

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



XÁC SUẤT VÀ THỐNG KÊ (MT2013)

Bài tập lớn

ĐÁNH GIÁ HIỆU NĂNG CPU_s

Giảng viên hướng dẫn:

Sinh viên thực hiện:

Nguyễn Kiều Dung

Nguyễn Ngô Uyên Nhi - 2312501 (*Nhóm trưởng*)

Bùi Hữu Lợi - 2311972

Bùi Trần Duy Khang - 2311402

Dương Hồ Nam - 2312153

Thân Thiên Kim - 2311795

Trần Khánh An - 2310037

Dương Khả Vân - 2313866

Trần Lê Gia Thoại - 2313323

Nhóm:

MT10 - L11

Thành phố Hồ Chí Minh, tháng 4 năm 2025



DANH SÁCH THÀNH VIÊN

STT	Họ và Tên	MSSV	Nhiệm vụ	% Thực hiện
1	Nguyễn Ngô Uyên Nhi	2312501	Kiến thức nền	100%
2	Bùi Hữu Lợi	2311972	Tổng quan dữ liệu	100%
3	Bùi Trần Duy Khang	2311402	Tiền xử lý số liệu	100%
4	Dương Hồ Nam	2312153	Thảo luận mở rộng và tổng hợp	100%
5	Thân Thiên Kim	2311795	Thống kê mô tả	100%
6	Trần Khánh An	2310037	Thống kê mô tả	100%
7	Dương Khả Vân	2313866	Thống kê suy diễn	100%
8	Trần Lê Gia Thoại	2313323	Thống kê suy diễn	100%

Bảng 1: Nhiệm vụ và đóng góp



Mục lục

Danh sách thành viên	1
Danh sách ảnh	5
Danh sách bảng	5
1 Tổng quan dữ liệu	5
1.1 Tổng quan bộ dữ liệu	5
1.2 Mô tả biến	5
1.3 Mục tiêu	6
1.4 Các bước tiến hành	6
1.4.1 Bài toán đặt ra	6
1.4.2 Các bước giải quyết	6
2 Kiến thức nền	7
2.1 Khái niệm về một số đại lượng cơ bản trong thống kê	7
2.2 Ước lượng và kiểm định	8
2.2.1 Ước lượng bằng khoảng tin cậy	8
2.2.2 Kiểm định giả thuyết	9
2.3 Phân tích phương sai – ANOVA một yếu tố	11
2.3.1 Mục tiêu	11
2.3.2 Giả định của mô hình	11
2.3.3 Các bước thực hiện ANOVA	11
2.3.4 Giải thích biến thiên	13
2.4 Hồi quy tuyến tính bội	13
2.4.1 Mô hình hồi quy	13
2.4.2 Ước lượng hệ số hồi quy	14
2.4.3 Hệ số xác định bội	14
2.4.4 Kiểm định sự ý nghĩa của mô hình hồi quy tuyến tính	15
2.4.5 Kiểm định sự phù hợp của các hệ số đường hồi quy tuyến tính	15
3 Tiền xử lý số liệu	15
3.1 Cấu trúc thư mục làm việc của toàn bộ Bài tập lớn	16
3.2 Đọc dữ liệu	16
3.3 Chọn lọc một số biến liên quan để phân tích	17
3.4 Làm sạch dữ liệu	18
3.4.1 Các hàm hỗ trợ	18
3.4.2 Hàm làm sạch dữ liệu	19
3.5 Kiểm tra số lượng và tỷ lệ dữ liệu bị khuyết	21
3.5.1 Đưa ra số liệu tổng quan về dữ liệu khuyết	21
3.5.2 Thay thế dữ liệu bị khuyết	22
3.6 Kiểm thử dữ liệu sau khi tiền xử lý	23
4 Thống kê mô tả	23
4.1 Tính toán thống kê mẫu	23
4.2 Mô tả dữ liệu bằng đồ thị	25
4.2.1 Đồ thị histogram	25
4.2.2 Đồ thị boxplot thể hiện phân phối	27



4.2.3	Đồ thị scatter thể hiện phân tán giữa TDP và các biến	30
4.3	Kiểm tra phân phối của các biến định lượng	35
4.4	Kiểm định tương quan	36
5	Thống kê suy diễn	37
5.1	Giới thiệu	37
5.2	Bài toán kiểm định một mẫu	38
5.3	Bài toán kiểm định hai mẫu	39
5.4	Mô hình ANOVA một nhân tố	42
5.5	Hồi quy	46
5.5.1	Phân chia dữ liệu	46
5.5.2	Mô hình hồi quy tuyến tính đa biến	46
5.5.3	Kiểm tra giả thiết thống kê	49
5.5.4	Dự báo	50
6	Thảo luận và mở rộng	53
6.1	Đánh giá kết quả phân tích	53
6.1.1	Khác biệt giữa các phân khúc CPUs	53
6.1.2	TDP tăng theo thời gian	53
6.2	Giới hạn của kiểm định	53
6.3	Mở rộng hướng kiểm định và nghiên cứu	53
6.3.1	Áp dụng mô hình học máy	53
6.3.2	Giảm chiều và phát hiện cộng tuyến	54
6.3.3	Phân tích phân cụm CPU	54
7	Nguồn dữ liệu và nguồn code	55
	Tài liệu tham khảo	56



Danh sách ảnh

1	Minh họa dữ liệu sau khi tiền xử lý được xuất ra tệp <code>processed_Intel_CPUs.csv</code> và đọc dữ liệu bằng phần mềm Microsoft Excel	23
2	Bảng kết quả tính toán các thống kê mô tả cho các biến định lượng	24
3	Đồ thị histogram của biến Lithography	26
4	Đồ thị histogram của biến nb_of_Cores	26
5	Đồ thị histogram của biến nb_of_Threads	26
6	Đồ thị histogram của biến Processor_Base_Frequency	26
7	Đồ thị histogram của biến Cache	27
8	Đồ thị histogram của biến TDP	27
9	Đồ thị histogram của biến Max_Memory_Bandwidth	27
10	Đồ thị boxplot giữa TDP với Vertical Segment	28
11	Đồ thị boxplot giữa TDP với Vertical Segment theo nhóm năm	30
12	Đồ thị Scatter của TDP với Vertical Segment	31
13	Đồ thị Scatter của TDP với Launch Year	31
14	Đồ thị Scatter của TDP với Lithography	32
15	Đồ thị Scatter của TDP với Number of Cores	32
16	Đồ thị Scatter của TDP với Number of Threads	33
17	Đồ thị Scatter của TDP với Processor Base Frequency	33
18	Đồ thị Scatter của TDP với Cache	34
19	Đồ thị Scatter của TDP với Embedded Options Availale	34
20	Đồ thị Scatter của TDP với Max Memory Banwidth	35
21	Phân phối của các biến định lượng	36
22	Ma trận tương quan	37
23	Đồ thị Q-Q plot đối với mẫu Embedded_Option	40
24	Đồ thị Q-Q plot đối với mẫu Non_Embedded_Option	41
25	Đồ thị Q-Q plot đối với các mẫu Vertical_Segment	43
26	Đồ thị so sánh bội giữa các cặp Vertical_Segment	46
27	Các đồ thị	49
28	Kết quả dự báo trên tập test.data	52
29	Kết quả đánh giá mô hình	53

Danh sách bảng

1	Nhiệm vụ và đóng góp	1
---	--------------------------------	---

1 Tổng quan dữ liệu

1.1 Tổng quan bộ dữ liệu

Bộ dữ liệu Computer Parts (CPUs and GPUs) được tác giả ILISSEK cung cấp trên nền tảng Kaggle, gồm thông tin chi tiết về các thành phần linh kiện máy tính, đặc biệt là CPU và GPU. Trong đó, nhóm tập trung vào tập dữ liệu *Intel_CPUs.csv* với quy mô 2.283 dòng tượng trưng cho 2.283 loại CPUs và 45 cột tượng trưng cho 45 thông số, thông tin kỹ thuật chi tiết về các dòng CPU Intel.

Link: https://www.kaggle.com/datasets/iliassekkaf/computerparts?select=Intel_CPUs.csv

Tập dữ liệu này bao gồm các thông tin chính:

- **Thông tin chung:** Bao gồm dòng sản phẩm (Product Collection), số hiệu bộ xử lý (Processor Number), trạng thái (Status), ngày ra mắt (Launch Date), và công nghệ chế tạo (Lithography). Đây là những thông tin cơ bản để xác định thế hệ và loại CPU.
- **Hiệu năng và đặc điểm:** Ghi nhận các thông số quan trọng như số lõi (*nb_of_Cores*), số luồng (*nb_of_Threads*), tần số cơ bản và tần số tối đa, tốc độ bus (Bus Speed), bộ nhớ đệm (Cache), công suất tiêu thụ (TDP). Ngoài ra, tập dữ liệu cũng cho biết khả năng hỗ trợ các công nghệ tiên tiến như Intel Hyper-Threading và Intel Virtualization.
- **Đồ họa và bộ nhớ:** Cung cấp thông tin về đồ họa tích hợp, loại bộ nhớ hỗ trợ, số kênh bộ nhớ tối đa, băng thông bộ nhớ, và khả năng hỗ trợ ECC (Error-Correcting Code), giúp cải thiện độ tin cậy trong xử lý dữ liệu.

Bộ dữ liệu này là nguồn tài nguyên hữu ích để phân tích hiệu năng và các yếu tố ảnh hưởng đến đặc tính kỹ thuật của CPU, từ đó hỗ trợ cho việc nghiên cứu và tối ưu hóa hiệu suất của các hệ thống máy tính.

1.2 Mô tả biến

Trong bài toán này, nhóm đã chọn ra được 10 biến liên quan từ tập dữ liệu *Intel_CPU.csv* để tiếp tục xử lý và phân tích. Bao gồm các biến phân loại, định lượng liên tục và định lượng rời rạc như sau:

STT	Tên biến (%)	Loại biến	Đơn vị	Chú thích
1	Vertical_Segment	Phân loại	–	Phân khúc máy tính
2	Launch_Date	Định lượng - Rời rạc	–	Ngày phát hành
3	Lithography	Định lượng - Liên tục	mm	Tiến trình chế tạo
4	nb_of_Cores	Định lượng - Rời rạc	core	Số nhân CPU
5	nb_of_Threads	Định lượng - Rời rạc	thread	Số luồng CPU
6	Processor_Base_Frequency	Định lượng - Liên tục	GHz, MHz	Tần số xung cơ bản
7	Cache	Định lượng - Liên tục	MB, KB	Bộ nhớ đệm
8	TDP	Định lượng - Liên tục	W	Công suất thiết kế nhiệt
9	Embedded_Options_Available	Phân loại	–	Tùy chọn nhúng
10	Max_Memory_Bandwidth	Định lượng - Liên tục	GB/s	Băng thông bộ nhớ tối đa

Các loại biến liên quan

1.3 Mục tiêu

Mục tiêu chính của nghiên cứu là dự đoán giá trị TDP (Thermal Design Power) của CPU dựa trên các thông số kỹ thuật khác. Thông qua việc phân tích dữ liệu, nhóm không chỉ xây dựng mô hình dự đoán TDP mà còn sử dụng các kỹ thuật thống kê để đánh giá mức độ ảnh hưởng và ý nghĩa của từng biến đầu vào trong việc giải thích sự biến thiên của TDP. Điều này giúp xác định những thông số kỹ thuật quan trọng nhất, từ đó cung cấp cái nhìn sâu sắc hơn về các yếu tố chính ảnh hưởng đến hiệu suất và khả năng tiêu thụ năng lượng của CPU. Nội dung cụ thể bao gồm các chi tiết chính:

- Dùng thống kê mô tả để lập bảng mô tả cho các chỉ số định lượng và kiểm tra phân phối đều. Đồng thời vẽ các đồ thị histogram, boxplot, scatter cho các biến.
- Dùng thống kê suy diễn kiểm tra ảnh hưởng của các biến đã chọn đến TDP bằng cách phương pháp như kiểm định hai mẫu, phân tích phương sai hay hồi quy sao cho phù hợp với từng biến cụ thể.
- Rút ra nhận xét về ảnh hưởng của các nhân tố đã nghiên cứu đến với TDP.

1.4 Các bước tiến hành

1.4.1 Bài toán đặt ra

Để đạt được mục tiêu đề ra, các bước tiến hành sẽ xử lý các loại dữ liệu độc lập nhau, đưa ra kết quả và nhận xét trong từng quá trình để đạt đến kết luận cuối cùng. Ta tổng quan xử lý ảnh hưởng của các biến độc lập đối với biến phụ thuộc là TDP thông qua 3 vấn đề chính sau:

- Kiểm định hai mẫu TDP dựa trên biến `Embedded_Options_Available`.
- Phân tích ANOVA cho biến phụ thuộc TDP dựa trên biến `Vertical_Segment` (phân thành 4 nhóm phân khúc).
- Hồi quy tuyến tính cho biến phụ thuộc TDP dựa trên các biến độc lập là `Lithography`, `nb_of_Cores`, `nb_of_Threads`, `Processor_Base_Frequency`, `Cache`, `Max_Memory_Bandwidth`.

1.4.2 Các bước giải quyết

Tiền xử lý dữ liệu:

- Lọc giá trị, chuyển đổi đơn vị và kiểu dữ liệu của biến định lượng.
- Nhận xét về trạng thái của tập dữ liệu trước và sau khi tiền xử lý.
- Lọc các giá trị khuyết và xử lý giá trị ngoại lệ.
- Chuẩn hóa và mã hóa biến phân loại cho phù hợp.

Thống kê mô tả:

- Tính toán thống kê các mẫu dữ liệu, bao gồm các thông số cơ bản và phân phối của các biến liên tục, tỷ lệ phần trăm các lớp trong biến phân loại.
- Mô tả dữ liệu bằng đồ thị và rút ra các nhận xét ban đầu.

Thống kê suy diễn:

- Thực hiện bài toán kiểm định hai mẫu và kiểm định ANOVA với các biến phân loại là biến độc lập.
- Thực hiện hồi quy tuyến tính với các biến định lượng là biến độc lập.
- Đánh giá các giả thiết hồi quy qua các kiểm định thống kê.
- Sử dụng mô hình để dự đoán giá trị TDP cho các mẫu CPU mới.

2 Kiến thức nền

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

– Kỳ vọng (Expectation/Mean)

Kỳ vọng của biến ngẫu nhiên X là giá trị trung bình theo xác suất của X .

Ký hiệu: $E(X)$ hoặc μ .

Công thức tính

- Đối với biến ngẫu nhiên rời rạc:

$$E(X) = \sum_i x_i p_i$$

- Đối với biến ngẫu nhiên liên tục:

$$E(X) = \int_{-\infty}^{+\infty} x f(x) dx$$

– Phương sai (Variance)

Phương sai của biến ngẫu nhiên X được định nghĩa bằng trung bình bình phương sai lệch giữa biến ngẫu nhiên với kỳ vọng của nó.

Ký hiệu: $V(X)$ hoặc σ^2 .

Công thức tính

$$V(X) = E[(X - E(X))^2] \quad \text{hay} \quad V(X) = E(X^2) - [E(X)]^2$$

- Nếu X là biến ngẫu nhiên rời rạc thì:

$$V(X) = \sum_i [x_i - E(X)]^2 p_i = \sum_i x_i^2 p_i - [E(X)]^2$$

- Nếu X là biến ngẫu nhiên liên tục thì:

$$V(X) = \int_{-\infty}^{+\infty} (x - E(X))^2 f(x) dx = \int_{-\infty}^{+\infty} x^2 f(x) dx - [E(X)]^2$$

– Độ lệch chuẩn (Standard Deviation)

Độ lệch chuẩn của biến ngẫu nhiên X (kí hiệu σ_X) là căn bậc hai của phương sai:

$$\sigma_X = \sigma(X) = \sqrt{V(X)}$$

– **Trung vị (Median)**

Trung vị của biến ngẫu nhiên X (ký hiệu $\text{Md}(X)$) là một giá trị sao cho:

$$P(X \leq \text{Md}) \geq 0.5 \quad \text{và} \quad P(X \geq \text{Md}) \geq 0.5$$

Khi X là biến ngẫu nhiên liên tục thì:

$$P(X \leq \text{Md}) = 0.5$$

– **Mốt (Mode)**

Mốt của biến ngẫu nhiên X (ký hiệu $\text{mod}(X)$) là:

- Đối với biến ngẫu nhiên rời rạc: giá trị của X tương ứng với xác suất lớn nhất.
- Đối với biến ngẫu nhiên liên tục: giá trị tương ứng với cực đại của hàm mật độ xác suất.

– **Cực tiểu (Minimum)**

Cực tiểu là giá trị nhỏ nhất trong toàn bộ các giá trị của một tập mẫu.

– **Cực đại (Maximum)**

Cực đại là giá trị lớn nhất trong toàn bộ các giá trị của một tập mẫu.

– **Tứ phân vị (Quartiles)**

Tứ phân vị là đại lượng mô tả sự phân bố và sự phân tán của tập dữ liệu. Có 3 giá trị tứ phân vị, chia tập dữ liệu đã sắp xếp theo thứ tự tăng dần thành 4 phần có số lượng quan sát bằng nhau:

- Tứ phân vị thứ hai Q_2 chính là trung vị.
- Tứ phân vị thứ nhất Q_1 là trung vị của phần dữ liệu nằm dưới Q_2 .
- Tứ phân vị thứ ba Q_3 là trung vị của phần dữ liệu nằm trên Q_2 .

– **Outliers**

Giá trị outlier là những giá trị dữ liệu bất thường, nằm ngoài khoảng phân bố chính của tập dữ liệu.

2.2 Ước lượng và kiểm định

2.2.1 Ước lượng bằng khoảng tin cậy

- **Khái niệm:** Khoảng tin cậy là khoảng giá trị (c, d) sao cho xác suất tham số tổng thể θ nằm trong khoảng này là $1 - \alpha$ (độ tin cậy của ước lượng, thường là 95% hoặc 99%).
- **Dạng đối xứng phổ biến:**
Nếu $\hat{\theta}$ là một ước lượng không chệch lệch của θ thì khoảng ước lượng của θ có dạng

$$(\hat{\theta} - \varepsilon, \hat{\theta} + \varepsilon)$$

Trong đó:

- $\hat{\theta}$: Ước lượng không chệch
- ε : Sai số cho phép (độ chính xác)

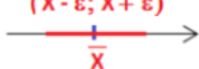
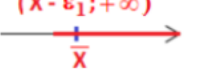
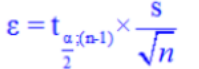
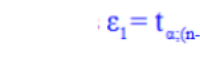
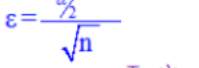
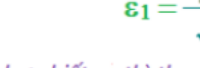
– Giả sử

$$(\hat{\theta} - \varepsilon, \hat{\theta} + \varepsilon)$$

là khoảng ước lượng đối xứng của θ với độ tin cậy $1 - \alpha$ thì:

$$P[\theta \in (\hat{\theta} - \varepsilon, \hat{\theta} + \varepsilon)] = P(|\theta - \hat{\theta}| < \varepsilon) = 1 - \alpha$$

• Các khoảng ước lượng thông dụng cho bài toán 1 mẫu:

		$(\bar{X} - \varepsilon; \bar{X} + \varepsilon)$	$(\bar{X} - \varepsilon_1; +\infty)$	$(-\infty; \bar{X} + \varepsilon_1)$
Trung bình tổng thể μ (2)	<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)	 $\varepsilon = \frac{t_{\alpha/2(n-1)} \times s}{\sqrt{n}}$	 $\varepsilon_1 = \frac{t_{\alpha(n-1)} \times s}{\sqrt{n}}$	
	<ul style="list-style-type: none"> Phân phối tùy ý Mẫu lớn ($n \geq 30$) (2c) <i>Sử dụng định lý giới hạn</i>	 $\varepsilon = \frac{Z_{\alpha/2} \times \sigma}{\sqrt{n}}$	 $\varepsilon_1 = \frac{Z_{\alpha} \times \sigma}{\sqrt{n}}$	

Trường hợp chưa biết σ thì thay thế bởi s

2.2.2 Kiểm định giả thuyết

• **Giả thuyết:**

- H_0 : Giả thuyết không (null hypothesis) – giả thuyết về yếu tố cần kiểm định của tổng thể ở trạng thái bình thường, không chịu tác động của các hiện tượng liên quan.
- H_1 : Giả thuyết đối (alternative hypothesis) – thể hiện mệnh đề mâu thuẫn với H_0 , mô tả xu hướng cần kiểm định.

• **Tiêu chuẩn kiểm định (G):** Một hàm thống kê

$$G = G(X_1, X_2, \dots, X_n, \theta_0)$$

dựa trên mẫu ngẫu nhiên

$$W = (X_1, X_2, \dots, X_n)$$

và tham số θ_0 , dùng để kiểm tra 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 (RR):** Miền giá trị mà nếu G rơi vào thì ta bác bỏ H_0 , với xác suất α (mức ý nghĩa, thường $< 10\%$).

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 $G \in RR$ có xác suất nhỏ nên với một mẫu thực nghiệm ngẫu nhiên, nó không thể xảy ra.

- Nếu $g_{qs} \in RR \Rightarrow$ bác bỏ H_0 , chấp nhận H_1 .
- Nếu $g_{qs} \notin RR \Rightarrow$ chưa đủ bằng chứng để bác bỏ H_0 , H_1 đúng.
- Miền chấp nhận AR : phần bù của miền bác bỏ RR trong R .
- **Kiểm định trung bình 2 mẫu:** So sánh trung bình của hai nhóm độc lập xem có khác biệt thống kê hay không.

TT	Phân bố của tổng thể	Gt H_0	Gt H_1	Miền bác bỏ RR	Tiêu chuẩn kiểm định
(4a)	<ul style="list-style-type: none"> * 2 mẫu độc lập * $X_1; X_2$ có pp chuẩn. * Đã biết σ_1^2 và σ_2^2 	$\mu_1 = \mu_2$	$\mu_1 \neq \mu_2$ $\mu_1 < \mu_2$ $\mu_1 > \mu_2$	$(-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; +\infty)$ $(-\infty; -z_{\alpha})$ $(z_{\alpha}; +\infty)$	$Z_{qs} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$ <p>z-test</p>
(4b)	<ul style="list-style-type: none"> * 2 mẫu độc lập * $X_1; X_2$ có pp chuẩn * Chưa biết $\sigma_1^2; \sigma_2^2$; gt $\sigma_1^2 = \sigma_2^2$ Dấu hiệu quy ước để nhận biết trường hợp (4b) từ mẫu: $\frac{s_1}{s_2} \in \left[\frac{1}{2}; 2\right]$ 	$\mu_1 = \mu_2$	$\mu_1 \neq \mu_2$ $\mu_1 < \mu_2$ $\mu_1 > \mu_2$	$(-\infty; -t_{\alpha/2; (n_1+n_2-2)}) \cup (t_{\alpha/2; (n_1+n_2-2)}; +\infty)$ $(-\infty; -t_{\alpha; (n_1+n_2-2)})$ $(t_{\alpha; (n_1+n_2-2)}; +\infty)$	$s_p^2 = \frac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1+n_2-2}$ $T_{qs} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_p^2}{n_1} + \frac{s_p^2}{n_2}}}$ <p>t-test</p>
(4c)	<ul style="list-style-type: none"> * 2 mẫu độc lập * $X_1; X_2$ có pp chuẩn * Chưa biết $\sigma_1^2; \sigma_2^2$; có thể giả thiết $\sigma_1^2 \neq \sigma_2^2$ Dấu hiệu quy ước để nhận biết trường hợp (4c) từ mẫu: $\frac{s_1}{s_2} \in \left[\frac{1}{2}; 2\right]$ 	$\mu_1 = \mu_2$	$\mu_1 \neq \mu_2$ $\mu_1 < \mu_2$ $\mu_1 > \mu_2$	$(-\infty; -t_{\alpha/2; (v)}) \cup (t_{\alpha/2; (v)}; +\infty)$ $(-\infty; -t_{\alpha; (v)})$ $(t_{\alpha; (v)}; +\infty)$ <i>bộ v làm tròn số nguyên</i>	$v = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{\left(\frac{s_1^2}{n_1}\right)^2}{n_1-1} + \frac{\left(\frac{s_2^2}{n_2}\right)^2}{n_2-1}}$ $T_{qs} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$ <p>t-test</p>

	Phân bố của tổng thể	Gt H_0	Gt H_1	Miền bác bỏ RR	Tiêu chuẩn kiểm định
(4d)	<ul style="list-style-type: none"> * 2 mẫu độc lập * $X_1; X_2$ có pp tùy ý. * 2 mẫu lớn: $n_1; n_2 \geq 30$ * Đã biết hoặc chưa biết $\sigma_1^2; \sigma_2^2$ 	$\mu_1 = \mu_2$	$\mu_1 \neq \mu_2$ $\mu_1 < \mu_2$ $\mu_1 > \mu_2$	$(-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; +\infty)$ $(-\infty; -z_{\alpha})$ $(z_{\alpha}; +\infty)$ <i>Nếu chưa biết σ_1^2 và σ_2^2 thì dùng s_1^2 và s_2^2 để thay thế.</i>	$Z_{qs} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$
(4e) ₀	<ul style="list-style-type: none"> * 2 mẫu phụ thuộc tương ứng theo cặp * $X_D = X_1 - X_2$ có pp chuẩn * Đã biết σ_D^2 	$\mu_1 = \mu_2$ hay $\mu_D = 0$	$\mu_1 \neq \mu_2$ $\mu_1 < \mu_2$ $\mu_1 > \mu_2$	$(-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; +\infty)$ $(-\infty; -z_{\alpha})$ $(z_{\alpha}; +\infty)$	$Z_{qs} = \frac{\bar{X}_D}{\frac{\sigma_D}{\sqrt{n}}}$ <p>(2a)</p> <p>z-test</p>
(4e)	<ul style="list-style-type: none"> * 2 mẫu phụ thuộc tương ứng theo cặp * $X_D = X_1 - X_2$ có pp chuẩn * Chưa biết σ_D^2 	$\mu_1 = \mu_2$ hay $\mu_D = 0$	$\mu_1 \neq \mu_2$ $\mu_1 < \mu_2$ $\mu_1 > \mu_2$	$(-\infty; -t_{\alpha/2; (n-1)}) \cup (t_{\alpha/2; (n-1)}; +\infty)$ $(-\infty; -t_{\alpha; (n-1)})$ $(t_{\alpha; (n-1)}; +\infty)$	$T_{qs} = \frac{\bar{X}_D}{\frac{s_D}{\sqrt{n}}}$ <p>(2b)</p> <p>t-test</p>
(4f)	<ul style="list-style-type: none"> * 2 mẫu pt t/đ theo cặp. * 2 mẫu lớn: $n \geq 30$ * D có phân phối tùy ý * Đã biết hoặc chưa biết σ_D^2 	$\mu_1 = \mu_2$ hay $\mu_D = 0$	$\mu_1 \neq \mu_2$ $\mu_1 < \mu_2$ $\mu_1 > \mu_2$	$(-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; +\infty)$ $(-\infty; -z_{\alpha})$ $(z_{\alpha}; +\infty)$	$Z_{qs} = \frac{\bar{X}_D}{\frac{\sigma_D}{\sqrt{n}}}$ <p>(2c)</p> <p><i>Nếu chưa biết σ_D^2 thì dùng s_D^2</i></p>

2.3 Phân tích phương sai – ANOVA một yếu tố

2.3.1 Mục tiêu

Phân tích phương sai một nhân tố là so sánh trung bình của nhiều yếu tố nguyên nhân (từ 3 yếu tố trở lên) đến một yếu tố kết quả (định lượng).

2.3.2 Giả định của mô hình

- Mỗi nhóm tuân theo phân phối chuẩn: $N(\mu_i, \sigma^2)$, $i = 1, 2, \dots, k$ (k là số tổng thể, thường $k \geq 3$).
- Các phương sai bằng nhau: $\sigma_1^2 = \sigma_2^2 = \dots = \sigma_k^2$
- Các mẫu được lấy độc lập

2.3.3 Các bước thực hiện ANOVA

1. Đặt giả thuyết:

- $H_0: \mu_1 = \mu_2 = \dots = \mu_k$
- H_1 : Tồn tại $i \neq j$ sao cho $\mu_i \neq \mu_j$

2. Tính thống kê kiểm định:

	Nhóm 1	Nhóm 2	...	Nhóm k
Các mẫu quan sát	x_{11} x_{21} ... $x_{n_1; 1}$	x_{12} x_{22} ... $x_{n_2; 2}$...	x_{1k} x_{2k} ... $x_{n_k; k}$
Kích thước từng mẫu	n_1	n_2		n_k
Trung bình từng mẫu	\bar{x}_1	\bar{x}_2		\bar{x}_k
Kích thước mẫu gộp	$N = n_1 + n_2 + \dots + n_k$			
Trung bình mẫu gộp	$\bar{x} = \sum_j \sum_i x_{ij} / N = (n_1 \bar{x}_1 + n_2 \bar{x}_2 + \dots + n_k \bar{x}_k) / N$			

SSB (SSTr)	Sum of squares between group	$SSB = \sum_{j=1}^k n_j \times (\bar{x}_j - \bar{x})^2$
SSW (SSE)	Sum of squares within group	$SSW = \sum_{j=1}^k \sum_{i=1}^{n_j} (x_{ij} - \bar{x}_j)^2$
SST	Total sum of squares	$SST = \sum_{j=1}^k \sum_{i=1}^{n_j} (x_{ij} - \bar{x})^2$

Source of Variation	Tổng Bình phương chênh lệch	Bậc tự do	Phương sai (Trung bình BPCL)	Tiêu chuẩn kiểm định F
Between Groups	SSB (SSTr) (3)	k - 1	$MSB = \frac{SSB}{k-1}$	$F_{qs} = \frac{MSB}{MSW}$
Within Groups	SSW (SSE) (4)	N - k	$MSW = \frac{SSW}{N-k}$	
Total	SST (6)	N - 1		

3. Miền bác bỏ:

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

4. **Kết luận:** Nếu bác bỏ $H_0 \Rightarrow$ yếu tố có ảnh hưởng đến biến phụ thuộc.

2.3.4 Giải thích biến thiên

- **SSB (SSTr):** Biến thiên của giá trị X do các mức độ của yếu tố đang xem xét tạo ra.
- **SSW (SSE):** Biến thiên của giá trị X do các yếu tố nào đó không được đề cập đến.
- **SST:** Tổng các biến thiên của X do tất cả các yếu tố tạo ra.

Hệ số xác định:

$$R^2 = \frac{SSB}{SST} \times 100\%$$

Hệ số xác định R^2 của mô hình Phân tích phương sai được sử dụng để đo mức độ ảnh hưởng của yếu tố được xem xét trong mô hình đối với sự biến động của các giá trị của biến ngẫu nhiên X quanh giá trị trung bình của nó.

$\Rightarrow R^2$ càng cao, mô hình càng phù hợp.

2.4 Hồi quy tuyến tính bội

2.4.1 Mô hình hồi quy

Bài toán hồi quy tuyến tính bội là bài toán nghiên cứu mối liên hệ phụ thuộc của một biến (gọi là biến phụ thuộc) vào nhiều biến khác (gọi là các biến độc lập), với ý tưởng ước lượng được giá trị trung bình (tổng thể) của biến phụ thuộc theo giá trị của các biến độc lập, dựa theo mẫu được biết trước. 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.

Nghiên cứu mối quan hệ giữa một biến phụ thuộc Y và $p - 1$ biến độc lập X_i .

- Dạng mô hình:

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

hay

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

- Trong đó:

- Y_i : Biến phụ thuộc
- X_{ik} : Biến độc lập, ghi chú cho biến i
- β_k : Hệ số của các biến độc lập, trong đó β_0 là hệ số hồi quy
- $\varepsilon_i \sim N(0, \sigma^2)$: Sai số ngẫu nhiên của mô hình hồi quy

- Giả định:

- ε_i có phân phối chuẩn và phương sai không đổi

$$\varepsilon_i \sim N(0, \sigma^2)$$

- Các biến X_{ik} độc lập với nhau.
- Kiểm định từng tham số hồi quy tổng thể

- * Nếu $\beta_k = 0 \Rightarrow$ không có mối tương quan giữa Y_i và X_{ik} . Nghĩa là, X_{ik} không ảnh hưởng đến Y_i .
 - * Nếu $\beta_k > 0 \Rightarrow Y_i$ và X_{ik} có quan hệ thuận chiều: X_{ik} tăng $\Rightarrow Y_i$ tăng (giả định các biến độc lập khác không đổi).
 - * Nếu $\beta_k < 0 \Rightarrow Y_i$ và X_{ik} có quan hệ nghịch chiều: X_{ik} tăng $\Rightarrow Y_i$ giảm (giả định các biến độc lập khác không đổi).
- Ma trận hồi quy:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1,p-1} \\ 1 & x_{21} & x_{22} & \cdots & x_{2,p-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{n,p-1} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{p-1} \end{bmatrix}, \quad \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Với dữ liệu:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

2.4.2 Ước lượng hệ số hồi quy

$$L = \sum_{i=1}^n \epsilon_i^2 = \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^n \beta_j X_{ij} \right)^2$$

Tìm giá trị nhỏ nhất của hàm L ta sẽ tìm được các hệ số hồi quy mẫu cho phương trình hồi quy. Trong khi xây dựng mô hình hồi quy tuyến tính bội, ta cần kiểm tra các giả thuyết như: hàm hồi quy là hàm tuyến tính theo các tham số, sai số ngẫu nhiên độc lập với nhau tuân theo phân phối chuẩn với kỳ vọng bằng 0 và phương sai là σ^2 .

2.4.3 Hệ số xác định bội

Để xác định được phần biến thiên trong biến phụ thuộc được giải thích bởi mối liên hệ giữa biến phụ thuộc và tất cả các biến độc lập trong mô hình, người ta đi xác định hệ số xác định bội R^2 ($0 \leq R^2 \leq 1$). Hệ số xác định bội sẽ giải thích trong 100% sự biến động của Y_i so với trung bình của nó thì có bao nhiêu % là do biến các biến X_j gây ra.

Hệ số xác định:

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

Trong đó:

- **SST** (Sum of Squares Total): tổng bình phương tất cả các sai lệch giữa các giá trị của biến phụ thuộc và giá trị trung bình.
- **SSR** (Sum of Squares in Regression): tổng bình phương của tất cả các sai lệch giữa các giá trị của biến phụ thuộc Y nhận được từ hàm hồi quy mẫu và giá trị trung bình của chúng.
- **SSE** (Sum of Squares for Error): đo sự chênh lệch giữa từng giá trị quan sát và giá trị dự đoán (sai số do những yếu tố ngoài X hoặc do lấy mẫu ngẫu nhiên).

- R^2 cao nghĩa là tỷ lệ phần trăm biến thiên của biến phụ thuộc Y so với trung bình của nó do biến X gây ra càng cao.

2.4.4 Kiểm định sự ý nghĩa của mô hình hồi quy tuyến tính

Kiểm định để xác định xem có tồn tại mối quan hệ tuyến tính hay không giữa biến phụ thuộc Y và tập con của các biến độc lập X_1, X_2, \dots, X_k . Như đã biết, hệ số xác định bội R^2 là đại lượng cho biết có bao nhiêu phần trăm biến thiên trong biến phụ thuộc có thể được giải thích bởi mô hình hồi quy, là một số thống kê trên mẫu có thể sử dụng để suy diễn về việc mô hình toàn diện có ý nghĩa và thống kê hay không trong việc giải thích cho biến thiên của biến phụ thuộc. Với ý tưởng này, nhóm tác giả đặt ra các giả thuyết kiểm định như sau:

- $H_0 : R^2 = 0$ - mô hình không đưa ra thông tin nào về sự thay đổi của biến phụ thuộc Y .
- $H_1 : R^2 \neq 0$ - mô hình phù hợp với sự thay đổi của biến phụ thuộc Y .

Tiêu chuẩn kiểm định:

$$F = \frac{MSR}{MSE}$$

Miền bác bỏ $RR = (f_{\alpha; (1, n-2)}; +\infty)$

2.4.5 Kiểm định sự phù hợp của các hệ số đường hồi quy tuyến tính

Dựa vào kết quả ước lượng với một mẫu cụ thể, ta có thể đánh giá được mối quan hệ giữa biến phụ thuộc và các biến độc lập trong mô hình một cách tương đối.

- Ở mức ý nghĩa α , giả thuyết vô hiệu H_0 được kiểm định:

$$H_0 : \beta_k = 0 \quad \text{và} \quad H_1 : \beta_k \neq 0$$

- Tiêu chuẩn kiểm định:

$$t = \frac{\hat{\beta}_k}{SE(\hat{\beta}_k)}$$

- $\hat{\beta}_k$: Hệ số hồi quy ước lượng từ mô hình.
- $SE(\hat{\beta}_k)$: Sai số chuẩn của hệ số $\hat{\beta}_k$.

- Miền bác bỏ:

$$RR = (-\infty, -t_{\alpha/2; (n-2)}) \cup (t_{\alpha/2; (n-2)}, +\infty)$$

3 Tiền xử lý số liệu

Tiền xử lý số liệu đóng vai trò là giai đoạn khởi đầu và vô cùng quan trọng trong quy trình phân tích và thống kê dữ liệu. Nó bao gồm một tập hợp các kỹ thuật thiết yếu nhằm làm sạch, chuyển đổi và tổ chức dữ liệu thô (ở đây là dữ liệu được cung cấp từ Giảng viên), đưa dữ liệu về một định dạng có cấu trúc và phù hợp trước khi được sử dụng cho các bước phân tích chuyên sâu tiếp theo.

3.1 Cấu trúc thư mục làm việc của toàn bộ Bài tập lớn

ProbabilityAndStatistics_Assignment_CSE_K23

```
|-- Data
|   |-- Intel_CPUs.csv
|-- Test
|   |-- processed_Intel_CPUs.csv
|-- main.R
```

Trong đó:

- **Data/Intel_CPUs.csv**: Tập tin dữ liệu gốc được sử dụng làm đầu vào, chứa thông tin chi tiết về các bộ vi xử lý Intel.
- **Test/processed_Intel_CPUs.csv**: Tập tin chứa kết quả sau khi xử lý và trích lọc dữ liệu, dùng để kiểm tra hoặc lưu trữ tạm thời.
- **main.R**: Tập tin chứa mã nguồn chính, thực hiện tất cả việc đọc, xử lý, phân tích và đánh giá dữ liệu.

Nhóm tác giả sử dụng ngôn ngữ lập trình **R** và trình soạn thảo tích hợp biên dịch **RStudio** để hiện thực các quá trình xử lý, phân tích dữ liệu.

3.2 Đọc dữ liệu

```
1 # Read the data from file
2 file_path <- "./Data/Intel_CPUs.csv"
3 CPU_data <- load_dataset(file_path)
4 head(CPU_data, 10)
```

Đoạn mã trên thực hiện các bước cơ bản để đọc dữ liệu từ tập tin có định dạng **.csv** (Comma Separated Values) chứa thông tin về các bộ vi xử lý Intel. Cụ thể:

- **file_path <- "./Data/Intel_CPUs.csv"**: Đặt đường dẫn đến tập tin dữ liệu CSV.
- **CPU_data <- read.csv(file_path)**: Đọc nội dung của tập tin CSV vào một biến có tên là **CPU_data**. Dữ liệu này sẽ được lưu dưới dạng một **data.frame** trong R.
- **head(CPU_data, 10)**: Lệnh in ra 10 dòng đầu tiên của dữ liệu đã được gán trong **CPU_data**.

Output

```
> head(CPU_data, 10)
# A tibble: 10 × 45
  Product_Collection Vertical_Segment Processor_Number Status Launch_Date Lithography
  <chr>                <chr>          <chr>          <chr>   <chr>      <chr>
1 7th Generation In... Mobile      i7-7Y75      Launc... Q3'16      14 nm
2 8th Generation In... Mobile      i5-8250U     Launc... Q3'17      14 nm
3 8th Generation In... Mobile      i7-8550U     Launc... Q3'17      14 nm
4 Intel® Core™ X-se... Desktop    i7-3820      End o... Q1'12      32 nm
5 7th Generation In... Mobile      i5-7Y57      Launc... Q1'17      14 nm
6 Intel® Celeron® P... Mobile      3205U        Launc... Q1'15      14 nm
7 Intel® Celeron® P... Mobile      N2805        Launc... Q3'13      22 nm
8 Intel® Celeron® P... Desktop    J1750        Launc... Q3'13      22 nm
9 Intel® Celeron® P... Desktop    G1610        Launc... Q1'13      22 nm
10 Legacy Intel® Pen... Mobile      518          End o... NA          90 nm
# 39 more variables: Recommended_Customer_Price <chr>, nb_of_Cores <dbl>,
```

```
# nb_of_Threads <dbl>, Processor_Base_Frequency <chr>, Max_Turbo_Frequency <chr>,  
# Cache <chr>, Bus_Speed <chr>, TDP <chr>, Embedded_Options_Available <chr>,  
# Conflict_Free <chr>, Max_Memory_Size <chr>, Memory_Types <chr>,  
# Max_nb_of_Memory_Channels <dbl>, Max_Memory_Bandwidth <chr>,  
# ECC_Memory_Supported <chr>, Processor_Graphics_ <lg1>,  
# Graphics_Base_Frequency <chr>, Graphics_Max_Dynamic_Frequency <chr>, ...
```

Trước khi tiến hành phân tích dữ liệu, nhóm tác giả đã thực hiện bước khảo sát sơ bộ để kiểm tra chất lượng và định dạng của các biến trong tập dữ liệu `Intel_CPUs.csv`. Dưới đây là một số nhận xét quan trọng:

- **Định dạng không phù hợp của các biến định lượng:** Một số biến đáng lẽ thuộc kiểu số (định lượng) lại đang được lưu dưới dạng chuỗi ký tự (**character**) do chứa đơn vị. Chúng cần được tách và chuyển đổi sang kiểu số (**numeric**) phục vụ cho các phân tích định lượng.
- **Thiếu dữ liệu (missing values):** Qua quan sát, một số biến có giá trị bị thiếu (hiển thị với các hình thức: là NA, N_A và cả chuỗi rỗng). Có thể xuất hiện các giá trị thiếu do dữ liệu không được công bố hoặc không áp dụng với một số dòng sản phẩm nhất định.
- **Chưa thống nhất đơn vị đo:** Một số biến định lượng chứa đơn vị đo khác nhau trong cùng một cột. Cần chuẩn hóa tất cả các giá trị về cùng một đơn vị chuẩn trong mỗi biến.
- **Các biến logic và nhị phân được biểu diễn dưới dạng chuỗi:** Một số biến như `Embedded_Options_Available`, `ECC_Memory_Supported`, `Conflict_Free` đang ở dạng **character** với giá trị "Yes" hoặc "No".
- **Dữ liệu văn bản chứa ký tự đặc biệt:** Một số biến văn bản như `Product_Collection` và `Processor_Number` có thể chứa các ký tự đặc biệt hoặc định dạng khác nhau giữa các dòng, cần được làm sạch để sử dụng cho mục đích phân loại hoặc tạo nhãn.

3.3 Chọn lọc một số biến liên quan để phân tích

```
1 # Define required columns  
2 selected_cols <- c(  
3   "Product_Collection", "Vertical_Segment", "Launch_Date", "Lithography",  
4   ↪ "nb_of_Cores",  
5   "nb_of_Threads", "Processor_Base_Frequency", "Cache", "Cache_Value_MB",  
6   ↪ "Cache_Type",  
7   "TDP", "Max_Memory_Bandwidth", "Embedded_Options_Available"  
8 )  
9 # Choose selected column and read data in these  
10 selected_data <- CPU_data[, intersect(selected_cols, names(CPU_data))]  
11 str(selected_data)
```

Ta chọn lọc tên của các cột cần phân tích và gán vào trong biến `selected_cols`. Sau đó, dùng hàm `intersect(selected_cols, names(CPU_data))` so sánh hai danh sách `selected_cols` và `names(CPU_data)` và trả về một danh sách các cột chung tồn tại trong cả hai.

Ở đây, nhóm tác giả chọn các cột có tên như trên để thực hiện các công việc tiếp theo.

Output

```
> str(selected_data)
tibble [2,283 × 11] (S3: tbl_df/tbl/data.frame)
 $ Product_Collection      : chr [1:2283] "7th Generation Intel® Core™ i7 Processors" ...
 $ Vertical_Segment        : chr [1:2283] "Mobile" "Mobile" "Mobile" "Desktop" ...
 $ Launch_Date             : chr [1:2283] "Q3'16" "Q3'17" "Q3'17" "Q1'12" ...
 $ Lithography             : chr [1:2283] "14 nm" "14 nm" "14 nm" "32 nm" ...
 $ nb_of_Cores             : num [1:2283] 2 4 4 4 2 2 2 2 1 ...
 $ nb_of_Threads           : num [1:2283] 4 8 8 8 4 2 2 2 NA ...
 $ Processor_Base_Frequency : chr [1:2283] "1.30 GHz" "1.60 GHz" "1.80 GHz" "3.60 GHz" ...
 $ Cache                  : chr [1:2283] "4 MB SmartCache" "6 MB SmartCache" "8 MB SmartCache" ...
 $ TDP                    : chr [1:2283] "4.5 W" "15 W" "15 W" "130 W" ...
 $ Max_Memory_Bandwidth    : chr [1:2283] "29.8 GB/s" "34.1 GB/s" "34.1 GB/s" "51.2 GB/s" ...
 $ Embedded_Options_Available: chr [1:2283] "No" "No" "No" "No" ...
```

3.4 Làm sạch dữ liệu

3.4.1 Các hàm hỗ trợ

```
1 # Select relevant columns from dataset
2 select_columns <- function(dataset, columns) {
3   return(dataset[, columns])
4 }
5 # Check for missing data
6 is_missing <- function(x) {
7   is.na(x) | x == "" | toupper(x) == "N/A"
8 }
9 # Filter strings from non-standard characters (ASCII)
10 clean_text_data <- function(text_vector) {
11   sapply(text_vector, function(text) {
12     if (is_missing(text)) return(NA)
13     else return(str_replace_all(text, "[^[:alnum:]]", " ") %>% str_squish
14       ↪ ())
15   })
16 }
17 # Convert the unit into standard
18 convert_unit <- function(x, base_unit = "MB") {
19   if (is_missing(x)) return(NA)
20   num <- as.numeric(str_extract(x, "[0-9.]+"))
21   unit <- str_to_upper(str_extract(x, regex("([kKmMgGtT][bB]|[mMgG][hH][
22     ↪ zZ]|[gG][tT]/[sS])", ignore_case = TRUE)))
23   if (is.na(num)) return(NA)
24   unit <- case_when(
25     unit %in% c("GHZ", "Ghz") ~ "GHZ",
26     unit %in% c("MHZ", "Mhz") ~ "MHZ",
27     unit %in% c("GT/S", "Gt/s") ~ "GT/s",
28     TRUE ~ unit
29   )
30   conversion_table <- list(
31     "MB" = list("KB" = 1/1024, "MB" = 1, "GB" = 1024, "TB" = 1024^2),
32     "GHZ" = list("MHZ" = 1/1000, "GHZ" = 1, "GT/s" = 1),
33     "MHZ" = list("MHZ" = 1, "GHZ" = 1000)
34   )
35   if (base_unit %in% names(conversion_table) && unit %in% names(
36     ↪ conversion_table[[base_unit]])) {
```

```
34   factor <- conversion_table[[base_unit]][[unit]]
35   return(round(num * factor, 2))
36 }
37 return(NA)
38 }
39 # Extract numeric value
40 extract_numeric <- function(x) {
41   if (is_missing(x)) return(NA)
42   num <- as.numeric(str_extract(x, "[0-9.]+"))
43   return(num)
44 }
45 # Normalize the format of data in Cache
46 normalize_cache <- function(cache_str) {
47   if (is_missing(cache_str)) return(list(Cache_Normalized = NA, Cache_
48     ↪ Value_MB = NA, Cache_Type = NA))
49   num <- convert_unit(cache_str, base_unit = "MB")
50   type <- str_extract(cache_str, "(SmartCache|L2|L3|Last Level Cache)")
51   result <- paste0(num, " MB")
52   if (!is.na(type)) result <- paste(result, type)
53   return(list(Cache_Normalized = result, Cache_Value_MB = num, Cache_Type
54     ↪ = type))
55 }
```

- `is_missing(x)`: Kiểm tra giá trị thiếu trong dữ liệu. Một giá trị được xem là thiếu nếu nó là NA, chuỗi rỗng hoặc bằng "N/A" (không phân biệt chữ hoa/thường).
- `clean_text_data(text_vector)`: Làm sạch dữ liệu văn bản bằng cách loại bỏ các ký tự không phải chữ và số, sau đó chuẩn hoá lại chuỗi.
- `convert_unit(x, base_unit)`: Trích xuất giá trị số và đơn vị từ chuỗi, sau đó quy đổi về đơn vị chuẩn (mặc định là MB).
- `extract_numeric(x)`: Trích xuất và chuyển đổi giá trị số đầu tiên trong chuỗi ký tự thành kiểu số thực. Dành cho những biến không cần quy đổi dữ liệu
- `normalize_cache(cache_str)`: Chuẩn hoá thông tin về bộ nhớ đệm (Cache), bao gồm:
 - Giá trị dung lượng sau khi quy đổi về MB.
 - Kiểu bộ nhớ đệm (L2, L3, SmartCache, Last Level Cache).

3.4.2 Hàm làm sạch dữ liệu

```
1 # Clean raw data and create new standardized columns
2 clean_data <- function(data) {
3   data <- data %>%
4     mutate(across(where(is.character), ~na_if(., "N/A")),
5             across(where(is.character), ~na_if(., ""))) %>%
6     mutate(
7       Product_Collection = if ("Product_Collection" %in% names(.)) {
8         clean_text_data(Product_Collection)
9       } else Product_Collection,
10 }
```

```
11 cache_data = map(Cache, normalize_cache),
12 Cache = map_chr(cache_data, "Cache_Normalized"),
13 Cache_Value_MB = map_dbl(cache_data, "Cache_Value_MB"),
14 Cache_Type = map_chr(cache_data, "Cache_Type"),
15
16 TDP = sapply(TDP, extract_numeric),
17 Lithography = sapply(Lithography, extract_numeric),
18 Max_Memory_Bandwidth = sapply(Max_Memory_Bandwidth, extract_numeric)
19 ↪ ,
20
21 Launch_Quarter = str_extract(Launch_Date, "Q[1-4]"),
22 Launch_Year = ifelse(!is.na(str_extract(Launch_Date, "\\d{2}$")),
23 ↪ paste0("20", str_extract(Launch_Date, "\\d{2}$"),
24 ↪ NA),
25 Launch_Year = as.numeric(Launch_Year),
26 Processor_Base_Frequency = sapply(Processor_Base_Frequency, convert
27 ↪ _unit, base_unit = "GHz")
28 ) %>%
29 filter(is_missing(Launch_Year) | Launch_Year <= 2025) %>% # filter
30 ↪ by the constraint that the years are smaller or equal than 2025
31 select(-cache_data)
32 desired_columns <- c(
33 "Product_Collection", "Vertical_Segment", "Launch_Date", "Launch_Year
34 ↪ ", "Launch_Quarter", "Lithography",
35 "nb_of_Cores", "nb_of_Threads", "Processor_Base_Frequency", "Cache_
36 ↪ Value_MB",
37 "Cache_Type", "TDP", "Max_Memory_Bandwidth", "Embedded_Options_
38 ↪ Available"
39 )
40 existing_columns <- desired_columns[desired_columns %in% names(data)]
41 data <- data %>% select(all_of(existing_columns))
42 return(data)
43 }
```

- Chuẩn hoá các trường văn bản: thay thế giá trị "N/A" và chuỗi rỗng bằng NA.
- Làm sạch và chuẩn hoá trường Product_Collection.
- Tách thông tin từ cột Cache thành hai phần: Cache_Value_MB, và Cache_Type.
- Trích xuất giá trị số từ các trường định lượng như TDP, Lithography, và Max_Memory_Bandwidth.
- Trích xuất quý (Q1 đến Q4) (Launch_Quarter) và năm (Launch_Year) từ trường Launch_Date. Chỉ lọc những giá trị từ năm 2025 trở về trước để đảm bảo tính hợp lý, từ đó giúp thống kê chính xác hơn.
- Chuyển đổi đơn vị của các biến về đơn vị chuẩn như: Cache_Value_MB (đơn vị chuẩn là MB), Processor_Base_Frequency (đơn vị chuẩn là GHz, ...
- Lọc và sắp xếp lại các cột cần thiết theo thứ tự mong muốn.

```
1 # Clean and transform the data
2 cleaned_data <- clean_data(selected_data)
```

Kết quả thu được là bộ dữ liệu `cleaned_data` đã được chuẩn hóa về mặt giá trị số học, đơn vị và ký tự.

3.5 Kiểm tra số lượng và tỷ lệ dữ liệu bị khuyết

3.5.1 Đưa ra số liệu tổng quan về dữ liệu khuyết

```
1 # Show summary of missing values
2 check_missing <- function(data) {
3   total_rows <- nrow(data)
4   missing_counts <- sapply(data, function(col) sum(is_missing(col)))
5   missing_percent <- round(missing_counts / total_rows * 100, 2)
6
7   return(data.frame(
8     Variable = names(data),
9     Missing_Count = missing_counts,
10    Missing_Percent = missing_percent
11  ) %>% arrange(desc(Missing_Count))) # We arrange the number of missing
12  }
    ↪ values in descending order
```

Output

```
> print(missing_summary)
```

	Variable	Missing_Count	Missing_Percent
Max_Memory_Bandwidth	Max_Memory_Bandwidth	1136	49.76
nb_of_Threads	nb_of_Threads	856	37.49
Launch_Date	Launch_Date	412	18.05
Launch_Year	Launch_Year	412	18.05
Cache_Type	Cache_Type	361	15.81
Lithography	Lithography	71	3.11
TDP	TDP	67	2.93
Processor_Base_Frequency	Processor_Base_Frequency	18	0.79
Cache	Cache	12	0.53
Cache_Value_MB	Cache_Value_MB	12	0.53
Embedded_Options_Available	Embedded_Options_Available	1	0.04
Product_Collection	Product_Collection	0	0.00
Vertical_Segment	Vertical_Segment	0	0.00
nb_of_Cores	nb_of_Cores	0	0.00

Nhận xét.

Dựa trên bảng thống kê tỷ lệ dữ liệu khuyết, có thể nhận thấy rằng một số biến như `Max_Memory_Bandwidth`, `nb_of_Threads` và `Launch_Date` có tỷ lệ thiếu dữ liệu tương đối cao, lần lượt là 49.76%, 37.49% và 18.05%. Tuy nhiên, phần lớn các biến còn lại có tỷ lệ khuyết không đáng kể, thậm chí một số biến hoàn toàn đầy đủ.

Do phần lớn dữ liệu bị khuyết không quá lớn (ngoại trừ một vài biến), nhóm quyết định áp dụng phương pháp:

- Thay thế giá trị bị khuyết bằng trung bình (mean cho các biến định lượng).
- Thay thế bằng **giá trị xuất hiện nhiều nhất (mode)** cho các biến định tính (biến phân loại).

Phương pháp mà nhóm tác giả đã chọn giúp giảm thiểu tổn thất thông tin do việc loại bỏ các quan sát chứa dữ liệu khuyết, đồng thời hạn chế sai lệch tiềm ẩn do sự không đồng đều trong phân bố dữ liệu sau khi loại bỏ.

3.5.2 Thay thế dữ liệu bị khuyết

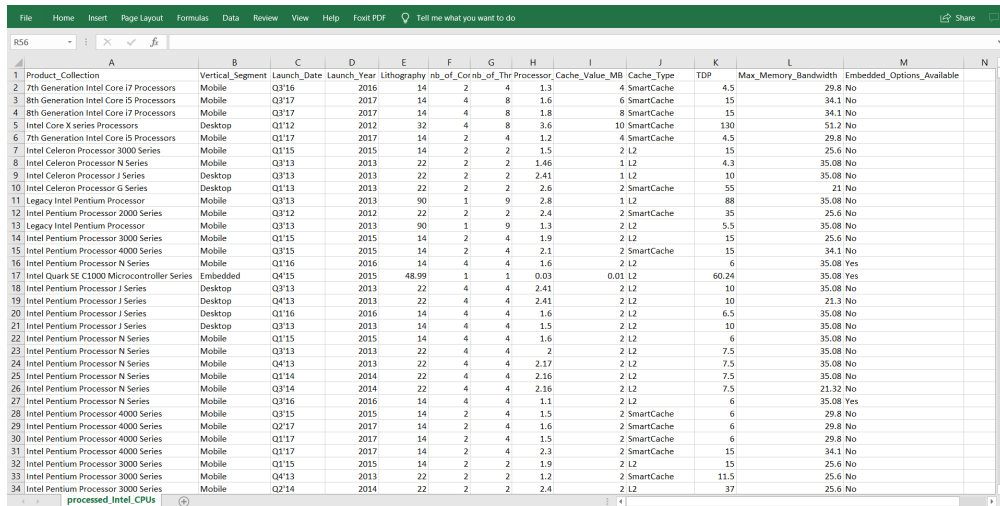
```
1 # Impute missing values: mean for numeric, mode for categorical
2 impute_missing <- function(data) {
3   integer_cols <- c("Launch_Year", "nb_of_Cores", "nb_of_Threads")
4   numeric_cols <- names(data)[sapply(data, is.numeric)]
5
6   for (col in numeric_cols) {
7     missing_idx <- is_missing(data[[col]])
8     if (any(missing_idx)) {
9       mean_val <- mean(data[[col]][!missing_idx], na.rm = TRUE)
10      if (col %in% integer_cols)
11        data[[col]][missing_idx] <- round(mean_val)
12      else
13        data[[col]][missing_idx] <- round(mean_val, 2)
14    }
15  }
16  char_cols <- names(data)[sapply(data, is.character)]
17  for (col in char_cols) {
18    missing_idx <- is_missing(data[[col]])
19    if (any(missing_idx)) {
20      non_missing <- data[[col]][!missing_idx]
21      mode_val <- names(which.max(table(non_missing)))
22      data[[col]][missing_idx] <- mode_val
23    }
24  }
25  return(data)
26 }
27 # Impute missing values if flag is TRUE
28 perform_imputation = TRUE
29 final_data <- if (perform_imputation) {
30   impute_missing(cleaned_data)
31 } else
32   cleaned_data
```

Đoạn mã trên định nghĩa một hàm `impute_missing` dùng để thay thế các giá trị bị thiếu trong dữ liệu:

- Với các cột kiểu số (`numeric`):
 - Nếu cột là số nguyên (ở đây nhóm tác giả lấy đặc trưng là dữ liệu ở 3 cột `Launch_Year`, `nb_of_Cores`, `nb_of_Threads`), các giá trị bị khuyết sẽ được thay bằng giá trị trung bình (mean), sau đó làm tròn.
 - Nếu là số thực, giá trị bị khuyết sẽ được thay bằng trung bình và làm tròn đến 2 chữ số thập phân.
- Với các cột kiểu ký tự (`character`): giá trị bị thiếu được thay bằng giá trị xuất hiện nhiều nhất (mode).
- Cuối cùng, nếu biến `perform_imputation` có giá trị `TRUE`, dữ liệu sau xử lý sẽ được gán vào `final_data`, ngược lại sử dụng dữ liệu gốc `cleaned_data`.

3.6 Kiểm thử dữ liệu sau khi tiền xử lý

Sau khi thực hiện các bước làm sạch và thay thế dữ liệu bị khuyết, nhóm tác giả muốn xuất ra tệp CSV để phục vụ cho việc kiểm thử kết quả một cách cẩn thận và rõ ràng, trực quan hơn khi dùng ứng dụng *Microsoft Excel*.



Product_Collection	Vertical_Segment	Launch_Date	Launch_Year	Lithography	nb_of_Cores	nb_of_Thr	Processor	Cache_Value_MB	Cache_Type	TDP	Max_Memory_Bandwidth	Embedded_Options_Available
7th Generation Intel Core i7 Processors	Mobile	Q3'16	2016	14	2	4	1.3	4	SmartCache	4.5	29.8	No
8th Generation Intel Core i5 Processors	Mobile	Q3'17	2017	14	4	8	1.6	6	SmartCache	15	34.1	No
8th Generation Intel Core i7 Processors	Mobile	Q3'17	2017	14	4	8	1.8	8	SmartCache	15	34.1	No
Intel Core X series Processors	Desktop	Q1'12	2012	32	4	8	3.6	10	SmartCache	130	51.2	No
7th Generation Intel Core i5 Processors	Mobile	Q1'17	2017	14	2	4	1.2	4	SmartCache	4.5	29.8	No
Intel Celeron Processor 3000 Series	Mobile	Q1'15	2015	14	2	2	1.5	2	L2	15	25.6	No
Intel Celeron Processor N Series	Mobile	Q3'13	2013	22	2	2	1.46	1	L2	4.3	35.08	No
Intel Celeron Processor J Series	Desktop	Q3'13	2013	22	2	2	2.41	1	L2	10	35.08	No
Intel Celeron Processor G Series	Desktop	Q1'13	2013	22	2	2	2.6	2	SmartCache	55	21	No
Legacy Intel Pentium Processor	Mobile	Q3'13	2013	90	1	9	2.8	1	L2	88	35.08	No
Intel Pentium Processor 2000 Series	Mobile	Q3'12	2012	22	2	2	2.4	2	SmartCache	35	25.6	No
Legacy Intel Pentium Processor	Mobile	Q3'13	2013	90	1	9	1.3	2	L2	5.5	35.08	No
Intel Pentium Processor 3000 Series	Mobile	Q1'15	2015	14	2	4	1.9	2	L2	15	25.6	No
Intel Pentium Processor 4000 Series	Mobile	Q3'15	2015	14	2	4	2.1	2	SmartCache	15	34.1	No
Intel Pentium Processor N Series	Mobile	Q1'16	2016	14	4	4	1.6	2	L2	6	35.08	Yes
Intel Quark SE C1000 Microcontroller Series	Embedded	Q4'15	2015	48.99	1	1	0.03	0.01	L2	60.24	35.08	Yes
Intel Pentium Processor J Series	Desktop	Q3'13	2013	22	4	4	2.41	2	L2	10	35.08	No
Intel Pentium Processor J Series	Desktop	Q4'13	2013	22	4	4	2.41	2	L2	10	21.3	No
Intel Pentium Processor J Series	Desktop	Q1'16	2016	14	4	4	1.6	2	L2	6.5	35.08	No
Intel Pentium Processor J Series	Desktop	Q3'13	2013	14	4	4	1.5	2	L2	10	35.08	No
Intel Pentium Processor N Series	Mobile	Q1'15	2015	14	4	4	1.6	2	L2	6	35.08	No
Intel Pentium Processor N Series	Mobile	Q3'13	2013	22	4	4	2	2	L2	7.5	35.08	No
Intel Pentium Processor N Series	Mobile	Q4'13	2013	22	4	4	2.17	2	L2	7.5	35.08	No
Intel Pentium Processor N Series	Mobile	Q1'14	2014	22	4	4	2.16	2	L2	7.5	35.08	No
Intel Pentium Processor N Series	Mobile	Q3'14	2014	22	4	4	2.16	2	L2	7.5	21.32	No
Intel Pentium Processor N Series	Mobile	Q3'16	2016	14	4	4	1.1	2	L2	6	35.08	Yes
Intel Pentium Processor 4000 Series	Mobile	Q3'15	2015	14	2	4	1.5	2	SmartCache	6	29.8	No
Intel Pentium Processor 4000 Series	Mobile	Q2'17	2017	14	2	4	1.6	2	SmartCache	6	29.8	No
Intel Pentium Processor 4000 Series	Mobile	Q1'17	2017	14	2	4	1.5	2	SmartCache	6	29.8	No
Intel Pentium Processor 4000 Series	Mobile	Q1'17	2017	14	2	4	2.3	2	SmartCache	15	34.1	No
Intel Pentium Processor 3000 Series	Mobile	Q1'15	2015	14	2	2	1.9	2	L2	15	25.6	No
Intel Pentium Processor 3000 Series	Mobile	Q4'13	2013	22	2	2	1.2	2	SmartCache	11.5	25.6	No
Intel Pentium Processor 3000 Series	Mobile	Q2'14	2014	22	2	2	2.4	2	L2	37	25.6	No

Ảnh 1: Minh họa dữ liệu sau khi tiền xử lý được xuất ra tệp `processed_Intel_CPUs.csv` và đọc dữ liệu bằng phần mềm *Microsoft Excel*

Nhận xét:

Nhìn chung, tập dữ liệu sau khi được tiền xử lý đã đáp ứng đầy đủ các yêu cầu về mặt cấu trúc và chất lượng. Các giá trị bị thiếu đã được xử lý một cách hợp lý, định dạng dữ liệu được chuẩn hóa, và các đơn vị đo lường đã được chuyển đổi thống nhất.

Tiền xử lý dữ liệu không chỉ giúp nâng cao độ tin cậy của kết quả phân tích, mà còn đảm bảo tính minh bạch và khả năng tái sử dụng trong các nghiên cứu của nhóm sau này.

4 Thống kê mô tả

4.1 Tính toán thống kê mẫu

Thực hiện tính toán thống kê các chỉ số min, max, trung bình, trung vị, tứ phân vị thứ nhất, tứ phân vị thứ ba, độ lệch chuẩn, độ nhọn, độ lệch bằng các lệnh sau:

```
1 for (column_name in names(columns)) {
2   column_data <- final_data[[column_name]]
3
4   if (!is.numeric(column_data) || length(na.omit(column_data)) < 2) {
5     cat("Skipping:", column_name, "- Not enough numeric data\n")
6     next
7   }
8
9   binwidth <- (max(column_data, na.rm = TRUE) - min(column_data, na.rm =
10  TRUE)) / 10
```



```

11 p <- ggplot(final_data, aes_string(x = column_name)) +
12   geom_histogram(binwidth = binwidth, fill = "skyblue", color = "black"
13   ↪ ) +
14   stat_bin(binwidth = binwidth, geom = "text",
15   ↪ aes(label = ..count.., y = ..count..), vjust = -0.5) +
16   labs(
17     title = paste("Histogram of", column_name),
18     x = columns[[column_name]],
19     y = "Frequency"
20   ) +
21   theme_minimal() +
22   theme(plot.title = element_text(hjust = 0.5, face = "bold"))
23
24 print(p)
25 }
26
27 rm(binwidth, column_data, p, column_name)

```

Column	Unit	Min	Mean	Max	Q1	Median	Q3	SD	Skewness	Kurtosis
Lithography	nm	14.0	49.0	250.0	22.0	32.0	65.0	44.6	2.0	4.1
nb_of_Cores	Core	1.0	4.1	72.0	1.0	2.0	4.0	6.3	6.5	54.6
nb_of_Threads	Thread	1.0	8.8	56.0	4.0	9.0	9.0	7.2	2.9	11.5
Processor_Base_Frequency	GHz	0.0	2.2	4.3	1.7	2.2	2.8	0.8	-0.3	-0.4
Cache_Value_MB	MB	0.0	7.1	60.0	2.0	3.0	8.0	9.2	2.4	6.3
TDP	W	0.0	60.2	300.0	26.8	51.0	84.0	44.2	1.2	2.2
Max_Memory_Bandwidth	GB/s	1.6	35.1	352.0	25.6	35.1	35.1	22.1	8.6	110.2

Ảnh 2: Bảng kết quả tính toán các thống kê mô tả cho các biến định lượng

Nhận xét:

- Lithography (nm): Lithography (nm): Giá trị trung bình là 49.0 nm, cho thấy nhiều CPU vẫn đang sử dụng công nghệ sản xuất với kích thước bóng bán dẫn khá lớn. Tuy nhiên, giá trị thấp nhất là 14.0 nm và cao nhất lên tới 250.0 nm, cho thấy có sự khác biệt rất lớn giữa các CPU, với một số sử dụng công nghệ rất cũ hoặc có kiến trúc đặc biệt. Độ lệch chuẩn (SD) là 44.6, skewness là 2.0 và kurtosis là 4.1, cho thấy dữ liệu bị lệch phải và có vài giá trị ngoại lai lớn.
- nb_of_Cores và nb_of_Threads: SD khá cao (6.3 và 7.2), và skewness lớn (6.5 và 2.9), phản ánh sự phân bố không đều – một số CPU có rất nhiều lõi/luồng. Q1, Median và Q3 của nb_of_Cores là 1.0, 2.0 và 4.0, cho thấy đa số CPU có ít lõi, nhưng có một số ít mẫu có giá trị rất cao (tối đa 72 Core), làm tăng độ lệch.
- Processor_Base_Frequency (GHz): Có giá trị từ 0.0 đến 4.3 GHz, với trung vị là 2.2 GHz và độ lệch chuẩn thấp (0.8), cho thấy tần số cơ bản của CPU khá đồng đều. Skewness là -0.3, cho thấy phân phối hơi lệch trái, nhưng gần như đối xứng.
- TDP (W): Có độ lệch chuẩn lớn (44.2) và giá trị từ 0 đến 300 W, cho thấy mức độ tiêu thụ điện năng giữa các CPU rất không đồng nhất. Skewness là 1.2, tức phân bố lệch phải. Điều này phù hợp với thực tế là có một số CPU hiệu năng cao tiêu thụ điện năng lớn.

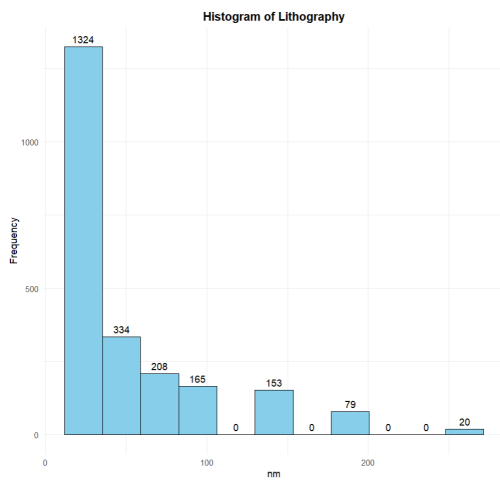
- Max_Memory_Bandwidth (GB/s): Với khoảng giá trị từ 1.6 đến 352.0, trung bình 35.1, SD 22.1, skewness lên tới 8.6, kurtosis rất cao 110.2, phản ánh sự phân bố rất lệch phải và có nhiều giá trị ngoại lệ lớn – có thể là các CPU chuyên dụng hoặc thể hệ mới.
- Độ phân tán dữ liệu: TDP và Lithography có độ lệch chuẩn cao, phản ánh sự khác biệt lớn giữa các dòng CPU. Ngược lại, Processor_Base_Frequency có SD nhỏ, cho thấy đặc điểm này tương đối đồng nhất giữa các mẫu. Các chỉ số skewness và kurtosis cho thấy phân phối không chuẩn ở nhiều thuộc tính, đặc biệt là nb_of_Cores, Max_Memory_Bandwidth.
- Ý nghĩa ứng dụng: Những thống kê này giúp phân loại và so sánh hiệu năng CPU, hỗ trợ chọn lựa phù hợp với nhu cầu: tiết kiệm điện, hiệu năng cao, hoặc cân bằng chi phí-hiệu suất. Ngoài ra, các đặc điểm thống kê có thể dùng để xây dựng mô hình dự đoán hiệu năng, hỗ trợ ra quyết định trong thiết kế hệ thống, tối ưu hóa chi phí hoặc sản xuất hàng loạt CPU theo từng phân khúc thị trường.

4.2 Mô tả dữ liệu bằng đồ thị

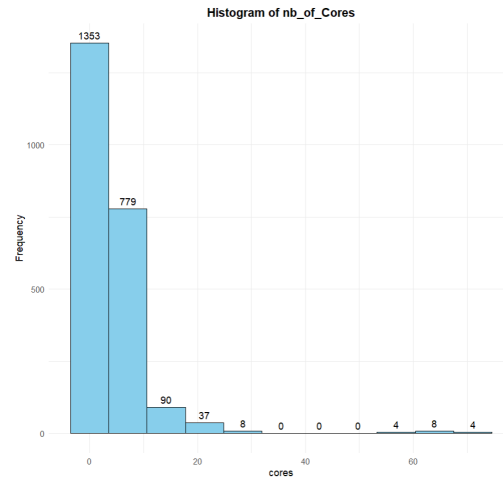
4.2.1 Đồ thị histogram

Vẽ các đồ thị histogram cho các biến định lượng tương ứng.

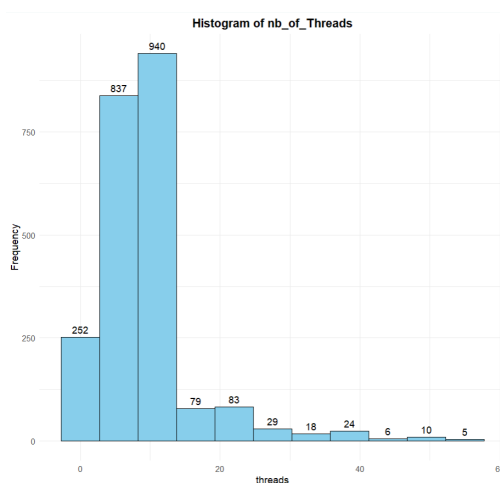
```
1 for (column_name in names(columns)) {  
2   column_data <- final_data[[column_name]]  
3   binwidth <- (max(column_data, na.rm = TRUE) - min(column_data, na.rm =  
4     ↪ TRUE)) / 10  
5  
6   p <- ggplot(final_data, aes_string(x = column_name)) +  
7     geom_histogram(binwidth = binwidth, fill = "skyblue", color = "black"  
8     ↪ ) +  
9     stat_bin(binwidth = binwidth, geom = "text", aes(label = ..count.., y  
10    ↪ = ..count..), vjust = -0.5) +  
11     labs(  
12       title = paste("Histogram of", column_name),  
13       x = columns[[column_name]],  
14       y = "Frequency"  
15     ) +  
16     theme_minimal() +  
17     theme(plot.title = element_text(hjust = 0.5, face = "bold"))  
18  
19   dev.new()  
20   print(p)  
21 }  
22 rm(binwidth, column_data, p)
```



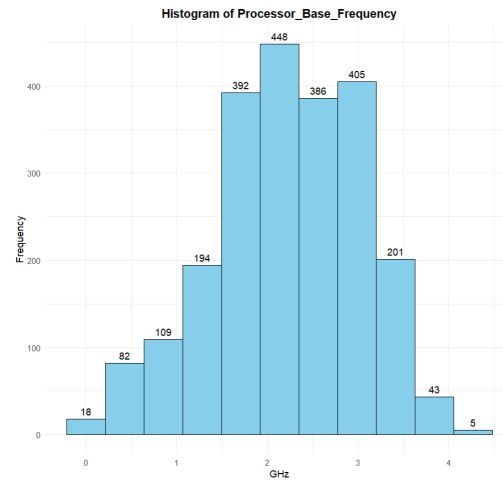
Ảnh 3: Đồ thị histogram của biến Lithography



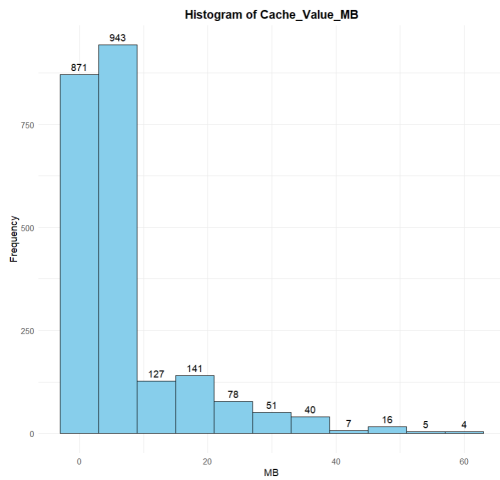
Ảnh 4: Đồ thị histogram của biến nb_of_Cores



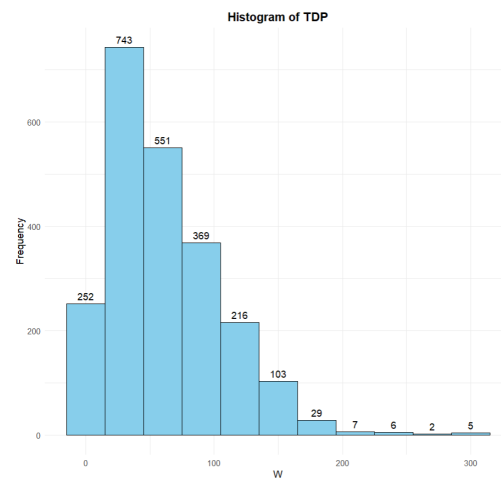
Ảnh 5: Đồ thị histogram của biến nb_of_Threads



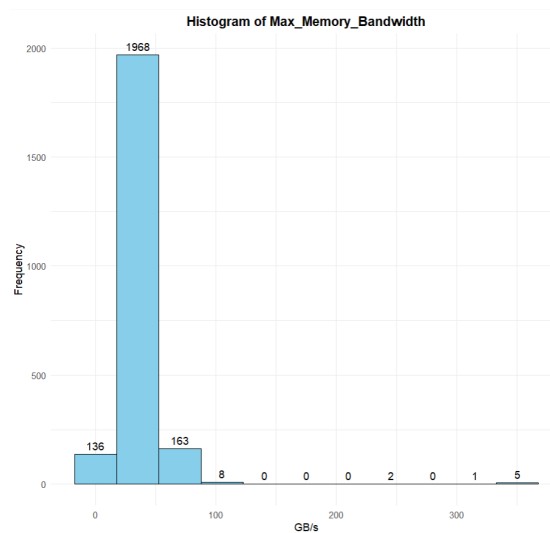
Ảnh 6: Đồ thị histogram của biến Processor_Base_Frequency



Ảnh 7: Đồ thị histogram của biến Cache



Ảnh 8: Đồ thị histogram của biến TDP



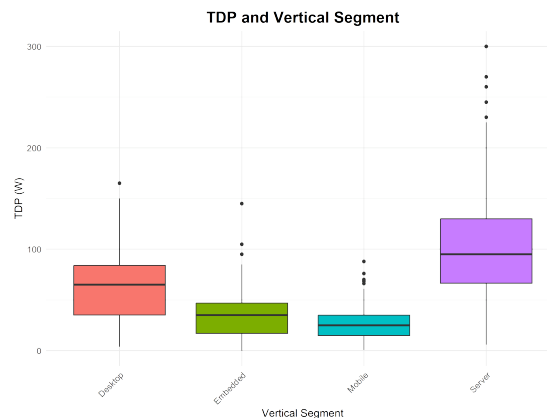
Ảnh 9: Đồ thị histogram của biến Max_Memory_Bandwidth

4.2.2 Đồ thị boxplot thể hiện phân phối

TDP và Vertical Segment

```
1 plot1 <- ggplot(final_data, aes(x = Vertical_Segment, y = TDP, fill =
2   ↪ Vertical_Segment)) +
3   geom_boxplot() +
4   theme_minimal() +
5   labs(title = "TDP and Vertical Segment",
6         x = "Vertical Segment",
7         y = "TDP (W)",
```

```
7     fill = "Vertical Segment") +  
8     theme(  
9         plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),  
10        axis.text.x = element_text(angle = 45, hjust = 1),  
11        legend.position = "none"  
12    )  
13  
14    print(plot1)
```



Ảnh 10: Đồ thị boxplot giữa TDP với Vertical Segment

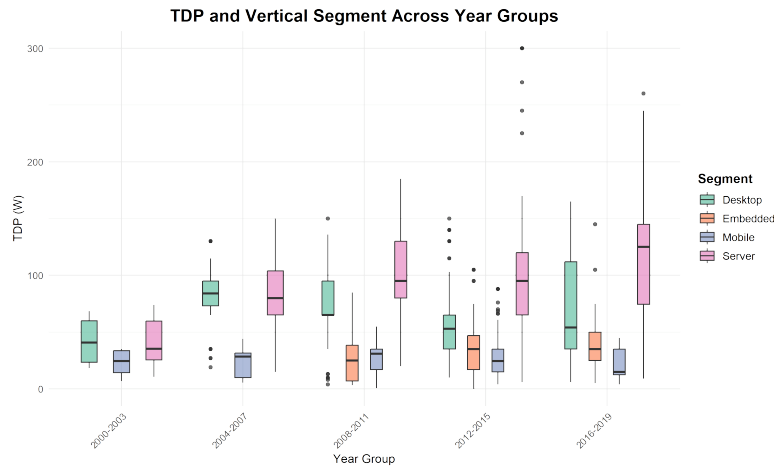
Nhận xét:

- **Phân bố giá trị TDP khác nhau giữa các phân khúc:**
 - **Server:** Độ trải IQR khoảng 70W (60W–130W), râu từ khoảng 5W đến 225W. Cho thấy TDP của Server có độ phân tán lớn, nhiều giá trị vượt trội (lên đến 300W). Server có sự đa dạng lớn về tiêu thụ năng lượng, phù hợp với các CPU hiệu năng cao dùng trong trung tâm dữ liệu.
 - **Desktop:** Độ trải IQR khoảng 50W (40W–90W), râu từ khoảng 5W đến 150W, có điểm ngoại lai lên đến 160W.
 - **Mobile:** Độ trải IQR khoảng 25W (15W–40W), râu từ 0W đến 60W, với điểm ngoại lai lên đến 90W. Thể hiện TDP của Mobile rất đồng nhất, phù hợp với mục đích tiết kiệm năng lượng cho thiết bị di động.
 - **Embedded:** Độ trải IQR khoảng 30W (18W–48W), râu từ 0W đến 90W, với điểm ngoại lai lên đến khoảng 150W. TDP của Embedded khá đồng nhất, phù hợp với các thiết bị yêu cầu tiêu thụ năng lượng thấp.
- **Server và Desktop có nhiều giá trị ngoại lai cao, thể hiện ở các điểm vượt ra khỏi phạm vi boxplot:**
 - **Server:** Có nhiều điểm ngoại lai lên đến 300W, cho thấy một số CPU Server có TDP vượt trội hơn phần còn lại, đồng thời phản ánh sự hiện diện của các CPU hiệu năng cao được thiết kế cho các tác vụ nặng, yêu cầu tiêu thụ năng lượng lớn.

- **Desktop:** Có một số điểm ngoại lai lên đến 160W, cho thấy một số CPU Desktop có TDP cao bất thường, có thể là các dòng CPU hiệu năng cao dành cho chơi game hoặc công việc chuyên sâu.
- **Embedded và Mobile:** Cũng có điểm ngoại lai (Embedded lên đến 150W, Mobile lên đến 90W), nhưng ít hơn và ở mức thấp hơn so với Server và Desktop. Điều này cho thấy sự hiện diện của một số CPU đặc biệt trong các phân khúc này, nhưng không phổ biến.

TDP và Vertical Segment theo nhóm năm

```
1 # Create year groups
2 data_cln <- final_data %>%
3   mutate(Year_Group = case_when(
4     Launch_Year >= 2000 & Launch_Year <= 2003 ~ "2000-2003",
5     Launch_Year >= 2004 & Launch_Year <= 2007 ~ "2004-2007",
6     Launch_Year >= 2008 & Launch_Year <= 2011 ~ "2008-2011",
7     Launch_Year >= 2012 & Launch_Year <= 2015 ~ "2012-2015",
8     Launch_Year >= 2016 & Launch_Year <= 2019 ~ "2016-2019",
9     TRUE ~ "Other"
10  ))
11
12 # Filter year groups
13 year_segment_counts <- data_cln %>%
14   group_by(Year_Group, Vertical_Segment) %>%
15   summarise(count = n(), .groups = "drop") %>%
16   filter(count >= 5)
17
18 # Filter the original dataset
19 data_filtered <- data_cln %>%
20   semi_join(year_segment_counts, by = c("Year_Group", "Vertical_Segment")
21   ↪ )
22
23 plot2 <- ggplot(data_filtered, aes(x = Year_Group, y = TDP, fill =
24   ↪ Vertical_Segment)) +
25   geom_boxplot(position = position_dodge(width = 0.8), width = 0.4, alpha
26   ↪ = 0.7) +
27   scale_fill_brewer(palette = "Set2", name = "Segment") +
28   theme_minimal() +
29   labs(
30     title = "TDP and Vertical Segment Across Year Groups",
31     x = "Year Group",
32     y = "TDP (W)"
33   ) +
34   theme(
35     plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
36     axis.text.x = element_text(angle = 45, hjust = 1),
37     legend.position = "right",
38     legend.title = element_text(size = 12, face = "bold"),
39     legend.text = element_text(size = 10)
40   )
41
42 print(plot2)
```



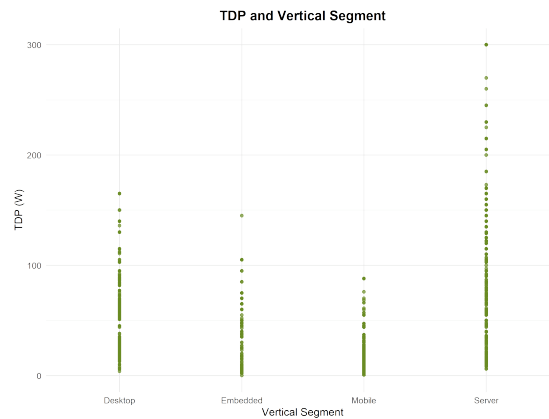
Ảnh 11: Đồ thị boxplot giữa TDP với Vertical Segment theo nhóm năm

Server có TDP cao nhất, biến thiên lớn, với giá trị ngoại lệ đạt mức tối đa 300W vào 2012-2015, thể hiện xu hướng tăng công suất. Desktop có TDP tăng nhẹ qua thời gian, kèm một số giá trị ngoại lệ. Mobile và Embedded duy trì TDP thấp (dưới 100W), ổn định nhất. Xu hướng này phản ánh sự phát triển công nghệ, với Server tập trung hiệu suất cao, còn các phân khúc khác ưu tiên tiết kiệm năng lượng.

4.2.3 Đồ thị scatter thể hiện phân tán giữa TDP và các biến

```
1 create_scatter_plot <- function(data, x_var, x_label, filename) {  
2   p <- ggplot(data, aes(x = .data[[x_var]], y = TDP)) +  
3     geom_point( color = "olivedrab4", alpha = 0.7) +  
4     theme_minimal() +  
5     labs(  
6       title = paste("TDP and", x_label),  
7       x = x_label,  
8       y = "TDP (W)"  
9     ) +  
10    theme(  
11      plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),  
12      legend.position = "none"  
13    )  
14    print(p)  
15  }
```

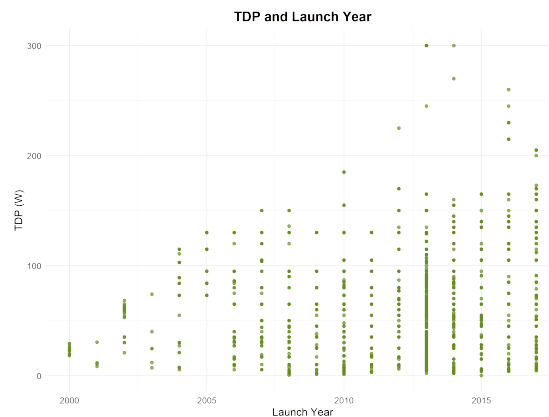
```
1 create_scatter_plot(final_data, "Vertical_Segment", "Vertical_Segment")
```



Ảnh 12: Đồ thị Scatter của TDP với Vertical Segment

Biểu đồ phân tán cho thấy sự phân tán TDP khác biệt rõ rệt giữa các phân khúc. Server và Desktop có TDP cao, phân tán rộng (lên đến 300W), phản ánh yêu cầu hiệu năng lớn. Ngược lại, Embedded và Mobile tập trung ở mức TDP thấp (dưới 100W), phù hợp với mục tiêu tiết kiệm năng lượng của các phân khúc này.

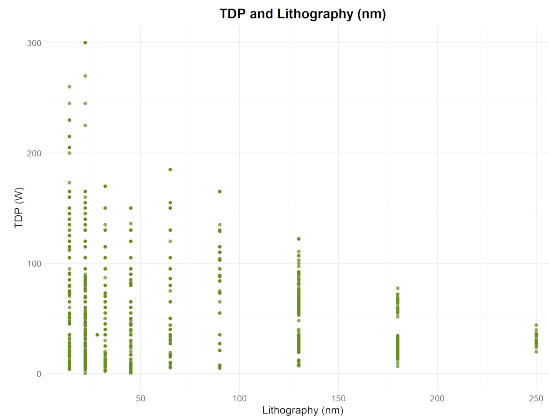
```
1 create_scatter_plot(final_data, "Launch_Year", "Launch Year")
```



Ảnh 13: Đồ thị Scatter của TDP với Launch Year

TDP của các thiết bị có xu hướng tăng theo thời gian, đặc biệt sau năm 2010. Các thiết bị ra mắt trong giai đoạn đầu (2000–2005) có TDP thấp và ít biến động, trong khi các thiết bị mới hơn (sau 2010) có TDP phân tán rộng với nhiều thiết bị có TDP cao (lên đến 300W).

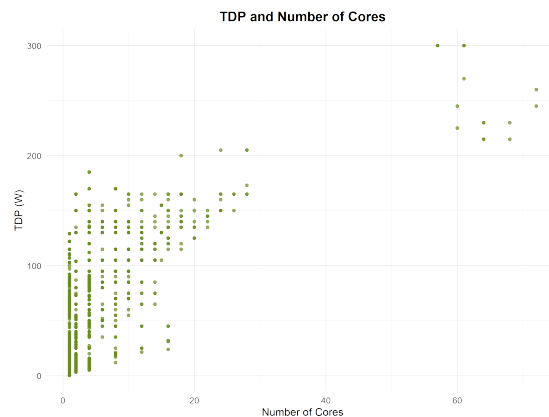
```
1 create_scatter_plot(final_data, "Lithography", "Lithography (nm)")
```

Ảnh 14: Đồ thị Scatter của TDP với Lithography

Các thiết bị sử dụng công nghệ sản xuất nhỏ (dưới 50 nm) có TDP phân tán rộng (0W-300 W), phản ánh sự đa dạng về hiệu năng và mức tiêu thụ năng lượng trong các thiết bị hiện đại. Ngược lại, các thiết bị sử dụng công nghệ sản xuất lớn hơn (trên 150 nm) thường có TDP thấp (dưới 100 W) và ít biến động, cho thấy hạn chế về hiệu năng ở các công nghệ cũ.

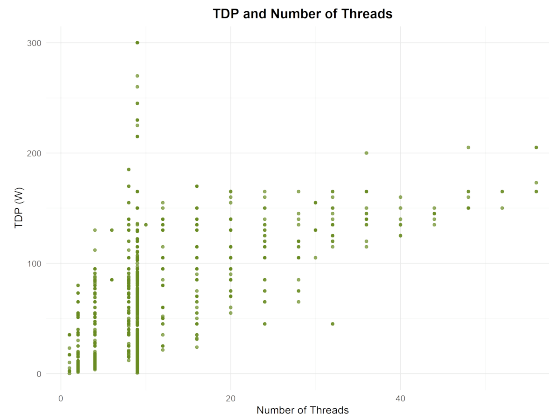
```
1 create_scatter_plot(final_data, "nb_of_Cores", "Number of Cores")
```



Ảnh 15: Đồ thị Scatter của TDP với Number of Cores

Số lượng nhân tăng, TDP có xu hướng tăng nhưng không rõ rệt. Phần lớn các thiết bị có số lượng nhân thấp (dưới 20) tập trung ở mức TDP thấp (0W-100W), trong khi các thiết bị có số lượng nhân lớn hơn (trên 40) có xu hướng tiêu thụ năng lượng cao hơn với TDP vượt 200W. Các giá trị ngoại lai ở mức TDP cao (trên 200W) cho hiệu năng cao không phụ thuộc hoàn toàn vào số lượng nhân.

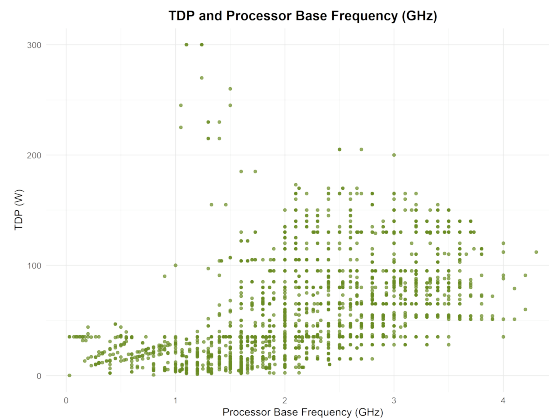
```
1 create_scatter_plot(final_data, "nb_of_Threads", "Number of Threads")
```



Ảnh 16: Đồ thị Scatter của TDP với Number of Threads

Phần lớn các thiết bị có số lượng luồng thấp (dưới 20) tập trung ở mức TDP thấp (0W-100W), trong khi các thiết bị có số lượng luồng lớn hơn (trên 40) có xu hướng tiêu thụ năng lượng cao hơn, với TDP thường vượt quá 100W. Các giá trị ngoại lai ở mức TDP cao (trên 200W) cho thấy hiệu năng cao không phụ thuộc hoàn toàn vào số lượng luồng.

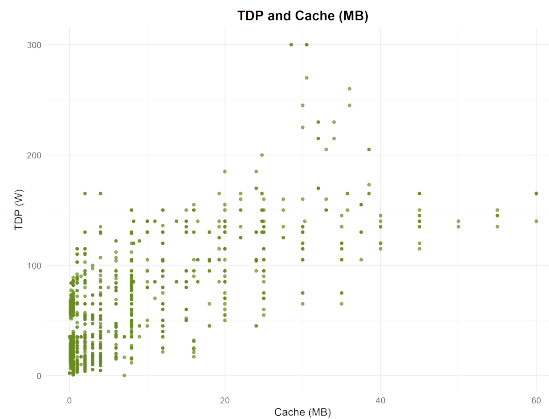
```
1 create_scatter_plot(final_data, "Processor_Base_Frequency", "Processor  
   ↳ Base Frequency (GHz)")
```



Ảnh 17: Đồ thị Scatter của TDP với Processor Base Frequency

Khi tần số xung cơ bản tăng, TDP có xu hướng tăng nhẹ. Phần lớn các thiết bị có tần số cơ bản thấp (dưới 1GHz) tập trung ở mức TDP thấp (0W-50W), trong khi tần số cao hơn (trên 2GHz) có xu hướng tiêu thụ năng lượng cao hơn với TDP từ 50 đến 150W. Sự phân tán của TDP lớn ở mọi mức tần số, đặc biệt ở khoảng 1GHz-3GHz cùng với các giá trị ngoại lai ở mức TDP trên 200W cho thấy hiệu năng cao không phụ thuộc hoàn toàn vào tần số xung cơ bản.

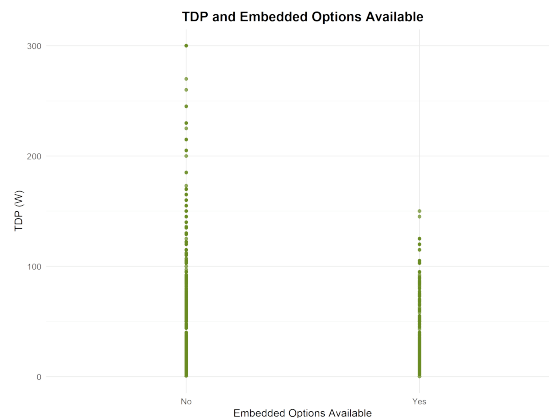
```
1 create_scatter_plot(final_data, "Cache_Value_MB", "Cache (MB)")
```



Ảnh 18: Đồ thị Scatter của TDP với Cache

Các thiết bị có bộ nhớ Cache lớn hơn 40 MB thường có TDP cao hơn, nhưng sự phân tán của TDP lớn ở mọi mức Cache. Phần lớn các thiết bị có Cache nhỏ (dưới 20MB) có TDP thấp (0W-150W), trong khi các thiết bị với Cache lớn hơn có xu hướng tiêu thụ năng lượng cao hơn (trên 100W). Các giá trị ngoại lai ở mức TDP cao (trên 200W) cho thấy hiệu năng cao không phụ thuộc hoàn toàn vào kích thước Cache.

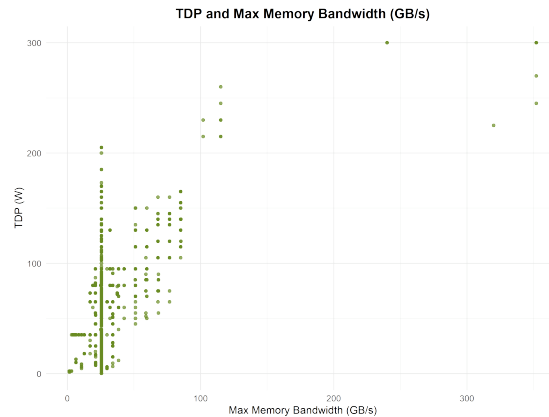
```
1 create_scatter_plot(final_data, "Embedded_Options_Available", "Embedded  
  ↳ Options Available")
```



Ảnh 19: Đồ thị Scatter của TDP với Embedded Options Availale

Có sự khác biệt rõ rệt về mức tiêu thụ năng lượng giữa hai nhóm: nhóm không có tùy chọn nhúng có TDP phân tán rộng (0W-300W), với nhiều giá trị ngoại lai ở mức cao (trên 200W), phản ánh sự đa dạng về hiệu năng. Ngược lại, nhóm có tùy chọn nhúng tập trung ở mức TDP thấp (0W-100W), với rất ít điểm vượt 100W, cho thấy chúng thường được thiết kế tiết kiệm năng lượng, phù hợp với phân khúc Embedded.

```
1 create_scatter_plot(final_data, "Max_Memory_Bandwidth", "Max Memory  
  ↳ Bandwidth (GB/s)")
```



Ảnh 20: Đồ thị Scatter của TDP với Max Memory Banwidth

Các thiết bị có băng thông bộ nhớ tối đa cao hơn (trên 200GB/s) thường có TDP lớn hơn (100W-200W), nhưng sự phân tán của TDP lớn ở mọi mức băng thông. Phần lớn các thiết bị với băng thông thấp (dưới 100GB/s) có TDP thấp (0W-100W), trong khi các thiết bị với băng thông cao hơn có xu hướng tiêu thụ năng lượng lớn hơn. Các giá trị ngoại lai ở mức TDP cao (hơn 200W) cho thấy hiệu năng cao không phụ thuộc hoàn toàn vào băng thông bộ nhớ.

4.3 Kiểm tra phân phối của các biến định lượng

```
1 for(column_name in names(columns)){
2
3 X<-final_data[[column_name]]
4 fit_norm<-fitdist(X,"norm")
5 aic_norm<-AIC(fit_norm)
6 fit_lognorm<-fitdist(X,"lnorm")
7 aic_lognorm<-AIC(fit_lognorm)
8 fit_exp<-fitdist(X,"exp")
9 aic_exp<-AIC(fit_exp)
10 if(all(X ==as.integer(X)&X>=0)){
11 fit_pois <-fitdist(X,"pois")
12 aic_pois <-AIC(fit_pois)
13 }else{
14 fit_pois =NULL
15 aic_pois <-Inf
16 }
17 fit_gamma<-fitdist(X,"gamma")
18 aic_gamma<-AIC(fit_gamma)
19 aic_values<-data.frame(
20 Distribution=c("Normal","Log-Normal","Exponential","Poisson","Gamma"),
21 AIC=c(aic_norm,aic_lognorm,aic_exp,aic_pois,aic_gamma)
22 )
23 best_fit <-aic_values[which.min(aic_values$AIC),]
24 tmp<-paste("Bestfitdistributionof",column_name,"is",best_fit$Distribution
25           ↪ )
26 print(tmp)
27 }
```

```
27 rm(X,tmp,fit_norm,aic_norm,fit_lognorm,aic_lognorm,fit_exp,aic_exp,  
28 fit_pois,aic_pois,fit_gamma,aic_gamma,best_fit,aic_values)
```

```
[1] "Bestfitdistributionof Lithography is Log-Normal"  
[1] "Bestfitdistributionof nb_of_Cores is Log-Normal"  
[1] "Bestfitdistributionof nb_of_Threads is Log-Normal"  
[1] "Bestfitdistributionof Processor_Base_Frequency is Normal"  
[1] "Bestfitdistributionof Cache_Value_MB is Log-Normal"  
[1] "Bestfitdistributionof TDP is Gamma"  
[1] "Bestfitdistributionof Max_Memory_Bandwidth is Log-Normal"
```

Ảnh 21: Phân phối của các biến định lượng

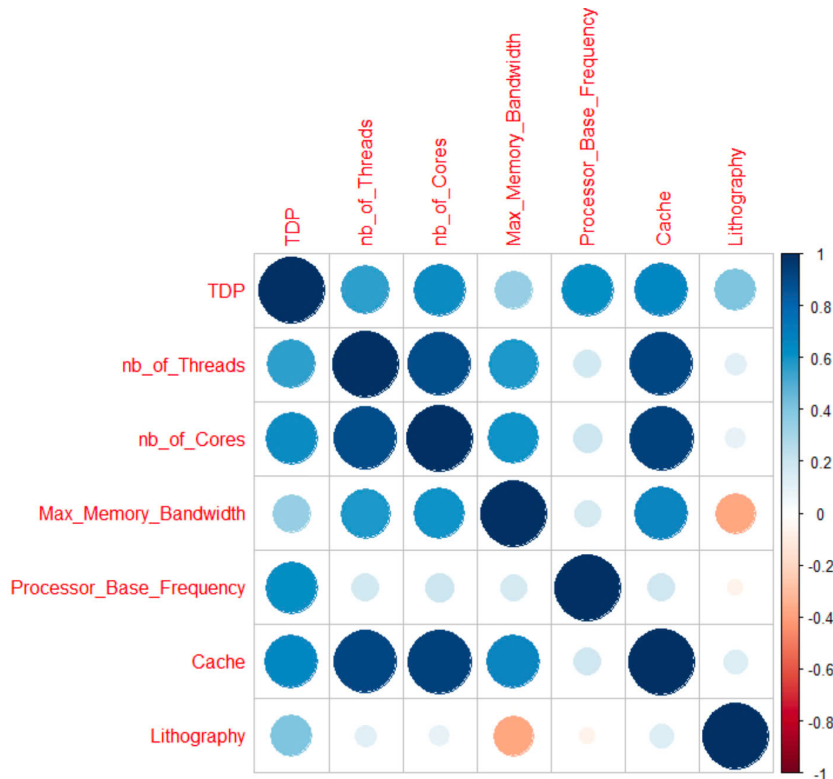
Nhận xét:

- Phân phối Log-Normal (Lithography, nb_of_Cores, nb_of_Threads, Cache, Max_Memory_Bandwidth): Đặc trưng bởi việc các giá trị bị giới hạn ở phía âm (không có giá trị âm) và có đuôi dài ở phía dương. Thực tế, Lithography với công nghệ mới thường tập trung quanh các giá trị nhỏ, trong khi nb_of_Cores, nb_of_Threads và Cache tăng mạnh ở các CPU hiệu năng cao, có đuôi dài về phía dương.
- Phân phối Normal – Phân phối chuẩn (Processor_Base_Frequency): Giá trị trung tâm chiếm ưu thế và phân phối đối xứng. Điều này là hợp lý vì tần số cơ bản của CPU thường được thiết kế xoay quanh một mức chuẩn cố định, với ít biến động lớn.
- Phân phối Gamma (TDP): Phân phối phù hợp với các biến luôn dương và có sự bất đối xứng rõ ràng, thường có đuôi dài ở phía dương. TDP có phân phối lệch phải rõ (Skewness > 0, SD cao = 29.0), thể hiện sự biến động lớn giữa các CPU từ tiết kiệm năng lượng đến hiệu năng cao.

4.4 Kiểm định tương quan

Vấn đề đặt ra: Xác định sự tương quan giữa các biến dự định sử dụng trong mô hình hồi quy: giữa biến phụ thuộc (TDP) và các biến độc lập (nb_of_Threads, nb_of_Cores, Max_Memory_Bandwidth, Processor_Base_Frequency, Cache, Lithography), cũng như khảo sát sự tương quan giữa các biến độc lập với nhau.

```
1 library(car)  
2 library(corrplot)  
3 cor <- cor(final_data[c("TDP","nb_of_Threads","nb_of_Cores","Max_  
   ↳ Memory_Bandwidth","Processor_Base_Frequency","Cache","  
   ↳ Lithography")])  
4 corrplot(cor, method="circle")
```



Ảnh 22: Ma trận tương quan

Nhận xét: Dựa vào kết quả trên ta có thể thấy được biến phụ thuộc TDP có mối tương quan dương với tất cả các biến độc lập. Phần lớn hệ số tương quan giữa TDP và các biến độc lập dao động từ mức 0.3 đến 0.7, mức tương quan dương vừa phải, phù hợp cho việc xây dựng mô hình. Bên cạnh đó, khi xem xét sự tương quan giữa các biến độc lập với nhau, ta có thể thấy được mối tương quan mạnh mẽ giữa các biến nb_of_Cores, nb_of_Threads và Cache (hệ số tương quan Pearson giữa các biến này rất cao, trong khoảng từ 0.89 đến 0.93). Vì vậy, cần loại bỏ 2 trong 3 biến này để tránh hiện tượng đa cộng hưởng xảy ra. Qua quan sát ma trận tương quan, ta thấy TDP có tương quan mạnh với cả 2 biến nb_of_Cores và Cache, nên ta sẽ chọn ngẫu nhiên một trong 2 biến giữ lại cho mô hình. Ở đây, ta sẽ giữ lại biến nb_of_Cores khi xây dựng mô hình.

5 Thống kê suy diễn

5.1 Giới thiệu

TDP (Thermal Design Power) là công suất nhiệt tối đa (tính bằng watt) mà CPU dự kiến sinh ra trong điều kiện hoạt động bình thường. Đây cũng là mức nhiệt mà hệ thống tản nhiệt cần xử lý hiệu quả để đảm bảo bộ vi xử lý vận hành ổn định và bền bỉ.

Giá trị TDP được xác định dựa trên nhiều thông số kỹ thuật và điều kiện hoạt động cụ thể của CPU, bao gồm: tần số xung nhịp, số nhân và luồng, kiến trúc vi xử lý, cùng các yếu tố như điện

áp hoạt động và tiến trình sản xuất. Mỗi hãng sản xuất có cách xác định TDP khác nhau, nên TDP không phản ánh chính xác mức tiêu thụ điện tối đa, mà chỉ mang tính định hướng cho việc thiết kế và lựa chọn giải pháp tản nhiệt phù hợp.

5.2 Bài toán kiểm định một mẫu

Bài 1: Có một bạn cho rằng chỉ số TDP (Thermal Design Power) sẽ không bé hơn 60W. Kiểm định xem nhận định này đúng hay không với mức ý nghĩa 5%.

Ta sử dụng lệnh kiểm định `shapiro.test()` để kiểm tra phân phối chuẩn:

```
1 shapiro.test(df$TDP)
```

Ta thấy rằng giá trị $p\text{-value}$ của kiểm định là $< 2.2 \times 10^{-16}$, bé hơn mức ý nghĩa 5%, nên dữ liệu không có phân phối chuẩn. Vậy bài toán thuộc dạng: Chưa biết σ tổng thể + phân phối bất kỳ.

Output

Shapiro-Wilk normality test

data: df\$TDP

W = 0.91343, p-value < 2.2e-16

Tính toán các đặc trưng:

```
1 n <- length(df$TDP)
2 x <- mean(df$TDP)
3 s <- sd(df$TDP)
4 print(data.frame(n, x, s))
```

Output

```
> print(data.frame(n, x, s))
```

	n	x	s
1	2283	60.24147	44.17117

Đặt giả thuyết kiểm định:

$$\begin{cases} H_0 : \mu \geq 60 \\ H_1 : \mu < 60 \end{cases}$$

Công thức tính thống kê kiểm định z_0 khi chưa biết σ tổng thể (n lớn):

$$z_0 = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$$

Tính toán trong R:

```
1 z0 <- (x - 60) / (s / sqrt(n))  
2 z_alpha <- qnorm(p = 0.05, lower.tail = TRUE)  
3 print(z0)  
4 print(z_alpha)
```

Output

```
z0 = 0.2612  
z_alpha = 1.6449
```

Kết luận:

Ta có:

$$RR = (-\infty, -z_\alpha) = (-\infty, -1.6449)$$

$$z_0 = 0.2612 \notin RR \Rightarrow \text{Không bác bỏ } H_0$$

Vậy nhận định của bạn kia là **đúng**.

5.3 Bài toán kiểm định hai mẫu

Đặt vấn đề: Bạn Hoàng thấy có 2 loại CPU là có hoặc không có Embedded, bạn ấy cho rằng TDP trung bình của CPU có Embedded giống với TDP trung bình của CPU không có Embedded. Kiểm định xem nhận định này đúng hay sai dựa trên hai mẫu dữ liệu của `Embedded_Option_Available` với mức ý nghĩa 5%.

Gọi μ_1 là giá trị TDP trung bình của CPU có Embedded, μ_2 là giá trị TDP trung bình của CPU không có Embedded.

Đặt giả thuyết kiểm định:

$$\begin{cases} H_0 : \mu_1 = \mu_2 \\ H_1 : \mu_1 \neq \mu_2 \end{cases}$$

Chia dữ liệu theo hai biến Yes và No của biến `Embedded_Option_Available`.

```
1 # Split data into 2 groups  
2 final_data$Group <- ifelse(final_data$Embedded_Options_Available == "Yes"  
3   ↪ , "Group_Yes", "Group_No")  
4 Embedded__Option <- subset(final_data, Group == "Group_Yes")  
5 Non_Embedded__Option <- subset(final_data, Group == "Group_No")  
6 # Summary statistics for the two groups:  
7 n1 <- length(Embedded__Option$TDP)  
8 x1 <- mean(Embedded__Option$TDP)  
9 s1 <- sd(Embedded__Option$TDP)  
10 n2 <- length(Non_Embedded__Option$TDP)  
11 x2 <- mean(Non_Embedded__Option$TDP)  
12 s2 <- sd(Non_Embedded__Option$TDP)  
13 data.frame(n1, x1, s1, n2, x2, s2)
```


Output

n1	x1	s1	n2	x2	s2
558	43.277	31.15567	1725	64.74877	46.69333

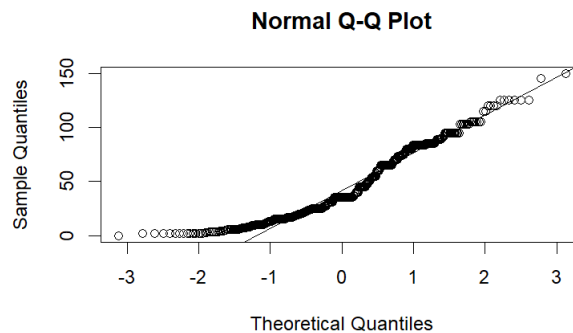
Để giải bài toán trên, cần xác định xem bài toán thuộc dạng toán nào, ở đây ta cần kiểm định giả định về phân phối chuẩn của hai mẫu tùy chọn nhúng.

Đặt giả thuyết kiểm định giả định về phân phối chuẩn của hai mẫu tùy chọn nhúng:

$$\begin{cases} H_0 : \text{Mẫu tuân theo phân phối chuẩn.} \\ H_1 : \text{Mẫu không tuân theo phân phối chuẩn.} \end{cases}$$

Đối với mẫu Embedded_Option:

```
1 qqnorm(Embedded__Option$TDP)
2 qqline(Embedded__Option$TDP)
```



Ảnh 23: Đồ thị Q-Q plot đối với mẫu Embedded_Option

Nhận xét: Dựa trên đồ thị QQ plot, ta nhận thấy các quan trắc nằm lệch khỏi đường thẳng kỳ vọng phân phối chuẩn. Do đó mẫu Embedded_Option không tuân theo phân phối chuẩn.

```
1 shapiro.test(Embedded__Option$TDP)
```

Output

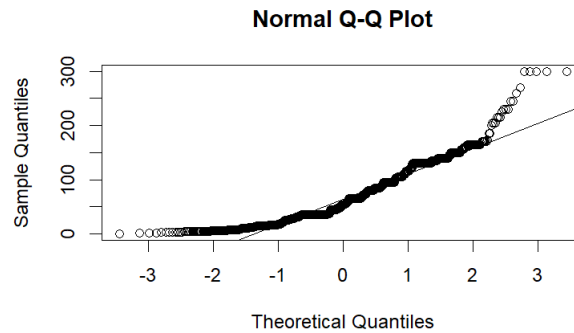
```
Shapiro-Wilk normality test

data:  Embedded__Option$TDP
W = 0.92603, p-value = 6.048e-16
```

Nhận xét: Dựa trên p-value = 6.048e-16 ở kiểm định shapiro.test rất bé, và nhỏ hơn mức ý nghĩa 5% nên ta bác bỏ H0, chấp nhận H1. Vậy mẫu Embedded_Option không tuân theo phân phối chuẩn.

Đối với mẫu Non_Embedded_Option:

```
1 qqnorm(Non_Embedded__Option$TDP)
2 qqline(Non_Embedded__Option$TDP)
```



Ảnh 24: Đồ thị Q-Q plot đối với mẫu Non_Embedded_Option

Nhận xét: Dựa trên đồ thị QQ plot, ta nhận thấy các quan trắc nằm lệch khỏi đường thẳng kỳ vọng phân phối chuẩn. Do đó mẫu Non_Embedded_Option không tuân theo phân phối chuẩn.

```
1 shapiro.test(Non_Embedded__Option$TDP)
```

Output

```
Shapiro-Wilk normality test

data:  Non_Embedded__Option$TDP
W = 0.91213, p-value < 2.2e-16
```

Nhận xét: Dựa trên p-value < 2.2e-16 ở kiểm định shapiro.test rất bé, và nhỏ hơn mức ý nghĩa 5% nên ta bác bỏ H_0 , chấp nhận H_1 . Vậy mẫu Non_Embedded_Option không tuân theo phân phối chuẩn.

Đây là dạng bài kiểm định trung bình 2 mẫu độc lập, 2 tổng thể có phân phối bất kỳ và 2 cỡ mẫu đều lớn hơn 30.

Tính giá trị kiểm định và xác định vùng bác bỏ

```
1 z0 <- (x1 - x2) / sqrt(s1^2/n1 + s2^2/n2)
2 z_half_alpha <- qnorm(p = 0.025, lower.tail = FALSE)
3 cat("z0 =", z0, " | z_alpha =", z_half_alpha, "\n")
```

Output

```
z0 = -12.38953 | z_alpha = 1.959964
```

Kiểm tra xem z0 có nằm trong miền bác bỏ không

```
1 if (abs(z0) > z_half_alpha) {
```

```
2   cat("Reject H0. There is a difference in TDP between the two groups.\n")
3   } else {
4     cat("Not enough evidence to reject H0. The average TDP may be the\n")
5   }
```

Output

Reject H0. There is a difference in TDP between the two groups.

Nhận xét: Vì $z_0 \in RR$ nên ta bác bỏ H_0 , chấp nhận H_1 .

Vậy ta có thể kết luận TDP trung bình của CPU có Embedded và CPU không có Embedded có sự chênh lệch.

5.4 Mô hình ANOVA một nhân tố

Đặt vấn đề : Phân tích và so sánh sự khác biệt trong giá trị TDP giữa các nhóm Vertical Segment trong dữ liệu, bao gồm các nhóm: Desktop, Embedded, Mobile, và Server. Sử dụng phân tích phương sai một nhân tố (ANOVA) để kiểm tra sự khác biệt về trung bình giữa các nhóm. Sau đó, áp dụng kiểm định Tukey HSD để xác định cặp nhóm nào có sự khác biệt có ý nghĩa thống kê.

Ta gọi $\mu_1, \mu_2, \mu_3, \mu_4$ lần lượt là giá trị TDP trung bình của các nhóm Mobile, Server, Desktop và Embedded.

Giả thuyết kiểm định:

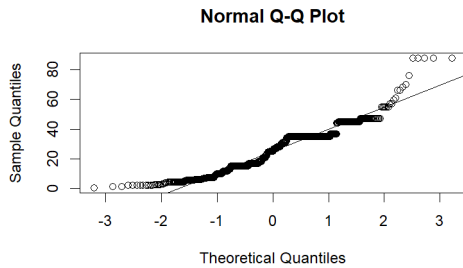
$$\begin{cases} H_0 : \mu_1 = \mu_2 = \mu_3 = \mu_4 \\ H_1 : \text{Tồn tại ít nhất một cặp } \mu_i \neq \mu_j \end{cases}$$

Các điều kiện cần kiểm tra trong mô hình ANOVA:

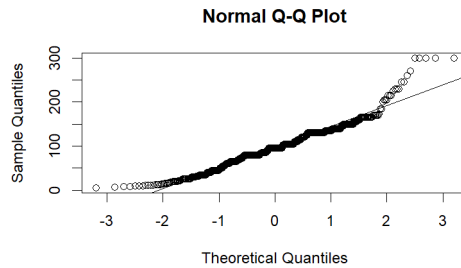
- **Giả định 1:** Dữ liệu trong mỗi nhóm tuân theo phân phối chuẩn.

```
1 # Mobile
2 Mobile_data <- subset(final_data, final_data$Vertical_Segment=="Mobile")
3 qqnorm(Mobile_data$TDP)
4 qqline(Mobile_data$TDP)
5 shapiro.test(Mobile_data$TDP)
6 # Server
7 Server_data <- subset(final_data, final_data$Vertical_Segment=="Server")
8 qqnorm(Server_data$TDP)
9 qqline(Server_data$TDP)
10 shapiro.test(Server_data$TDP)
11 # Desktop
12 Desktop_data <- subset(final_data, final_data$Vertical_Segment=="Desktop")
13 qqnorm(Desktop_data$TDP)
14 qqline(Desktop_data$TDP)
15 shapiro.test(Desktop_data$TDP)
16 # Embedded
```

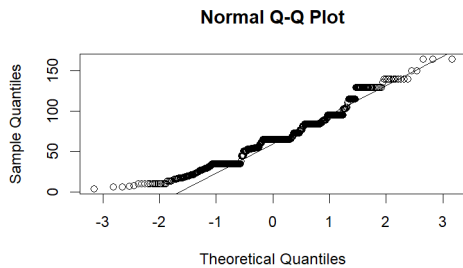
```
17 Embedded_data <- subset(final_data, final_data$Vertical_Segment=="Embedded  
↪ ")  
18 qqnorm(Embedded_data$TDP)  
19 qqline(Embedded_data$TDP)  
20 shapiro.test(Embedded_data$TDP)
```



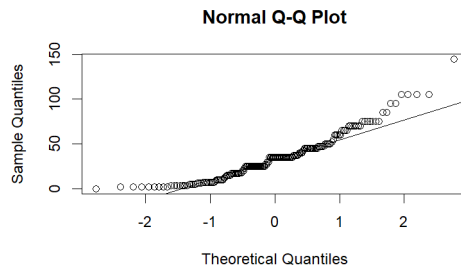
(a) Mobile



(b) Server



(c) Desktop



(d) Embedded

Ảnh 25: Đồ thị Q-Q plot đối với các mẫu Vertical_Segment

Nhận xét: Dựa trên đồ thị QQ plot, ta nhận thấy các quan trắc nằm lệch khỏi đường thẳng kỳ vọng phân phối chuẩn. Do đó các mẫu Vertical_Segment không tuân theo phân phối chuẩn.

Output

```
Shapiro-Wilk normality test  
  
data: Mobile_data$TDP  
W = 0.93376, p-value < 2.2e-16  
data: Server_data$TDP  
W = 0.95489, p-value = 4.972e-14  
data: Desktop_data$TDP  
W = 0.9611, p-value = 7.708e-12  
data: Embedded_data$TDP  
W = 0.92165, p-value = 3.735e-08
```

Nhận xét: Dựa trên p-value ở kiểm định shapiro.test rất bé. Vậy các mẫu Vertical_Segment không tuân theo phân phối chuẩn.

- **Giả định 2:** Các phương sai của các nhóm là bằng nhau (đồng nhất phương sai).



```
1 library(car)
2 leveneTest(TDP~as.factor(Vertical_Segment),final_data)
```

Output

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value    Pr(>F)
group   3   146.8 < 2.2e-16 ***
      2279
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nhận xét: p-value $< 2.2e - 16$ ở kiểm định leveneTest cho thấy p-value này rất bé. Vậy phương sai giữa các nhóm là khác nhau.

Từ việc kiểm tra các giả định đều không thoả, việc thực hiện mô hình Anova bên dưới chỉ mang tính chất tham khảo.

Giả sử các điều kiện thoả mãn, ta thực hiện mô hình Anova:

```
1 anova_vs <- aov(TDP ~ Vertical_Segment, data = final_data)
2 summary(anova_vs)
```

Output

```
              Df Sum Sq Mean Sq F value Pr(>F)
Vertical_Segment  3 2016140  672047   618.2 <2e-16 ***
Residuals      2279 2477688    1087
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nhận xét:

- **Kiểm định ý nghĩa:** Giá trị p-value $< 2 \times 10^{-16}$ (rất nhỏ) cho thấy rằng sự khác biệt giữa các nhóm trong yếu tố **Vertical_Segment** là có ý nghĩa thống kê. Nói cách khác, có sự khác biệt đáng kể về giá trị trung bình của TDP giữa các nhóm phân khúc (**Desktop**, **Embedded**, **Mobile**, **Server**).
- **Phân tích phương sai:** Tổng phương sai được chia thành hai phần: phương sai giữa các nhóm ($Sum\ Sq = 2,016,140$) và phương sai trong nhóm ($Residuals = 2,477,688$). Tỷ lệ giữa phương sai giữa nhóm và phương sai trong nhóm, thông qua giá trị F ($F = 618.2$), rất lớn, khẳng định thêm rằng yếu tố **Vertical_Segment** giải thích được phần lớn sự khác biệt.

Thực hiện so sánh bội:

```
1 tukey_result <- TukeyHSD(anova_vs)
2 print(tukey_result)
```

Output

```
Tukey multiple comparisons of means
 95% family-wise confidence level

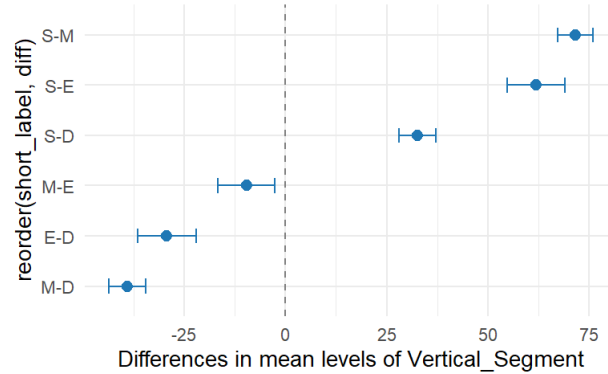
Fit: aov(formula = TDP ~ Vertical_Segment, data = final_data)
```

\$Vertical_Segment	diff	lwr	upr	p adj
Embedded-Desktop	-29.370907	-36.58485	-22.156967	0.0000000
Mobile-Desktop	-39.040669	-43.61207	-34.469270	0.0000000
Server-Desktop	32.592056	27.96056	37.223554	0.0000000
Mobile-Embedded	-9.669762	-16.74461	-2.594909	0.0025378
Server-Embedded	61.962963	54.84913	69.076797	0.0000000
Server-Mobile	71.632725	67.22099	76.044458	0.0000000

Nhận xét: Tất cả các sự khác biệt đều có ý nghĩa thống kê (p-adj rất nhỏ), cho thấy TDP giữa các nhóm Vertical Segment là khác biệt rõ rệt. CPU nhóm Server có TDP trung bình cao nhất, tiếp theo là Desktop, rồi đến Mobile và Embedded. Điều này phù hợp với đặc tính sử dụng: Server yêu cầu hiệu năng cao, Desktop cân bằng giữa hiệu năng và điện năng, còn Mobile và Embedded ưu tiên tiết kiệm điện.

Vẽ đồ thị so sánh bội

```
1 # Create a dataframe from the Tukey HSD test results
2 tukey_df <- as.data.frame(tukey_result$Vertical_Segment)
3 tukey_df$comparison <- rownames(tukey_result$Vertical_Segment)
4
5 # Simplify group names for easier visualization (Server -> S, Embedded ->
  ↪ E, Mobile -> M, Desktop -> D)
6 tukey_df$short_label <- tukey_df$comparison
7 tukey_df$short_label <- gsub("Server", "S", tukey_df$short_label)
8 tukey_df$short_label <- gsub("Embedded", "E", tukey_df$short_label)
9 tukey_df$short_label <- gsub("Mobile", "M", tukey_df$short_label)
10 tukey_df$short_label <- gsub("Desktop", "D", tukey_df$short_label)
11
12 # Create a horizontal plot showing the mean differences and confidence
  ↪ intervals
13 ggplot(tukey_df, aes(x = diff, y = reorder(short_label, diff))) +
14   geom_point(size = 3, color = "#1f77b4") + # Add points for mean
  ↪ differences
15   geom_errorbarh(aes(xmin = lwr, xmax = upr), height = 0.3, color = "#1
  ↪ f77b4") + # Add horizontal error bars
16   geom_vline(xintercept = 0, linetype = "dashed", color = "gray50") + #
  ↪ Reference line at 0 for no difference
17   labs(
18     x = "Differences in mean levels of Vertical_Segment" # X-axis label
19   ) +
20   theme_minimal(base_size = 14) # Use a clean, minimal theme with larger
  ↪ base font
```



Ảnh 26: Đồ thị so sánh bội giữa các cặp Vertical_Segment

Nhận xét:

- Đồ thị thể hiện khoảng tin cậy 95% cho các cặp so sánh: Tất cả các khoảng tin cậy đều không chứa giá trị 0, điều này khẳng định rằng sự khác biệt giữa các nhóm là có ý nghĩa thống kê.
- Đồ thị minh họa rằng sự khác biệt lớn nhất nằm giữa nhóm Server-Mobile, trong khi sự khác biệt giữa Embedded-Desktop là nhỏ nhất.

5.5 Hồi quy

5.5.1 Phân chia dữ liệu

Từ kết quả ở phần kiểm định tương quan, nhóm tiến hành giữ lại các biến sau để xây dựng mô hình: TDP, nb_of_Cores, Max_Memory_Bandwidth, Processor_Base_Frequency, Lithography.

5.5.2 Mô hình hồi quy tuyến tính đa biến

Bài toán hồi quy tuyến tính dự báo TDP của CPU

Mục tiêu là xây dựng mô hình hồi quy tuyến tính đa biến để dự báo công suất thiết kế nhiệt (TDP) của CPU.

- Biến phụ thuộc: $Y = \text{TDP}$
- Các biến độc lập:
 - $X_1 = \text{nb_of_Cores}$
 - $X_2 = \text{Max_Memory_Bandwidth}$
 - $X_3 = \text{Processor_Base_Frequency}$
 - $X_4 = \text{Lithography}$

Mô hình hồi quy tuyến tính đa biến có dạng như sau:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \varepsilon \quad (1)$$

Trong đó:

- β_0 : Hệ số chặn (intercept)
- $\beta_1, \beta_2, \beta_3, \beta_4$: Các hệ số hồi quy tương ứng với các biến độc lập
- ε : Sai số ngẫu nhiên

Thực hiện ước lượng với các hệ số β_i với $i = 0, 1, 2, 3, 4$.

```
lm1 <- lm(train.data$TDP ~ ., data = train.data)
print(summary(lm1))
```

Output

```
Call:
lm(formula = TDP ~ ., data = train.data)

Residuals:
    Min       1Q   Median       3Q      Max
-103.985  -16.510   -5.525   13.946   139.499

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -43.35695     2.39326  -18.116 < 2e-16 ***
nb_of_Cores     4.43586     0.15201   29.180 < 2e-16 ***
Max_Memory_Bandwidth  0.21764     0.04123    5.278 1.46e-07 ***
Processor_Base_Frequency 29.61320     0.78382   37.781 < 2e-16 ***
Lithography     0.24768     0.01532   16.170 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 25.63 on 1823 degrees of freedom
Multiple R-squared:  0.6626,    Adjusted R-squared:  0.6619
F-statistic: 895 on 4 and 1823 DF,  p-value: < 2.2e-16
```

Nhận xét mô hình hồi quy tuyến tính đa biến:

Từ kết quả ước lượng mô hình, ta thu được các hệ số như sau:

$$\beta_0 = -43.35695$$

$$\beta_1 = 4.43586$$

$$\beta_2 = 0.21764$$

$$\beta_3 = 29.61320$$

$$\beta_4 = 0.24768$$

Do đó, phương trình hồi quy ước lượng là:

$$Y = -43.35695 + 4.43586X_1 + 0.21764X_2 + 29.61320X_3 + 0.24768X_4 \quad (7)$$

Trong đó:

- X_1 là *nb_of_Cores*
- X_2 là *Max_Memory_Bandwidth*
- X_3 là *Processor_Base_Frequency*
- X_4 là *Lithography*

Nhận xét:

- Hệ số $\beta_1 = 4.43586$ cho thấy: Khi số lõi (*nb_of_Cores*) tăng thêm 1 đơn vị (giữ các biến khác không đổi), thì TDP kỳ vọng sẽ tăng thêm khoảng 4.43586 đơn vị.
- Hệ số $\beta_2 = 0.21764$ cho thấy: Khi băng thông bộ nhớ tối đa (*Max_Memory_Bandwidth*) tăng thêm 1 đơn vị, thì TDP kỳ vọng sẽ tăng khoảng 0.21764 đơn vị, với điều kiện các biến khác giữ nguyên.
- Hệ số $\beta_3 = 29.6132$: Khi *Processor_Base_Frequency* (X_3) tăng 1 đơn vị (các biến còn lại không đổi), TDP kỳ vọng tăng 29.6132 đơn vị.
- Hệ số $\beta_4 = 0.24768$: Khi *Lithography* (X_4) tăng 1 đơn vị, TDP kỳ vọng tăng 0.24768 đơn vị.

Ví dụ:

Giả sử có một CPU với các thông số:

$$X_1 = 4, \quad X_2 = 25.6, \quad X_3 = 3.1, \quad X_4 = 22$$

Khi đó, TDP ước lượng:

$$\hat{Y} = 77.207954 \text{ (W)}$$

Nếu TDP thực tế là $Y = 77$, sai số là:

$$\varepsilon = Y - \hat{Y} = 77 - 77.207954 = -0.207954$$

Kiểm định ý nghĩa thống kê của các hệ số β_i :

- Giả thuyết H_0 : $\beta_i = 0$ với $i = 1, 2, 3, 4$
- Giả thuyết H_1 : $\beta_i \neq 0$

Dựa vào giá trị **p-value** $< 2.2 \times 10^{-16}$ cho tất cả các hệ số, nhỏ hơn rất nhiều so với mức ý nghĩa 0.1%, ta bác bỏ giả thuyết H_0 , chấp nhận H_1 . Do đó, các hệ số β_i đều có ý nghĩa thống kê, góp phần đáng kể trong dự báo TDP.

Hệ số xác định $R^2 = 0.6626$ cho thấy mô hình có thể giải thích 66.26% phương sai của biến TDP, cho thấy mô hình có độ phù hợp cao.

Kiểm định sự phù hợp của mô hình hồi quy:

- Giả thuyết H_0 : $\beta_1 = \beta_2 = \beta_3 = \beta_4 = 0$ hay $R^2 = 0$ (phương trình hồi quy không thích hợp)
- Giả thuyết H_1 : tồn tại ít nhất một $\beta_i \neq 0$ hay $R^2 \neq 0$

Dựa vào kiểm định F với **p-value** $< 2.2 \times 10^{-16}$, nhỏ hơn nhiều so với mức ý nghĩa 0.1%, ta bác bỏ H_0 , chấp nhận H_1 . Vậy phương trình hồi quy là **thích hợp**.

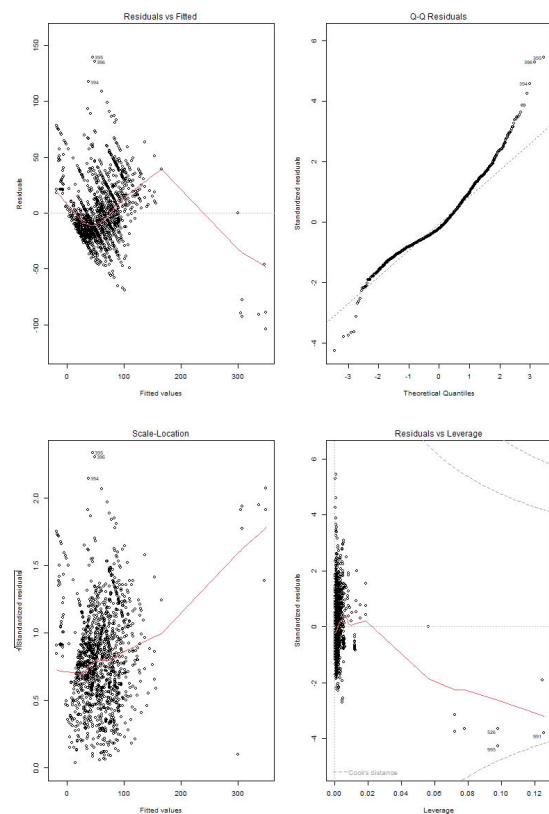
Đánh giá mức độ ảnh hưởng của các biến độc lập đến TDP:

- Ảnh hưởng lớn nhất: **Processor_Base_Frequency** (X_3)
- Tiếp theo: **nb_of_Cores** (X_1)
- Kế đến: **Lithography** (X_4)
- Ảnh hưởng ít nhất: **Max_Memory_Bandwidth** (X_2)

5.5.3 Kiểm tra giả thiết thống kê

Từ kết quả xây dựng mô hình 2. ta cần kiểm định lại các giả thiết của mô hình:

- Tính tuyến tính của dữ liệu
- Sai số có phân phối chuẩn
- Phương sai của các sai số là hằng số
- Các sai số độc lập với nhau



Ảnh 27: Các đồ thị

Đồ thị **Residuals vs Fitted**: Dùng để kiểm tra giả thuyết 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 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. Mặt 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.

Nhận xét: Đường thẳng màu đỏ trên đồ thị không phải là một đường thẳng, tức là mối quan hệ giữa các biến dự báo X và biến phụ thuộc Y không thể được xem như là tuyến tính, không thỏa mãn giả định tuyến tính của dữ liệu. Giá trị thặng dư (sai số) phân tán tương đối đều xung quanh đường thẳng $y = 0$ (ngoài trừ một số giá trị ngoại lai) nên giả định về phương sai của các sai số là hằng số thỏa mãn.

Đồ thị **Normal Q-Q**: Kiểm tra giả định về phân phối chuẩn của các sai số. Nếu các điểm thặng dư nằm trên cùng một đường thẳng thì điều kiện về phân phối chuẩn được thỏa mãn.

Nhận xét: Biểu đồ Q-Q Residuals cho thấy các điểm sai số không nằm trên đường chéo, đặc biệt ở hai đầu, tạo thành hình dạng cong giống chữ "S", điều này cho thấy sai số không tuân theo phân phối chuẩn – một giả định quan trọng trong hồi quy tuyến tính. Ngoài ra, sự xuất hiện của các điểm ngoại lai như 394, 395, 396 với residuals lớn hơn 4 cho thấy mô hình đang bị ảnh hưởng bởi các điểm dữ liệu bất thường.

Đồ thị **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 dự đoán của 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 mãn.

Nhận xét: Biểu đồ Scale-Location cho thấy phương sai phần dư tăng dần theo giá trị dự đoán, thể hiện hiện tượng heteroscedasticity (phương sai thay đổi).

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

Nhận xét: Biểu đồ Residuals vs Leverage cho thấy các điểm 526, 991 và 995 có leverage cao và nằm gần hoặc vượt ngưỡng đường Cook's distance, cho thấy chúng có ảnh hưởng lớn đến mô hình.

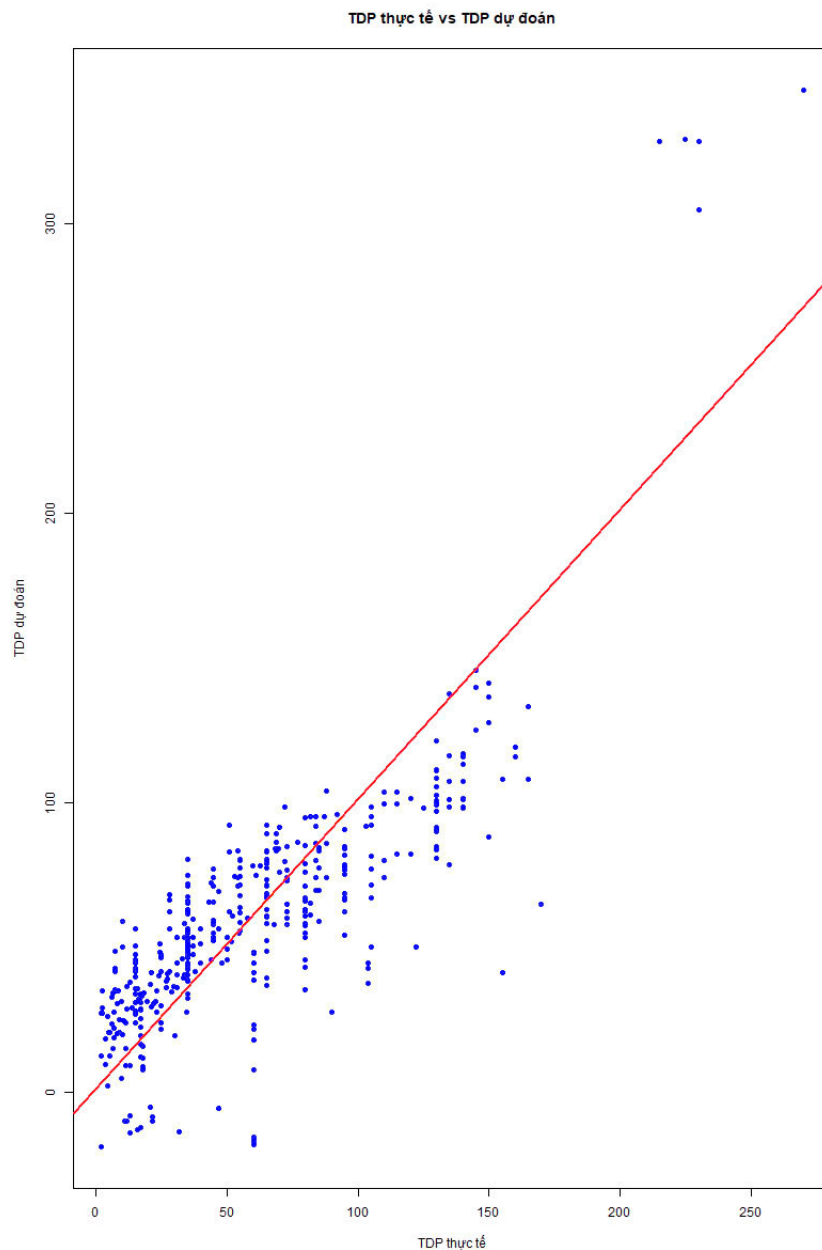
5.5.4 Dự báo

Từ mô hình đã xây. Ta thực hiện dự đoán và kiểm tra mô hình dựa trên tập `test.data` đã phân ban đầu.

```
1 prediction <- predict(lm1, test.data)
2
3 plot(test.data$TDP, prediction,
4      xlab = "TDP output",
5      ylab = "TDP expected",
6      main = "TDP output vs TDP expected",
7      pch = 19, col = "blue")
8 abline(a = 1, b = 1, col = "red", lwd = 2)
```



```
9  
10 results <- data.frame(  
11   R2    = round(R2(prediction, test.data$TDP), 4),  
12   RMSE = round(rmse(test.data$TDP, prediction), 4),  
13   MAE   = round(mae(test.data$TDP, prediction), 4)  
14 )
```



Ảnh 28: Kết quả dự báo trên tập test.data

Dựa vào đồ thị và giá trị $R^2 = 0,6358$ cho thấy mô hình hồi quy đa biến có hiệu quả khá cao. Mặc dù vẫn còn một số điểm có độ chênh lệch khá lớn, nhưng với kết quả MAE (sai số trung bình tuyệt đối) = 21,2158 và RMSE (độ lệch chuẩn của các phần dư) = 27,4847 đều ở mức trung bình so với khoảng giá trị của TDP, thì mô hình vẫn cho thấy được sự tin cậy và ổn định tốt.

R2	RMSE	MAE
0.6358	27.4847	21.2158

Ảnh 29: Kết quả đánh giá mô hình

6 Thảo luận và mở rộng

6.1 Đánh giá kết quả phân tích

6.1.1 Khác biệt giữa các phân khúc CPUs

Phân tích các kiểm định thống kê và biểu đồ đã chỉ ra đặc trưng tiêu thụ riêng của từng phân khúc:

- Phân khúc Server có TDP cao với biên độ biến động lớn, phù hợp với nhu cầu sử dụng của hệ thống lớn, xử lý cao.
- Phân khúc Desktop có TDP tầm trung, thể hiện đặc điểm thiết kế cân bằng giữa hiệu năng và tiết kiệm điện.
- Phân khúc Mobile, Embedding có TDP thấp với biên độ biến động nhỏ, chứng tỏ thiết kế bền vững và chú trọng hiệu suất năng lượng.

6.1.2 TDP tăng theo thời gian

Trong kết quả của các kiểm định, TDP có xu hướng tăng theo từng năm. Tuy nhiên, sự tăng này không hoàn toàn tuyến tính và có biến động. Sự tăng trưởng của TDP còn tuân theo các bước nhảy tương ứng với sự thay đổi trong công nghệ.

6.2 Giới hạn của kiểm định

Thiếu dữ liệu quan trọng: Trong quá trình chọn lọc dữ liệu, các thông số quan trọng trong việc đánh giá TDP không có trong tập dữ liệu/ dữ liệu khuyết thiếu (Công nghệ tản nhiệt khuyết dữ liệu, các thông số quan trọng khác như điện áp hoạt động, kiến trúc microarchitecture không có trong tập dữ liệu).

Phân phối lệch: Một số biến như Cache và Max Mem Bandwidth có phân phối lệch, có thể cần phải sử dụng các phương pháp kiểm định khác thay cho truyền thống.

Đa cộng tuyến (Multicollinearity): Có thể xảy ra ở biến kỹ thuật Cache và Thread, ảnh hưởng tới kết quả kiểm định do làm sai lệch ước lượng trong hồi quy tuyến tính bội.

6.3 Mở rộng hướng kiểm định và nghiên cứu

6.3.1 Áp dụng mô hình học máy

Đối với việc các biến có phân phối lệch, việc quan sát qua mô hình phi tuyến / học máy sẽ có thể đem lại kết quả tốt hơn.

Output

```
# Ví dụ sử dụng mô hình Random Forest
library(randomForest)
rf_model <- randomForest(TDP ~ ., data = data_clean, ntree = 100)
print(rf_model)
```

Ngoài ra, sử dụng mô hình Random Forest còn hỗ trợ xác định độ quan trọng của các biến.

6.3.2 Giảm chiều và phát hiện cộng tuyến

Có thể sử dụng PCA (Phân tích thành phần chính) hoặc kiểm tra hệ số VIF (Variance Inflation Factor) để kiểm soát đa cộng tuyến:

Output

```
# Tính VIF để kiểm tra multicollinearity
library(car)
lm_model <- lm(TDP ~ ., data = data_clean)
vif(lm_model)
```

6.3.3 Phân tích phân cụm CPU

Thay vì chỉ phân tích theo Vertical Segment, có thể thực hiện phân cụm (clustering) để phát hiện các nhóm CPU ẩn theo đặc trưng kỹ thuật library(cluster)

Output

```
data_numeric <- data_clean[, sapply(data_clean, is.numeric)]
kmeans_result <- kmeans(scale(data_numeric), centers = 4)
fviz_cluster(kmeans_result, data = scale(data_numeric))
```



7 Nguồn dữ liệu và nguồn code

- Nguồn dữ liệu về linh kiện máy tính (CPU và GPU): https://www.kaggle.com/datasets/iliassekkaf/computerparts/?select=All_GPUs.csv
- Nguồn mã R: <https://drive.google.com/drive/folders/11251VNvmzHFwhmTvPu7gcxzNIiUYwHv?usp=sharing>



Tài liệu tham khảo

- Nguyễn Tiến Dũng, *Xác suất - Thống kê & Phân tích số liệu*, NXB. Đại học Quốc gia Thành phố Hồ Chí Minh.
- Nguyễn Văn Tuấn, 2014, Suy diễn thống kê và ngôn ngữ R (4): Phân tích phương sai (ANOVA), <https://www.slideshare.net/slideshow/r-chap11-anova/36094530>
- Navidi, 2021, *Statistics for Engineers and Scientists*, <https://www.mheducation.com/highered/product/statistics-engineers-scientists-navidi/M9781260696093.html>
- Snedecor, G. W., Cochran, W. G., 1989, *Statistical Methods*, <https://www.wiley.com/en-us/Statistical+Methods-p-9780813815619>
- Tham khảo các lệnh code R, <https://www.w3schools.com/r/>
- Tham khảo một số chương trình sử dụng code R để phân tích, <https://www.kaggle.com/datasets/iliassekkaf/computerparts/code>