2 2 1 ...
 $ nb_of_Threads : int 4 8 8 8 4 2 2 2 2 NA ...
 $ Processor_Base_Frequency : chr "1.30 GHz" "1.60 GHz" "1.80 GHz" "3.60 GHz" ...
 $ Max_Turbo_Frequency : chr "3.60 GHz" "3.40 GHz" "4.00 GHz" "3.80 GHz" ...
 $ Cache : chr "4 MB SmartCache" "6 MB SmartCache" "8 MB SmartCache" "10 MB SmartCache" ...
 $ Bus_Speed : chr "4 GT/s OPI" "4 GT/s OPI" "4 GT/s OPI" "5 GT/s DMI2" ...
 $ TDP : chr "4.5 W" "15 W" "15 W" "130 W" ...
 $ Embedded_Options_Available : chr "No" "No" "No" "No" ...
 $ Conflict_Free : chr "Yes" "Yes" "Yes" "" ...
 $ Max_Memory_Size : chr "16 GB" "32 GB" "32 GB" "64.23 GB" ...
 $ Memory_Types : chr "LPDDR3-1866, DDR3L-1600" "DDR4-2400, LPDDR3-2133" "DDR4-2400, LPDDR3-2133" "DDR3 1066/1333/1600" ...
 $ Max_nb_of_Memory_Channels : int 2 2 2 4 2 2 1 2 2 NA ...
 $ Max_Memory_Bandwidth : chr "29.8 GB/s" "34.1 GB/s" "34.1 GB/s" "51.2 GB/s" ...
```

Hình 4: Sử dụng hàm `str ()` để xem cấu trúc của dữ liệu nhập

3.2 Trích xuất dữ liệu

Bước 1: Đặt đường dẫn đến tệp `Intel_CPUs.csv` vào biến `file_path`

Bước 2: Dùng toán tử `<data_frame>[, c(<column_indices>)]` để trích xuất những biến đã chọn từ `data` sang frame `new_data`

Bước 3: Dùng lệnh `View()` để tổng quan bảng dữ liệu mới được trích xuất

Từ 45 thuộc tính có trong tập dữ liệu, nhóm tác giả chọn ra 14 thuộc tính mà nhóm tác giả quan tâm và lưu vào data frame `new_data` để xử lý: `Product_Collection`, `Launch_Date`, `Bus_Speed`, `Cache`, `Lithography`, `Max_Memory_Bandwidth`, `Max_nb_of_Memory_Channels`, `Max_Memory_Size`, `nb_of_Cores`, `Processor_Base_Frequency`, `Recommended_Customer_Price`, `TDP`, `DirectX_Support` và `PCI_Express_Revision`.

Product_Collection	Launch_Date	Bus_Speed	Cache	Lithography	Max_Memory_Bandwidth	Max_nb_of_Memory_Channels	Max_Memory_Size	nb_of_Cores
1 7th Generation Intel® Core™ i7 Processors	Q3'16	4 GT/s OPI	4 MB SmartCache	14 nm	29.8 GB/s	2	16 GB	
2 8th Generation Intel® Core™ i5 Processors	Q3'17	4 GT/s OPI	6 MB SmartCache	14 nm	34.1 GB/s	2	32 GB	
3 8th Generation Intel® Core™ i7 Processors	Q3'17	4 GT/s OPI	8 MB SmartCache	14 nm	34.1 GB/s	2	32 GB	
4 Intel® Core™ X-series Processors	Q1'12	5 GT/s DMI2	10 MB SmartCache	32 nm	51.2 GB/s	4	64.23 GB	
5 7th Generation Intel® Core™ i5 Processors	Q1'17	4 GT/s OPI	4 MB SmartCache	14 nm	29.8 GB/s	2	16 GB	
6 Intel® Celeron® Processor 3000 Series	Q1'15	5 GT/s DMI2	2 MB	14 nm	25.6 GB/s	2	16 GB	
7 Intel® Celeron® Processor N Series	Q3'13		1 MB	22 nm		1	4 GB	
8 Intel® Celeron® Processor J Series	Q3'13		1 MB L2	22 nm		2	8 GB	
9 Intel® Celeron® Processor G Series	Q1'13	5 GT/s DMI	2 MB SmartCache	22 nm	21 GB/s	2	32 GB	
10 Legacy Intel® Pentium® Processor		533 MHz FSB	1 MB L2	90 nm		N/A		
11 Intel® Pentium® Processor 2000 Series	Q3'12	5 GT/s DMI	2 MB SmartCache	22 nm	25.6 GB/s	2	32 GB	
12 Legacy Intel® Pentium® Processor		400 MHz FSB	2 MB L2	90 nm		N/A		
13 Intel® Pentium® Processor 3000 Series	Q1'15	5 GT/s DMI2	2 MB	14 nm	25.6 GB/s	2	16 GB	
14 Intel® Pentium® Processor 4000 Series	Q3'15	4 GT/s OPI	2 MB SmartCache	14 nm	34.1 GB/s	2	32 GB	
15 Intel® Pentium® Processor N Series	Q1'16		2 MB L2	14 nm		2	8 GB	
16 Intel® Quark™ SE C1000 Microcontroller Series	Q4'15		8 KB			N/A		
17 Intel® Pentium® Processor J Series	Q3'13		2 MB L2	22 nm		2	8 GB	
18 Intel® Pentium® Processor J Series	Q4'13		2 MB L2	22 nm	21.3 GB/s	2	8 GB	
19 Intel® Pentium® Processor J Series	Q1'16		2 MB L2	14 nm		2	8 GB	

Showing 1 to 19 of 2,283 entries, 14 total columns

Hình 5: Sử dụng lệnh `View()` để xem một số phần tử đầu tiên của frame dữ liệu mới được trích xuất

3.3 Làm sạch dữ liệu

3.3.1 Kiểm tra các dữ liệu bị khuyết

Nhóm tác giả coi các dữ liệu bị khuyết là các dữ liệu rỗng, hoặc chứa giá trị NA, hoặc chứa giá trị NULL hoặc chứa chuỗi "N/A".

Nhóm tác giả thống kê tổng số và tỉ lệ của các giá trị bị khuyết trong `new_data`. Nhóm tác giả thu được các thông số tương ứng với biến đã chọn như sau:

```
> # Count missing data on new data label
> missing_data <- new_data[is.na(new_data)]
      Product_Collection Launch_Date Bus_Speed Cache Lithography
      0                  412         294      12         71
      Max_Memory_Bandwidth Max_nb_of_Memory_Channels Max_Memory_Size nb_of_Cores Processor_Base_Frequency
      1136                  869                880          0              18
Recommended_Customer_Price TDP DirectX_Support PCI_Express_Revision
      982                  67         1888         1015
```

Hình 6: Tổng số giá trị bị khuyết của các biến được chọn

```
> missing_data_frequency <- missing_data %>%
+   summarise(frequency = sum(is.na(new_data))) %>%
+   print()
      Product_Collection Launch_Date Bus_Speed Cache Lithography
      0.0000000000      0.180464301      0.128777924      0.005256242      0.031099431
      Max_Memory_Bandwidth Max_nb_of_Memory_Channels Max_Memory_Size nb_of_Cores Processor_Base_Frequency
      0.497590889      0.380639509      0.385457731      0.000000000      0.007884363
Recommended_Customer_Price TDP DirectX_Support PCI_Express_Revision
      0.430135786      0.029347350      0.826982041      0.444590451
```

Hình 7: Tỉ lệ giá trị bị khuyết của các biến được chọn

Nhận xét: Nhóm tác giả nhận thấy có 12/14 biến bị khuyết dữ liệu. Nhóm tác giả tiến hành xử lý tất cả các biến theo các quy tắc nhất định và xử lý dữ liệu bị khuyết

của các biến nếu có.

3.3.2 Xử lý dữ liệu

3.3.2.1 *Product_Collection*

Nhóm tác giả loại bỏ tất cả các ký tự đặc biệt (trừ chữ cái, chữ số, dấu gạch chéo "/" và khoảng trắng) khỏi các chuỗi trong cột *Product_Collection*.

	<i>Product_Collection</i>		<i>Product_Collection</i>
1	7th Generation Intel® Core™ i7 Processors	1	7th Generation Intel Core i7 Processors

Hình 8: Một ví dụ về dữ liệu của *Product_Collection* biến sau khi được xử lý
(*Bên trái: Dữ liệu gốc; Bên phải: Dữ liệu đã được xử lý*)

3.3.2.2 *Launch_Date*

Đầu tiên, nhóm tác giả xử lý các giá trị trong cột *Launch_Date* bằng cách loại bỏ các ký tự không phải số và lưu vào biến *years*.

Vì tập dữ liệu dừng lại ở giá trị 2022 nên nhóm đã làm như sau:

- Nếu giá trị của *years* ≤ 22 thì *years+=2000*, và coi như nó thuộc những năm của thế kỷ XXI.
- Nếu không, *years+=1900*, và coi như nó thuộc những năm của thế kỷ XX.

Tiếp theo, nhóm tác giả chuyển ký tự đầu tiên trong *years* thành số và lưu vào biến *quart*.

Cuối cùng, giá trị mới sẽ được xác định lại theo quy tắc sau:

$$\text{Giá trị mới} = \text{years} + (\text{quart} - 1)/4$$

Trong đó, $(\text{quart} - 1)/4$ là biểu thức quy số quý về tháng đầu tiên của quý đó. Tương ứng như sau:

- Quý 1: Tháng 1
- Quý 2: Tháng 4
- Quý 3: Tháng 7
- Quý 4: Tháng 10

Ví dụ: "Q3'16" = $16 + 2000 + (3-1)/4 = 2016.5$, tương ứng là Tháng 7 năm 2016

3.3.2.3 *Lithography*

Nhóm tác giả chuyển kiểu dữ liệu `char` thành `numeric` với đơn vị là *nm*.

Lithography	Lithography
14 nm	14

Hình 9: Một ví dụ về dữ liệu của `Lithography` biến sau khi được xử lý
(Bên trái: Dữ liệu gốc; Bên phải: Dữ liệu đã được xử lý)

3.3.2.4 *Recommended_Customer_Price*

Bước 1: Dữ liệu lần lượt được sắp xếp theo thứ tự tăng dần dựa trên các cột `Launch_Date`, `Product_Collection`, và `Vertical_Segment`.

Bước 2: Loại bỏ ký hiệu \$ và ra khỏi các giá trị, chuyển các giá trị "N/A" trong cột thành NA (biến các giá trị bất hợp lệ này thành giá trị khuyết).

Bước 3: Với các giá trị đặc biệt mang định dạng khoảng giá trị (các giá trị chứa dấu gạch ngang "-"), tách giá trị đó thành hai số, tính trung bình của chúng và làm tròn kết quả. Nếu không thì giữ nguyên.

Bước 4: Chuyển đổi các giá trị trong cột thành kiểu dữ liệu số thực (`numeric`).

Recommended_Customer_Price	Recommended_Customer_Price
\$393.00	393.0
Recommended_Customer_Price	Recommended_Customer_Price
\$62.00 - \$72.00	67.0

Hình 10: Một ví dụ về dữ liệu của `Product_Collection` biến sau khi được xử lý
(Bên trái: Dữ liệu gốc; Bên phải: Dữ liệu đã được xử lý)

3.3.2.5 *nb_of_Cores*

Kiểu dữ liệu gốc của biến này là `int` nên nhóm tác giả không có thao tác xử lý gì với biến này.

3.3.2.6 *Processor_Base_Frequency*

Nhóm tác giả chuyển kiểu dữ liệu `char` thành `numeric` với đơn vị là *GHz*.

Processor_Base_Frequency	Processor_Base_Frequency
950 MHz	0.95

Hình 11: Một ví dụ về dữ liệu của **TDP** biến sau khi được xử lý
(Bên trái: Dữ liệu gốc; Bên phải: Dữ liệu đã được xử lý)

3.3.2.7 Cache

Nhóm tác giả chuyển kiểu dữ liệu **char** thành **numeric** với đơn vị là **KB**.

Cache	Cache
8 MB SmartCache	8000

Hình 12: Một ví dụ về dữ liệu của **Cache** biến sau khi được xử lý
(Bên trái: Dữ liệu gốc; Bên phải: Dữ liệu đã được xử lý)

3.3.2.8 TDP

Nhóm tác giả chuyển kiểu dữ liệu **char** thành **numeric** với đơn vị là **W**.

TDP	TDP
20.8 W	20.8

Hình 13: Một ví dụ về dữ liệu của **TDP** biến sau khi được xử lý
(Bên trái: Dữ liệu gốc; Bên phải: Dữ liệu đã được xử lý)

3.3.2.9 Bus_Speed

Bước 1: Tách cột **Bus_Speed** thành ba cột riêng biệt: **Bus_Speed** (giá trị tốc độ), **Speed_Unit** (đơn vị tốc độ), và **Bus_Type** (loại bus) để phục vụ các quá trình sau.

Bước 2: Chuyển đổi tất cả giá trị tốc độ về đơn vị là **MHz**.

3.3.2.10 Max_nb_of_Memory_Channels

Từ giá trị của cột **Bus_Type** đã tách ra ở trên, nhóm tác giả tiến hành xử lý **Max_nb_of_Memory_Channels**.

Bước 1: Nếu **Bus_Type** là "FSB", đặt giá trị **Max_nb_of_Memory_Channels** thành 1, vì chuẩn FSB chỉ có một kênh nhận nên mặc định là 1.

Bước 2: Trích xuất và chuyển kiểu dữ liệu **char** thành **numeric**.

3.3.2.11 *Max_Memory_Bandwidth*

Từ giá trị của cột **Bus_Type** đã tách ra ở trên, nhóm tác giả tiến hành xử lý **Max_Memory_Bandwidth**.

Bước 1: Nếu **Bus_Type** là "FSB", giá trị của **Max_Memory_Bandwidth** được tính lại theo công thức **Bus_Type** * 4 / 1000.

Bước 2: Trích xuất và chuyển kiểu dữ liệu **char** thành **numeric** với đơn vị là GB/s.

Bus_Speed	Max_Memory_Bandwidth	Max_nb_of_Memory_Channels
1333 MHz FSB		NA
Bus_Speed	Max_Memory_Bandwidth	Max_nb_of_Memory_Channels
1333	5.332	1

Hình 14: Một ví dụ về dữ liệu của ba biến **Bus_Speed**, **Max_nb_of_Memory_Channels** và **Max_Memory_Bandwidth** sau khi được xử lý
(Bên trên: Dữ liệu gốc; Bên dưới: Dữ liệu đã được xử lý)

3.3.2.12 *DirectX_Support*

Nhóm tác giả giới hạn phiên bản DirectX được hỗ trợ mang giá trị hợp lệ trong khoảng 1.0-12.2, vì hiện tại DirectX chỉ có những phiên bản như thế.

3.3.2.13 *PCI_Express_Revision*

Nhóm tác giả giới hạn phiên bản PCI Express được hỗ trợ mang giá trị hợp lệ trong khoảng 1.0-6.0, vì hiện tại PCI Express chỉ có những phiên bản như thế.

3.3.3 Xử lý các biến có dữ liệu khuyết

Dữ liệu bị khuyết là hiện tượng xảy ra khi một hoặc nhiều giá trị trong tập dữ liệu không được ghi nhận hoặc bị mất. Nguyên nhân của dữ liệu bị khuyết có thể bao gồm lỗi trong quá trình thu thập dữ liệu, như thiết bị đo lường gặp sự cố hoặc lỗi phần mềm, dẫn đến việc không ghi nhận được giá trị. Ngoài ra, dữ liệu có thể bị thiếu do các yếu tố bên ngoài như người cung cấp dữ liệu quên hoặc từ chối trả lời một câu hỏi

trong khảo sát. Việc xử lý dữ liệu bị khuyết cần thiết để đảm bảo tính chính xác và đáng tin cậy của các phân tích tiếp theo, bởi dữ liệu bị khuyết có thể gây ra sai lệch trong mô hình dự đoán hoặc làm giảm độ tin cậy của các kết luận thống kê.

Có một số phương pháp xử lý dữ liệu bị khuyết sau:

- Thay giá trị khuyết bằng giá trị trung bình các phần tử không khuyết của mẫu.
(1)
- Thay giá trị khuyết bằng giá trị mod của thuộc tính tương ứng trong mẫu. (2)
- Thay giá trị khuyết bằng giá trị trung vị của thuộc tính tương ứng trong mẫu.
(2)

Nhóm tác giả đã áp dụng phương pháp (2) lên các biến `Launch_Date`, `Bus_Speed`, `Lithography`, `Max_nb_of_Memory_Channels`, `Max_Memory_Bandwidth`, `Processor_Base_Frequency`, `TDP`, `DirectX_Support`, `PCI_Express_Revision` bằng hàm `CleanData_f_name_mod`.

Nhóm tác giả đã áp dụng phương pháp (1) lên các biến `Cache` và `Max_Memory_Size` bằng hàm `CleanData_f_name_avr`, biến `Recommended_Customer_Price` với cách hiện thực khác sẽ được nhóm tác giả đề cập trong mục 3.3.3.3.

Cả 2 phương pháp trên đều lấy cột `Product_Collection` làm cột mẫu, nghĩa là sẽ lấy giá trị xuất hiện nhiều nhất/ giá trị trung bình trong một nhóm có cùng tên của cột `Product_Collection`.

3.3.3.1 `CleanData_f_name_mod`

Hiện thực bằng R:

```
1 #-----
2 # Brief: Fill the missing values by the most repeated value
3 #       of Sample_Vector name
4 # Arguments: data - table data; column_names - working column;
5 #            sample_column_name - column to fill based on same value
6 # e.g: new_data <- CleanData_f_name_mod(new_data, Cache,
7 #    ↪ Product_Collection)
7 CleanData_f_name_mod <- function(data, column_name, sample_column_name) {
```

```
8 data %>%
9   # groups the data frame by the sample column.
10  group_by({{ sample_column_name }}) %>%
11  mutate({{ column_name }} := ifelse(
12    {{ column_name }} %in% c(NA, "N/A", ""),
13    {{ column_name }} %>%
14      get_num() %>%
15      mfv1(na.rm = TRUE),
16    {{ column_name }}
17  )) %>%
18  ungroup()
19 }
```

3.3.3.2 *CleanData_f_name_avr*

Hiện thực bằng R:

```
1 # Brief: Fill the missing values by average of Sample_Vector name
2 # Arguments: data - table data; column_names - working column;
3 #           sample_column_name - column to fill based on same value
4 # e.g: new_data <- CleanData_f_name_avr(new_data, Cache,
5     ↪ Product_Collection)
6 CleanData_f_name_avr <- function(data, column_name, sample_column_name) {
7   data %>%
8     # groups the data frame by the sample column.
9     group_by({{ sample_column_name }}) %>%
10    mutate({{ column_name }} := ifelse(
11      {{ column_name }} %in% c(NA, "N/A", ""),
12      # Replace NA values with the first non-NA value from
13      # the sample column.
14      {{ column_name }} %>%
15        get_num() %>%
16        mean(na.rm = TRUE),
```

```
16     # else just keep its value
17     {{ column_name }}
18 )) %>%
19 ungroup()
20 }
```

3.3.3.3 Trường hợp khác dùng để xử lý biến `Customer_Recommended_Price`

```
1  ### Recommended_Customer_Price ###
2  new_data <- new_data[order(new_data$Launch_Date,
   → new_data$Product_Collection, new_data$Vertical_Segment), ]
3
4  new_data$Recommended_Customer_Price <- gsub("\\$", "",
   → new_data$Recommended_Customer_Price)
5  new_data$Recommended_Customer_Price <- gsub(",", "",
   → new_data$Recommended_Customer_Price)
6  new_data$Recommended_Customer_Price <-
   → ifelse(new_data$Recommended_Customer_Price == "N/A", NA,
   → new_data$Recommended_Customer_Price)
7  new_data$Recommended_Customer_Price <-
   → sapply(new_data$Recommended_Customer_Price, function(price_range) {
8    if (grepl("-", price_range)) {
9      range <- strsplit(price_range, "-")[1]
10     return((as.double(range[1]) + as.double(range[2])) / 2)
11   }
12   return(price_range)
13 })
14 new_data$Recommended_Customer_Price <-
   → as.numeric(new_data$Recommended_Customer_Price)
15 new_data <- new_data %>%
16   group_by(Product_Collection) %>%
17   fill(Recommended_Customer_Price, .direction = "downup") %>%
18   ungroup(Product_Collection)
```

Khi sử dụng hàm `fill` để điền giá trị thiếu trong cột `Recommended_Customer_Price`, dữ liệu cần được sắp xếp theo thứ tự hợp lý để việc điền vào giá trị khuyết có tính logic và nhất quán.

Việc sắp xếp dữ liệu theo các cột `Launch_Date`, `Product_Collection`, và `Vertical_Segment` đảm bảo rằng các giá trị được điền dựa trên các sản phẩm tương tự, ra mắt gần nhau hoặc trong cùng một phân khúc thị trường.

3.3.3.4 Kết quả

Sau khi đã hoàn thành việc xử lý thô và xử lý dữ liệu khuyết của các biến đã chọn, nhóm tác giả thu được kết quả sau:

```
> # Print the results
> print(new_missing_data)
      Product_Collection      Vertical_Segment      Launch_Date      Bus_Speed      Cache
      0                  0                  0                  0                  0
      Lithography      Max_Memory_Bandwidth      Max_nb_of_Memory_channels      Max_Memory_Size      nb_of_Cores
      0                  0                  0                  0                  0
      Processor_Base_Frequency      Recommended_Customer_Price      TDP      DirectX_Support      PCI_Express_Revision
      0                  0                  0                  0                  0
```

Hình 15: Tổng số giá trị bị khuyết của các biến sau khi xử lý

Nhận xét: Như vậy, các biến đã được xử lý và không còn biến nào trong `new_data` có dữ liệu bị khuyết, ta chuyển sang bước tiếp theo

3.4 Xác định ngoại lai

Một trong những cách xác định các giá trị ngoại lai là **phương pháp IQR**.

IQR là sự chênh lệch giữa tứ phân vị thứ nhất **Q1** và tứ phân vị thứ ba **Q3**:

$$IQR = Q3 - Q1$$

Giá trị IQR có thể sử dụng để xác định outliers bằng cách thiết lập các giá trị biên Upper/Lower như sau: Nếu chúng ta trừ đi $k \times IQR$ từ tứ phân vị đầu tiên Q1, bất kỳ giá trị dữ liệu nào nhỏ hơn con số này được coi là giá trị ngoại lai. Tương tự như vậy, nếu chúng ta thêm $k \times IQR$ đến tứ phân vị thứ ba Q3, bất kỳ giá trị dữ liệu nào lớn hơn con số này được coi là ngoại lai. Giá trị k thường được chọn là 1.5.

Nhóm thống kê được số giá trị ngoại lai của mỗi biến như sau:

▶ resultBus_Speed	0 obs. of 14 variables
▶ resultCache	169 obs. of 14 variables
▶ resultLaunch_Date	0 obs. of 14 variables
▶ resultLithography	253 obs. of 14 variables
▶ resultMax_Memory_Bandwidth	125 obs. of 14 variables
▶ resultMax_Memory_Size	250 obs. of 14 variables
▶ resultMax_nb_of_Memory_Channels	224 obs. of 14 variables
▶ resultnb_of_Cores	117 obs. of 14 variables
▶ resultProcessor_Base_Frequency	8 obs. of 14 variables
▶ resultRecommended_Customer_Price	916 obs. of 14 variables
▶ resultTDP	0 obs. of 14 variables

Hình 16: Tổng số giá trị ngoại lệ các biến được chọn

4 Thống kê mô tả

Đầu tiên, chia các biến thành hai nhóm: Biến định tính và Biến định lượng

- Biến định tính: `Launch_Date`, `Product_Collection`, `DirectX_Support` và `PCI_Express`
- Biến định lượng: `Bus_Speed`, `Cache`, `Lithography`, `Max_Memory_Bandwidth`, `Max_nb_of_Memory_Channels`, `Max_Memory_Size`, `nb_of_Cores`, `Processor_Base_Frequency`, `Recommended_Customer_Price` và `TDP`

Sau đó, thực hiện thống kê mô tả.

4.1 Các giá trị đặc trưng của mẫu đối với biến định lượng

Sử dụng hàm `apply()` để xuất ra màn hình các giá trị: `mean`, `median`, `min`, `max`, `Q1`, `Q3`, `sd` để thống kê các đặc trưng cơ bản của các biến định lượng.

```
1 #####
2 #      Descriptive statistics
3 #####
4 # -----
5 ### Summary statistics ###
6 # !Add more specific
7 summary_stats <- new_data[, c(
8   "Bus_Speed",
9   "Cache",
10  "Lithography",
11  "Max_Memory_Bandwidth",
12  "Max_nb_of_Memory_Channels",
13  "Max_Memory_Size",
14  "nb_of_Cores",
15  "Processor_Base_Frequency",
16  "Recommended_Customer_Price",
17  "TDP"
18 )]
19 Mean <- apply(summary_stats, 2, mean) # Tính
    ↪ trung bình
```



```
20 SD <- apply (summary_stats ,2 , sd) #  
    ↪ Tính độ lệch chuẩn  
21 Median <- apply (summary_stats ,2 , median ) # Tính trung  
    ↪ vi  
22 Q1 <- apply (summary_stats ,2 , quantile , probs =0.25) # Tính  
    ↪ phần vi 25% (Q1)  
23 Q3 <- apply (summary_stats ,2 , quantile , probs =0.75) # Tính  
    ↪ phần vi 75% (Q3)  
24 Min <- apply (summary_stats ,2 , min ) # Tính giá  
    ↪ trị nhỏ nhất  
25 Max <- apply (summary_stats ,2 , max ) # Tính giá  
    ↪ trị lớn nhất  
26 #Tạo dataframe  
27 # Tạo một bản sao của tên hàng (rownames) trước khi áp dụng lapply  
28 stats_df <- data.frame(Mean,SD,Q1,Median,Q3,Min,Max)  
29 print(stats_df)  
30 rownames_stats_df <- rownames(stats_df)  
31  
32 # Áp dụng formatC để rút gọn số 0 không cần thiết  
33 stats_df <- data.frame(lapply(stats_df, function(x) formatC(x, format =  
    ↪ "f", digits = 4, drop0trailing = TRUE)))  
34  
35 # Gán lại tên hàng ban đầu cho data frame sau khi định dạng  
36 rownames(stats_df) <- rownames_stats_df  
37  
38 # Xem kết quả  
39 print(stats_df)
```

	Mean	SD	Q1	Median	Q3	Min	Max
Bus_Speed	3820.5274	2911.2554	800	5000	5000	0	9600
Cache	7068.1029	9245.7687	2000	3000	8000	8	60000
Max_Memory_Bandwidth	20.8606	18.9254	3.2	25.6	25.6	0.2	85.3
Max_nb_of_Memory_Channels	1.7902	0.9307	1	2	2	1	4
Max_Memory_Size	248.756	470.6355	20.5007	32.77	288	1	4100
nb_of_Cores	4.0666	6.3299	1	2	4	1	72
Processor_Base_Frequency	2.2174	0.8244	1.6	2.2	2.8	0.03	4.3
Recommended_Customer_Price	337.5097	365.2626	276.5	307.1083	307.1083	2.54	7408
TDP	59.7176	44.3392	26.8	47	84	0.025	300

Hình 17: Kết quả thống kê mô tả của từng biến ngẫu nhiên đã chọn lọc (đã làm tròn 4 chữ số)

4.2 Thống kê số lượng đối với các biến định tính

Đối với biến định tính là `Product_Collection`, ta dùng `table()` để thống kê số lượng cho từng dòng CPU:

```
> table(new_data$Product_Collection)
```

Intel Atom Processors	Intel Celeron Processor	Intel Core Processors	Intel Pentium Processor
46	261	684	385
Intel Xeon Processors	597		

Hình 18: Thống kê số lượng cho biến `Product_Collection`

Đối với biến định tính là `Launch_Date`, ta dùng `table()` để thống kê số lượng các ngày ra mắt các loại CPU:

```
> table(new_data$Launch_Date)
```

1999	1999.25	2000	2001.75	2002	2002.5	2002.75	2003.25	2003.5	2003.75	2004	2004.25	2004.5	2004.75	2005	2005.25
2	1	10	4	16	1	1	4	1	1	6	283	2	15	5	7
2005.5	2005.75	2006	2006.25	2006.5	2006.75	2007	2007.25	2007.5	2007.75	2008	2008.25	2008.5	2008.75	2009	2009.25
8	8	19	10	23	6	15	14	29	21	36	17	44	13	35	15
2009.5	2009.75	2010	2010.25	2010.5	2010.75	2011	2011.25	2011.5	2011.75	2012	2012.25	2012.5	2012.75	2013	2013.25
26	3	187	25	32	7	57	39	21	17	35	120	34	1	34	49
2013.5	2013.75	2014	2014.25	2014.5	2015	2015.25	2015.5	2015.75	2016	2016.25	2016.5	2016.75	2017	2017.25	2017.5
90	16	61	62	50	35	41	48	31	38	38	7	3	65	13	16

Hình 19: Thống kê số lượng cho biến `Launch_Date`

Đối với biến định tính là `DirectX_Support`, ta dùng `table()` để thống kê số lượng của các phiên bản DirectX được hỗ trợ:

```
> table(new_data$DirectX_Support)
```

0	11.1	11.2	12
1534	59	192	188

Hình 20: Thống kê số lượng cho biến `DirectX_Support`

Đối với biến định tính là `PCI_Express_Revision`, ta dùng `table()` để thống kê số lượng của các phiên bản PCI Express:

```
> table(new_data$PCI_Express_Revision)
```

0	1	2	3
3	43	1257	670

Hình 21: Thống kê số lượng cho biến `PCI_Express_Revision`

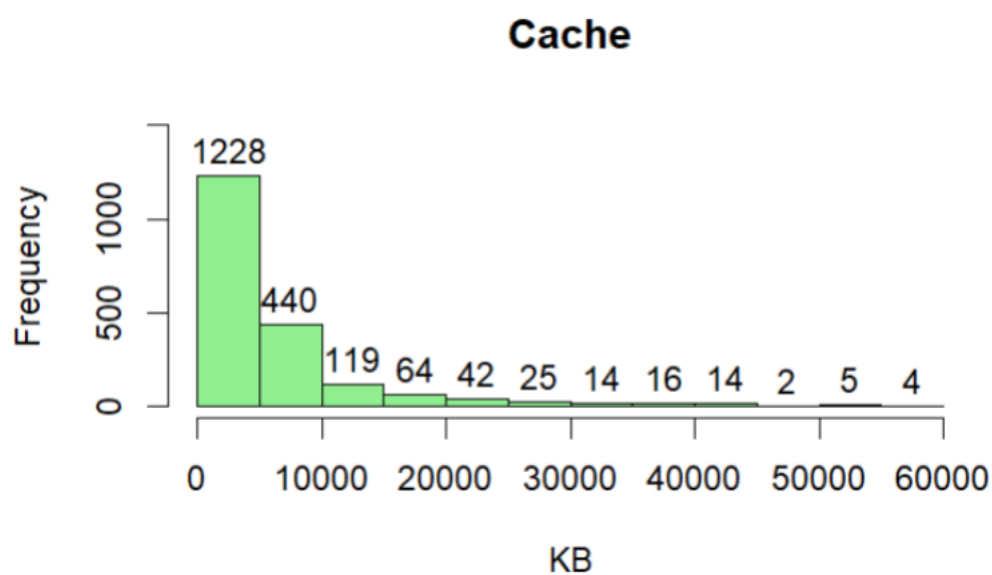
4.3 Đồ thị mô tả dữ liệu

4.3.1 Biểu đồ phân phối tần số của các biến định lượng

Vẽ đồng thời biểu đồ histogram và biểu đồ boxplot để xử lý các biến ngẫu nhiên.

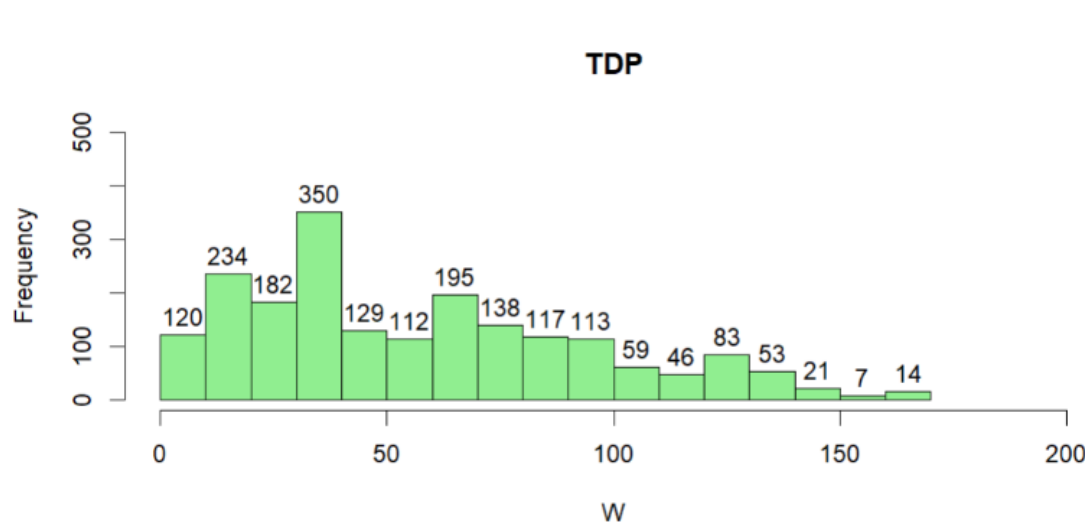
4.3.1.1 Biểu đồ Histogram

Ở đây, nhóm vẽ biểu đồ histogram cho 3 biến `Cache`, `TDP` và `Processor_Base_Frequency`.

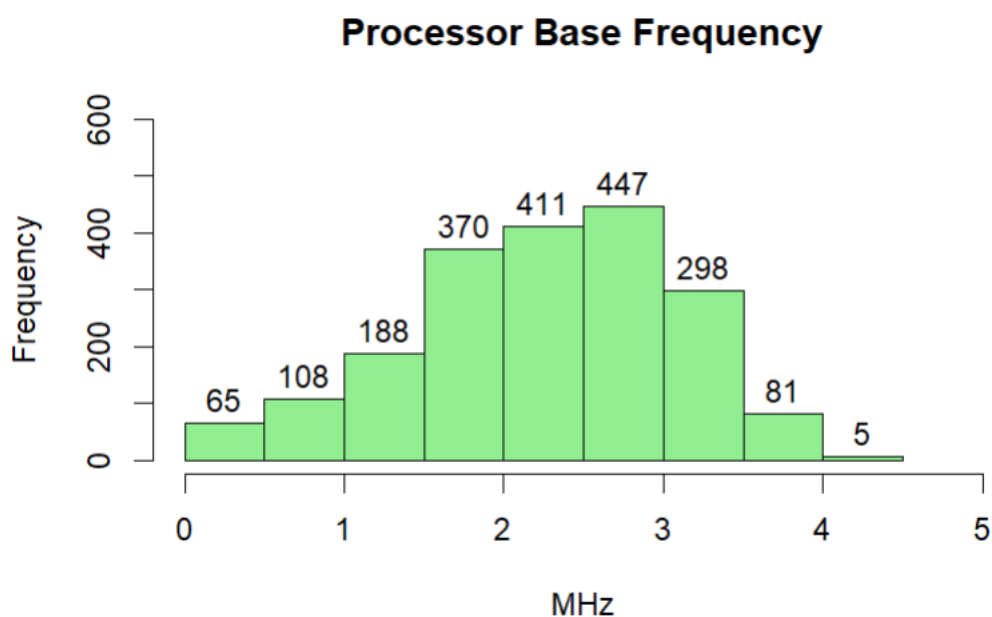


Hình 22: Biểu đồ phân phối tần số mẫu của `Cache`

Nhận xét: Đồ thị về Dung lượng `Cache` ở Hình 22 có hình dáng phân phối lệch phải. Đa phần bộ nhớ cache có dung lượng dưới 5000 MB. Dung lượng của bộ nhớ cache tập trung dao động từ 0 tới 10000 MB, chỉ có một số ít có dung lượng trên 10000 MB.



Hình 23: Biểu đồ phân phối tần số mẫu của TDP

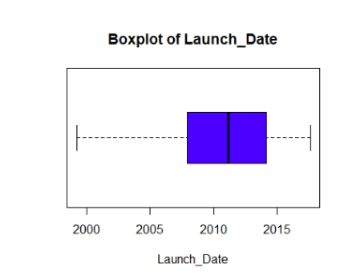


Hình 24: Biểu đồ phân phối tần số mẫu Processor_Base_Frequency

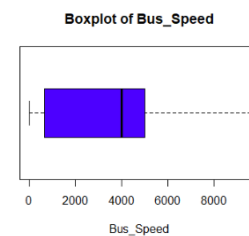
Đồ thị về Công suất thiết kế nhiệt **TDP** ở Hình 23 có hình dáng phân phối không quá lệch. CPU có công suất thiết kế nhiệt nhiều nhất là trong khoảng 30-40W. Công suất thiết kế nhiệt của các CPU dao động từ 0 tới 100W, chỉ có một số ít có công suất thiết kế nhiệt trên 100W.

Đồ thị về **Processor_Base_Frequency** (PBF) ở Hình 24 có hình dáng phân phối gần như đối xứng. Các CPU có PBF đạt tần số nhiều nhất là trong khoảng 1.5-3.5GHz. PBF của các CPU tập trung dao động từ 0.5 tới 3.5 GHz, chỉ có một số ít có PBF dưới 0.5 hay trên 3.5GHz.

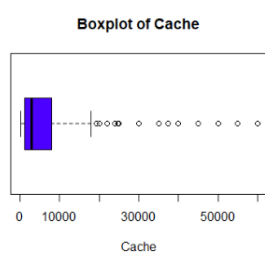
4.3.1.2 Biểu đồ Boxplot



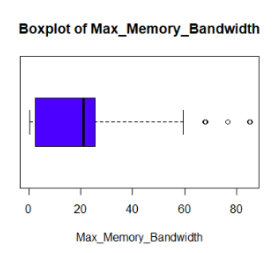
Hình 25: Biểu đồ Boxplot của
Launch_Date



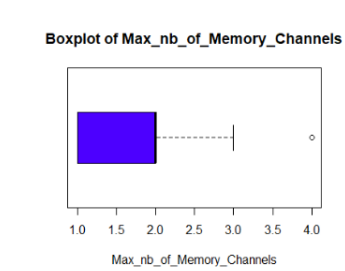
Hình 26: Biểu đồ Boxplot của
Bus_Speed



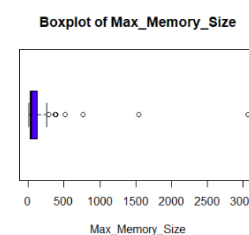
Hình 27: Biểu đồ Boxplot của
Cache



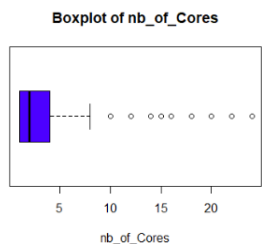
Hình 28: Biểu đồ Boxplot của
Max_Memory_Bandwidth



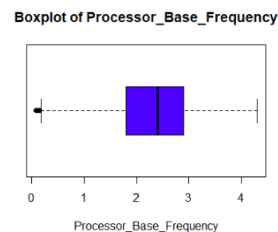
Hình 29: Biểu đồ Boxplot của
Max_nb_of_Memory_Channels



Hình 30: Biểu đồ Boxplot của
Max_Memory_Size



Hình 31: Biểu đồ Boxplot của
`nb_of_Cores`



Hình 32: Biểu đồ Boxplot của
`Processor_Base_Frequency`



Hình 33: Biểu đồ Boxplot của `Recommended_Customer_Price`

Từ các biểu đồ trên, nhóm tác giả rút được vài điều như sau:

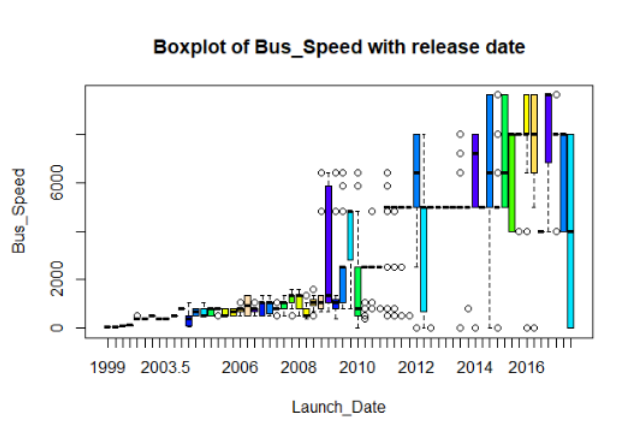
- Dựa vào Hình 25, ta thấy được Giai đoạn phát triển mạnh nhất của CPU nằm trong khoảng 2011 - 2013.
- Dựa vào Hình 26, ta thấy được Tốc độ truyền dẫn phần lớn nằm trong khoảng 4000MHz trở xuống.
- Dựa vào Hình 27, ta thấy được Để phù hợp với chức năng của cache, dung lượng bộ nhớ chỉ tập trung trong một khoảng nhỏ.
- Dựa vào Hình 29, ta thấy được Đa phần CPU có khoảng hai kênh truyền nhận đến các vùng bộ nhớ khác.
- Dựa vào Hình 33, ta thấy được Phần lớn CPU có mức giá cố định trong một khoảng nhỏ so với tổng thể.

Nhận xét chung: Dựa vào đồ thị boxplot trên ta thấy CPU phát triển không ổn định và chỉ mới phát triển mạnh sau năm 2010. Các đồ thị biểu diễn tốc độ xử lý,

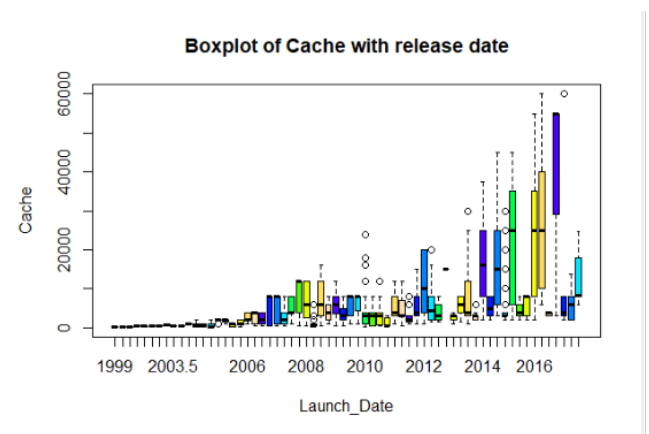
truyền nhận tín hiệu và khả năng lưu trữ đa phần lệch phải, tập trung trong một khoảng giá trị nhỏ nhất định. Đặc biệt đồ thị giá cả được đề xuất cho thấy phần lớn CPU có mức giá cố định trong một khoảng rất nhỏ so với tổng thể, phân bố tập trung ở một mức giá phù hợp.

4.3.1.3 Sự tương quan giữa một vài biến với biến `Launch_Date`

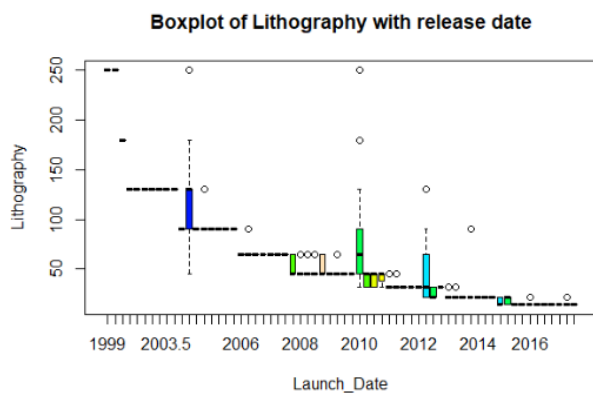
Nhóm tác giả xem xét sự ảnh hưởng của ngày ra mắt sản phẩm đến các thông số của CPU đó. Đầu tiên, ta sử dụng lệnh `by()` để thống kê. Sau đó, vẽ biểu đồ Boxplot thể hiện phân phối của biến `Bus_Speed`, biến `Cache`, biến `Lithography` và biến `Processor_Base_Frequency` theo phân loại biến `Launch_Date` theo Hình 17.



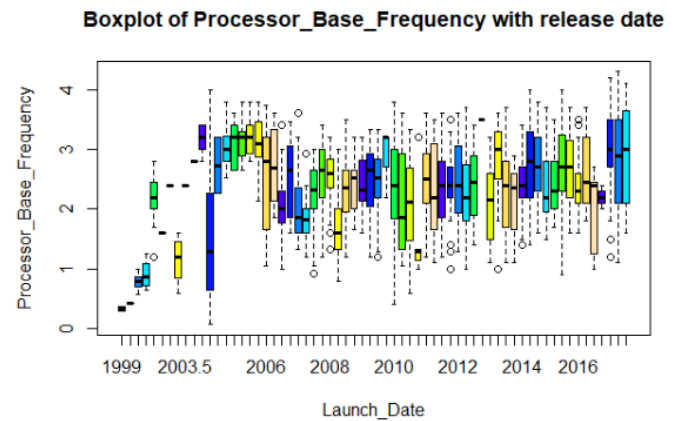
Hình 34: Biểu đồ Boxplot của
`Bus_Speed`



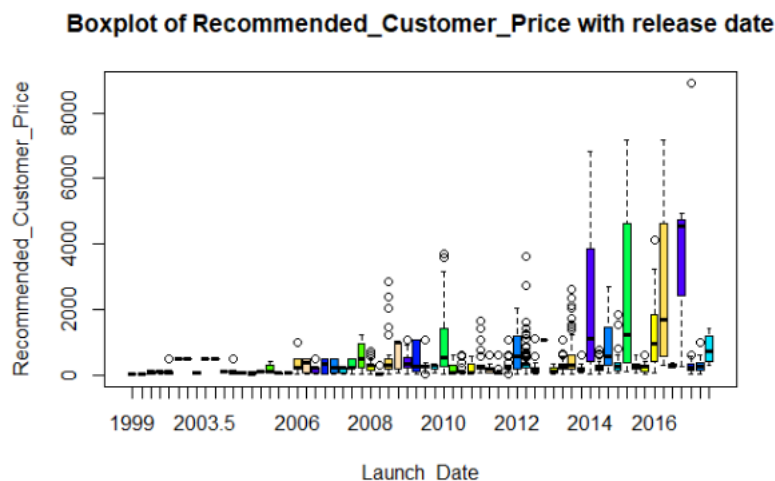
Hình 35: Biểu đồ Boxplot của
`Cache`



Hình 36: Biểu đồ Boxplot của
Lithography



Hình 37: Biểu đồ Boxplot của
Processor_Base_Frequency



Hình 38: Đồ thị Boxplot của Recommend_Customer_Price

Hiện thực bằng R:

```
1 for (i in colnames(summary_stats)) {
2   if (i != "Launch_Date") {
3     boxplot(new_data[[i]] ~ new_data$Launch_Date,
4             xlab = "Launch_Date",
5             ylab = i,
6             col = topo.colors(10),
7             main = paste("Boxplot of", i, "with release date"))
  }
```



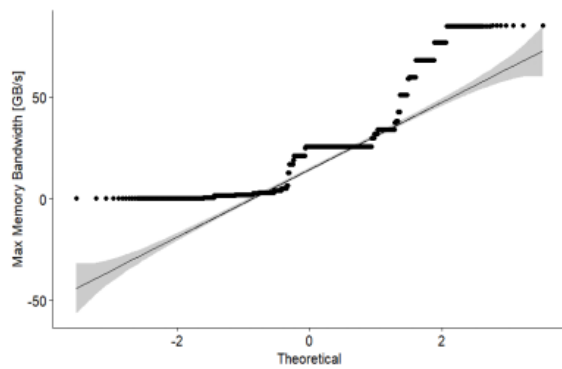
```
8      )  
9      }  
10     }
```

Nhận xét:

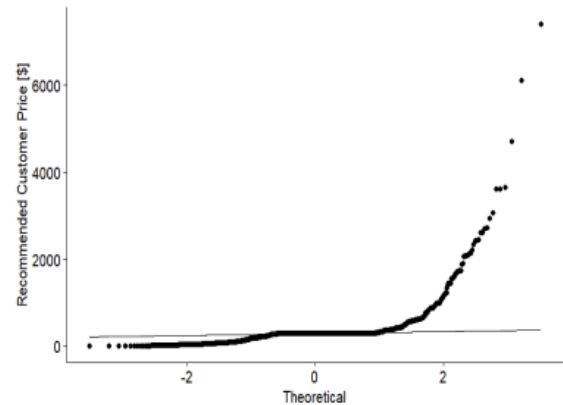
- Qua biểu đồ boxplot `Bus_Speed ~ Launch_Date` và `Cache ~ Launch_Date` trên cho thấy trước những năm 10 của thế kỷ XXI công nghệ chế tạo CPU còn hạn chế, khả năng xử lý của CPU còn thấp. Nhưng từ năm 2009 tốc độ truyền nhận thông tin và dung lượng Cache tăng vọt và được cải thiện đáng kể, cho thấy CPU phát triển mạnh trong giai đoạn này.
- Biểu đồ `Lithography ~ Launch_Date` cho thấy kỹ thuật in mạch CPU nhìn chung đang được cải thiện đáng kể theo thời gian khi độ dày bản in giảm mạnh qua những giai đoạn, đồng nghĩa công nghệ in được nâng cấp liên tục qua năm tháng.
- Bên cạnh đó, biểu đồ `Processor_Base_Frequency ~ Launch_Date` tăng nhanh từ rất sớm trong những năm đầu của thế kỷ 21 và bão hòa ổn định trong khoảng thời gian còn lại, nhìn chung có độ trải giữa khá lớn qua các năm. Mặt khác biểu đồ `Recommend_Customer_Price ~ Launch_Date` lại cho một góc nhìn ổn định về giá cả mặt bằng chung của CPU khi có ít biến động qua năm tháng trừ những CPU đặc thù (giá trị ngoại lai).

4.4 Phân phối chuẩn

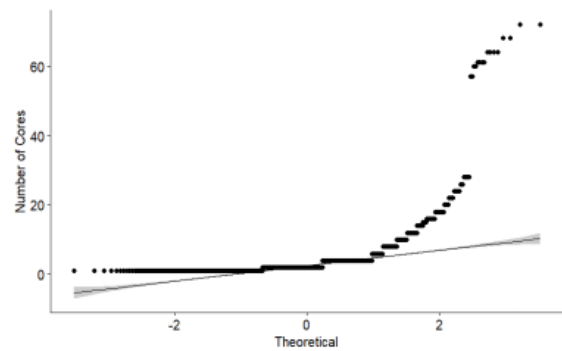
Phân phối chuẩn là giả thuyết rất quan trọng cho một số mô hình. Vì thế, cần kiểm tra xem các yếu tố đã trích xuất có (hoặc có gần với) phân phối chuẩn hay không. Để làm điều đó, ta dùng hàm `ggqqplot()` trong thư viện `ggpubr` để vẽ biểu đồ QQ-plot (biểu đồ kiểm tra mẫu có phân phối chuẩn hay không) cho các mẫu định lượng.



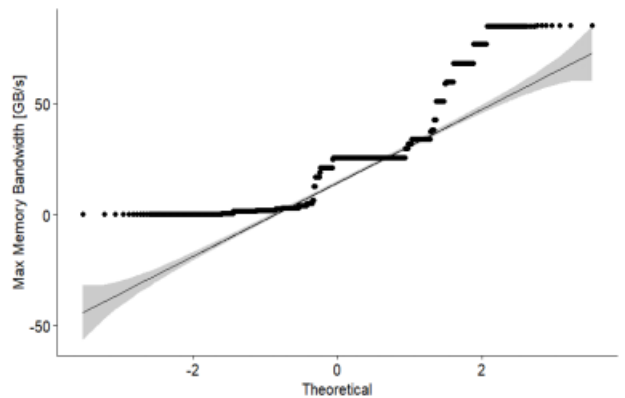
Hình 39: Biểu đồ phân phối chuẩn của `Max_Memory_Bandwidth`



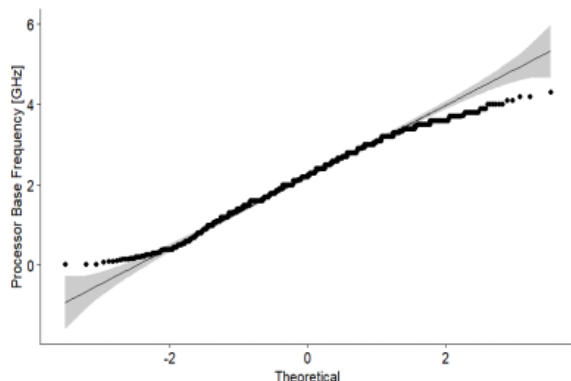
Hình 40: Biểu đồ phân phối chuẩn của `Recommended_Customer_Price`



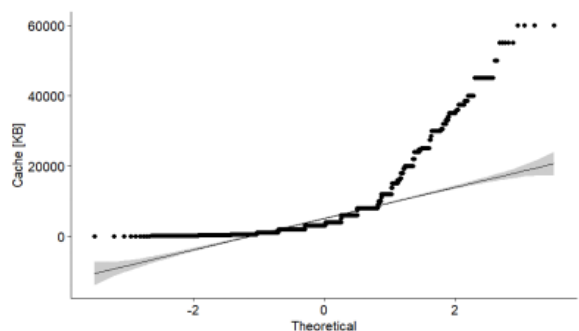
Hình 41: Biểu đồ phân phối chuẩn của `nb_of_Cores`



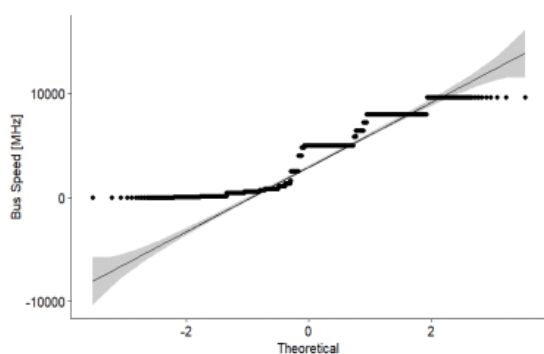
Hình 42: Biểu đồ phân phối chuẩn của `Max_Memory_Bandwidth`



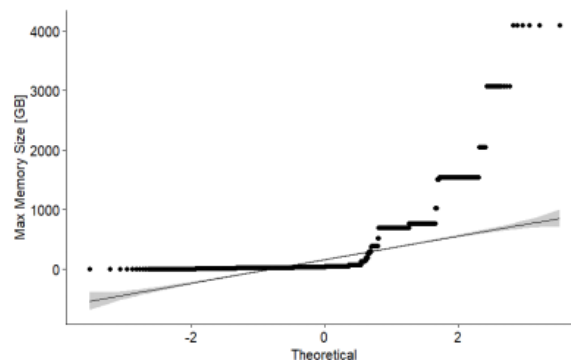
Hình 43: Biểu đồ phân phối chuẩn của `Processor_Base_Frequency`



Hình 44: Biểu đồ phân phối chuẩn của `Cache`



Hình 45: Biểu đồ phân phối chuẩn của `Bus_Speed`



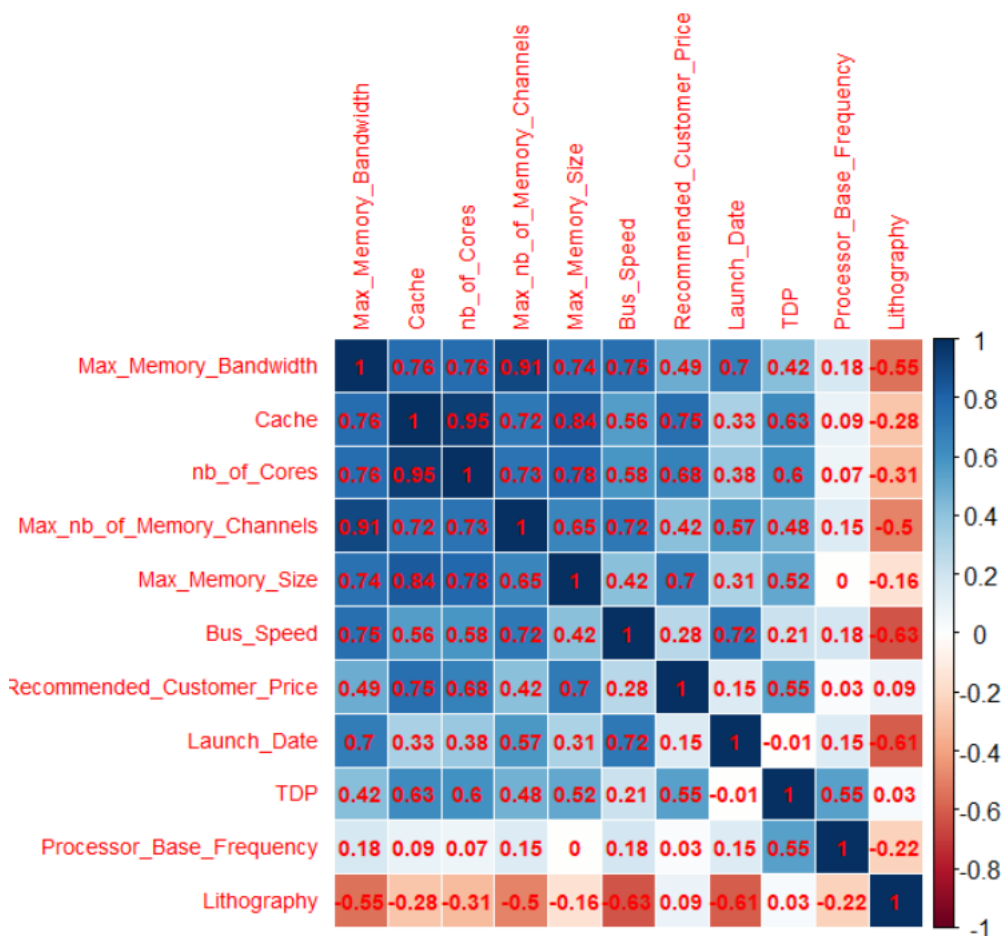
Hình 46: Biểu đồ phân phối chuẩn của `Max_Memory_Size`

Nhận xét: Từ các biểu đồ ở trên, ta có thể thấy được những giá trị quan sát của mẫu `Processor_Base_Frequency` phần lớn tập trung ở đường thẳng kì vọng của phân phối chuẩn, do đó có thể xem mẫu này có phân phối xấp xỉ chuẩn. Trong khi đó, các giá trị quan sát của các mẫu còn lại tập trung ít và lệch xa đường kì vọng, do đó các mẫu này có khả năng không tuân theo phân phối chuẩn.

Và cùng với biểu đồ histogram của mẫu `Processor_Base_Frequency` ở phần trên ta cũng có thể thấy biểu đồ có dạng hình chuông đối xứng với tần số cao nhất nằm ở giữa và các tần số thấp dần về hai phía. Vì vậy, ta có thể coi `Processor_Base_Frequency` tuân theo phân phối chuẩn.

4.5 Mối liên hệ giữa các biến

Để xác định mối liên hệ giữa các biến, nhóm sử dụng hàm `corrplot()` để vẽ biểu đồ tương quan và lập bảng hệ số tương quan giữa các biến nhằm trực quan hóa sự phụ thuộc tuyến tính giữa các biến, được thể hiện qua Hình 47.



Hình 47: Biểu đồ tương quan với hệ số tương quan giữa từng cặp biến

Qua các dữ liệu thu được từ Hình 47, nhóm tác giả đưa ra nhận xét như sau:

- **Lithography** : Biến hầu như không có tương quan với các biến khác và có tương quan nghịch với một số biến như **Launch_Date** , **Bus_Speed** với hệ số tương quan ở mức trung bình và biến **Max_Memory_Bandwidth** và **Max_nb_of_Memory_Channels** với hệ số tương quan yếu, còn lại là hệ số tương quan gần như bằng 0. Do đó kích thước node size (kích thước giữa các đặc trưng nhỏ nhất trên vi mạch) càng nhỏ thì tốc độ bus cũng như hiệu suất của CPU tăng đáng kể, còn lại là tăng không đáng kể hoặc không ảnh hưởng.
- **Launch_Date** : Biến có tương quan thuận với hầu hết các biến nhưng đa số với hệ số tương quan là yếu, chỉ có biến **Lithography** là có tương quan nghịch với hệ số mức trung bình. Do đó khi các CPU ra mắt thì chỉ ảnh hưởng đến kích thước node size, còn lại ảnh hưởng không đáng kể.
- **Bus_Speed** : Biến có tương quan thuận với hầu hết các biến với hệ số

tương quan ở mức trung bình đến yếu, đa số ở các biến như `Launch_Date`, `Max_Memory_Bandwidth`, `Max_nb_of_Memory_Channels` là có hệ số tương quan cao nhất; chỉ có biến `Lithography` là có tương quan nghịch với hệ số mức trung bình. Do đó một CPU có hiệu suất cao, băng thông tối đa hỗ trợ cao cũng như số cổng tối đa từ CPU sang thiết bị khác càng lớn và kích thước node size càng nhỏ thì tốc độ truyền dữ liệu tối đa càng lớn.

- `Processor_Base_Frequency`: Biến hầu như không có tương quan với các biến còn lại, hệ số tương quan hầu như gần bằng 0 chỉ trừ có biến `Lithography` là có tương quan nghịch với hệ số mức trung bình. Nên có thể kết luận rằng nếu tăng tần số cơ bản của CPU thì các thành phần còn lại của CPU cũng không thay đổi.
- `Max_Memory_Bandwidth` và `Max_nb_of_Memory_Channels`: cả hai biến có tương quan thuận với hệ số trung bình với hầu hết các biến, đặc biệt có hệ số tương quan thuận khá cao so với biến còn lại và đều có tương quan nghịch với biến `Lithography`. Do đó khi tăng băng thông bộ nhớ hoặc số lượng cổng tối đa trên CPU thì các chỉ số còn lại hầu như sẽ tăng và đặc biệt là chỉ số còn lại trong 2 biến.
- `Max_Memory_Size`: hai biến có tương quan thuận với hệ số trung bình với hầu hết các biến, đặc biệt có hệ số tương quan thuận khá cao so với biến `Cache` và đều có tương quan nghịch với biến `Lithography`. Do đó khi tăng dung lượng bộ nhớ Cache và giảm kích thước node size thì dung lượng bộ nhớ tối đa cũng tăng theo.
- `nb_of_Cores` và `Cache`: cả hai biến có tương quan thuận với hệ số trung bình với hầu hết các biến, đặc biệt có hệ số tương quan thuận khá cao so với biến còn lại trừ các biến `Lithography` và `Processor_Base_Frequency` với hệ số tương quan gần như bằng 0. Do đó khi tăng dung lượng bộ nhớ `Cache` và số cores thì sẽ tương tự như tăng băng thông bộ nhớ và số cổng trên CPU, thậm chí các chỉ số còn có phần tăng mạnh hơn.
- `Recommended_Customer_Price`, và `TDP`: cả 2 biến có tương quan thuận với hầu hết các biến với hệ số tương quan từ trung bình đến yếu. Do đó khi tăng thước

đo nhiệt tối đa của CPU cũng như giá khuyến nghị thì các chỉ số cũng sẽ tăng nhưng không đáng kể.

- **Max_Memory_Size** và **Max_Memory_Bandwidth**: Cả 2 biến ngoài không tương quan với biến **Processor_Base_Frequency** thì tương quan thuận với các biến còn lại với hệ số tương quan ở mức yếu đến trung bình. Vì vậy, nếu tăng giá trị của kích thước tối đa của bộ nhớ và băng thông của bộ nhớ thì các giá trị khác của CPU cũng sẽ tăng theo.

Quan sát các biểu đồ và từ các nhận xét trên, nhóm tác giả nhận thấy hệ số tương quan đa số đều là dương nhưng có **mức độ dao động thất thường** (từ 0.95 cho tới -0.63), chứng tỏ **khả năng xảy ra đa cộng tuyến khá cao**. Từ các nhận xét trên ta cũng thấy được đa số các biến tương quan thuận khá cao, đặc biệt là biến **Cache**, chỉ riêng biến **Lithography** là tương quan nghịch với các biến còn lại.

5 Thống kê suy diễn

5.1 Bài toán Kiểm định 1 mẫu

5.1.1 Mục tiêu

- Nắm được các khái niệm liên quan đến lý thuyết ước lượng, bài toán ước lượng khoảng tin cậy trung bình tổng thể của một mẫu.
- Biết cách giải một bài toán tìm khoảng tin cậy một mẫu.
- Tìm hiểu và hiện thực các lệnh của ngôn ngữ R để giải quyết bài toán.

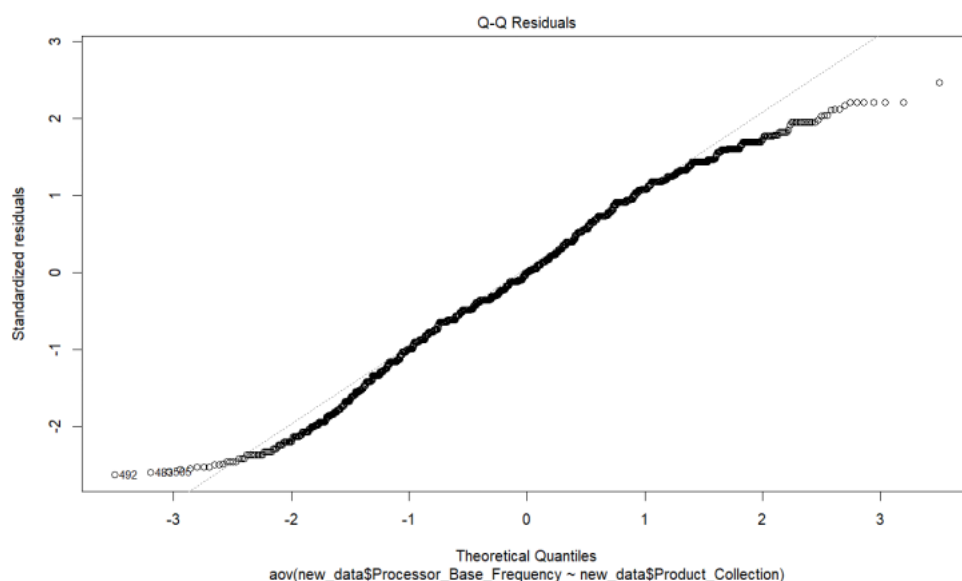
5.1.2 Bài toán

Xét mẫu `Processor_Base_Frequency`) trong dữ liệu đã trích xuất và tiền xử lý.

Tiến hành ước lượng khoảng tin cậy trung bình tổng thể của mẫu này với độ tin cậy 95%.

5.1.3 Tiến hành

Ta kiểm tra mẫu `Processor_Base_Frequency` và thấy rằng kích thước của mẫu lớn, tổng thể của mẫu này có dạng phân phối tương đối chuẩn, chưa biết phương sai tổng thể.



Hình 48: Kiểm tra mẫu `Processor_Base_Frequency`

Để chắc chắn hơn, ta tiến hành dùng kiểm định Shapiro để kiểm định mẫu `Processor_Base_Frequency` thêm một lần nữa.

Shapiro-wilk normality test

```
data: new_data$Processor_Base_Frequency
W = 0.98961, p-value = 2.706e-11
```

Hình 49: Kết quả khi kiểm định Shapiro của mẫu `Processor_Base_Frequency`

Nhóm tác giả nhận thấy, giá trị của `p-value` nhỏ hơn 0.05 rất nhiều, cho nên có thể nói rằng phân phối của `Processor_Base_Frequency` không tuân theo quy luật phân phối chuẩn. Vì thế, bài toán trở thành: **Mẫu dữ liệu có phân phối tùy ý, mẫu lớn.**

Phần giải tay của nhóm:

gày No.

Về lượng trung bình tổng thể Processor Base Freq

Mẫu dữ liệu Base Frequency có phân phối tùy ý, mẫu lớn nên ta dùng công thức sau:

Sử dụng phân mềm biết:

$\bar{X} = 2,2882$	$\varepsilon = \frac{Z_{\alpha/2} \cdot s}{\sqrt{n}}$
$s = 0,8362$	Lấy độ tin cậy 95% ta có
$n = 1973$	

$\varepsilon = \frac{1,96 \times 0,8362}{\sqrt{1973}} = 0,0369$

KVL: $(2,2882 - 0,0369 ; 2,2882 + 0,0369)$

$\Rightarrow (2,2513 ; 2,3251)$

Được quét bằng CamScanner

Hình 50: Lời giải cho bài toán Kiểm định 1 mẫu

Sau đó, nhóm sử dụng R để tìm khoảng ước lượng của trung bình tổng thể như sau:

```
> CIm_ProcFreq <- data.frame(lower_bound, upper_bound) %>% print()
  lower_bound upper_bound
1    2.251326    2.325167

> # Re-check
> t.test(
+   proc_freq,
+   conf.level = 0.95
+ ) %>% print()

      one sample t-test

data:  proc_freq
t = 121.55, df = 1972, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 2.251326 2.325167
sample estimates:
mean of x
 2.288246
```

Hình 51: Kết quả khi tìm khoảng tin cậy của mẫu `Processor_Base_Frequency` khi **không** sử dụng `t.test()` và khi sử dụng `t.test()`

5.1.4 Kết luận

Khoảng ước lượng trung bình tổng thể của tần số CPU nằm trong khoảng 2.2513GHz đến 2.3251GHz với độ tin cậy 95%.

$$(2.2513; 2.3251)$$

Nhận xét: Khoảng từ 2.2513 GHz đến 2.3251 GHz là tương đối hẹp, ta thấy được có sự biến thiên nhỏ trong tần số CPU và dữ liệu có thể đã được thu thập từ một mẫu tương đối đồng nhất.

5.2 Bài toán Kiểm định 2 mẫu

5.2.1 Mục tiêu

- Nắm được các khái niệm liên quan đến lý thuyết ước lượng, bài toán ước lượng khoảng tin cậy thông số của hai tổng thể.
- Biết cách giải một bài toán kiểm định tỷ lệ hai mẫu.
- Tìm hiểu và hiện thực các lệnh của ngôn ngữ R để giải quyết bài toán.

5.2.2 Bài toán

TDP là năng lượng tối đa mà một CPU có thể tiêu tốn để vận hành. Thông thường, chúng ta sẽ nghĩ rằng năng lượng tiêu thụ của CPU 1 nhân sẽ thấp hơn so với CPU 4 nhân.

Bài toán đặt ra: Liệu tỷ lệ CPU 1 nhân có TDP > 50 có thật sự thấp hơn tỷ lệ CPU 4 nhân có TDP > 50 hay không? Biết mức ý nghĩa là 5%.

5.2.3 Kiểm định giả thiết

- Giả thiết không H_0 : Tỷ lệ TDP của CPU 1 nhân và TDP của CPU 4 nhân có giá trị lớn hơn 50 là bằng nhau.
- Giả thiết đối H_1 : Tỷ lệ TDP của CPU 1 nhân có giá trị lớn hơn 50 thấp hơn tỷ lệ TDP của CPU 4 nhân có giá trị lớn hơn 50.

5.2.4 Tiến hành

Gọi p_1 là tỷ lệ CPU 1 nhân có giá trị TDP lớn hơn 50. Gọi p_2 là tỷ lệ CPU 4 nhân có giá trị TDP lớn hơn 50.

Ta tìm các thông số để có thể tính được hai giá trị trên bằng R như sau:

```
> print(core1_size)
[1] 546

> print(core4_size)
[1] 463

> print(core1_50_size)
[1] 204

> print(core4_50_size)
[1] 299
```

Hình 52: Các thông số cần có để tính p_1, p_2

Tiếp theo, nhóm tác giả tiến hành giải tay tính toán kết quả:

We make it work.

p_1 : Tỷ lệ cpu 1 core có TDP > 50
 p_2 : Tỷ lệ cpu 4 core có TDP > 50

$H_0: p_1 = p_2$ ($p_1 > p_2$) $\alpha = 5\%$
 $H_1: p_1 < p_2$ $RR = (-\infty, -z_\alpha)$
 $= (-\infty, -1,6449)$

$F_1 = \frac{m_1}{n_1} = \frac{204}{546}$ $F_2 = \frac{m_2}{n_2} = \frac{299}{463}$
 $\bar{f} = \frac{m_1 + m_2}{n_1 + n_2} = \frac{204 + 299}{546 + 463} = \frac{503}{1009}$

$Z_{qs} = \frac{F_1 - F_2}{\sqrt{\bar{f}(1-\bar{f})(\frac{1}{n_1} + \frac{1}{n_2})}} = \frac{\frac{204}{546} - \frac{299}{463}}{\sqrt{\frac{503}{1009}(1-\frac{503}{1009})(\frac{1}{546} + \frac{1}{463})}}$
 $\approx -8,6159$
 $Z_{qs} \in RR \Rightarrow p_1 < p_2$
 $p\text{-value} = 1 - \Phi(|Z_{qs}|) = 1 - \Phi(8,6159)$
 $= 3,4698 \cdot 10^{-18}$

Được quét bằng CamScanner

Hình 53: Lời giải cho bài toán Kiểm định Tỷ lệ 2 mẫu

Sau đó, nhóm sử dụng công cụ `prop.test()` trong R để giải bài toán đã đặt ra:

```
core1 = subset(new_data, nb_of_Cores == "1") $ TDP
core4 = subset(new_data, nb_of_Cores == "4") $ TDP
core1_50 = subset(core1, core1 > 50)
core4_50 = subset(core4, core4 > 50)
core1_size = length(core1)
core4_size = length(core4)
core1_50_size = length(core1_50)
core4_50_size = length(core4_50)
result <- prop.test(x = c(core1_50_size, core4_50_size),
                    n = c(core1_size, core4_size),
                    alternative = "less",
                    correct = FALSE)
print(result)
```

Hình 54: Code R cho bài toán Kiểm định Tỷ lệ 2 mẫu

Sau khi chạy code R, nhóm tác giả thu được kết quả sau:

```
2-sample test for equality of proportions without continuity correction

data:  c(core1_50_size, core4_50_size) out of c(core1_size, core4_size)
X-squared = 74.234, df = 1, p-value < 2.2e-16
alternative hypothesis: less
95 percent confidence interval:
 -1.0000000 -0.2221985
sample estimates:
  prop 1    prop 2 
0.3736264 0.6457883
```

Hình 55: Kết quả code R cho bài toán tỷ lệ 2 mẫu

5.2.5 Giải thích kết quả

Giải thích các thông số đã chạy R:

- **X-squared**: Bình phương của Z_{qs}
- **p-value**: Dùng để so sánh với mức ý nghĩa đã quy định. Ở đây nhóm lấy mức ý nghĩa $\alpha = 5\%$, ta thấy **p-value < α** \rightarrow Kết quả nhận được là “**alternative hypothesis**” (Giả thiết thay thế, chính là H_1 mà nhóm đã đặt là “**less**”).
- **95 percent confidence interval: -1.0000000 -0.2221985**: Đây là khoảng tin cậy 95% cho sự khác biệt giữa hai tỷ lệ ($p_1 - p_2$). Vì khoảng này nằm bên trái so với 0, cho nên ta có thể nói $p_1 < p_2$.
- **sample estimates: prop 1 = 0.3736264, prop 2 = 0.6457883**: **prop 1** và **prop 2** là các tỷ lệ mẫu được ước lượng từ dữ liệu.

5.2.6 Kết luận

Từ việc kết quả giải tay cũng như kết quả thu được sau khi chạy code R là như nhau, nhóm tác giả kết luận: **Tỷ lệ TDP của CPU 1 nhân có giá trị lớn hơn 50 thấp hơn tỷ lệ TDP của CPU 4 nhân có giá trị lớn hơn 50.**

Vì **p-value** cực kỳ nhỏ cho nên độ tin cậy của kết luận rất cao.

5.3 Phân tích phương sai

5.3.1 Mục tiêu

- Nắm được các khái niệm liên quan đến phân tích phương sai một yếu tố.
- Biết cách giải một bài toán phân tích phương sai một yếu tố.
- Tìm hiểu và hiện thực các lệnh của ngôn ngữ R để giải quyết bài toán.

5.3.2 Bài toán

Thông thường, chúng ta cảm thấy số nhân càng nhiều thì giá bán lẻ càng cao.

Dùng mô hình ANOVA kiểm định tỷ lệ của yếu tố `Recommend_Customer_Price` giữa các nhóm có `nb_of_Cores` khác nhau với mức ý nghĩa $\alpha = 5\%$, đồng thời so sánh giá bán lẻ của các dòng CPU.

5.3.3 Kiểm tra giả thuyết

Ta cần kiểm tra các giả định sau:

- Xác định các mẫu có độc lập với nhau hay không.
- Kiểm tra xem giá bán được đề nghị cho khách hàng theo số nhân có tuân theo phân phối chuẩn hay không. Ở đây nhóm dùng kiểm định Shapiro để kiểm tra.
- Kiểm tra sự tương đồng về phương sai mẫu giữa các nhóm quan sát. Để làm điều đó, nhóm dùng kiểm định Levene.

5.3.4 Tiến hành

5.3.4.1 Kiểm tra và xử lý mẫu phân loại

Ta chỉ lấy mẫu đối với những loại nhân có tần số trên 30 để đảm bảo về độ lớn.

Tiến hành lọc mẫu, ta có được 6 mẫu để xét như hình dưới:

```
  1   2   4   6   8  10  12  14  15  16  18  20  22  24
546 712 463 79  56  37  24  15  10   9  13   2   5   2

> new_data_filtered <- new_data[new_data$nb_of_Cores %in% names(name_table[name_table > 30]), ]
> print(name_table_filtered <- table(new_data_filtered$nb_of_Cores))

  1   2   4   6   8  10
546 712 463 79  56  37
```

Hình 56: Kết quả sau khi tiến hành lọc mẫu

5.3.4.2 Kiểm tra giả thiết để áp dụng mô hình ANOVA một yếu tố

Kiểm tra tính độc lập:

- Các mẫu có tính độc lập bởi vì số lượng các mẫu là khác nhau.
- Nhóm không lấy thống kê trong các mẫu theo một quy chuẩn nào cả.
- Tính ngẫu nhiên chưa được đảm bảo vì nhóm chỉ lấy dữ liệu từ file `cpu.csv`.

Kiểm tra phân phối chuẩn: Nhóm sử dụng kiểm định Shapiro, nhóm lấy mức ý nghĩa là 5% để so sánh với **p-value**. Nếu **p-value** > 5% nghĩa là mẫu có phân phối chuẩn và ngược lại thì không đảm bảo phân phối chuẩn.

Kết quả chạy các mẫu như sau:

```
data: cores_1$Recommended_Customer_Price
W = 0.26095, p-value < 2.2e-16
data: cores_2$Recommended_Customer_Price
W = 0.80269, p-value < 2.2e-16
data: cores_4$Recommended_Customer_Price
W = 0.78818, p-value < 2.2e-16
data: cores_6$Recommended_Customer_Price
W = 0.8395, p-value = 8.428e-08
data: cores_8$Recommended_Customer_Price
W = 0.72514, p-value = 6.465e-09
data: cores_10$Recommended_Customer_Price
W = 0.68102, p-value = 1.091e-07
```

Hình 57: Kết quả sau khi chạy với các mẫu đã lọc phía trên

Ta nhận thấy **p-value** của các mẫu trên đều mang giá trị lớn hơn 5%, từ đó xác định các mẫu đều không tuân theo phân phối chuẩn. Tuy nhiên vì tần số mẫu lớn (>30) nên vẫn có ý nghĩa tham khảo.

Tiếp theo, ta cần đánh giá về tính đồng nhất phương sai giữa các nhóm. Ta sử dụng kiểm định Levene bằng cách sử dụng hàm **leveneTest()** để kiểm tra.

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value    Pr(>F)
group  4 29.864 < 2.2e-16 ***
1888
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Hình 58: Kết quả sau khi chạy với các mẫu đã lọc phía trên

Dựa vào kết quả kiểm định với **p-value** mang giá trị rất nhỏ (<0.05), ta có thể bác bỏ giả thuyết các nhóm có phương sai đồng nhất. Tuy nhiên số lượng mẫu giữa các nhóm lớn nên có thể tiếp tục sử dụng Anova.

5.3.4.3 Tiến hành phân tích phương sai

Nhóm giải quyết bài toán đã đặt ra bằng ANOVA, giả thiết như sau

Giả thiết không H_0 : Giá bán lẻ không phụ thuộc vào số nhân của CPU.

Giả thiết đối H_1 : Giá bán lẻ phụ thuộc vào số nhân của CPU.

Nhóm tác giả sử dụng hàm `aov()` kết hợp với hàm `summary()` cùng những tham số phù hợp để tiến hành áp dụng mô hình phân tích phương sai một yếu tố cho mẫu `Recommended_Customer_Price`, phân loại theo `nb_of_Cores` đã lọc.

```
aov1 <- aov(Recommended_Customer_Price~as.factor(nb_of_Cores), new_data_filtered)
summary(aov1)
```

Hình 59: Code R cho việc áp dụng mô hình phân tích phương sai một yếu tố

```
> summary(aov1)
              Df    Sum Sq Mean Sq F value Pr(>F)
as.factor(nb_of_Cores)    5 30996853 6199371   150.8 <2e-16 ***
Residuals              1887 77583136   41115
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Hình 60: Kết quả phân tích phương sai một yếu tố

Dựa vào Hình 61, ta có thể thống kê các giá trị đặc trưng của mô hình ANOVA.

Các giá trị cụ thể được thể hiện ở Bảng 2.

Đại lượng	Bậc tự do		SSB	SSW	MSB	MSW	F value	p -value
	$df_B = k - 1$	$df_W = N - k$						
Giá trị	5	1887	30996853	77583136	6199371	41115	150.8	$< 2 \times 10^{-16}$

Bảng 2: Các giá trị đặc trưng của mô hình ANOVA

Từ Bảng 2, ta có một số nhận xét như sau:

- Giá trị thống kê kiểm định F (hay F value) biểu thị tỷ lệ bình phương trung bình giữa các nhóm với bình phương trung bình trong mỗi nhóm. Giá trị F rất cao, cho thấy trung bình của nhóm không bằng nhau.
- Bên cạnh đó, giá trị **p-value** thấp hơn rất nhiều so với mức ý nghĩa $\alpha = 5\%$, cho thấy rõ ràng sự khác biệt về trung bình giữa các nhóm được quan sát là có ý nghĩa đối với thống kê.

Kết luận: Ta thấy **p-value** nhỏ hơn 5% rất nhiều, nên ta có thể bác bỏ H_0 , chấp nhận H_1 . Hay nói rõ hơn, giá bán lẻ phụ thuộc vào số nhân của CPU.

5.3.4.4 Phân tích hậu định

Sau khi thực hiện phân tích ANOVA, ta có nhu cầu kiểm tra xem giữa các nhóm có sự khác nhau như thế nào, cũng như kiểm tra xem có ý nghĩa thống kê giữa các nhóm hay không. Ta có thể thực hiện bằng cách sử dụng hàm `TukeyHSD()`.

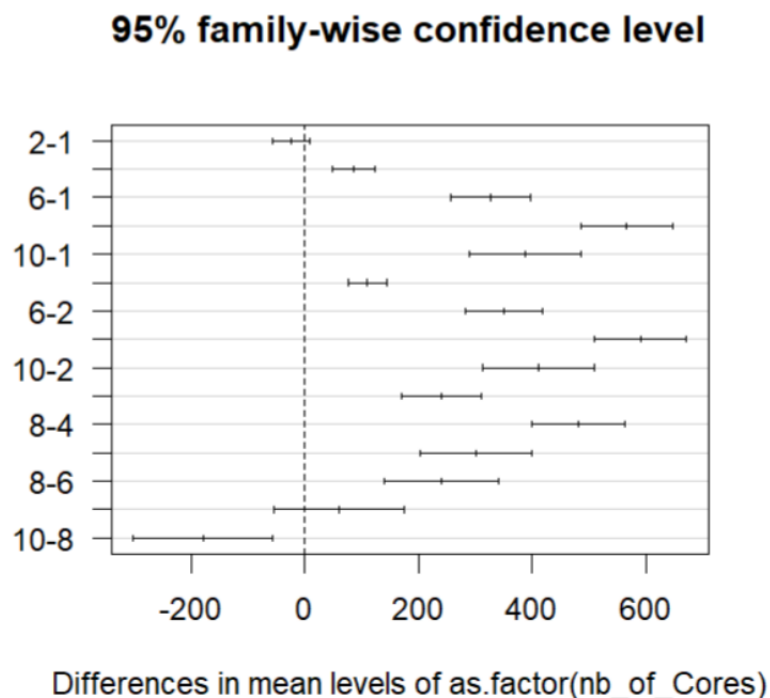
```
> TukeyHSD(aov1)
Tukey multiple comparisons of means
 95% family-wise confidence level

Fit: aov(formula = Recommended_Customer_Price ~ as.factor(nb_of_Cores), data = new_data_filtered)

$`as.factor(nb_of_Cores)`
      diff      lwr      upr    p adj
2-1  -23.97741  -56.88144   8.926606 0.2988543
4-1   85.60570   49.06274  122.148658 0.0000000
6-1  326.31115  256.68456  395.937733 0.0000000
8-1  566.25514  485.09311  647.417165 0.0000000
10-1 386.58965  288.32845  484.850849 0.0000000
4-2  109.58311   75.05018  144.116040 0.0000000
6-2  350.28856  281.69558  418.881540 0.0000000
8-2  590.23255  509.95547  670.509634 0.0000000
10-2 410.56706  313.03554  508.098589 0.0000000
6-4  240.70545  170.29444  311.116458 0.0000000
8-4  480.64944  398.81349  562.485394 0.0000000
10-4 300.98395  202.16537  399.802533 0.0000000
8-6  239.94399  138.90144  340.986543 0.0000000
10-6  60.27850  -54.94990  175.506910 0.6693128
10-8 -179.66549 -302.20935 -57.121627 0.0004323
```

Hình 61: Code R và Kết quả kiểm tra khi sử dụng hàm `TukeyHSD()`

Để dễ dàng kiểm tra hơn, ta sử dụng biểu đồ:



Hình 62: Biểu đồ kiểm tra sự khác nhau giữa các nhóm

Nhận xét: Ta có thể thấy xu hướng số nhân càng tăng thì giá bán lẻ sẽ tăng. Tuy

nhiên vẫn tồn tại một số ngoại lệ, cho nên không thể hoàn toàn khẳng định rằng số nhân càng cao thì giá thành càng cao, bởi giá tiền còn phụ thuộc vào rất nhiều yếu tố khác nữa.

5.3.5 Kết luận

- Dựa vào kết quả phân tích Anova, ta có thể thấy có sự khác biệt lớn về giá tiền giữa các nhóm CPU có số nhân khác nhau.
- Tuy nhiên, do số lượng mẫu nhiều nhưng không có phân phối chuẩn và không đồng nhất về phương sai nên kết quả phân tích Anova có thể không chính xác và đáng tin cậy. Vì vậy, mô hình Anova không hoạt động tốt trong trường hợp này và kết quả phân tích trên chỉ có giá trị tham khảo.

5.4 Phân tích hồi quy tuyến tính

5.4.1 Mục tiêu

- Tìm hiểu được các khái niệm liên quan đến mô hình hồi quy tuyến tính.
- Biết cách xử lý một bài toán phân tích hồi quy tuyến tính đơn, hồi quy tuyến tính bội.
- Tìm hiểu và hiện thực các câu lệnh của ngôn ngữ R để giải quyết các bài toán. Đồng thời biết cách đọc, giải thích và vẽ được các biểu đồ minh họa cho kết quả thu được

5.4.2 Bài toán

Từ các yếu tố đã trích xuất, hãy xây dựng mô hình hồi quy tuyến tính đánh giá các yếu tố ảnh hưởng đến `Bus_Speed` của CPU với mức ý nghĩa (significant level) là 5%.

5.4.3 Tiến hành

5.4.3.1 Xử lý dữ liệu

Ta cần trích xuất ra các mẫu định lượng trong tập dữ liệu, đồng thời lọc ra những quan trắc có yếu tố `Bus_Speed` trong tập dữ liệu, đồng thời bỏ đi các quan trắc bị khuyết. Bước này đã được thực hiện ở Mục 3.

5.4.3.2 Xây dựng mô hình

Để tiến hành xây dựng mô hình, ta dùng hàm `lm()` để tiến hành áp dụng mô hình hồi quy tuyến tính cho bộ dữ liệu đã xử lý, với biến `Bus_Speed` là biến phụ thuộc và các biến còn lại là các biến độc lập. Hình 60 thể hiện kết quả sau khi xây dựng mô hình hồi quy tuyến tính.

```
1 reg<-lm(formula(new_data$Bus_Speed ~ new_data$TDP + new_data$Cache +
  ↳ new_data$Launch_Date + new_data$Lithography + new_data$nb_of_Cores +
  ↳ new_data$Max_Memory_Size + new_data$Max_Memory_Bandwidth +
  ↳ new_data$Processor_Base_Frequency +
  ↳ new_data$Max_nb_of_Memory_Channels +
  ↳ new_data$Recommended_Customer_Price))
2 summary(reg)
```

Call:

```
lm(formula = formula(new_data$Bus_Speed ~ new_data$TDP + new_data$Cache +
  new_data$Launch_Date + new_data$Lithography + new_data$nb_of_Cores +
  new_data$Max_Memory_Size + new_data$Max_Memory_Bandwidth +
  new_data$Processor_Base_Frequency + new_data$Max_nb_of_Memory_Channels +
  new_data$Recommended_Customer_Price))
```

Residuals:

Min	1Q	Median	3Q	Max
-9824.5	-526.0	108.3	614.1	3963.8

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-4.500e+05	3.235e+04	-13.910	< 2e-16 ***
new_data\$TDP	-1.530e+01	1.770e+00	-8.641	< 2e-16 ***
new_data\$Cache	1.268e-01	1.504e-02	8.428	< 2e-16 ***
new_data\$Launch_Date	2.241e+02	1.610e+01	13.925	< 2e-16 ***
new_data\$Lithography	-6.093e-01	1.236e+00	-0.493	0.6221
new_data\$nb_of_Cores	3.809e+01	3.399e+01	1.121	0.2626
new_data\$Max_Memory_Size	-1.689e+00	1.720e-01	-9.821	< 2e-16 ***
new_data\$Max_Memory_Bandwidth	3.289e+01	5.807e+00	5.664	1.70e-08 ***
new_data\$Processor_Base_Frequency	4.739e+02	6.985e+01	6.784	1.54e-11 ***
new_data\$Max_nb_of_Memory_Channels	7.857e+02	9.624e+01	8.163	5.77e-16 ***
new_data\$Recommended_Customer_Price	-1.395e-01	5.646e-02	-2.470	0.0136 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1374 on 1962 degrees of freedom

Multiple R-squared: 0.7768, Adjusted R-squared: 0.7757

F-statistic: 682.9 on 10 and 1962 DF, p-value: < 2.2e-16

Hình 63: Kết quả khi dùng hàm `lm()` xây dựng mô hình hồi quy tuyến tính

Với giả thiết H_0 là biến độc lập không có ý nghĩa với mô hình, ta tiến hành xem xét giá trị $\Pr(>|t|)$ và so sánh với mức ý nghĩa 5% để kết luận về giả thiết H_0 cho từng biến. Dựa vào Hình 56, ta nhận thấy giá trị $\Pr(>|t|)$ của 2 biến **Lithography** và **nb_of_Cores** lần lượt là 0.6221 (62.21%) và 0.2626 (26.26%) lớn hơn rất nhiều so với mức ý nghĩa 5% nên ta sẽ loại đi 2 biến này, đồng thời giữ lại các biến còn lại.

Kế tiếp, ta gọi lại hàm `lm()` để xây dựng mô hình hồi quy tuyến tính với các biến có nghĩa. Hình 61 thể hiện kết quả sau khi xây dựng mô hình với các biến có nghĩa.

```
Call:
lm(formula = formula(new_data$Bus_Speed ~ new_data$TDP + new_data$Cache +
  new_data$Launch_Date + new_data$Max_Memory_Size + new_data$Max_Memory_Bandwidth +
  new_data$Processor_Base_Frequency + new_data$Max_nb_of_Memory_Channels +
  new_data$Recommended_Customer_Price))

Residuals:
    Min       1Q   Median       3Q      Max
-9568.6  -524.1   107.9   612.7  3998.7

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    -4.628e+05  2.755e+04 -16.798  < 2e-16 ***
new_data$TDP    -1.510e+01  1.685e+00  -8.963  < 2e-16 ***
new_data$Cache    1.411e-01  9.301e-03  15.167  < 2e-16 ***
new_data$Launch_Date 2.305e+02  1.374e+01  16.777  < 2e-16 ***
new_data$Max_Memory_Size -1.720e+00  1.700e-01 -10.118  < 2e-16 ***
new_data$Max_Memory_Bandwidth 3.319e+01  5.774e+00  5.748 1.04e-08 ***
new_data$Processor_Base_Frequency 4.688e+02  6.396e+01  7.330 3.35e-13 ***
new_data$Max_nb_of_Memory_Channels 7.915e+02  9.565e+01  8.274 2.35e-16 ***
new_data$Recommended_Customer_Price -1.546e-01  5.437e-02  -2.843  0.00451 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1374 on 1964 degrees of freedom
Multiple R-squared:  0.7766,    Adjusted R-squared:  0.7757
F-statistic: 853.6 on 8 and 1964 DF,  p-value: < 2.2e-16
```

Hình 64: Kết quả khi dùng hàm `lm()` xây dựng mô hình hồi quy tuyến tính với các biến có nghĩa

Lúc này, các hệ số trong mô hình đều có ý nghĩa. Do đó, mô hình hồi quy tuyến tính cho biến **Bus_Speed** phụ thuộc vào các biến độc lập có ý nghĩa đã được xây dựng thành công. Đồng thời, kết quả trên cũng mang đến một số thông tin như sau:

- Dựa vào cột **Estimate**, ta có các hệ số $\beta_0, \beta_1, \beta_2, \dots$ của mô hình. Từ đó ta có thể xác định được phương trình hồi quy tuyến tính mẫu.
- Hệ số xác định hiệu chỉnh $R^2 = 0.7757 = 77.57\%$. Nó thể hiện mức độ phù hợp của mô hình với dữ liệu. Đây là một tỷ lệ tương đối cao, cho thấy các biến độc lập đưa vào phân tích hồi quy giải thích được 77.57% sự biến thiên của biến phụ thuộc,

phần còn lại (22.43%) có thể được giải thích bởi phần dư gồm các biến độc lập ngoài mô hình và sai số ngẫu nhiên.

- Phần dư nằm trong khoảng từ 9568.6 đến 3998.7 (với Quartile 1 (1Q): 524.1, Trung vị (Median): 107.9 và Quartile 3 (3Q): 612.7), cho thấy có sự phân tán đáng kể trong các sai số dự đoán, bao gồm một số sai số dương và âm đặc biệt lớn.
- Các hệ số ước lượng (hoặc tham số) của mô hình thể hiện hướng và độ lớn của mối quan hệ giữa các biến độc lập và biến phụ thuộc. Trong trường hợp này, tất cả các biến dự đoán đều có mối quan hệ dương với biến phụ thuộc ứng. Đồng thời, các biến phụ thuộc cho ta thấy giá trị kiểm định $\Pr(>|t|)$ thấp hơn 0.05, cho thấy chúng là các yếu tố dự báo có ý nghĩa thống kê.
- Giá trị F và p (**F-statistic** / **p-value**): sử dụng để kiểm tra tính đáng tin cậy của mô hình hồi quy tuyến tính. **F-statistic** đo lường sự khác biệt giữa mô hình tuyến tính và mô hình không có biến độc lập. Giá trị **F-statistic** càng cao và **p-value** càng nhỏ, càng có mối quan hệ tuyến tính giữa biến phụ thuộc và biến độc lập. Mô hình có **F-statistic** cao (853.6) và **p-value** rất nhỏ ($< 2.2e - 16$) điều này cho thấy mô hình có mối quan hệ tuyến tính giữa biến phụ thuộc và các biến độc lập.
- Sai số chuẩn dư (RSE) của mô hình là khoảng 1374. Giá trị này là thước đo chắc chắn về độ lệch điển hình của giá trị thực tế so với giá trị dự đoán.

5.4.4 Ước lượng khoảng tin cậy các hệ số

Sau khi xây dựng mô hình hồi quy, ta dùng hàm `confint()` để tìm khoảng ước lượng cho các hệ số $\beta_0, \beta_1, \beta_2, \dots$ của mô hình hồi quy tuyến tính trên cho biến độc lập Cache với độ tin cậy 95%. Hình 65 thể hiện kết quả sau khi thực hiện đoạn code bên dưới để tìm khoảng ước lượng cho các hệ số

```
1 confint(reg, level = 0.95) #alpha = 0.05
```

```
> confint(reg,level = 0.95)#alpha=.05
              2.5 %      97.5 %
(Intercept) -5.168761e+05 -4.088054e+05
new_data$TDP -1.840300e+01 -1.179518e+01
new_data$Cache 1.228183e-01 1.592984e-01
new_data$Launch_Date 2.035501e+02 2.574383e+02
new_data$Max_Memory_Size -2.053448e+00 -1.386664e+00
new_data$Max_Memory_Bandwidth 2.186422e+01 4.451045e+01
new_data$Processor_Base_Frequency 3.433701e+02 5.942313e+02
new_data$Max_nb_of_Memory_Channels 6.038719e+02 9.790482e+02
new_data$Recommended_Customer_Price -2.612448e-01 -4.796623e-02
> |
```

Hình 65: Kết quả khi dùng hàm `confint()` để tìm khoảng ước lượng

Từ kết quả thu được ở Hình 68, ta xác định được các khoảng ước lượng cho hệ số chặn β_0 cũng như các hệ số của các biến độc lập trong mô hình với độ tin cậy 95%. Hình 66 thể hiện khoảng ước lượng của hệ số chặn β_0 và các hệ số của các biến độc lập.

```
(-5.168761x10^05, -4.088054x10^05) B0
(-1.840300x10^01, -1.179518x10^01) TBD
(1.228183x10^-01, 1.592984x10^-01) Cache
(2.035501x10^02, 2.574383x10^02) v.v
(-2.053448x10^00, -1.386664x10^00)
(2.186422x10^01, 4.451045x10^01)
(3.433701x10^02, 5.942313x10^02)
(6.038719x10^02, 9.790482x10^02)
(-2.612448x10^-01, -4.796623x10^-02)
```

Hình 66: Khoảng ước lượng β_0, β_1
giữa các biến.

5.4.5 Kiểm tra giả thiết

5.4.5.1 Kiểm định qua mô hình

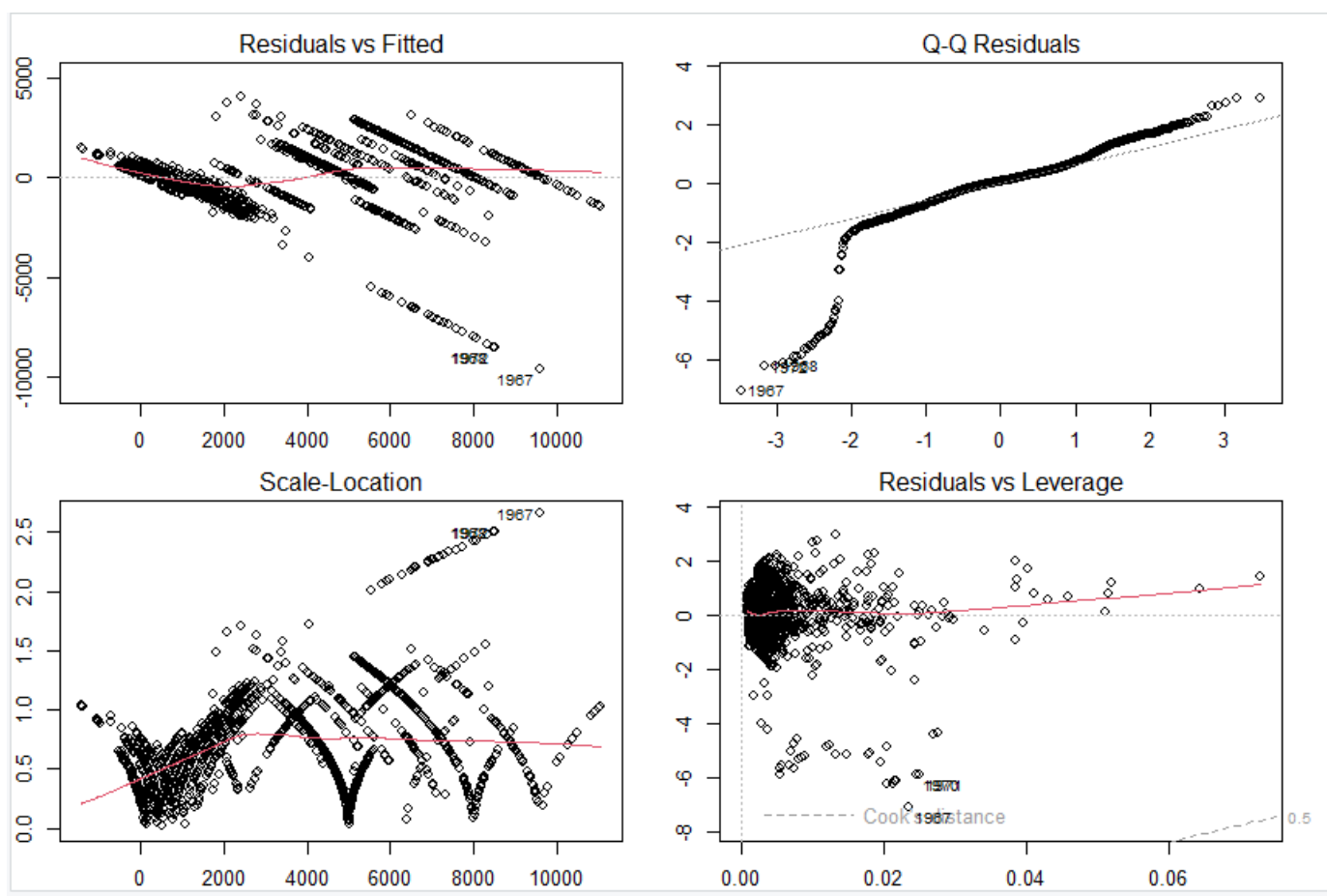
Từ kết quả có được từ mô hình hồi quy đã xây dựng, ta cần phải kiểm định lại các giả thiết của mô hình, cụ thể như sau:

- (1) Tính tuyến tính có dữ liệu.
- (2) Phần dư có phân phối chuẩn.
- (3) Phần dư có trung bình bằng 0.
- (4) Phần dư có phương sai không đổi.

(5) Giữa các biến độc lập không có mối quan hệ đa cộng tuyến hoàn hảo.

Đầu tiên, ta sử dụng hàm `plot()` để vẽ biểu đồ phần dư và kiểm tra các giả thiết về phần dư dựa trên mô hình tuyến tính đã xây dựng. Kết quả thu được thể hiện qua Hình 67.

```
1 par(mfrow = c(2, 2)) #Set the position of the graph
2 plot(reg)             #Use plot() function
```



Hình 67: Kết quả khi thực thi hàm `plot()`

Phân tích các biểu đồ:

- **Biểu đồ Residuals vs Fitted:** Là biểu đồ dùng để kiểm tra giả thiết tuyến tính của dữ liệu và giả thiết phần dư có trung bình bằng 0. Trục tung biểu thị giá trị của phần dư, trục hoành biểu thị giá trị tiên lượng \hat{y}_i của biến phụ thuộc. Nếu đường màu đỏ trên biểu đồ càng có dạng một đường thẳng nằm ngang, điều đó càng chứng tỏ tính tuyến tính của dữ liệu càng cao. Mặc khác, giả thiết phần dư

có trung bình bằng 0 thỏa mãn nếu đường màu đỏ gần với đường nét đứt nằm ngang (ứng với phần dư bằng 0) trên biểu đồ.

- **Biểu đồ Q-Q Residuals:** Dùng để kiểm tra giả thiết phần dư có phân phối chuẩn. Nếu các điểm thặng dư có xu hướng phân bố trên một đường thẳng thì điều kiện về phân phối chuẩn của phần dư được thỏa.
- **Biểu đồ Scale - Location:** Dùng để kiểm định giả thiết phương sai của phần dư là không đổi. Trục tung là căn bậc hai của phần dư (đã được chuẩn hóa), trục hoành là giá trị tiên lượng \hat{y}_i của các biến phụ thuộc. Nếu 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 thiết về phương sai của phần dư được thỏa.
- **Biểu đồ Residuals vs Leverage:** Mặc dù không dùng để kiểm định giả thiết cho mô hình, tuy nhiên biểu đồ này giúp xác định những điểm ngoại lai. Những điểm ngoại lai sẽ cách xa đường màu đỏ trên biểu đồ.

Nhận xét:

- **Biểu đồ Residuals vs Fitted:** cho thấy giả thiết (1) về tính tuyến tính của dữ liệu hơi vi phạm. Tuy nhiên, đối với giả thiết (3) về trung bình của phần dư có thể coi là thỏa mãn.
- **Biểu đồ Q-Q Residuals:** cho thấy giả thiết (2) về phần dư có phân phối chuẩn có thể coi là thỏa mãn.
- **Biểu đồ Scale - Location:** cho thấy đường màu đỏ có độ dốc, không thẳng. Đồng thời các điểm thặng dư phân tán không đều xung quanh đường thẳng này. Do đó, giả thiết (4) về tính đồng nhất của phương sai đối với mô hình này bị vi phạm.
- **Biểu đồ Residuals vs Leverage:** cho thấy không có điểm quan trắc nào gây ảnh hưởng tới mô hình hồi quy.

Đối với giả thiết cuối cùng, giả thiết (5), ta kiểm định thông qua việc đánh giá **Hệ số lạm phát phương sai - VIF**.

- Nếu $VIF_j = 1$, hệ số cho thấy không có mối tương quan giữa X_j với các biến độc lập trong mô hình. Tức là giả thiết (5) thỏa mãn.

- Nếu $1 < VIF_j < 5$, hệ số biểu thị mối tương quan vừa phải giữa X_j với các biến độc lập còn lại trong mô hình nhưng vẫn có thể chấp nhận được. Ta vẫn có thể cho rằng giả thiết (5) thoả mãn.
- Nếu $VIF_j > 5$, hệ số cho thấy mối tương quan mật thiết giữa X_j với các biến độc lập còn lại trong mô hình. Trong trường hợp này, giả thiết (5) không thoả mãn đối với mô hình.

Tính trực tiếp các giá trị VIF_j thông qua hàm `vif()` cho mô hình đã xây dựng. Kết quả và biểu đồ sau khi thực thi đoạn code bên dưới được thể hiện qua Hình 68 và Hình 69.

```

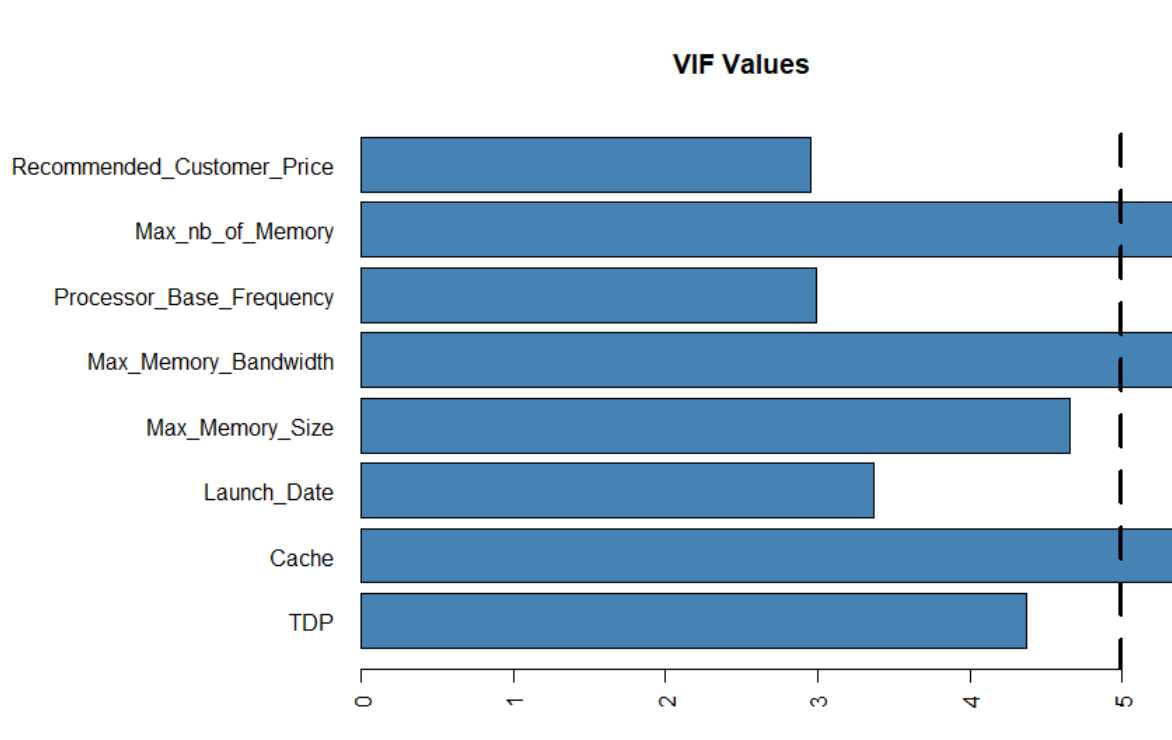
1  #Calculate VIF values
2  vif(reg)
3  #Create vector of VIF values
4  vif_values <- vif(reg)
5  vp0 <- viewport(x = .15, y = 0, just = c("left", "bottom"), width = 0.85,
6                , height = 1)
7  pushViewport(vp0)
8  #Add barplot
9  par(new = TRUE, fig = gridFIG())
10 par(mar = c(5, 14, 4, 2))
11 barplot(vif_values, main = "VIF Values", horiz = TRUE, col = "steelblue",
12         xlim = c(0,5), las = 2, names.arg = c("TDP","Cache",
13         ↪ "Launch_Date" , "Max_Memory_Size", "Max_Memory_Bandwidth",
14         ↪ "Processor_Base_Frequency", "Max_nb_of_Memory",
15         ↪ "Recommended_Customer_Price" ))
13 abline(v = 5, lwd = 5, lty = 2)

```



```
> vif(reg)
               new_data$TDP               new_data$Cache
               4.372014               6.754059
    new_data$Launch_Date    new_data$Max_Memory_Size
               3.366806               4.654758
    new_data$Max_Memory_Bandwidth  new_data$Processor_Base_Frequency
               14.610410               2.986520
    new_data$Max_nb_of_Memory_Channels  new_data$Recommended_Customer_Price
               8.765673               2.954315
> vif values <- vif(reg)
```

Hình 68: Kết quả khi thực thi hàm `vif()`



Hình 69: Biểu đồ thể hiện các giá trị VIF_j của các biến độc lập

Từ hình, ta nhận thấy các hệ số lạm phát phương sai của các biến độc lập có 3 giá trị lớn hơn 5 của 3 biến `Max_nb_of_Memory`, `Max_Memory_Bandwidth` và `Cache`, tức là vẫn có sự đa cộng tuyến với nhau giữa 3 biến này. Do đó, giả thiết (5) chưa thỏa mãn. Vậy cần loại bỏ một hoặc hai biến trên để loại bỏ đa cộng tuyến khi xây dựng mô hình.

5.4.5.2 Kiểm định sự phù hợp của hệ số HQT

Dựa vào kết quả phần trên, ta có thể dự đoán dữ liệu chứa các giá trị ngoại lai (outlier) có ảnh hưởng lớn đến kết quả và độ chính xác khi xây dựng mô hình hồi quy (phần dư sai số chuẩn rất lớn và hệ số xác định bội (R^2) còn thấp). Đồng thời, các thông số đa cộng tuyến ảnh hưởng đến việc hiểu biến độc lập nào góp phần vào phương sai được giải thích trong biến độc lập, cũng như sai lệch trong kết quả của mô hình hồi quy tuyến tính bội.

Từ kết quả các phần trên, ta loại bỏ các thông số `Vertical_Segment`, `Launch_Date`, `Lithography`, `Cache` và `Max_nb_of_Memory_Channels`. Đồng thời, xóa các giá trị ngoại lai của các thông số còn lại để cải thiện độ chính xác của mô hình.

```
1  ### Regression test ###
2  df <- summary_stats %>%
3    outlier_rm(Launch_Date) %>%
4    outlier_rm(TDP) %>%
5    outlier_rm(Bus_Speed) %>%
6    outlier_rm(Max_Memory_Size) %>%
7    outlier_rm(Processor_Base_Frequency) %>%
8    outlier_rm(Recommended_Customer_Price) %>%
9    outlier_rm(Max_Memory_Bandwidth) %>%
10   outlier_rm(nb_of_Cores) %>%
11   subset(select = -c(
12     Cache,
13     Max_nb_of_Memory_Channels,
14     Lithography
15   )
16 )
```

Dùng phương pháp kiểm định phương sai (ANOVA) một yếu tố để kiểm định F với mức ý nghĩa `p-value < 0.05`

Bằng cách so sánh bình phương trung bình của mô hình (MSM) với bình phương trung bình của sai số (MSE), ANOVA giúp đánh giá tầm quan trọng tổng thể của mô

hình hồi quy. Sự so sánh này được thực hiện bằng cách sử dụng thống kê F, là tỷ lệ giữa MSM và MSE. Giá trị F cao hơn cho thấy mô hình giải thích được phần lớn sự biến thiên của biến phản hồi

Giả thuyết H_0 : Tất cả các hệ số hồi quy bằng 0 (mô hình không phù hợp).

Giả thuyết H_1 : Ít nhất một hệ số hồi quy khác 0 (mô hình phù hợp).

```
> anova(model) %>% print()
Analysis of Variance Table

Response: Bus_Speed
Df      Sum Sq   Mean Sq  F value    Pr(>F)
Launch_Date      1 5820560108 5820560108 6385.0822 < 2.2e-16 ***
Max_Memory_Bandwidth 1 942906126 942906126 1034.3563 < 2.2e-16 ***
Max_Memory_Size    1 19393996 19393996 21.2750 4.393e-06 ***
nb_of_Cores        1 5924271 5924271 6.4989 0.010914 *
Processor_Base_Frequency 1 7545060 7545060 8.2768 0.004084 **
Recommended_Customer_Price 1 40553309 40553309 44.4865 3.855e-11 ***
TDP               1 10035837 10035837 11.0092 0.000933 ***
Residuals       1235 1125810362 911587
---
signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Hình 70: Kết quả khi thực thi hàm `anova()`

Vậy R^2 khác 0 một cách có ý nghĩa thống kê, mô hình hồi quy tuyến tính phù hợp với tập dữ liệu.

Kiểm định t bằng bảng Coefficients với mức ý nghĩa $t\text{-value} < 0.05$

Giả thuyết H_0 : Hệ số hồi quy của biến độc lập bằng 0 (biến độc lập không có ý nghĩa).

Giả thuyết H_1 : Hệ số hồi quy của biến độc lập khác 0 (biến độc lập có ý nghĩa).

```
> model <- lm(Bus_Speed ~ ., data = df_train)
> summary(model) %>% print()

Call:
lm(formula = Bus_Speed ~ ., data = df_train)

Residuals:
    Min       1Q   Median       3Q      Max
-10503.6  -309.8    20.0    345.0   4129.7

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.649e+05  2.942e+04  -9.002  < 2e-16 ***
Launch_Date   1.320e+02  1.468e+01   8.995  < 2e-16 ***
Max_Memory_Bandwidth 1.331e+02  4.561e+00  29.177  < 2e-16 ***
Max_Memory_Size    5.200e+00  8.273e-01   6.285 4.53e-10 ***
nb_of_Cores        2.207e+02  4.121e+01   5.355 1.02e-07 ***
Processor_Base_Frequency 2.602e+02  5.756e+01   4.520 6.78e-06 ***
Recommended_Customer_Price -1.873e+00  2.637e-01  -7.101 2.08e-12 ***
TDP              -7.007e+00  1.800e+00  -3.893 0.000104 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 922.5 on 1235 degrees of freedom
Multiple R-squared:  0.8664,    Adjusted R-squared:  0.8657
F-statistic: 1144 on 7 and 1235 DF,  p-value: < 2.2e-16
```

Hình 71: Kết quả bảng Coefficients

Vậy, mô hình hồi quy có các hệ số hồi quy của biến X_i khác 0 một cách có ý nghĩa thống kê và có tác động lên biến phụ thuộc **Bus_Speed**.

Bên cạnh đó, hệ số xác định điều chỉnh R^2 tương đối cao (khoảng 86.57%), cho thấy mô hình tương thích tốt với tập dữ liệu, cho thấy hơn 86% biến phụ thuộc được giải thích bằng các biến độc lập.

5.4.6 Dự đoán

Từ mô hình hồi quy tuyến tính đã xây dựng, ta có thể thực hiện việc dự đoán giá trị của biến **Bus_Speed** thông qua các giá trị của các biến độc lập trong mô hình.

Đầu tiên, nhóm tiến hành chia tập dữ liệu thành hai phần:

- Tập dữ liệu huấn luyện **df_train** : chiếm 80% tập dữ liệu, được sử dụng để xây dựng mô hình.
- Tập dữ liệu kiểm tra **df_test** : chiếm 20% tập dữ liệu, được dùng để so sánh tương quan với giá trị thu được sau khi dự đoán.

Sau khi xây dựng mô hình hồi quy tuyến tính bởi bằng tập dữ liệu **df_train** , nhóm sử dụng làm **predict** để dự đoán kết quả và thu được kết quả dưới đây:

```
> y_pred <- predict(model, newdata = x_test, interval = "confidence")
> mse <- mean((y_test - y_pred)^2)
> mae <- mean(abs(y_test - y_pred))
> rmse <- sqrt(mse)
> # Calculate R-squared
> rss <- sum((y_test - y_pred)^2)
> tss <- sum((y_test - mean(y_test))^2)
> r_squared <- 1 - (rss / tss)
> data.frame(mse, mae, rmse, r_squared) %>% print()
      mse      mae      rmse r_squared
1 382411.6 439.4899 618.3943 0.7921963
> |
```

Hình 72: Kết quả thực thi hàm `predict`

Nhận xét: Từ dự đoán trên ta có thể rút ra rằng mô hình còn sai lệch lớn với giá trị thực. Trung bình bình phương sai số (MSE) và trung bình độ lệch (MAE) cho thấy giá trị dự đoán còn sai lệch rất lớn so với giá trị thực, đồng thời căn bậc hai của mức trung bình của các sai số bình phương (RMSE) là thước đo mức độ dàn trải của những phần dư và hệ số xác định R^2 còn cho thấy mô hình chưa thực sự dự đoán được tổng thể toàn bộ dữ liệu. Điều này có thể được cải thiện độ chính xác bằng cách tăng số lượng mẫu dùng để dự đoán hoặc sử dụng một mô hình hồi quy khác.

5.4.7 Kết luận

Nhóm đã thành công trong việc nghiên cứu các lý thuyết về mô hình hồi quy tuyến tính bội, tìm hiểu các lệnh của ngôn ngữ R để thực hiện việc tính toán, xây dựng và kiểm định giả thiết cho mô hình. Hơn nữa, từ mô hình đã xây dựng, nhóm cũng đã đưa ra được những dự đoán cho biến phụ thuộc.

Tuy nhiên, kết quả nhóm trình bày có thể chưa thật sự chính xác. Nguyên nhân có thể đến từ nhiều yếu tố như sau

- Tỷ lệ khuyết dữ liệu lớn, ảnh hưởng của việc loại bỏ các quan sát khuyết trong quá trình tiền xử lý số liệu.
- Sự hạn chế về mặt dữ liệu, các mẫu dữ liệu không thoả mãn các giả thiết cần của mô hình, dẫn đến kết quả chưa thật sự có ý nghĩa lớn về mặt thống kê.



Chính vì những hạn chế trên, nhìn chung kết quả mà nhóm tìm được có thể có sai khác so với kết quả chính xác.

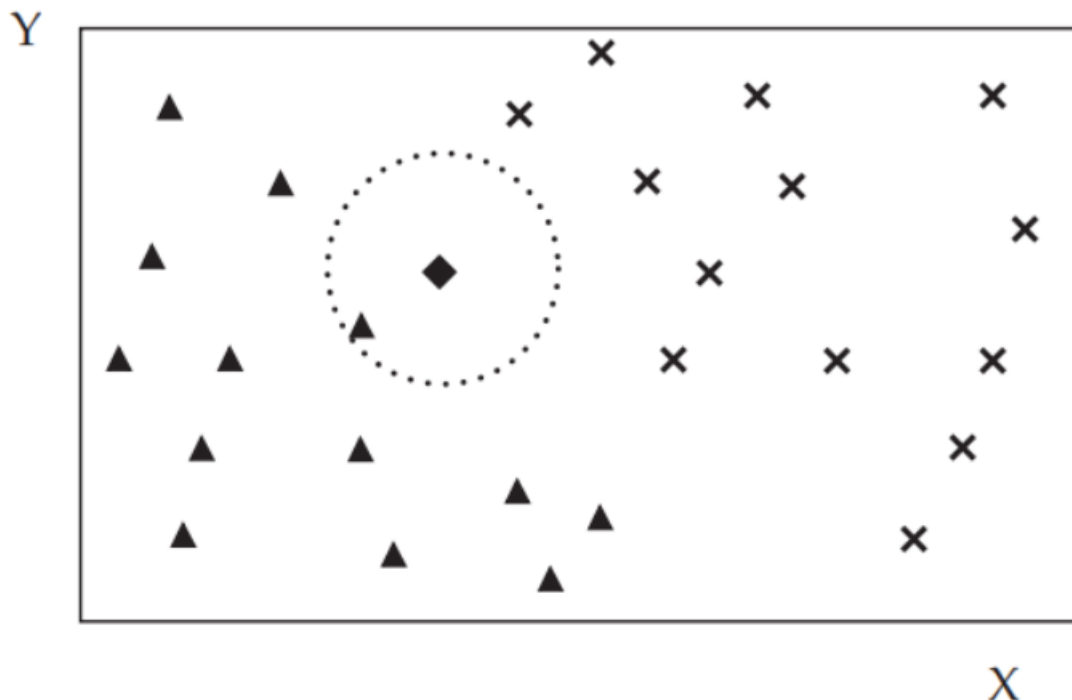
6 Thảo luận và mở rộng

Sử dụng thuật toán k -nearest neighbors để dự đoán thay cho hồi quy tuyến tính bội.

Thuật toán k -nearest neighbors :

Thuật toán kNN là một thuật toán phân loại đơn giản nhưng hiệu quả, dựa trên việc tìm kiếm k điểm dữ liệu gần nhất với điểm dữ liệu mới cần phân loại. Đây là một mô hình tuy đơn giản nhưng phù hợp với bài toán đặt ra.

Thuật toán kNN dự đoán nhãn cho một điểm dữ liệu mới dựa trên nhãn của các hàng xóm của nó. kNN dựa vào giả định rằng các điểm dữ liệu tương tự sẽ nằm gần nhau trong các biến tương quan khác trong tập dữ liệu, hay nói cách khác là “tọa độ” của nó trong không gian dữ liệu. Chỉ số k trong kNN là số lượng hàng xóm gần nhất mà chúng ta xem xét để thực hiện dự đoán. Chúng ta xác định sự gần nhau của một điểm dựa trên khoảng cách của nó (Ví dụ: Euclidean, Manhattan, vv) với các điểm đang xem xét. Ví dụ, nếu $k = 5$, chúng ta xem xét 5 điểm gần nhất và lấy nhãn của đa số trong 5 điểm này làm nhãn được dự đoán.



Để dễ hình dung hơn, ta quay lại với bài toán dự đoán giá trị của `Bus_Speed` thông

qua các biến `TDP`, `Cache`, `Launch_Date`, `Max_Memory_Size`, `Max_Memory_Bandwidth`, `Processor_Base_Frequency`, `Max_nb_of_Memory_Channels` và `Recommended_Customer_Price` CPU khi chế tạo sẽ có một chuẩn chung để xác định các thông số quan trọng như Interface giữa các bộ phận (bus hoặc interconnect), đồng nghĩa đa phần các thông số của các CPU có càng nhiều thông số giống nhau sẽ có *Bus Speed* gần tương đương hay bằng nhau. Từ ý tưởng đó, nhóm sử dụng mô hình phân loại `kNN` với biến dự đoán là giá trị của `Bus_Speed` và $k = 5$. Theo thuật toán `kNN`, giá trị của `Bus_Speed` chính là nhãn mà ta cần dự đoán. Những biến `Bus_Speed` thông qua các biến `TDP`, `Cache`, `Launch_Date`, `Max_Memory_Size`, `Max_Memory_Bandwidth`, `Processor_Base_Frequency`, `Max_nb_of_Memory_Channels` và `Recommended_Customer_Price` chính là từng điểm tọa độ đại diện cho nhãn. Ta sẽ có bảng dữ liệu với cột đầu tiên là cột nhãn (`Bus_Speed`), những cột tiếp theo là đại lượng định lượng của các biến dùng để dự đoán.

Mô hình về mặt lý thuyết tối ưu hơn mô hình hồi quy tuyến tính bởi vì giá trị được suy ra là giá trị được làm tròn quy ước do nhà sản xuất công bố theo từng dòng CPU và độ chênh lệch tốc độ truyền giữa các CPU cùng dòng tương đối thấp, độ chính xác cao đến rất cao. Tuy nhiên mô hình sẽ gặp bất lợi nếu dòng CPU được dùng để kiểm tra là dòng CPU có quá ít mẫu hoặc không tồn tại trong tập dữ liệu khi xây dựng mô hình.

Kết quả chạy thử bằng `C++` sử dụng thuật toán `kNN`, với $k = 5$, dữ liệu dự đoán bằng 0,5 độ lớn dữ liệu huấn luyện. Dưới đây là so sánh kết quả dự đoán của 25 giá trị đầu tiên.


```
Bus_Speed
5000
5000
5000
5000
5000
5000
5000
5000
5000
5000
5000
2500
2500
2500
4800
4800
4800
5860
5860
6400
6400
6400
6400
6400
5000
5000
```

Hình 73: Bus_Speed_test

```
Bus_Speed
5000
5000
5000
5000
5000
5000
5000
5000
5000
5000
5000
2500
2500
2500
2500
1333
4800
4800
4800
5860
5860
5860
6400
6400
6400
1066
6400
5000
5000
```

Hình 74: Bus_Speed_pred

Accuracy: 0.828774

Hình 75: Tỷ lệ đúng của kết quả dự đoán, thể hiện độ trùng khớp của dữ liệu dự đoán với dữ liệu thực tế

Nhật xét: Tỷ lệ 0,828774 là một con số rất cao, biểu thị thuật toán này rất có giá trị dự đoán và sử dụng thay cho hồi quy tuyến tính bội đối với những biến không phụ thuộc tuyến tính mà theo từng cụm.

7 Nguồn dữ liệu và nguồn code

- Nguồn file `Intel_CPU.csv`: <https://www.kaggle.com/datasets/iliassekka/f/computerparts/>
- Nguồn code: https://drive.google.com/file/d/1r_OmJiIH0HdWTJnDEHUqgkv36RNl5jnZ/view?usp=sharing

Tài liệu

- [1] Nguyễn Kiều Dung, *Slide bài giảng Xác suất và Thống kê*.
- [2] Nguyễn Đình Huy, *Giáo trình Xác suất và Thống kê*, Nhà xuất bản Đại học quốc gia Thành phố Hồ Chí Minh.
- [3] Patterson, D. A., & Hennessy, J. L. (2004), *Computer Organization and Design: The Hardware/Software Interface*.
- [4] Algorithmica, *RAM & CPU Caches: Memory Bandwidth*. Truy cập tại <https://en.algorithmica.org/hpc/cpu-cache/bandwidth/>.
- [5] Green, O., Fox, J., Young, J., Shirako, J., & Bader, D, (2019), *Performance Impact of Memory Channels on Sparse and Irregular Algorithms*. Truy cập tại <https://arxiv.org/pdf/1910.03679.pdf>.
- [6] H.Kwak, B.Lee, A.R. Hurson, Suk-Han Yoon, Woo-Jong Hahn, (1999), *Effects of multithreading on cache performance*. Truy cập tại <https://ieeexplore.ieee.org/document/752659>.
- [7] HNC (27/09/2021), *Ý nghĩa của giá trị R bình phương hiệu chỉnh trong hồi quy*. Truy cập tại <https://hocnghienccu.com/y-nghia-cua-gia-tri-r-binh-phuong-hieu-chinh-trong-hoi-quy/>.
- [8] Prisaganec, Milco & Mitrevski, Pece, (2016), *Reducing Competitive Cache Misses in Modern Processor Architectures*. Truy cập tại https://www.researchgate.net/publication/303895846_Reducing_Competitive_Cache_Misses_in_Modern_Processor_Architectures.