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

BÁO CÁO BÀI TẬP LỚN

Nhóm: 12 - L16

XÁC SUẤT THỐNG KÊ

Computer Parts (CPUs and GPUs)
Khoa Khoa học và Kỹ thuật Máy tính

GVHD: Nguyễn Thị Kiều Ân
Lớp: L16

Thành phố Hồ Chí Minh, 11/2024

DANH SÁCH THÀNH VIÊN

STT	Họ và tên	MSSV
1	Trương Thái Nguyên	2312391
2	Lê Trần Bảo Nhi	2312499
3	Bùi Đình Phúc	2312664
4	Đặng Minh Quang	2312785
5	Đỗ Minh Sang	2312930

MÔ TẢ CÔNG VIỆC

STT	Họ và tên	MSSV	Mô tả công việc
1	Đỗ Minh Sang	2312930	Latex, Code R, Tổng hợp lý thuyết, Tiền xử lý, Thống kê mô tả, tổng hợp
2	Bùi Đình Phúc	2312664	Latex, Code R, 1 mẫu, 2 mẫu , Hồi quy đa biến, Mở rộng, hổ trợ tiền xử lý
3	Lê Trần Bảo Nhi	2312399	Latex, Hỗ trợ thống kê mô tả, slide, thuyết trình, tổng hợp
4	Đặng Minh Quang	2312785	Latex, Code R, ANOVA, Hồi quy đa biến
5	Trương Thái Nguyên	2312391	Slide

Mục lục

1 Tổng quan dữ liệu

1. **Architecture:** Kiến trúc cơ bản của GPU, quyết định khả năng xử lý và hiệu suất, cùng các tính năng kỹ thuật của nó.
2. **Best_Resolution:** Độ phân giải cao nhất mà GPU có thể xử lý trong khi duy trì hiệu suất ổn định và chất lượng hình ảnh tốt.
3. **Boost_Clock:** Tốc độ tối đa mà GPU có thể đạt được trong những tình huống tải cao, khi hệ thống tản nhiệt cho phép.
4. **Core_Speed:** Tốc độ hoạt động bình thường của lõi GPU, thường được đo bằng Megahertz (MHz) hoặc Gigahertz (GHz).
5. **DVI_Connection:** Số lượng cổng DVI trên GPU, cho phép kết nối với màn hình sử dụng giao diện Digital Visual Interface.
6. **Dedicated:** Chỉ ra GPU là phần cứng đồ họa chuyên dụng, không phụ thuộc vào CPU, nhằm tối ưu hóa cho xử lý hình ảnh.
7. **Direct_X:** Bộ API của Microsoft được sử dụng để xử lý các tác vụ đa phương tiện, bao gồm đồ họa và âm thanh, chủ yếu trên nền tảng Windows.
8. **DisplayPort_Connection:**Số lượng cổng DisplayPort có trên GPU, cho phép kết nối với màn hình hỗ trợ giao diện DisplayPort.
9. **HDMI_Connection:** Số lượng cổng HDMI, dùng để truyền tín hiệu hình ảnh và âm thanh đến các thiết bị như màn hình hoặc TV.
10. **Integrated:** GPU tích hợp, nằm trong cùng một gói với CPU và sử dụng chung tài nguyên hệ thống, như RAM.
11. **L2_Cache:** Bộ nhớ đệm cấp 2 giúp GPU truy cập nhanh hơn vào dữ liệu được sử dụng thường xuyên, tối ưu hóa hiệu suất.
12. **Manufacturer:** Tên nhà sản xuất của GPU, ví dụ như NVIDIA, AMD, hoặc Intel.
13. **Max_Power:** Công suất điện tối đa mà GPU tiêu thụ khi hoạt động với hiệu suất cao nhất.
14. **Memory:** Dung lượng bộ nhớ đồ họa (VRAM) của GPU, quyết định khả năng xử lý và lưu trữ dữ liệu hình ảnh.
15. **Memory_Bandwidth:** Lượng dữ liệu tối đa mà bộ nhớ GPU có thể truyền tải trong mỗi giây, đo bằng Gigabyte/giây (GB/s).
16. **Memory_Bus:** Độ rộng của bus bộ nhớ, ảnh hưởng đến tốc độ truy cập và hiệu suất của bộ nhớ GPU.
17. **Memory_Speed:** Tốc độ đọc và ghi của bộ nhớ GPU, thường được đo bằng Megahertz (MHz).
18. **Memory_Type:** Loại bộ nhớ VRAM được sử dụng, ví dụ như GDDR5, GDDR6, ảnh hưởng đến hiệu suất và tốc độ của GPU.
19. **Name:** Tên sản phẩm của GPU, dùng để nhận diện mô hình cụ thể.
20. **Notebook_GPU:** Chỉ ra rằng GPU được thiết kế cho máy tính xách tay, thường có khả năng tiết kiệm năng lượng tốt hơn so với GPU dành cho máy tính để bàn.
21. **Open_GL:** Một tiêu chuẩn lập trình đồ họa, được sử dụng để phát triển ứng dụng đồ họa và trò chơi trên nhiều nền tảng.
22. **PSU:** Yêu cầu công suất nguồn tối thiểu mà GPU cần để hoạt động hiệu quả.
23. **Pixel_Rate:** Tốc độ xử lý điểm ảnh của GPU, thường được đo bằng số pixel/giây.
24. **Power_Connector:** Số lượng và loại cổng kết nối nguồn mà GPU cần để nhận đủ năng lượng.

25. **Process:** Kích thước của các transistor trên GPU, đo bằng nanomet, ảnh hưởng đến hiệu suất và hiệu quả sử dụng điện.
26. **ROPs:** Các đơn vị xử lý đầu ra trong GPU, quyết định số lượng pixel mà GPU có thể xử lý trong mỗi chu kỳ.
27. **Release_Date:** Ngày ra mắt sản phẩm GPU trên thị trường.
28. **Release_Price:** Giá bán ra của GPU vào thời điểm ra mắt.
29. **Resolution_WxH:** Độ phân giải tối đa mà GPU hỗ trợ, biểu thị bằng số pixel chiều rộng và chiều cao.
30. **SLI_Crossfire:** Công nghệ cho phép kết nối nhiều GPU để tăng cường hiệu suất xử lý đồ họa.
31. **Shader:** Các chương trình nhỏ được thực thi trên GPU để tạo ra các hiệu ứng hình ảnh và xử lý đồ họa.
32. **TMUs:** Các đơn vị ánh xạ texture của GPU, chịu trách nhiệm xử lý và áp dụng texture lên các bề mặt 3D.
33. **Texture_Rate:** Tốc độ xử lý texture của GPU, được đo bằng số texel/giây.
34. **VGA_Connection:** Số lượng cổng VGA trên GPU, dùng để kết nối với màn hình qua tín hiệu tương tự (analog).

2 Kiến thức nền

2.1 Thống kê mô tả và thống kê suy diễn

Thống kê mô tả (descriptive statistics): là quá trình thu thập, biểu diễn, tổng hợp và xử lý dữ liệu để biến đổi dữ liệu thành thông tin.

- Thu thập dữ liệu: khảo sát, đo đạc, ...
- Biểu diễn dữ liệu: dùng bảng và đồ thị.
- Tổng hợp và xử lý dữ liệu: tính các tham số mẫu như trung bình mẫu (sample mean), phương sai mẫu (sample variance), trung vị (median),...

Thống kê suy diễn (Inferential statistics): xử lý các thông tin có được từ thống kê mô tả, từ đó đưa ra các cơ sở cho những dự đoán (predictions), dự báo (forecasts) và các ước lượng (estimations).

- Ước lượng: tham số thống kê (trung bình, tỷ lệ, phương sai).
- Kiểm định giả thuyết: tham số thống kê (trung bình, tỷ lệ, phương sai), quy luật phân phối xác suất (chuẩn, poisson,...), tính độc lập,...

2.2 Các đặc trưng của tổng thể và mẫu

2.2.1 Các khái niệm

Tổng thể: (population) là tập hợp hoàn chỉnh của tất cả các đơn vị hoặc cá thể mà người nghiên cứu muốn thu thập, quan sát. Tổng thể chứa toàn bộ dữ liệu có thể về một đối tượng nghiên cứu nhất định, có thể là một nhóm dân số, một tập hợp các sản phẩm, một tập hợp các văn bản, hoặc bất kỳ thực thể nào khác mà nghiên cứu muốn xem xét. Ví dụ, nếu quan tâm đến chiều cao của tất cả học sinh trong một trường học, thì tổng thể sẽ là tất cả các học sinh trong trường.

Mẫu: (sample) là một phần của tổng thể được lựa chọn để nghiên cứu. Mục tiêu của việc chọn một mẫu là để thu thập dữ liệu có thể phản ánh hoặc đại diện cho toàn bộ tổng thể mà không cần phải khảo sát toàn bộ cá nhân trong tổng thể đó. Điều này giúp giảm chi phí và thời gian cần thiết cho nghiên cứu. Mẫu cần được lựa chọn một cách ngẫu nhiên và đại diện cho tổng thể để kết quả của nghiên cứu có thể được áp dụng cho tổng thể. Tiếp tục ví dụ trên, nếu chỉ chọn một số học sinh từ trường để đo chiều cao, thì đó sẽ là mẫu

2.2.2 Tỷ lệ

Với một tổng thể có N phần tử và M phần tử mang tính chất A nào đó. Tỷ lệ tổng thể (kí hiệu: p) được tính bởi công thức:

$$p = \frac{M}{N}$$

Với một mẫu có n phần tử và có m phần tử mang tính chất A nào đó. Tỷ lệ mẫu (kí hiệu: f hay $\bar{p}$) được tính bởi công thức:

$$f = \frac{m}{n}$$

2.2.3 Trung bình

Trung bình (mean): là đại lượng thường được sử dụng nhất để đo giá trị trung tâm của dữ liệu. Với một tổng thể có N phần tử, trung bình tổng thể (kí hiệu: μ hay $\bar{x}$) tính bởi công thức:

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_1 + x_2 + \cdots + x_N}{N}$$

Với một mẫu có n phần tử, trung bình mẫu (kí hiệu: $\bar{x}$) tính bởi công thức:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Trong trường hợp X có bảng phân phối tần số như sau:

X	x_1	x_2	x_3	...	x_k
Tần số	n_1	n_2	n_3	...	n_k

Ta lại có trung bình mẫu tính bởi công thức:

$$\bar{x} = \frac{\sum_{i=1}^{k} x_i \cdot n_i}{n} = \frac{x_1 \cdot n_1 + x_2 \cdot n_2 + \cdots + x_k \cdot n_k}{n}$$

Lưu ý: Trung bình bị ảnh hưởng bởi các giá trị ngoại lai (outliers).

2.2.4 Phương sai, độ lệch chuẩn

Phương sai (Variance): là trung bình của bình phương độ lệch các giá trị so với trung bình. Phương sai phản ánh độ phân tán hay sự biến thiên của dữ liệu.

Độ lệch chuẩn (Standard deviation): Độ lệch chuẩn (Standard deviation) là căn bậc hai của phương sai. Độ lệch chuẩn dùng để đo sự biến thiên, biểu diễn sự biến thiên xung quanh trung bình và cũng có cùng đơn vị đo với dữ liệu gốc.

Với một tổng thể có N phần tử, phương sai tổng thể (kí hiệu: σ^2) tính bởi công thức:

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N} = \frac{\sum_{i=1}^{N} x_i^2 - N \cdot \mu^2}{N} = \frac{\sum_{i=1}^{N} x_i^2}{N} - \mu^2.$$

Khi đó: σ được gọi là độ lệch chuẩn của tổng thể. Với một mẫu có n phần tử, phương sai mẫu (kí hiệu: s^2) tính bởi công thức:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1} = \frac{\sum_{i=1}^{n} x_i^2 - n \cdot \bar{x}^2}{n-1} = \frac{\sum_{i=1}^{n} x_i^2}{n-1} - \frac{n \cdot \bar{x}^2}{n-1}.$$

Khi đó: s được gọi là độ lệch mẫu.

Trong trường hợp X có bảng phân phối tần số như sau:

X	x_1	x_2	x_3	...	x_k
Tần số	n_1	n_2	n_3	...	n_k

Ta lại có phương sai mẫu tính bởi công thức:

$$s^2 = \frac{n}{n-1} \cdot \frac{s^2}{n} = \frac{1}{n-1} \sum_{i=1}^{k} (x_i - \bar{x})^2 \cdot n_i$$

$$= \frac{1}{n-1} \left[(x_1 - \bar{x})^2 \cdot n_1 + (x_2 - \bar{x})^2 \cdot n_2 + \cdots + (x_k - \bar{x})^2 \cdot n_k \right].$$

Khi đó: s được gọi là độ lệch mẫu.

2.2.5 Các đặc trưng khác

Yếu vị (Mode):là giá trị của phần tử có số lần xuất hiện lớn nhất trong mẫu. Yếu vị không bị ảnh hưởng bởi các điểm ngoại lai.

Hệ số biến thiên (Coefficient of variation): được sử dụng để so sánh sự biến thiên của hai hay nhiều tập dữ liệu, có thể đo ở các đơn vị khác nhau. Đơn vị tính bằng %.

$$CV(\text{tổng thể}) = \frac{\sigma}{\mu} \times 100\%$$

$$CV(\text{mẫu}) = \frac{s}{\bar{x}} \times 100\%$$

Sai số chuẩn (Standard Error): là giá trị đại diện cho độ lệch chuẩn của giá trị trung bình trong tập dữ liệu. Nó phục vụ như một thước đo biến động cho các biến ngẫu nhiên hay độ lượn độ phân tán. Độ phân tán càng nhỏ, dữ liệu càng chính xác.

$$SE(\text{tổng thể}) = \frac{\sigma}{\sqrt{N}}$$

$$SE(\text{mẫu}) = \frac{s}{\sqrt{n}}$$

Trung vị (Median) Giả sử X có N quan sát, sắp các quan sát này theo thứ tự tăng dần. Trung vị là giá trị nằm chính giữa dãy số này và chia nó thành 2 phần bằng nhau. Cụ thể, nếu dữ liệu có dạng $x_1 < \cdots < x_{2k+1}$ thì trung vị là x_{k+1}; nếu mẫu có dạng $x_1 < \cdots < x_{2k}$ thì trung vị là trung bình cộng $\left(\frac{x_k + x_{k+1}}{2}\right)$. Trung vị không bị ảnh hưởng bởi các điểm ngoại lai (outliers).

Phân vị (Quartile)

- Điểm tứ phân vị dưới (Q1) là trung vị của nửa dữ liệu nhỏ, là giá trị chia dữ liệu thành 2 phần sao cho phần trái chiếm 25% của dữ liệu.
- Điểm tứ phân vị trên (Q3) là trung vị của nửa dữ liệu lớn, là giá trị chia dữ liệu thành 2 phần sao cho phần trái chiếm 75% của dữ liệu.
- Các điểm Q1, Q2 và Q3 được gọi là các điểm tứ phân vị.
- Hiệu IQR = Q3 − Q1 được gọi là khoảng tứ phân vị hay độ trải giữa.

Điểm Outlier: gọi là điểm dị biệt, điểm ngoại lai. Đó là các phần tử của dữ liệu nằm ngoài khoảng $(Q1 - 1.5 \cdot \text{IQR}; Q3 + 1.5 \cdot \text{IQR})$. Trong trường hợp mạnh hơn ta nói ngoài khoảng $(Q1 - 3.0 \cdot \text{IQR}; Q3 + 3.0 \cdot \text{IQR})$ thì đó là các ngoại lai xa.

2.3 Ước lượng

2.3.1 Các khái niệm

Lí thuyết ước lượng là một phần quan trọng trong lĩnh vực thống kê và nghiên cứu khoa học, chủ yếu liên quan đến việc tìm cách xác định giá trị của các tham số (parameters) của một quần thể dựa trên các mẫu (samples) được rút ra từ quần thể đó. Thông thường ta cần ước lượng trung bình tổng thể (μ), phương sai tổng thể (σ^2) và tỉ lệ các phần tử có tính chất nào đó trong tổng thể (p).

- **Khoảng Tin Cậy (Confidence Interval - CI)**: Là một loại ước lượng khoảng được sử dụng để chỉ ra phạm vi mà ta tin rằng tham số của tổng thể nằm trong đó. Khoảng tin cậy thường được xác định bởi hai giới hạn: giới hạn dưới và giới hạn trên. Ví dụ, một khoảng tin cậy 95% cho trung bình tổng thể có thể là (20, 30), nghĩa là ta tin rằng với độ tin cậy 95%, trung bình thực sự của tổng thể nằm trong khoảng từ 20 đến 30.
- **Mức Ý Nghĩa (Significance Level - α)**: Là ngưỡng mà ta chọn để quyết định ý nghĩa. Ví dụ mức ý nghĩa thường được chọn ở mức 0.05 - nghĩa là khả năng kết quả quan sát sự khác biệt được nhìn thấy trên số liệu là ngẫu nhiên chỉ là 5%.
- **Độ Tin Cậy (Confidence Level)**: Được biểu thị dưới dạng một tỷ lệ phần trăm chỉ mức độ tin tưởng hoặc sự chắc chắn mà khoảng tin cậy ước lượng của chúng ta bao gồm tham số tổng thể thực sự. Ví dụ: nếu ta xây dựng khoảng tin cậy với mức tin cậy 95%, ta tin chắc rằng 95 trên 100 lần ước tính sẽ nằm giữa giá trị trên và giá trị dưới được chỉ định bởi khoảng tin cậy.

$$\text{Confidence level} = 1 - \alpha$$

Có hai phương pháp ước lượng thường được sử dụng là ước lượng điểm (point estimation) và ước lượng khoảng (interval estimation).

- **Ước lượng điểm (Point Estimation)**: Là phương pháp tìm một giá trị duy nhất (một điểm) dùng làm ước lượng tốt nhất cho tham số của quần thể. Kết quả cần được ước lượng cho bởi một số:

$$\mu \approx \bar{x}$$

$$\sigma^2 \approx s^2$$

$$p \approx f$$

- **Ước lượng khoảng (Interval Estimation)**: Thay vì cung cấp một giá trị duy nhất như ước lượng điểm, ước lượng khoảng cho biết phạm vi giá trị mà tham số có thể nằm trong với một mức độ tin cậy nhất định (ví dụ, 95% hoặc 99% tin cậy). Kết quả ước lượng được cho bởi một khoảng (a,b).

$$(\bar{x} - \varepsilon, \bar{x} + \varepsilon)$$

2.3.2 Ước lượng điểm

Một **ước lượng (estimator)** của một tham số (của tổng thể): là một biến ngẫu nhiên có giá trị phụ thuộc vào thông tin của mẫu, giá trị của nó là một xấp xỉ cho tham số chưa biết của tổng thể. Một giá trị cụ thể của biến ngẫu nhiên này gọi là một giá trị ước lượng điểm.

Xét đại lượng ngẫu nhiên X có phân phối $F(x;\theta)$ với tham số θ chưa biết.
Chọn một mẫu ngẫu nhiên cỡ n từ X: $X_1, X_2, \ldots, X_n$.
Thống kê $\hat{\Theta} = h(X_1, X_2, \ldots, X_n)$ gọi là một ước lượng điểm cho θ.
Với một mẫu cụ thể $(x_1, x_2, \ldots, x_n)$, ta gọi $\hat{\theta} = h(x_1, x_2, \ldots, x_n)$ là một giá trị ước lượng điểm cụ thể cho θ.

2.3.3 Ước lượng khoảng tin cậy

Cho tham số θ của tổng thể và $X_1, X_2, \ldots, X_n$ là các quan sát ngẫu nhiên. Ta gọi khoảng (c,d) là khoảng ước lượng (hay khoảng tin cậy) của tham số θ với độ tin cậy γ nếu: $P(\theta \in (c,d)) = \gamma$. Có thể nói, độ tin cậy γ cho khoảng ước lượng của tham số θ chính là xác suất để ta đúng khi ước lượng tham số θ bằng khoảng (c,d). Ngược lại, xác suất mà ta cho phép sai khi ước lượng θ được gọi là mức ý nghĩa. Kí hiệu là α. Ta có $\alpha + \gamma = 1$.

Xác định khoảng ước lượng đối xứng của trung bình tổng thể dựa vào một mẫu đã cho, với kích thước là n, trung bình mẫu là $\bar{x}$ và phương sai mẫu là s^2 hoặc phương sai tổng thể là σ^2. Mục tiêu là xác định một khoảng ước lượng đối xứng xung quanh μ của mẫu với một mức độ tin cậy cụ thể.

Để giải quyết vấn đề này, ta cần đi đến việc xác định epsilon (ε) - sai số ước lượng, dựa trên các thông tin đã biết về mẫu. Khoảng tin cậy sẽ được biểu diễn bằng khoảng $\bar{x} \pm \varepsilon$. Tùy thuộc vào giả định về phân phối của dữ liệu và các thông tin đã biết về phương sai, cách tính ε sẽ thay đổi như sau:

Bảng tóm tắt các bài toán tìm khoảng tin cậy (tỷ lệ và trung bình):

Dạng	Giả định	Loại	Ngưỡng sai số	Khoảng tin cậy
Tỷ lệ	$n > 30$	Đối xứng	$\varepsilon = z_{\alpha/2} \cdot \sqrt{\frac{f(1-f)}{n}}$	$f - \varepsilon < p < f + \varepsilon$
		Bên phải	$\varepsilon = z_{\alpha/2} \cdot \sqrt{\frac{f(1-f)}{n}}$	$0 < p < f + z_\alpha \cdot \sqrt{\frac{f(1-f)}{n}}$
		Bên trái	$\varepsilon = z_{\alpha/2} \cdot \sqrt{\frac{f(1-f)}{n}}$	$f - z_\alpha \cdot \sqrt{\frac{f(1-f)}{n}} < p < 1$
Trung bình	**(1)**	Đối xứng	$\varepsilon = z_{\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}$	$\bar{x} - \varepsilon < \mu < \bar{x} + \varepsilon$
		Bên phải	$\varepsilon = z_{\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}$	$-\infty < \mu < \bar{x} + z_\alpha \cdot \frac{\sigma}{\sqrt{n}}$
		Bên trái	$\varepsilon = z_{\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}$	$\bar{x} - z_\alpha \cdot \frac{\sigma}{\sqrt{n}} < \mu < \infty$
	(2)	Đối xứng	$\varepsilon = t_{\alpha/2;n-1} \cdot \frac{s}{\sqrt{n}}$	$\bar{x} - \varepsilon < \mu < \bar{x} + \varepsilon$
		Bên phải	$\varepsilon = t_{\alpha/2;n-1} \cdot \frac{s}{\sqrt{n}}$	$-\infty < \mu < \bar{x} + t_{\alpha;n-1} \cdot \frac{s}{\sqrt{n}}$
		Bên trái	$\varepsilon = t_{\alpha/2;n-1} \cdot \frac{s}{\sqrt{n}}$	$\bar{x} - t_{\alpha;n-1} \cdot \frac{s}{\sqrt{n}} < \mu < \infty$
	(3)	Đối xứng	$\varepsilon = z_{\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}$	$\bar{x} - \varepsilon < \mu < \bar{x} + \varepsilon$
		Bên phải	$\varepsilon = z_{\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}$	$-\infty < \mu < \bar{x} + z_\alpha \cdot \frac{\sigma}{\sqrt{n}}$
		Bên trái	$\varepsilon = z_{\alpha/2} \cdot \frac{\sigma}{\sqrt{n}}$	$\bar{x} - z_\alpha \cdot \frac{\sigma}{\sqrt{n}} < \mu < \infty$

- Giả định (1): $X_i \sim N(\mu, \sigma^2)$, đã biết σ
- Giả định (2): $X_i \sim N(\mu, \sigma^2)$, chưa biết σ
- Giả định (3): Phân phối tùy ý, mẫu lớn $(n \geq 30)$

Lưu ý: Đối với bài toán ước lượng trung bình dạng (3), trường hợp chưa biết σ thì thay bằng s.

2.3.4 Ước lượng khoảng tin cậy - bài toán 2 mẫu

Dạng	Loại	Ngưỡng sai số	Khoảng tin cậy
$p_1 - p_2$	Đối xứng	$\varepsilon = z_{\alpha/2} \cdot \sqrt{\frac{f_1(1-f_1)}{n_1} + \frac{f_2(1-f_2)}{n_2}}$	$(f_1 - f_2 - \varepsilon; f_1 - f_2 + \varepsilon)$
	Bên trái	$\varepsilon_1 = z_\alpha \cdot \sqrt{\frac{f_1(1-f_1)}{n_1} + \frac{f_2(1-f_2)}{n_2}}$	$(f_1 - f_2 - \varepsilon_1; +\infty)$
	Bên phải	$\varepsilon_1 = z_\alpha \cdot \sqrt{\frac{f_1(1-f_1)}{n_1} + \frac{f_2(1-f_2)}{n_2}}$	$(-\infty; f_1 - f_2 + \varepsilon_1)$
$\mu_1 - \mu_2(1)$	Đối xứng	$\varepsilon = z_{\alpha/2} \cdot \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$	$(\bar{x}_1 - \bar{x}_2 - \varepsilon; \bar{x}_1 - \bar{x}_2 + \varepsilon)$
	Bên trái	$\varepsilon_1 = z_\alpha \cdot \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$	$(\bar{x}_1 - \bar{x}_2 - \varepsilon_1; +\infty)$
	Bên phải	$\varepsilon_1 = z_\alpha \cdot \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$	$(-\infty; \bar{x}_1 - \bar{x}_2 + \varepsilon_1)$
$\mu_1 - \mu_2(2)$	Đối xứng	$\varepsilon = t_{(n_1+n_2-2),\alpha/2} \cdot \sqrt{\frac{S_p^2}{n_1} + \frac{S_p^2}{n_2}}$	$(\bar{x}_1 - \bar{x}_2 - \varepsilon; \bar{x}_1 - \bar{x}_2 + \varepsilon)$
	Bên trái	$\varepsilon_1 = t_{(n_1+n_2-2),\alpha} \cdot \sqrt{\frac{S_p^2}{n_1} + \frac{S_p^2}{n_2}}$	$(\bar{x}_1 - \bar{x}_2 - \varepsilon_1; +\infty)$
	Bên phải	với $S_p^2 = \frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2}$	$(-\infty; \bar{x}_1 - \bar{x}_2 + \varepsilon_1)$

Dạng	Loại	Ngưỡng sai số	Khoảng tin cậy
$\mu_1-\mu_2(3)$	Đối xứng	$\varepsilon = t^{(v)}_{\alpha/2}\cdot\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}$	$(\bar{x}_1-\bar{x}_2-\varepsilon;\bar{x}_1-\bar{x}_2+\varepsilon)$
	Bên trái	$\varepsilon_1 = t^{(v)}_{\alpha}\cdot\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}$	$(\bar{x}_1-\bar{x}_2-\varepsilon_1;+\infty)$
	Bên phải	$\varepsilon_1 = t^{(v)}_{\alpha}\cdot\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}$ với $v=\frac{\left(\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}\right)^2}{\frac{\left(\frac{s_1^2}{n_1}\right)^2}{n_1-1}+\frac{\left(\frac{s_2^2}{n_2}\right)^2}{n_2-1}}$	$(-\infty;\bar{x}_1-\bar{x}_2+\varepsilon_1)$
$\mu_1-\mu_2(4)$	Đối xứng	$\varepsilon = z_{\alpha/2}\cdot\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}$	$(\bar{x}_1-\bar{x}_2-\varepsilon;\bar{x}_1-\bar{x}_2+\varepsilon)$
	Bên trái	$\varepsilon_1 = z_{\alpha}\cdot\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}$	$(\bar{x}_1-\bar{x}_2-\varepsilon_1;+\infty)$
	Bên phải	$\varepsilon_1 = z_{\alpha}\cdot\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}$	$(-\infty;\bar{x}_1-\bar{x}_2+\varepsilon_1)$
$\mu_1-\mu_2(5)$	Đối xứng	$\varepsilon = z_{\alpha/2}\cdot\frac{\sigma_D}{\sqrt{n}}$	$(\bar{x}_1-\bar{x}_2-\varepsilon;\bar{x}_1-\bar{x}_2+\varepsilon)$
	Bên trái	$\varepsilon_1 = z_{\alpha}\cdot\frac{\sigma_D}{\sqrt{n}}$	$(\bar{x}_1-\bar{x}_2-\varepsilon_1;+\infty)$
	Bên phải	$\varepsilon_1 = z_{\alpha}\cdot\frac{\sigma_D}{\sqrt{n}}$	$(-\infty;\bar{x}_1-\bar{x}_2+\varepsilon_1)$
$\mu_1-\mu_2(6)$	Đối xứng	$\varepsilon = t^{(n-1)}_{\alpha/2}\cdot\frac{s_D}{\sqrt{n}}$	$(\bar{x}_1-\bar{x}_2-\varepsilon;\bar{x}_1-\bar{x}_2+\varepsilon)$
	Bên trái	$\varepsilon_1 = t^{(n-1)}_{\alpha}\cdot\frac{s_D}{\sqrt{n}}$	$(\bar{x}_1-\bar{x}_2-\varepsilon_1;+\infty)$
	Bên phải	$\varepsilon_1 = t^{(n-1)}_{\alpha}\cdot\frac{s_D}{\sqrt{n}}$	$(-\infty;\bar{x}_1-\bar{x}_2+\varepsilon_1)$
$\mu_1-\mu_2(7)$	Đối xứng	$\varepsilon = z_{\alpha/2}\cdot\frac{\sigma_D}{\sqrt{n}}$	$(\bar{x}_1-\bar{x}_2-\varepsilon;\bar{x}_1-\bar{x}_2+\varepsilon)$
	Bên trái	$\varepsilon_1 = z_{\alpha}\cdot\frac{\sigma_D}{\sqrt{n}}$	$(\bar{x}_1-\bar{x}_2-\varepsilon_1;+\infty)$
	Bên phải	$\varepsilon_1 = z_{\alpha}\cdot\frac{\sigma_D}{\sqrt{n}}$	$(-\infty;\bar{x}_1-\bar{x}_2+\varepsilon_1)$

Ghi chú:

- Đối với dạng (4): TH chưa biết σ_1,σ_2, ta thay bằng s_1,s_2 trong công thức sai số.
- Đối với dạng (7): Trường hợp chưa biết σ_D, ta thay bằng s_D trong công thức tính sai số.

Quy ước các trường hợp ước lượng cho hiệu hai trung bình:

(1) 2 mẫu độc lập, X_1,X_2 có phân phối Chuẩn, đã biết σ_1^2,σ_2^2

(2) 2 mẫu độc lập, X_1,X_2 có phân phối Chuẩn, chưa biết σ_1^2,σ_2^2, với giả thiết $\sigma_1^2=\sigma_2^2$

(3) 2 mẫu độc lập, X_1,X_2 có phân phối Chuẩn, chưa biết σ_1^2,σ_2^2, với giả thiết $\sigma_1^2\neq\sigma_2^2$

(4) 2 mẫu độc lập, X_1,X_2 có phân phối tùy ý, $n_1\geq 30$ và $n_2\geq 30$

(5) 2 mẫu phụ thuộc tương ứng theo cặp, X_1,X_2 có phân phối Chuẩn, đã biết σ_D^2

(6) 2 mẫu phụ thuộc tương ứng theo cặp, X_1,X_2 có phân phối Chuẩn, chưa biết σ_D^2

(7) 2 mẫu phụ thuộc tương ứng theo cặp, X_1,X_2 có phân phối tùy ý, $n_1\geq 30$ và $n_2\geq 30$

2.4 Kiểm định giả thuyết thống kê

2.4.1 Khái niệm chung về kiểm định

Trong thống kê, kiểm định (*hypothesis testing*) là quá trình đánh giá một giả thuyết về dữ liệu để xác định xem liệu có đủ bằng chứng để chấp nhận hay bác bỏ giả thuyết đó. Mục tiêu của kiểm định là đưa ra quyết định dựa trên dữ liệu mẫu có sẵn để rút ra những kết luận về tổng thể.

Quá trình kiểm định thường bắt đầu bằng việc xây dựng hai giả thuyết:

- **Giả thuyết không (null hypothesis, ký hiệu** H_0**)**: Đây là giả thuyết mà chúng ta muốn kiểm tra hoặc bác bỏ. Thông thường, giả thuyết không cho rằng không có sự khác biệt hoặc tác động đáng kể giữa các nhóm hoặc biến số trong tổng thể.
- **Giả thuyết thay thế (alternative hypothesis, ký hiệu** H_1**)**: Đây là giả thuyết mà chúng ta muốn chấp nhận nếu có đủ bằng chứng để bác bỏ giả thuyết H_0. Giả thuyết thay thế thường cho rằng có sự khác biệt hoặc tác động đáng kể giữa các nhóm hoặc biến số trong tổng thể.

Miền Chấp Nhận (Acceptance Region): Là phạm vi giá trị của thống kê kiểm định mà ở đó, giả thuyết H_0 được chấp nhận. Nói cách khác, không có bằng chứng đủ mạnh để bác bỏ giả thuyết H_0.

Miền Bác Bỏ (Rejection Region): Là phạm vi giá trị của thống kê kiểm định mà ở đó, giả thuyết H_0 được bác bỏ và giả thuyết đối (H_1) được chấp nhận. Các giá trị trong miền này chỉ ra rằng có sự khác biệt đáng kể so với những gì giả định bởi H_0.

2.4.2 Tiêu chuẩn kiểm định - Miền bác bỏ

Xét một bài toán kiểm định giả thuyết H_0 và đối thuyết H_1. Giả sử rằng H_0, từ mẫu ngẫu nhiên $X = (X_1, X_2, \ldots, X_n)$. Ta chọn hàm $Z = h(X_1, X_2, \ldots, X_n; \theta_0)$ sao cho với số $\alpha > 0$ bất kỳ ta có thể tìm được tập hợp W_α thỏa điều kiện:

$$P(Z \in W_\alpha) = \alpha$$

Đại lượng ngẫu nhiên $Z = h(X_1, X_2, \ldots, X_n; \theta_0)$ được gọi là tiêu chuẩn kiểm định giả thuyết H_0 (hay còn gọi là giá trị quan sát). Tập hợp W_α được gọi là miền bác bỏ giả thuyết H_0 (có thể hiểu là tập hợp những tiêu chuẩn kiểm định mà xảy ra thì khi đó giả thuyết H_0 bị bác bỏ), phần còn lại của W_α được gọi là miền chấp nhận. Giá trị α được gọi là mức ý nghĩa của bài toán kiểm định.

Một ký hiệu khác của miền bác bỏ giả thuyết H_0: **RR (Reject Region)**

2.4.3 Các sai lầm trong bài toán kiểm định

Trong bài toán kiểm định giả thuyết thống kê, ta có thể mắc phải các sai lầm sau:

- **Sai lầm loại I**: là sai lầm mắc phải khi ta bác bỏ giả thuyết H_0 trong khi thực tế giả thuyết H_0 đúng. Sai lầm loại I ký hiệu là α, chính là mức ý nghĩa của bài toán kiểm định.
- **Sai lầm loại II**: là sai lầm mắc phải khi ta chấp nhận giả thuyết H_0 trong khi thực tế giả thuyết H_0 sai. Sai lầm loại II ký hiệu là β.

	H_0 đúng	H_0 sai
Không bác bỏ H_0	Không có sai lầm $(1-\alpha)$	Sai lầm loại II β
Bác bỏ H_0	Sai lầm loại I α	Không có sai lầm $(1-\beta)$

2.4.4 Các bước thực hiện kiểm định

1. Phát biểu giả thuyết và đối thuyết của bài toán.
2. Tính giá trị thống kê kiểm định (tiêu chuẩn kiểm định) cho bài toán.
3. Xác định miền bác bỏ tốt nhất cho bài toán.
4. Đưa ra kết luận.

2.4.5 Kiểm định một mẫu

Kiểm định tỷ lệ 1 mẫu

Nếu ta so sánh tỷ lệ tổng thể p với giá trị cho trước gọi là p_0 thì bài toán này được gọi là kiểm định tỷ lệ 1 mẫu (Có nghĩa là dựa vào mẫu thu thập được, kiểm định để so sánh tỷ lệ tổng thể p với p_0 cho trước)

Điều kiện: $n > 30$	**Tiêu chuẩn kiểm định**	**Miền bác bỏ**
$H_0 : p = p_0$ $H_1 : p \neq p_0$	$z = \dfrac{p - p_0}{\sqrt{\dfrac{p_0(1-p_0)}{n}}}$	$RR = (-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; \infty)$
$H_0 : p \geq p_0$ $H_1 : p < p_0$	$z = \dfrac{p - p_0}{\sqrt{\dfrac{p_0(1-p_0)}{n}}}$	$RR = (-\infty; -z_{\alpha})$
$H_0 : p \leq p_0$ $H_1 : p > p_0$	$z = \dfrac{p - p_0}{\sqrt{\dfrac{p_0(1-p_0)}{n}}}$	$RR = (z_{\alpha}; \infty)$

Với $np \geq 5$ và $n(1-p) \geq 5$ thì ta có phân phối (xấp xỉ) của tỷ lệ mẫu

$$\hat{P} \approx N\left(p, \frac{p(1-p)}{n}\right)$$

Kiểm định trung bình 1 mẫu

Nếu ta so sánh trung bình tổng thể μ với giá trị cho trước gọi là μ_0 thì bài toán này được gọi là kiểm định trung bình 1 mẫu (Có nghĩa là dựa vào mẫu thu thập được, kiểm định để so sánh trung bình tổng thể μ với μ_0 cho trước).

.

H_0	H_1	TCKĐ	Miền bác bỏ
(1) Trường hợp tổng thể có phân phối chuẩn, đã biết σ^2			
$\mu = \mu_0$	$\mu \neq \mu_0$	$z_{qs} = \dfrac{\bar{x} - \mu_0}{\sigma/\sqrt{n}}$	$RR = (-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; +\infty)$
$\mu = \mu_0$	$\mu < \mu_0$		$RR = (-\infty; -z_{\alpha})$
$\mu = \mu_0$	$\mu > \mu_0$		$RR = (z_{\alpha}; +\infty)$
(2) Trường hợp tổng thể có phân phối chuẩn, chưa biết σ^2			
$\mu = \mu_0$	$\mu \neq \mu_0$	$t_{qs} = \dfrac{\bar{x} - \mu_0}{s/\sqrt{n}}$	$RR = (-\infty; -t_{\alpha/2;n-1}) \cup (t_{\alpha/2;n-1}; +\infty)$
$\mu = \mu_0$	$\mu < \mu_0$		$RR = (-\infty; -t_{\alpha;n-1})$
$\mu = \mu_0$	$\mu > \mu_0$		$RR = (t_{\alpha;n-1}; +\infty)$
(3) Trường hợp tổng thể có phân phối tùy ý, $n \geq 30$			
$\mu = \mu_0$	$\mu \neq \mu_0$	$z_{qs} = \dfrac{\bar{x} - \mu_0}{\sigma/\sqrt{n}}$	$RR = (-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; +\infty)$
$\mu = \mu_0$	$\mu < \mu_0$		$RR = (-\infty; -z_{\alpha})$
$\mu = \mu_0$	$\mu > \mu_0$		$RR = (z_{\alpha}; +\infty)$

2.4.6 Kiểm định hai mẫu

Kiểm định tỷ lệ hai mẫu:

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

Tính thống kê kiểm định:

$$z_0 = \frac{f_1 - f_2}{\sqrt{\frac{\bar{f}(1-\bar{f})}{\tilde{n}}}}$$

với

$$\bar{f} = \frac{m_1 + m_2}{n_1 + n_2}, \quad \tilde{n} = \frac{n_1 \times n_2}{n_1 + n_2}$$

Kiểm định trung bình 2 mẫu:

H_0	H_1	TCKĐ	Miền bác bỏ
(1) Kiểm định trung bình 2 mẫu - trường hợp đã biết σ_1^2, σ_2^2			
$\mu = \mu_0$	$\mu \neq \mu_0$	$z_0 = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$	$RR = (-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; +\infty)$
$\mu = \mu_0$	$\mu < \mu_0$		$RR = (-\infty; -z_\alpha)$
$\mu = \mu_0$	$\mu > \mu_0$		$RR = (z_\alpha; +\infty)$
(2) Kiểm định trung bình 2 mẫu, mẫu lớn $n_1 \geq 30$ và $n_2 \geq 30$			
$\mu = \mu_0$	$\mu \neq \mu_0$	$z_0 = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$	$RR = (-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; +\infty)$
$\mu = \mu_0$	$\mu < \mu_0$		$RR = (-\infty; -z_\alpha)$
$\mu = \mu_0$	$\mu > \mu_0$		$RR = (z_\alpha; +\infty)$
(3) Kiểm định trung bình 2 mẫu - trường hợp chưa biết σ_1^2, σ_2^2 hoặc $n_1 \leq 30$ và $n_2 \leq 30$			
Trường hợp 1: $\sigma_1^2 = \sigma_2^2$			
$H_0 : \mu_1 = \mu_2$	$H_1 : \mu_1 \neq \mu_2$	$t_0 = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s^2}{n_1} + \frac{s^2}{n_2}}}$	$RR = (-\infty; -t_{\alpha/2;df}) \cup (t_{\alpha/2;df}; +\infty)$
$H_0 : \mu_1 = \mu_2$	$H_1 : \mu_1 < \mu_2$		$RR = (-\infty; -t_{\alpha;df})$
$H_0 : \mu_1 = \mu_2$	$H_1 : \mu_1 > \mu_2$		$RR = (t_{\alpha;df}; +\infty)$

Với: $df = n_1 + n_2 - 2$

$$t_0 = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \quad \text{với } S^2 = \frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1 + n_2 - 2}$$

Với:

$$S^2 = \frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1 + n_2 - 2} \quad \text{và} \quad df = n_1 + n_2 - 2$$

Trường hợp 2: $\sigma_1^2 \neq \sigma_2^2$

$$df = \frac{\left(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}\right)^2}{\frac{\left(\frac{S_1^2}{n_1}\right)^2}{n_1 - 1} + \frac{\left(\frac{S_2^2}{n_2}\right)^2}{n_2 - 1}}$$

lấy df làm tròn số nguyên theo nguyên tắc quá bán.

$$t_0 = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$$

(4) Kiểm định trung bình 2 mẫu không độc lập

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$$

$$s_d^2 = \frac{1}{n-1}\sum_{i=1}^{n}(d_i - \bar{d})^2$$

Giả thuyết	Miền bác bỏ (1)
$H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 \neq \mu_2$	$RR = (-\infty; -t_{\alpha/2;df}) \cup (t_{\alpha/2;df}; +\infty)$
$H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 < \mu_2$	$RR = (-\infty; -t_{\alpha;df})$
$H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 > \mu_2$	$RR = (t_{\alpha;df}; +\infty)$

Giả thuyết	Miền bác bỏ (1)
$H_0 : \mu_D = D_0$ $H_1 : \mu_D \neq D_0$	$RR = (-\infty; -t_{\alpha/2,n-1}) \cup (t_{\alpha/2,n-1}; +\infty)$
$H_0 : \mu_D = D_0$ $H_1 : \mu_D > D_0$	$RR = (-\infty; -t_{\alpha;n-1})$
$H_0 : \mu_D = D_0$ $H_1 : \mu_D < D_0$	$RR = (t_{\alpha;n-1}; +\infty)$

Thông thường: $D_0 = 0$

$$t_0 = \frac{\bar{d} - D_0}{\frac{s_d}{\sqrt{n}}}$$

với

$$d_i = x_i - y_i \quad ; \quad \bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i$$

$$s_d^2 = \frac{1}{n-1}\sum_{i=1}^{n}(d_i - \bar{d})^2$$

2.5 Phân tích phương sai (ANOVA)

Mục tiêu của phân tích phương sai là so sánh trung bình của nhiều nhóm (tổngthể) dựa trên các số trung bình của các mẫu quan sát từ các nhóm này và thông qua kiểm định giả thuyết để kết luận về sự bằng nhau của các số trung bình này. Trong nghiên cứu, phân tích phương sai được dùng như là một công cụ để xem xét ảnh hưởng của một hay một số yếu tố nguyên nhân (định tính) đến một yếu tố kết quả (định lượng).

Ta có các mô hình phân tích phương sai như sau: phân tích phương sai một nhân tố, 2 nhân tố và 3 nhân tố. Cụm từ nhân tố cho ta số lượng nhân tố nguyên nhân ảnh hưởng đến kết quả ta nghiên cứu.

Phân tích phương sai một yếu tố là phân tích ảnh hưởng của một yếu tố nguyên nhân (dạng biến định tính) đến một yếu tố kết quả (dạng biến định lượng) đang nghiên cứu.

Giả sử cần so sánh số trung bình của k tổng thể độc lập trên những mẫu ngẫu nhiên và độc lập $n_1, n_2, \ldots, n_k$ quan sát từ k tổng thể này. Cần ghi nhớ ba giả định sau đây để tiến hành phân tích Anova:

- Các tổng thể này có phân phối chuẩn.
- Các phương sai của tổng thể bằng nhau.
- Các quan sát được lấy mẫu là độc lập.

Nếu trung bình các tổng thể được ký hiệu là $\mu_1, \mu_2, \ldots, \mu_k$ thì ta có các giả thiết trong mô hình phân tích phương sai như sau:

$$H_0 : \mu_1 = \mu_2 = \mu_3 = \cdots = \mu_k$$

(Giả thiết H_0 cho rằng trung bình của k tổng thể bằng nhau, tức nhân tố nguyên nhân không có tác động gì đến nhân tố kết quả ta đang nghiên cứu).

$$H_1 : \exists \mu_i \neq \mu_j \quad (i \neq j)$$

(Giả thiết H_1 cho rằng có ít nhất 2 giá trị trung bình ở các tổng thể khác nhau, tức nhân tố nguyên nhân có tác động đến nhân tố kết quả ta đang nghiên cứu). Các bước tiến hành phân tích phương sai một nhân tố:

Bước 1: Tính trung bình mẫu của các nhóm $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_k$ theo công thức:

$$\bar{x}_i = \frac{\sum_{j=1}^{n_i} x_{ij}}{n_i}, \quad (i = 1, 2, \ldots, k)$$

Trung bình chung của k mẫu được tính theo công thức:

$$\bar{x} = \frac{\sum_{i=1}^{k} n_i \bar{x}_i}{\sum_{i=1}^{k} n_i} = \frac{\sum_{i=1}^{k} n_i \bar{x}_i}{n_1 + n_2 + \cdots + n_k}$$

Bước 2: Tính tổng các bình phương lệch (tổng bình phương):

- Tổng các độ lệch bình phương trong nội bộ nhóm (SSW):

$$\text{SSW} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)^2$$

- Tổng các độ lệch bình phương giữa các nhóm (SSB):

$$\text{SSB} = \sum_{i=1}^{k} n_i (\bar{x}_i - \bar{x})^2$$

- Tổng các độ lệch bình phương của toàn bộ tổng thể (SST):

$$\text{SST} = \text{SSW} + \text{SSB} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \bar{x})^2$$

Bước 3: Tính các phương sai:

$$\text{MSW} = \frac{\text{SSW}}{N-k}, \quad \text{MSB} = \frac{\text{SSB}}{k-1}$$

Bước 4: Tính thống kê kiểm định (tiêu chuẩn kiểm định, giá trị quan sát):

$$F = \frac{\text{MSB}}{\text{MSW}}$$

Bước 5: Xác định miền bác bỏ của bài toán:

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

Tìm giá trị $F_{\alpha;k-1;N-k}$: tra bảng Fisher mức ý nghĩa α và cột $k-1$ và dòng $N-k$.

Bước 6: Đưa ra kết luận:

$$\text{Nếu } F > F_{\alpha;k-1;N-k} \iff F \in RR \Rightarrow \text{Bác bỏ } H_0, \text{ chấp nhận } H_1$$

$$\text{Nếu } F < F_{\alpha;k-1;N-k} \iff F \notin RR \Rightarrow \text{không bác bỏ } H_0 \text{ (chưa bác bỏ được } H_0, \text{ chấp nhận } H_0)$$

Nguồn của sự biến thiên	SS	df	MS	F
Giữa các nhóm	SSB	$k-1$	MSB	$F = \frac{\text{MSB}}{\text{MSW}}$
Trong từng nhóm	SSW	$N-k$	MSW	
Toàn bộ	SST	$N-1$		

2.6 Hồi quy tuyến tính bội

Hồi quy tuyến tính là một phương pháp phân tích quan hệ giữa biến phụ thuộc Y với một hay nhiều biến độc lập X. Mô hình hóa sử dụng hàm tuyến tính. Các tham số của mô hình (hay hàm số) được ước lượng từ dữ liệu.

Trong hồi quy tuyến tính đơn, chỉ có một biến độc lập được sử dụng để dự đoán biến phụ thuộc. Tuy nhiên, trong hồi quy tuyến tính bội, có nhiều hơn một biến độc lập được sử dụng.

Mô hình hồi quy tuyến tính bội giả định rằng có mối quan hệ tuyến tính giữa biến phụ thuộc và các biến độc lập. Mục tiêu của hồi quy tuyến tính bội là tìm ra một phương trình tuyến tính tốt nhất để dự đoán giá trị của biến phụ thuộc dựa trên các giá trị của các biến độc lập.

Phương trình hồi quy tuyến tính bội có dạng:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon$$

Trong đó:

- Y là biến phụ thuộc (dependent variable): Đây là biến chúng ta muốn dự đoán hoặc giải thích. Trong một mô hình hồi quy tuyến tính bội, biến này là cố gắng tìm một phương trình tuyến tính để dự đoán giá trị của biến phụ thuộc Y dựa trên các biến độc lập $X_1, X_2, \ldots, X_n$.
- $X_1, X_2, \ldots, X_n$ là các biến độc lập (independent variables): Đây là các đại lượng mà chúng ta sử dụng để dự đoán biến phụ thuộc Y. Mỗi biến độc lập X_i (i = 1 đến n) có thể là một thuộc tính, một đặc trưng hoặc một biến số mà chúng ta cho là có ảnh hưởng đến biến phụ thuộc.
- $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ là các hệ số hồi quy (coefficients): Đây là các tham số trong phương trình tuyến tính. Các hệ số hồi quy đại diện cho mức độ ảnh hưởng của từng biến độc lập đến biến phụ thuộc. Hệ số β_0 (cũng được gọi là hệ số điều chỉnh) là hệ số mức độ tự do của mô hình, tức là giá trị dự đoán của biến phụ thuộc khi tất cả các biến độc lập đều bằng 0.
- ε là sai số ngẫu nhiên (error term): Đây là thành phần không thể giải thích bằng các biến độc lập trong mô hình. Nó đại diện cho sự biến động tự nhiên và các yếu tố không xác định khác mà có thể ảnh hưởng đến biến phụ thuộc. Sai số ngẫu nhiên ε được giả định tuân theo phân phối chuẩn với giá trị trung bình bằng 0.

2.6.1 Hệ số xác định (R-square)

Hệ số xác định (R-square) được biểu diễn bằng công thức toán học sau đây:

$$R^2 = \frac{\text{Sum of Squares Regression}}{\text{Sum of Squares Total}}$$

Trong đó:

- **SSR** (Sum of Squares Regression): SSR đo lường tổng phương sai giữa dữ liệu quan sát và giá trị dự đoán bởi mô hình hồi quy tuyến tính. Nó đại diện cho phần biệt chênh lệch giữa dữ liệu thực tế và dự đoán của mô hình. SSR thường được sử dụng để đo lường mức độ phù hợp của mô hình hồi quy với dữ liệu.
- **SST** (Sum of Squares Total): SST đo lường tổng phương sai của dữ liệu quan sát so với giá trị trung bình của dữ liệu. Nó đại diện cho phần biệt chênh lệch giữa dữ liệu thực tế và giá trị trung bình. SST thường được sử dụng để đo lường phạm vi biến thiên của dữ liệu ban đầu.

R-square có ý nghĩa như sau:

- R-square = 0: Điều này cho thấy không có sự biến đổi nào trong biến phần hồi có thể được giải thích bởi các biến giải thích trong mô hình. Mô hình không thể dự đoán kết quả.

- *R*-square = 1: Điều này cho thấy toàn bộ sự biến thiên trong biến phần hồi có thể được giải thích bởi các biến giải thích trong mô hình. Mô hình có thể dự đoán kết quả một cách chính xác mà không có sai số từ các biến độc lập.
- *R*-square nằm giữa 0 và 1: Điều này cho thấy một phần của sự biến thiên trong biến phần hồi có thể được giải thích bởi các biến giải thích trong mô hình. Giá trị càng gần 1 thì mô hình càng tốt trong việc giải thích biến đổi của biến phần hồi. Tuy nhiên, *R*-square có hạn chế vì nó không đánh giá tính chất hay ý nghĩa của các biến độc lập. Nó chỉ đo lường mức độ biến thiên có thể được giải thích bởi mô hình, không xác định xem các biến giải thích nào nên được bao gồm trong mô hình. Do đó, việc đánh giá tính phù hợp của mô hình cần dựa vào *R*-square bằng cách thêm các biến giải thích.

3 Tiền xử lí số liệu

3.1 Đọc dữ liệu

Đọc dữ liệu từ file "All_GPUs.csv" vào gpu_data, sau đó in ra 3 dòng đầu của mỗi cột bằng head:

Code

```
gpu_data <- read.csv("D:/BTLXSTK/All_GPUs.csv")
head(gpu_data, 3)
```

Result

```
  Architecture Best_Resolution Boost_Clock Core_Speed DVI_Connection Dedicated Direct_X DisplayPort_Connection HDMI_Connection Integrated L2_Cache Manufacturer
1  Tesla G92b                                    738 MHz             2       Yes  DX 10.0                    NA                0          No       0KB         Nvidia
2     R600 XT       1366 x 768                    \n-                 2       Yes    DX 10                    NA                0          No       0KB            AMD
3    R600 PRO       1366 x 768                    \n-                 2       Yes    DX 10                    NA                0          No       0KB            AMD
  Max_Power   Memory Memory_Bandwidth Memory_Bus Memory_Speed Memory_Type                Name Notebook_GPU Open_GL                                  PSU  Pixel_Rate Power_Connector
1 141 Watts 1024 MB          64GB/sec   256 Bit      1000 MHz        GDDR3      GeForce GTS 150           No      3.3 450 Watt & 38 Amps 12 GPixel/s            None
2 215 Watts  512 MB         106GB/sec   512 Bit       828 MHz        GDDR3 Radeon HD 2900 XT 512MB           No      3.1 550 Watt & 35 Amps 12 GPixel/s            None
3 200 Watts  512 MB        51.2GB/sec   256 Bit       800 MHz        GDDR3      Radeon HD 2900 Pro           No      3.1 550 Watt & 35 Amps 10 GPixel/s            None
  Process ROPs   Release_Date Release_Price Resolution_WxH SLI_Crossfire Shader TMUs Texture_Rate VGA_Connection
1    55nm   16 \n01-Mar-2009                    2560x1600            Yes      4   64  47 GTexel/s              0
2    80nm   16 \n14-May-2007                    2560x1600            Yes      4   16  12 GTexel/s              0
3    80nm   16 \n07-Dec-2007                    2560x1600            Yes      4   16  10 GTexel/s              0
```

Tiếp theo, ta tìm số quan sát và số cột (biến) của dữ liệu:

Code

```
> dim(gpu_data)
[1] 3406   34
```

Ta thấy dữ liệu có 3406 quan sát và 34 cột.

3.2 Làm sạch dữ liệu

3.2.1 Chọn lọc dữ liệu

Ta chọn các biến quan trọng và lưu nó vào data:

Code

```
data<-gpu_data[, c("Manufacturer", "Memory_Bandwidth",
      "Release_Date","Memory_Bus", "Texture_Rate", "Memory_Speed")]
```

Kết quả thu được:

Result

	Manufacturer	Memory_Bandwidth	Release_Date	Memory_Bus	Texture_Rate	Memory_Speed
1	Nvidia	64GB/sec	01-Mar-2009	256 Bit	47 GTexel/s	1000 MHz
2	AMD	106GB/sec	14-May-2007	512 Bit	12 GTexel/s	828 MHz
3	AMD	51.2GB/sec	07-Dec-2007	256 Bit	10 GTexel/s	800 MHz
4	AMD	36.8GB/sec	01-Jul-2007	128 Bit	7 GTexel/s	1150 MHz
5	AMD	22.4GB/sec	28-Jun-2007	128 Bit	6 GTexel/s	700 MHz
6	AMD	35.2GB/sec	26-Jun-2007	128 Bit	6 GTexel/s	1100 MHz
7	AMD	134.4GB/sec	13-Jul-2009	256 Bit	35 GTexel/s	1050 MHz
8	AMD	51.2GB/sec	06-Nov-2007	256 Bit	7 GTexel/s	800 MHz
9	AMD	160GB/sec	18-Jan-2014	256 Bit	62 GTexel/s	1250 MHz
10	AMD	2.9GB/sec	02-Jan-2001	64 Bit		366 MHz
11	Nvidia	5.2GB/sec	01-Nov-2002	128 Bit	2 GTexel/s	325 MHz
12	Nvidia	177.6GB/sec	25-Jul-2011	384 Bit	36 GTexel/s	925 MHz
13	Nvidia	168GB/sec	01-Nov-2012	320 Bit		1050 MHz
14	Nvidia	288.4GB/sec	12-Nov-2013	384 Bit	169 GTexel/s	1502 MHz
15	AMD	5.8GB/sec	22-Jan-2002	128 Bit		360 MHz
16	AMD	57.6GB/sec	28-May-2015	128 Bit	62 GTexel/s	900 MHz
17	Nvidia	320GB/sec	15-May-2012	512 Bit		1250 MHz
18	Nvidia	249.6GB/sec	12-Nov-2012	384 Bit		1300 MHz
19	Intel	34.1GB/sec	01-Sep-2015	128 Bit	8 GTexel/s	1067 MHz

3.2.2 Thêm và xóa biến

Ta xóa biến Release_Date ra khỏi data, thay vào đó, ta thêm 1 biến Year chỉ để lưu năm ra mắt, dễ dàng xử lý số liệu hơn

Code

```
library(lubridate)
data$Year <- year(dmy(data$Release_Date))
data$Release_Date <- NULL
View(data)
```

Kết quả thu được:

Result

	Manufacturer	Memory_Bandwidth	Memory_Bus	Texture_Rate	Memory_Speed	Year
1	Nvidia	64GB/sec	256 Bit	47 GTexel/s	1000 MHz	2009
2	AMD	106GB/sec	512 Bit	12 GTexel/s	828 MHz	2007
3	AMD	51.2GB/sec	256 Bit	10 GTexel/s	800 MHz	2007
4	AMD	36.8GB/sec	128 Bit	7 GTexel/s	1150 MHz	2007
5	AMD	22.4GB/sec	128 Bit	6 GTexel/s	700 MHz	2007
6	AMD	35.2GB/sec	128 Bit	6 GTexel/s	1100 MHz	2007
7	AMD	134.4GB/sec	256 Bit	35 GTexel/s	1050 MHz	2009
8	AMD	51.2GB/sec	256 Bit	7 GTexel/s	800 MHz	2007
9	AMD	160GB/sec	256 Bit	62 GTexel/s	1250 MHz	2014
10	AMD	2.9GB/sec	64 Bit		366 MHz	2001
11	Nvidia	5.2GB/sec	128 Bit	2 GTexel/s	325 MHz	2002
12	Nvidia	177.6GB/sec	384 Bit	36 GTexel/s	925 MHz	2011
13	Nvidia	168GB/sec	320 Bit		1050 MHz	2012
14	Nvidia	288.4GB/sec	384 Bit	169 GTexel/s	1502 MHz	2013
15	AMD	5.8GB/sec	128 Bit		360 MHz	2002
16	AMD	57.6GB/sec	128 Bit	62 GTexel/s	900 MHz	2015
17	Nvidia	320GB/sec	512 Bit		1250 MHz	2012
18	Nvidia	249.6GB/sec	384 Bit		1300 MHz	2012

3.2.3 Xử lý đơn vị trong data

Trước tiên, ta xem xét các loại đơn vị trong từng cột, ta tạo 1 hàm get_unique_units để xử lý và chỉ trả về đơn vị của 1 cột

Code

```
get_unique_units <- function(column) {
  cleaned_column <- gsub("\\.", "", column)  # Loại bỏ dấu chấm
  cleaned_column <- gsub("\\s+", "", cleaned_column)  # Loại bỏ khoảng trắng
  cleaned_column <- gsub("[0-9]", "", cleaned_column)  # Loại bỏ số
  return(unique(cleaned_column[cleaned_column != ""]))  # Trả về các giá trị duy nhất không rỗng
}
```

Sau đó ta in ra các đơn vị có trong từng cột của data

Code

```
unique_units_list <- lapply(data, get_unique_units)
print(unique_units_list)
```

Kết quả thu được đối với 4 biến:

Result

```
$Memory_Bandwidth
[1] "GB/sec" "MB/sec"

$Memory_Bus
[1] "Bit"

$Texture_Rate
[1] "GTexel/s"

$Memory_Speed
[1] "MHz"
```

Ta thấy Memory_Bandwidth có xuất hiện 2 đơn vị là “GB/sec” và “MB/sec”, ta phải đưa về cùng 1 đơn vị để khi xử lý dữ liệu chính xác hơn.
Ta in ra số đơn vị “GB/sec” và “MB/sec” của Memory_Bandwidth theo lệnh:

Code

```
mb_sec_count <- length(grep("MB/sec", data$Memory_Bandwidth))
gb_sec_count <- length(grep("GB/sec", data$Memory_Bandwidth))
cat("Số lượng giá trị có đơn vị MB/sec:", mb_sec_count, "\n")
cat("Số lượng giá trị có đơn vị GB/sec:", gb_sec_count, "\n")
```

Kết quả:

Result

```
> cat("Số lượng giá trị có đơn vị MB/sec:", mb_sec_count, "\n")
Số lượng giá trị có đơn vị MB/sec: 4
> cat("Số lượng giá trị có đơn vị GB/sec:", gb_sec_count, "\n")
Số lượng giá trị có đơn vị GB/sec: 3281
```

Ở đây, vì số lượng đơn vị GB/sec nhiều hơn nên ta đưa về 1 kiểu là “GB/sec”.Ta tạo 1 hàm convert_memory_bandwidth để chuyển đổi từ “MB/sec” sang “GB/sec”:

Code

```
convert_memory_bandwidth <- function(bandwidth) {
  matches <- regmatches(bandwidth, regexec("([0-9.]+)([A-Za-z/]+)", bandwidth))
  if (length(matches) > 0 && length(matches[[1]]) == 3) {
    value <- as.numeric(matches[[1]][2])
    unit <- matches[[1]][3]
    if (unit == "MB/sec") {
      return(paste(value / 1024, "GB/sec", sep = ""))
    } else if (unit == "GB/sec") {
      return(bandwidth)
    }
  }
  return(NA)
}
```

Áp dụng vào Memory_Bandwidth:

Code

```
data$Memory_Bandwidth <- sapply(data$Memory_Bandwidth, convert_memory_bandwidth)
View(data)
```

Kết quả thu được:

Result

	Manufacturer	Memory_Bandwidth	Memory_Bus	Texture_Rate	Memory_Speed	Year
518	AMD	179.2GB/sec	512 Bit	22 GTexel/s	700 MHz	2009
519	AMD	*NA*				2011
520	Nvidia	*NA*	256 Bit			2010
521	Nvidia	16GB/sec	128 Bit		500 MHz	2006
522	Nvidia	22.4GB/sec	128 Bit		700 MHz	2007
523	Nvidia	11.2GB/sec	64 Bit		700 MHz	2008
524	AMD	21.3GB/sec	128 Bit	15 GTexel/s	667 MHz	2013
525	AMD	25.6GB/sec	128 Bit	16 GTexel/s	800 MHz	2012
526	Nvidia	11.2GB/sec	64 Bit		700 MHz	2008
527	AMD	29.9GB/sec	128 Bit	17 GTexel/s	933 MHz	2013
528	Nvidia	28.8GB/sec	128 Bit		900 MHz	2012
529	Nvidia	1.1GB/sec	32 Bit		275 MHz	2002
530	AMD	*NA*	128 Bit			2011
531	AMD	1.6GB/sec	64 Bit		200 MHz	2001
532	Nvidia	51.2GB/sec	256 Bit		800 MHz	2008
533	Nvidia	102.4GB/sec	256 Bit		800 MHz	2008

Sau khi đồng nhất đơn vị, ta sẽ xóa các đơn vị của các biến, để dễ dàng thao tác và xử lý dữ liệu:

Code

```
data$Memory_Bandwidth <- as.numeric(gsub("GB/sec", "", data$Memory_Bandwidth))
data$Memory_Bus <- as.numeric(gsub("Bit", "", data$Memory_Bus))
data$Memory_Speed <- as.numeric(gsub("MHz", "", data$Memory_Speed))
data$Texture_Rate <- as.numeric(gsub("GTexel/s", "", data$Texture_Rate))
```

Kết quả thu được:

Result

	Manufacturer	Memory_Bandwidth	Memory_Bus	Texture_Rate	Memory_Speed	Year
1	Nvidia	64.0	256	47	1000	2009
2	AMD	106.0	512	12	828	2007
3	AMD	51.2	256	10	800	2007
4	AMD	36.8	128	7	1150	2007
5	AMD	22.4	128	6	700	2007
6	AMD	35.2	128	6	1100	2007
7	AMD	134.4	256	35	1050	2009
8	AMD	51.2	256	7	800	2007
9	AMD	160.0	256	62	1250	2014
10	AMD	2.9	64	NA	366	2001
11	Nvidia	5.2	128	2	325	2002
12	Nvidia	177.6	384	36	925	2011
13	Nvidia	168.0	320	NA	1050	2012
14	Nvidia	288.4	384	169	1502	2013

3.3 Xử lý dữ liệu khuyết

Trước tiên, ta xem có bao nhiêu các dữ liệu khuyết và tỉ lệ của chúng trong từng biến:

Code

```
missing_counts <- colSums(is.na(data))
missing_percentage <- (missing_counts / nrow(data))
missing_data <- data.frame(
  Tenbien = names(data),
  soluong = missing_counts,
  tile = missing_percentage
)
print(missing_data)
```

Kết quả:

Result

```
                            Tenbien soluong       tile
Manufacturer           Manufacturer       0 0.000000000
Memory_Bandwidth Memory_Bandwidth     121 0.035525543
Memory_Bus             Memory_Bus      62 0.018203171
Texture_Rate         Texture_Rate     544 0.159718144
Memory_Speed         Memory_Speed     105 0.030827951
Year                         Year      30 0.008807986
```

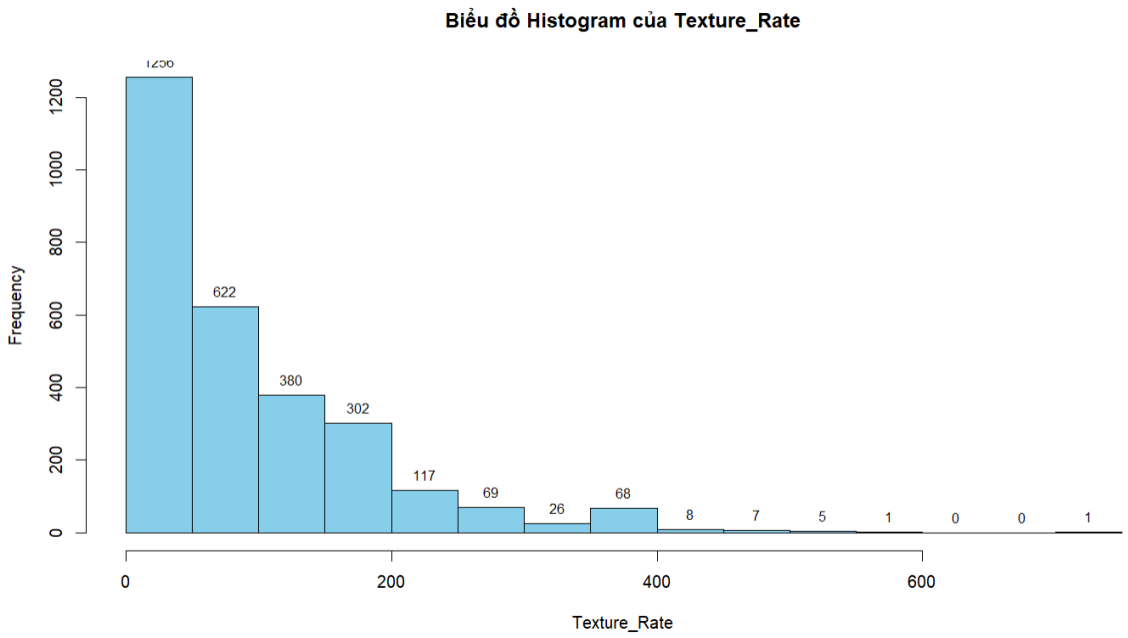
- Qua đó ta có thể thấy rằng đối với các biến "Manufacturer", "Memory_Bandwidth", "Memory_Bus","Memory_Speed" và “Year” thì ta có thể xóa các hàng chứa dữ liệu khuyết, vì tỉ lệ của dữ liệu khuyết là không đáng kể.
- Đối với biến "Texture_Rate", vì tỉ lệ dữ liệu khuyết >15% nên ta sẽ dùng phương pháp thay thế cho từng giá trị “NA”. Ta vẽ biểu đồ Histogram cho "Texture_Rate" như sau:

Code

```
hist_data <- hist(data$Texture_Rate,
                  main = "Biểu đồ Histogram của Texture_Rate",
                  xlab = "Texture_Rate",
                  col = "skyblue",
                  border = "black",
                  ylim = c(0, 1250))  # Tăng giới hạn y

text(hist_data$mids, hist_data$counts, labels = hist_data$counts, pos = 3, cex = 0.8, col = "black")
```

Kết quả:

Result



Từ biểu đồ trên, ta thấy rằng biểu đồ lệch phải, nên để tối ưu độ chính xác, ta thế giá trị trung vị của các hàng còn lại thay vào các giá trị “NA”

Result

```
median_texture_rate <- median(data$Texture_Rate, na.rm = TRUE)
data$Texture_Rate[is.na(data$Texture_Rate)] <- median_texture_rate
```

Kết quả:

Result

```
median_texture_rate <- median(data$Texture_Rate, na.rm = TRUE)
data$Texture_Rate[is.na(data$Texture_Rate)] <- median_texture_rate
```

Result

	Manufacturer	Memory_Bandwidth	Memory_Bus	Texture_Rate	Memory_Speed	Year
1	Nvidia	64.0	256	47	1000	2009
2	AMD	106.0	512	12	828	2007
3	AMD	51.2	256	10	800	2007
4	AMD	36.8	128	7	1150	2007
5	AMD	22.4	128	6	700	2007
6	AMD	35.2	128	6	1100	2007
7	AMD	134.4	256	35	1050	2009
8	AMD	51.2	256	7	800	2007
9	AMD	160.0	256	62	1250	2014
10	AMD	2.9	64	60	366	2001
11	Nvidia	5.2	128	2	325	2002
12	Nvidia	177.6	384	36	925	2011
13	Nvidia	168.0	320	60	1050	2012
14	Nvidia	288.4	384	169	1502	2013
15	AMD	5.8	128	60	360	2002
16	AMD	57.6	128	62	900	2015
17	Nvidia	320.0	512	60	1250	2012
18	Nvidia	249.6	384	60	1300	2012

Result

```
                Tenbien soluong        tile
Manufacturer                  0 0.000000000
Memory_Bandwidth            121 0.035525543
Memory_Bus                   62 0.018203171
Texture_Rate                  0 0.000000000
Memory_Speed                105 0.030827951
Year                         30 0.008807986
```

Các giá trị “NA” của biến "Texture_Rate" đã được thay bằng giá trị trung vị (60)
Sau đó, ta xóa giá trị “NA” đối với các biến còn lại:

Code

```
data <- na.omit(data)
```

Kết quả:

Result

```
                          Tenbien soluong tile
Manufacturer          Manufacturer       0    0
Memory_Bandwidth Memory_Bandwidth       0    0
Memory_Bus             Memory_Bus       0    0
Texture_Rate         Texture_Rate       0    0
Memory_Speed         Memory_Speed       0    0
Year                         Year       0    0
```

Như vậy, sau khi xử lý dữ liệu xong:

Result

```
> dim(data)
[1] 3257    6
```

Ta có tổng cộng 3257 quan sát với 6 biến trong dữ liệu data

4 Thống kê mô tả

4.1 Tổng quan dữ liệu

Ta tạo 1 hàm func để tạo 1 dữ liệu chứa các thông số:

Code

```
func <- function(x) {
  c(GTNN = min(x),
    GTLN = max(x),
    GTTB = mean(x),
    Dolechchuan = sd(x),
    Phuongsai = var(x),
    Trungvi = median(x))
}
```

Sau đó tạo bộ dữ liệu với các biến là số:

Code

```
dulieuso <- sapply(data, is.numeric)
kequa <- apply(data[, dulieuso], 2, func)
print(result)
```

Kết quả:

Result

	Memory_Bandwidth	Memory_Bus	Texture_Rate	Memory_Speed	Year
GTNN	2.7000	32.0000	1.00000	166.0000	2002.000000
GTLN	1280.0000	8192.0000	717.00000	2127.0000	2017.000000
GTTB	154.6952	217.3938	90.91896	1260.1660	2012.918600
Dolechchuan	137.3717	274.1965	91.00426	400.6702	2.462827
Phuongsai	18870.9798	75183.7024	8281.77593	160536.6121	6.065515
Trungvi	115.2000	192.0000	60.00000	1250.0000	2013.000000

4.2 Đồ thị histogram thể hiện phân phối của Memory_Bandwidth

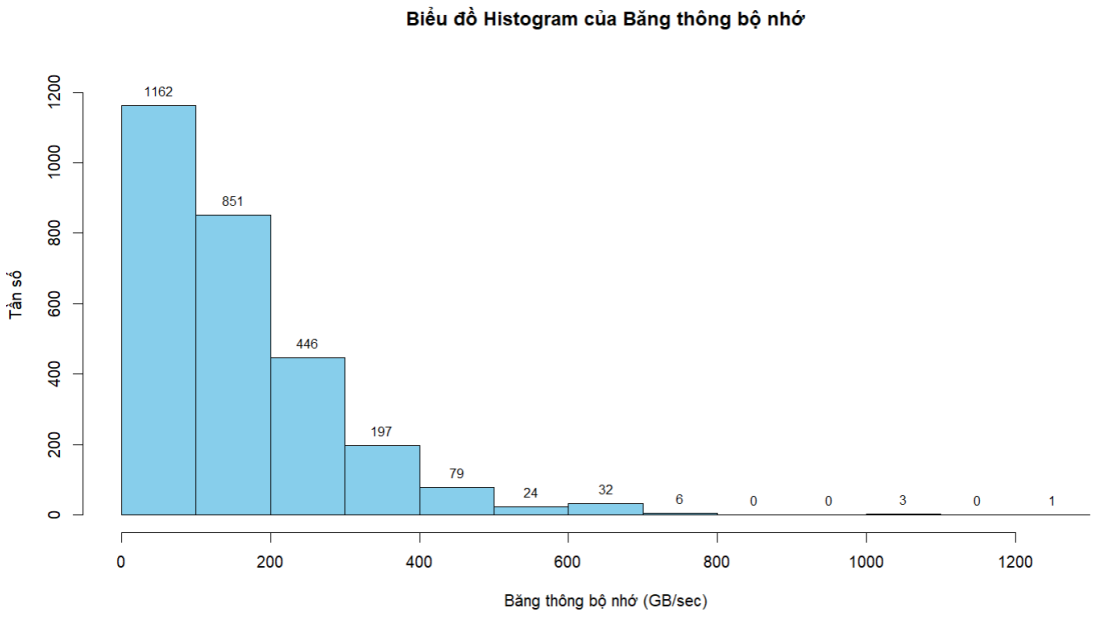
Ta vẽ đồ thị histogram với các thuộc tính sau:

Code

```
hist_data <- hist(data$Memory_Bandwidth,
                  main = "Biểu đồ Histogram của Băng thông bộ nhớ",
                  xlab = "Băng thông bộ nhớ (GB/sec)",
                  ylab="Tần số",
                  col = "skyblue",
                  border = "black",
                  ylim = c(0, 1250))  # Tăng giới hạn y

text(hist_data$mids, hist_data$counts, labels = hist_data$counts, pos = 3, cex = 0.8, col = "black")
```

Kết quả:

Result



Đồ thị cho ta thấy băng thông bộ nhớ không có phân phối chuẩn mà có phân phối lệch phải. Đa số các chip GPU có băng thông chủ yếu trong khoảng <400 GB/s.

4.3 Biểu đồ tròn thể hiện thị phần của GPU theo Manufacturer

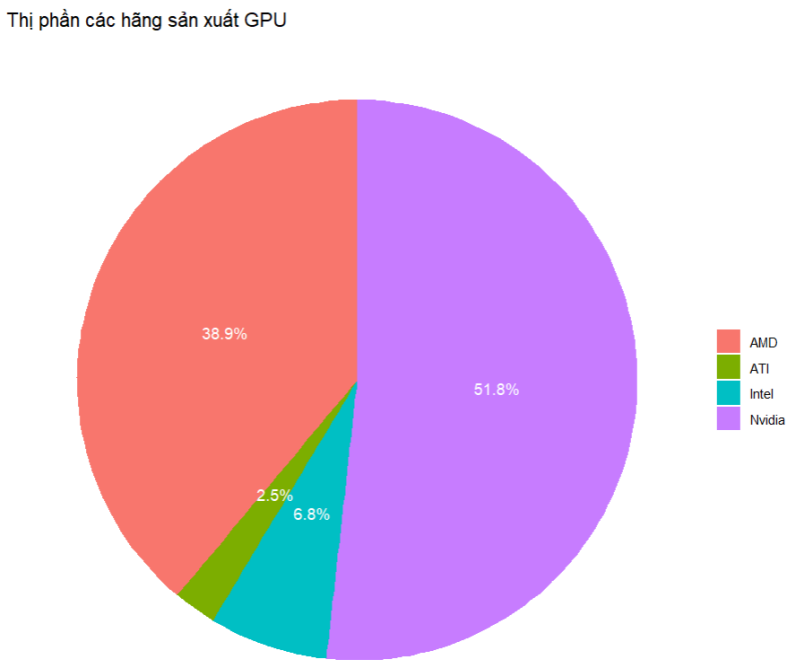
Để biết được hãng nào chiếm thị phần GPU lớn trên thị trường ta vẽ biểu đồ tròn:

Code

```
manufacturer_counts <- table(data$Manufacturer)
manufacturer_data <- as.data.frame(manufacturer_counts)
manufacturer_data$Percentage <- manufacturer_data$Freq / sum(manufacturer_data$Freq) * 100
library(ggplot2)
ggplot(manufacturer_data, aes(x = "", y = Freq, fill = Var1)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y", start = 0) +
  labs(title = "Thị phần các hãng sản xuất GPU", fill = "Hãng sản xuất") +
  theme_void() +
  theme(legend.title = element_blank()) +
  geom_text(aes(label = paste0(round(Percentage, 1), "%")),
            position = position_stack(vjust = 0.5),
            color = "white")
```

Kết quả:

Result



Từ biểu đồ trên, hãng Nvidia chiếm tỉ lệ lớn nhất, hơn tất cả hãng còn lại, cho thấy Nvidia là nhà sản xuất dẫn đầu trong GPU, theo sau là AMD cũng chiếm tỉ lệ lớn trên thị trường.

4.4 Đồ thị thể hiện phân phối của Memory_Bandwidth theo Manufacturer

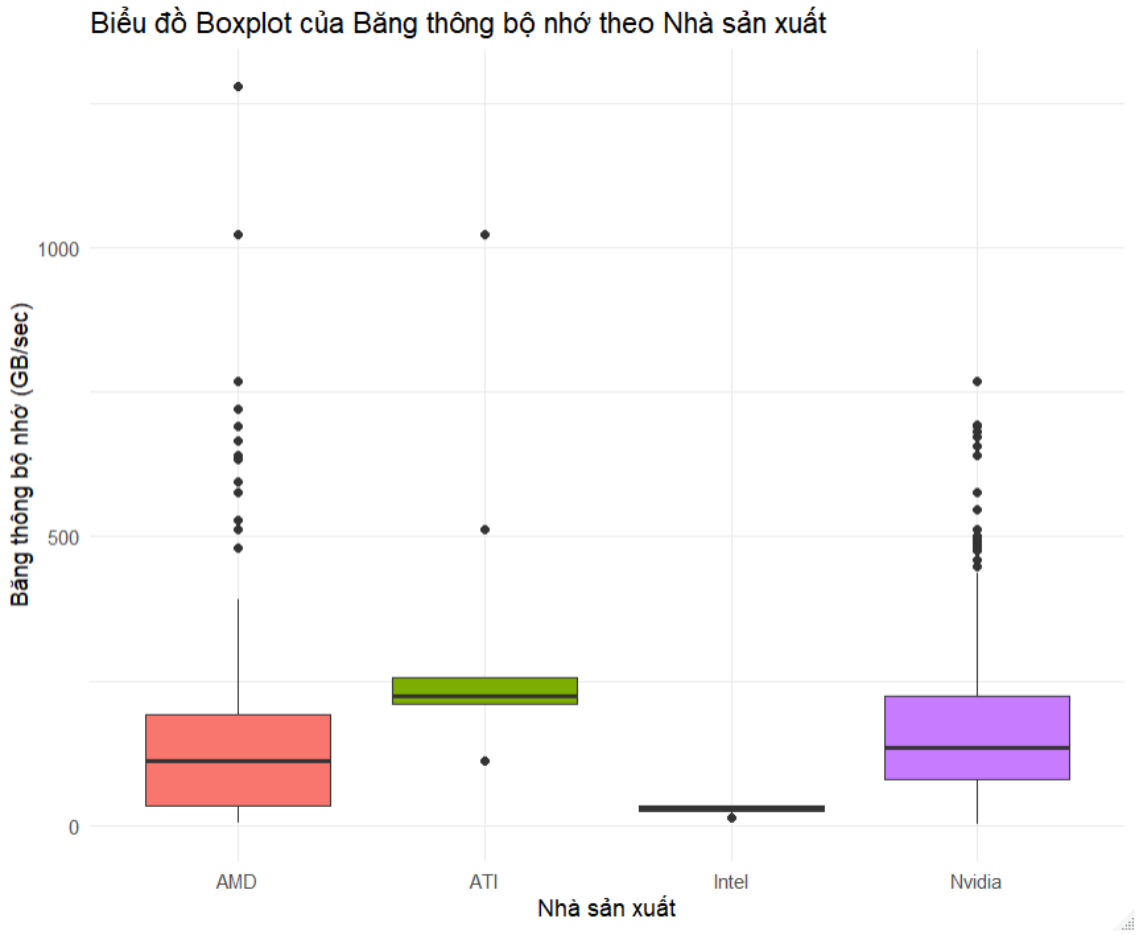
Vì Manufacturer là biến phân loại nên để xem mối liên hệ của Memory_Bandwidth theo Manufacturer ta sử dụng đồ thị Boxplot:

Code

```
ggplot(data, aes(x = Manufacturer, y = Memory_Bandwidth, fill = Manufacturer)) +
  geom_boxplot() +
  labs(title = "Biểu đồ Boxplot của Băng thông bộ nhớ theo Nhà sản xuất",
       x = "Nhà sản xuất",
       y = "Băng thông bộ nhớ (GB/sec)") +
  theme_minimal() +
  theme(legend.position = "none")
```

Kết quả:

Result



Từ biểu đồ trên, ta thấy được sự khác biệt về băng thông bộ nhớ trên GPUs ở các hãng, cụ thể hãng Intel thấp hơn các hãng còn lại.

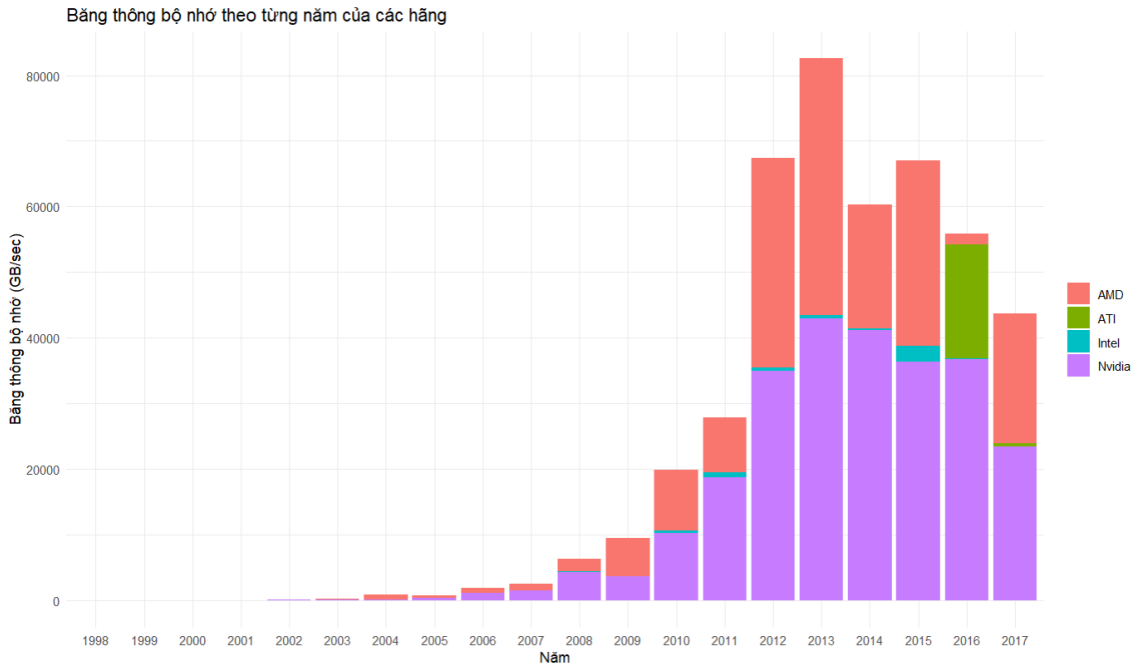
Để rõ ràng hơn, ta sẽ vẽ biểu đồ cột theo từng năm và hãng:

Code

```
library(ggplot2)
ggplot(data, aes(x = as.factor(Year), y = Memory_Bandwidth, fill = Manufacturer)) +
  geom_bar(stat = "identity", position = "stack") +
  labs(title = "Băng thông bộ nhớ theo từng năm của các hãng",
       x = "Năm",
       y = "Băng thông bộ nhớ (GB/sec)") +
  theme_minimal() +
  theme(legend.title = element_blank())
```

Kết quả:

Result



Từ đó, ta dễ dàng thấy rằng 2 hãng AMD và hãng Nvidia chiếm hầu hết về tổng băng thông bộ nhớ, Intel chiếm tỉ lệ rất nhỏ. ATI chỉ chiếm tỉ lệ cao trong năm 2016.

4.5 Vẽ đồ thị phân tán của Memory_Bandwidth theo các biến Memory_Bus, Texture_Rate, Memory_Speed, Year

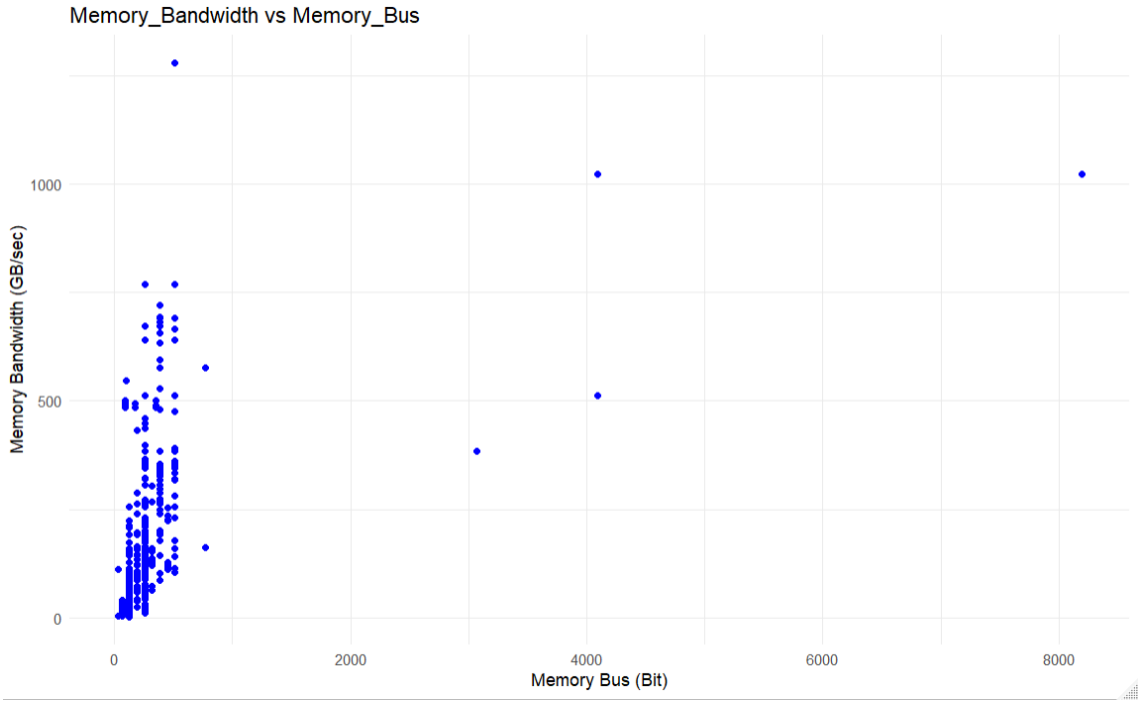
Vì cả 4 biến là biến định lượng nên ta vẽ đồ thị phân tán:

Code

```
p1 <- ggplot(data, aes(x = Memory_Bus, y = Memory_Bandwidth)) +
  geom_point(color = "blue") +
  labs(title = "Memory_Bandwidth vs Memory_Bus",
       x = "Memory Bus (Bit)",
       y = "Memory Bandwidth (GB/sec)") +
  theme_minimal()

p2 <- ggplot(data, aes(x = Texture_Rate, y = Memory_Bandwidth)) +
  geom_point(color = "green") +
  labs(title = "Memory_Bandwidth vs Texture_Rate",
       x = "Texture Rate (GTexel/s)",
       y = "Memory Bandwidth (GB/sec)") +
  theme_minimal()

p3 <- ggplot(data, aes(x = Memory_Speed, y = Memory_Bandwidth)) +
  geom_point(color = "purple") +
  labs(title = "Memory_Bandwidth vs Memory_Speed",
       x = "Memory Speed (MHz)",
       y = "Memory Bandwidth (GB/sec)") +
  theme_minimal()

p4 <- ggplot(data, aes(x = Year, y = Memory_Bandwidth)) +
  geom_point(color = "red") +
  labs(title = "Memory_Bandwidth vs Year",
       x = "Year",
       y = "Memory Bandwidth (GB/sec)") +
  theme_minimal()
print(p1)
```

4.5.1 Memory_Bandwidth theo Memory_Bus:

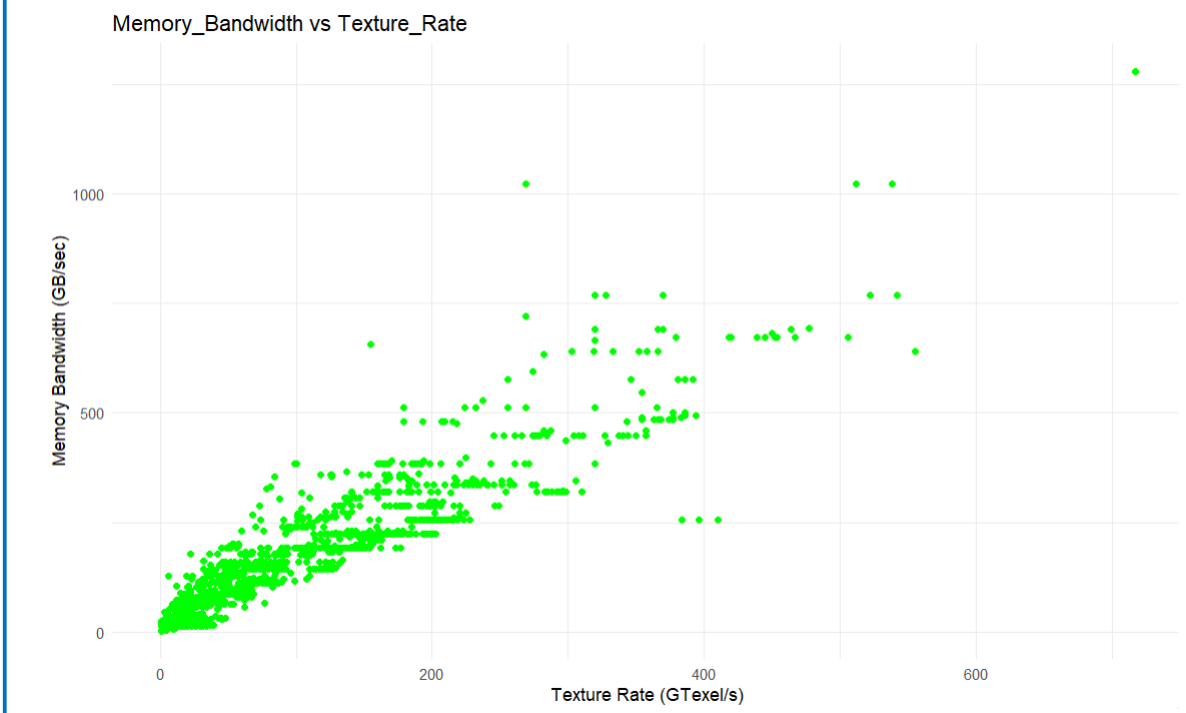
Result



Biểu đồ cho thấy quan hệ không tuyến tính rõ ràng, chủ yếu tập trung ở vùng giá trị nhỏ, và mối quan hệ có thể rất yếu giữa Memory_Bus và Memory_Bandwidth.

4.5.2 Memory_Bandwidth theo Texture_Rate:

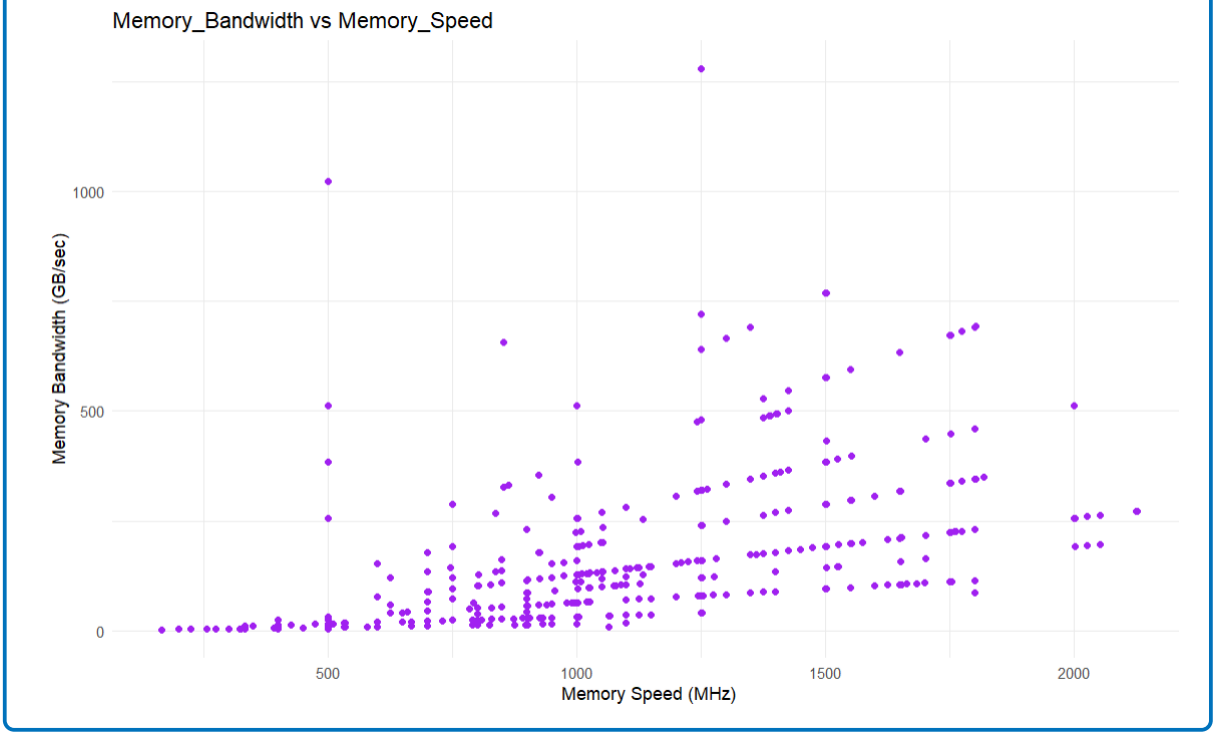
Result



Có một xu hướng tăng tuyến tính giữa Memory_Bandwidth và Texture_Rate ở các giá trị nhỏ đến trung bình. Tuy nhiên, xu hướng này có thể không duy trì ở các giá trị Texture_Rate rất cao, nơi mà dữ liệu trở nên thưa thớt hơn nên không đủ dữ liệu để suy ra mối quan hệ chắc chắn.

4.5.3 Memory_Bandwidth theo Memory_Speed:

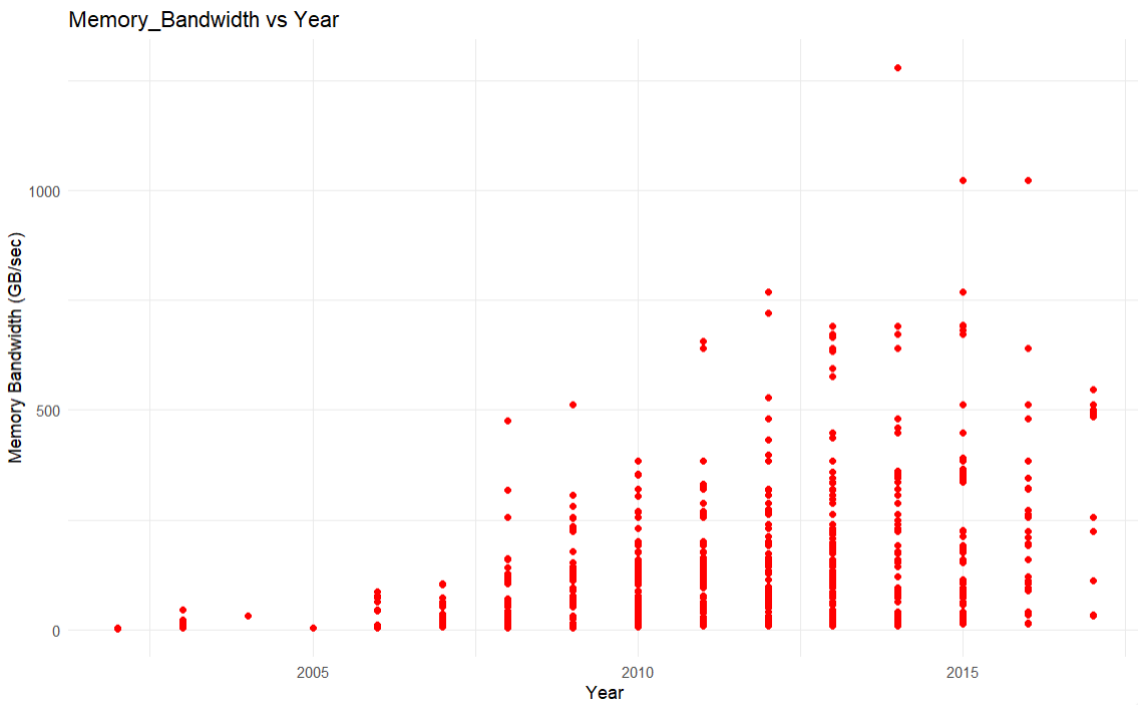
Result



Khi Memory_Speed tăng, Memory_Bandwidth cũng có xu hướng tăng theo. Chủ yếu phân bố theo một mối quan hệ tương quan tuyến tính dương .Dữ liệu có sự phân tán khá lớn, đặc biệt ở các mức Memory Speed cao hơn, cho thấy có các yếu tố khác ngoài tốc độ bộ nhớ ảnh hưởng đến băng thông bộ nhớ.

4.5.4 Memory_Bandwidth theo Year:

Result



Có xu hướng tăng dần qua các năm, thể hiện sự tiến bộ công nghệ trong lĩnh vực bộ nhớ.Có một số điểm vượt trội về băng thông (outliers), đặc biệt sau năm 2010. Các giá trị này thường vượt xa phần còn lại, phản ánh những sản phẩm cao cấp hoặc công nghệ đặc thù

4.6 Vẽ đồ thị tương quan giữa các biến

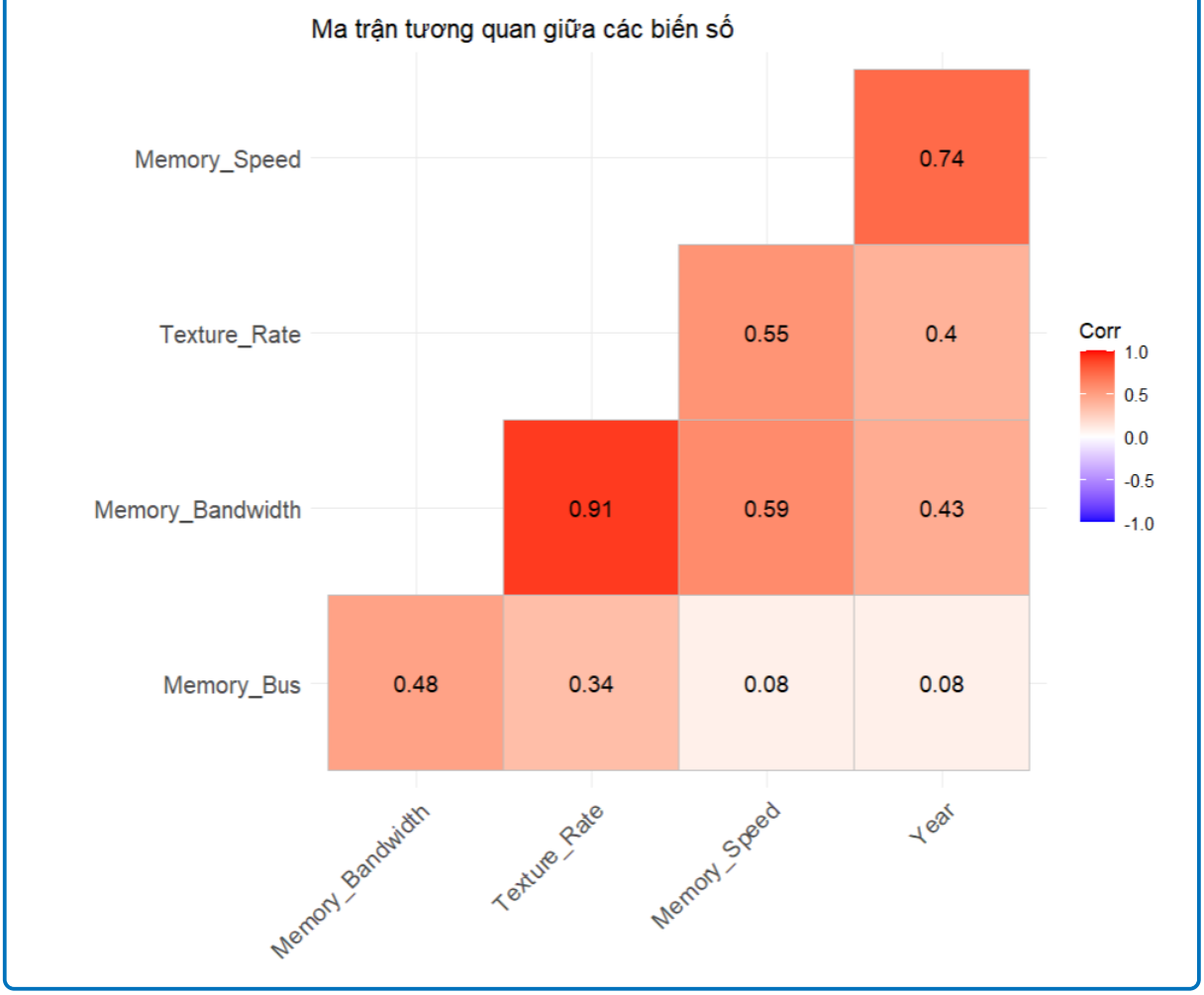
Để biết được cụ thể mức độ tương quan giữa các biến trong data, ta vẽ ma trận tương quan:

Code

```
library(ggcorrplot)
numeric_data <- data[, sapply(data, is.numeric)]
correlation_matrix <- cor(numeric_data, use = "complete.obs")
ggcorrplot(correlation_matrix, lab = TRUE,
           title = "Ma trận tương quan giữa các biến số",
           hc.order = TRUE,
           type = "lower")
```

Kết quả:

Result



Vì các biến có mức độ tương quan nhìn chung khá cao, có thể gây ra hiện tượng đa cộng tuyến, gây khó khăn khi tính hệ số hồi quy trong mô hình.

5 Thống kê suy diễn

5.1 Bài toán 1 mẫu

Ta đặt bài toán: Với độ tin cậy 95%, hãy ước lượng khả năng truyền tải dữ liệu của bộ nhớ GPU trung bình (Memory_Bandwidth) dựa theo bộ dữ liệu trên? Gọi X là khả năng truyền tải dữ liệu của bộ nhớ.

Đầu tiên, ta tính thống kê mẫu với biến đang xét là Memory_Bandwidth:

Code

```
n <- length(data$Memory_Bandwidth)
xtb <- mean(data$Memory_Bandwidth)
sx <- sd(data$Memory_Bandwidth)
```

Kết quả thu được:

Result

Table: Thống kê mẫu của biến Memory_Bandwidth

Số.lượng.mẫu..n.	Giá.trị.trung.bình..x..	Độ.lệch.chuẩn..s.
3257	137.24	135.52

Tiếp theo, ta xem X có phân phối chuẩn hay không bằng các cách sau:

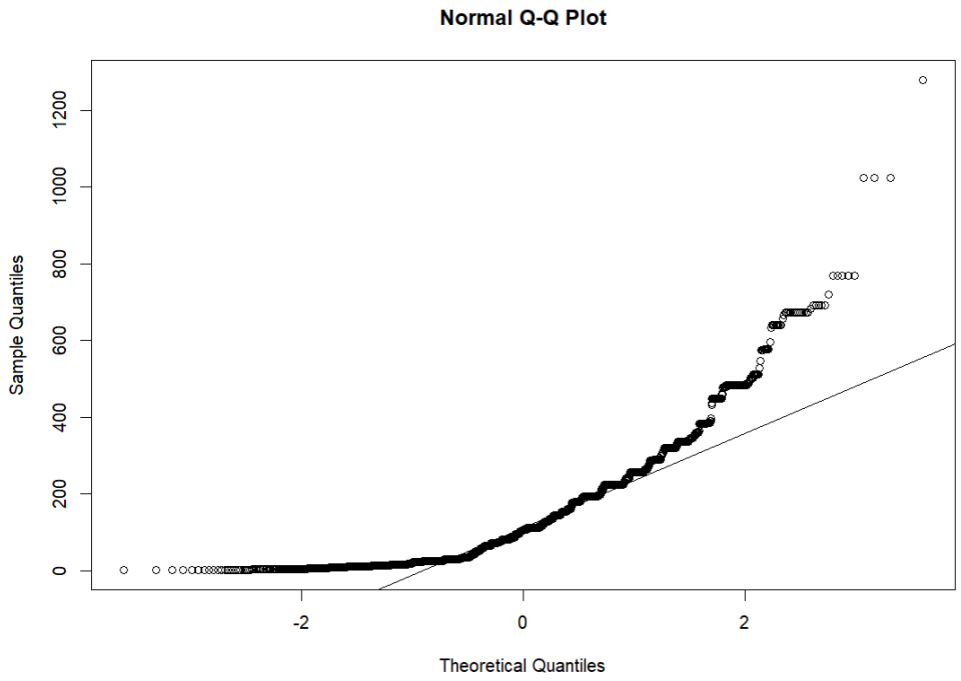
- Cách 1: Vẽ biểu đồ
  - Vẽ biểu đồ Q-Q (Quantile-Quantile Plot) bằng câu lệnh sau:

Code

```
qqnorm(data$Memory_Bandwidth)
qqline(data$Memory_Bandwidth)
```

Kết quả:

Result



Ta thấy nhiều điểm lệch xa khỏi đường thẳng, tức là khác so với phân phối chuẩn, X không theo phân phối chuẩn.

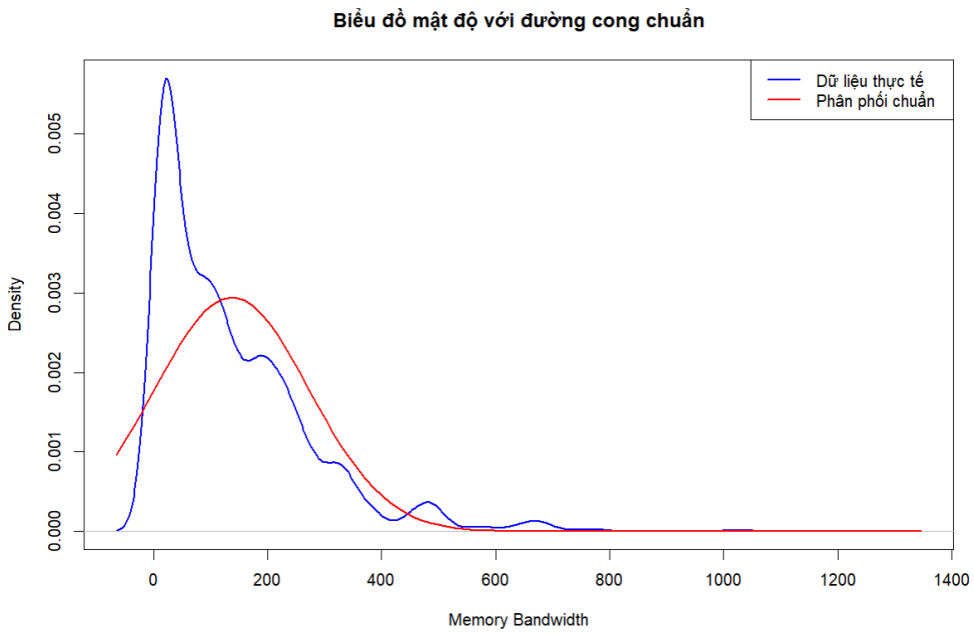
  - Để trực quan hóa dữ liệu so với phân phối chuẩn trên cùng một biểu đồ, có thể vẽ **biểu đồ mật độ (density plot)**:

Code

```
plot(density(data$Memory_Bandwidth), main="Biểu đồ mật độ với đường cong chuẩn",
     xlab="Memory Bandwidth", col="blue", lwd=2)

curve(dnorm(x, mean=xtb, sd=sx), col="red", lwd=2, add=TRUE)

legend("topright", legend=c("Dữ liệu thực tế", "Phân phối chuẩn"), col=c("blue", "red"), lwd=2)
```

Kết quả:

Result



Ta thấy dữ liệu thực tế của X lệch nhiều so với phân phối chuẩn nên ta xem X không có phân phối chuẩn.

- Cách 2: Kiểm định bằng cách tính p-value (Kiểm định Shapiro-Wilk) ($\alpha = 5\%$)
  - Đặt giả thuyết:
    H0: X tuân theo phân phối chuẩn
    H1: X không tuân theo phân phối chuẩn
  - Tiếp theo ta tìm p-value:

Code

```
result <- shapiro.test(data$Memory_Bandwidth)
print(result)
```

Kết quả:

Result

```
data:  data$Memory_Bandwidth
W = 0.83281, p-value < 2.2e-16
```

  - Theo kết quả trên, W đo mức độ phù hợp của dữ liệu với phân phối chuẩn, thấy lệch khá xa so với 1, giá trị p-value < 2.2e-16 bé hơn nhiều so với α (5%), ta đủ cơ sở để bác bỏ giả thuyết H0.

- Vậy X không có phân phối chuẩn với mức ý nghĩa 5%.

Đây là bài toán ước lượng trung bình, trong trường hợp tổng thể có phân phối bất kỳ và cỡ mẫu lớn.

- Tính $z_{\frac{\alpha}{2}}$ cho độ tin cậy 95%:

Code

```
z <- qnorm(1 - 0.05 / 2)
```

- Tính sai số Ước lượng:

Result

```
E <- z * sx / sqrt(n)
```

- Tính khoảng tin cậy và in ra màn hình:

Code

```
duoi <- xtb - E
tren <- xtb + E
cat("Khoảng tin cậy 95% cho trung bình là: (", duoi, ", ", tren, ")\n")
```

Kết quả:

Result

```
> cat("Khoảng tin cậy 95% cho trung bình là: (", duoi, ", ", tren, ")\n")
Khoảng tin cậy 95% cho trung bình là: ( 132.5889 ,  141.8974 )
```

Kết luận: Với độ tin cậy 95%, khả năng truyền tải dữ liệu của bộ nhớ GPU trung bình dựa theo bộ dữ liệu trên: $(\bar{x}-\varepsilon;\bar{x}+\varepsilon) \leftrightarrow (132.5889;141.8974)$

5.2 Bài toán 2 mẫu

Để biết được khả năng truyền tải dữ liệu (Memory_Bandwidth) sau năm 2013 có sự gia tăng đáng kể so với trước đó hay không, ta thực hiện bài toán kiểm định 2 mẫu. Ta đặt ra bài toán sau:
Với mức ý nghĩa 1% có thể kết luận rằng khả năng truyền tải bộ nhớ GPU ở nhóm 2 (sau năm 2013) cao hơn so với nhóm 1 (trước năm 2013) hay không?

- Nhóm 1: Year < 2013
- Nhóm 2: Year >= 2013

Đầu tiên, ta chia thành 2 nhóm dữ liệu và tính các thống kê mẫu cho từng nhóm:

Code

```
data_before_2013 <- subset(data, Year < 2013)$Memory_Bandwidth
data_after_2013 <- subset(data, Year >= 2013)$Memory_Bandwidth

# Tính các thống kê mẫu cho nhóm trước năm 2013
n_before <- length(data_before_2013)
xtb_before <- mean(data_before_2013)
sx_before <- sd(data_before_2013)

# Tính các thống kê mẫu cho nhóm sau năm 2013
n_after <- length(data_after_2013)
xtb_after <- mean(data_after_2013)
sx_after <- sd(data_after_2013)
```

Kết quả:

Result

Nhóm	n	xtb	sx
1	1605	85.70037	91.58179
2	1652	187.31955	151.59449

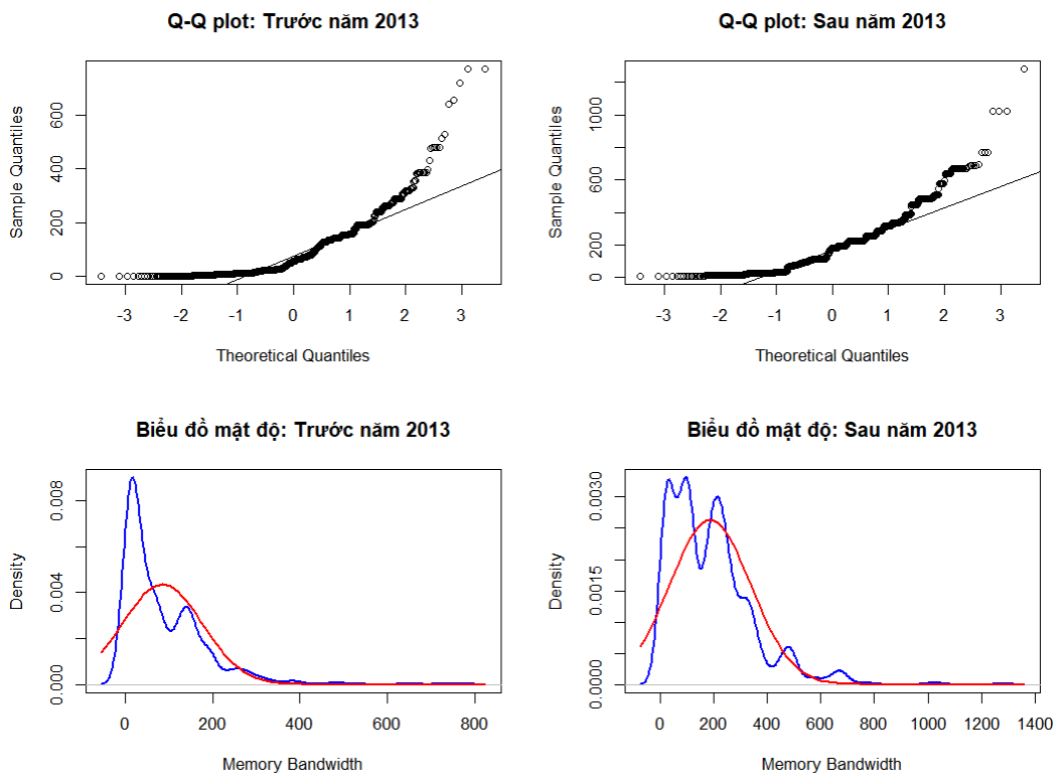
Đặt μ1 và μ2 là khả năng truyền tải bộ nhớ trung bình của nhóm 1 và nhóm 2. Ta đặt giả thuyết sau:

- H0: μ1 = μ2 (khả năng truyền tải bộ nhớ trung bình ở nhóm 1 và nhóm 2 là như nhau)
- H1: μ1 < μ2 (khả năng truyền tải bộ nhớ trung bình của nhóm 1 thấp hơn nhóm 2)

Sau khi đặt giả thuyết, ta kiểm định xem nhóm 1 và nhóm 2 có phân phối chuẩn hay không:

- Vẽ đồ thị: Tương tự phần trước, ta cũng vẽ 2 loại biểu đồ: biểu đồ Q-Q và biểu đồ mật độ cho cả 2 nhóm, ta thu được kết quả sau:

Result



Ta thấy rằng, cả 2 nhóm đều không có phân phối chuẩn.

- Để xác định rõ hơn, ta dùng phương pháp p-value (Kiểm định Shapiro-Wilk) cho 2 nhóm với mức ý nghĩa 1%:

Code

```
result_before <- shapiro.test(data_before_2013)
result_after <- shapiro.test(data_after_2013)
```

Kết quả thu được:

Result

	Group	W	p_value
1	1	0.7951797	5.714171e-41
2	2	0.8809839	8.330432e-34

Ta thấy cả 2 nhóm giá trị W đều nhỏ và p-value đều bé hơn mức 1% nên cả 2 nhóm đều không có phân phối chuẩn với mức ý nghĩa 1%.

Bài toán của ta có mẫu có phân phối bất kỳ, cỡ mẫu lớn:

- Tính giá trị kiểm định và miền bác bỏ:

Code

```
z0 <- (xtb_before - xtb_after) / sqrt((sx_before^2 / n_before) + (sx_after^2 / n_after))
RR <- qnorm(p = 1 - 0.01 / 2)
```

- Kết quả:

Result

```
> cat("z0:", z0, "\n")
z0: -23.22966
> cat("RR:", RR, "\n")
RR: 2.575829
```

Miền bác bỏ: RR = (-∞;-2.5758)
Giá trị kiểm định: z0 = -23.22966
Vì z0 ∈ R, nên bác bỏ H0 và chấp nhận H1.

Kết luận: Khả năng truyền tải bộ nhớ sau năm 2013 cao hơn so với trước năm với mức ý nghĩa 1%.
Ta thấy được khả năng truyền tải bộ nhớ của GPU sau năm 2013 có sự cải thiện rõ rệt so với trước năm 2013.

5.3 Mô hình ANOVA

Ta sử dụng mô hình ANOVA để đánh giá sự khác nhau giữa các băng thông bộ nhớ trong GPU giữa các nhãn hàng (Intel, AMD, ATI, Nvidia).

Tuân theo quy trình của mô hình ANOVA, trước tiên ta cần kiểm tra các giả định:

- Giả định phân phối chuẩn: Băng thông bộ nhớ của các nhãn hàng tuân theo phân phối chuẩn.
- Phương sai giữa băng thông bộ nhớ ở các nhãn hàng bằng nhau.

Sau đó, ta đặt giả thuyết:

- **Giả thuyết H_0:** Băng thông bộ nhớ trung bình giữa các nhãn hàng bằng nhau.
- **Giả thuyết đối H_1:** Có ít nhất 2 nhãn hàng có băng thông trong bộ nhớ trung bình khác nhau.

5.3.1 Các bước thực hiện

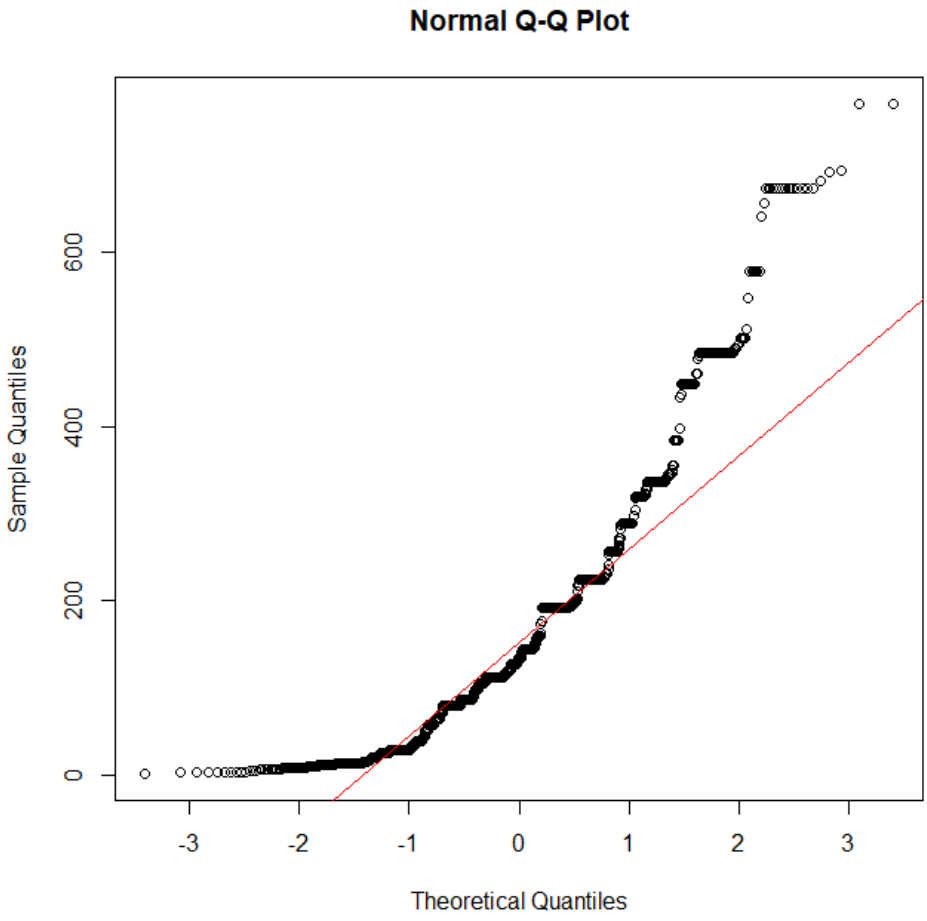
- **Bước 1:** Kiểm tra giả định phân phối chuẩn. Sử dụng các câu lệnh sau trong RStudio:

Code

```
#Kiem tra cac gia dinh anova
Nvidia= subset(selected_data , selected_data$Manufacturer == "Nvidia")
qqnorm(Nvidia$Memory_Bandwidth) #kiem tra Bandwidth cua Nvidia co thuoc phan phoi chuan hay khong
qqline(Nvidia$Memory_Bandwidth)
ad.test(Nvidia$Memory_Bandwidth) #su dung Anderson-Darling normality test
                                 #de kiem tra tap du lieu co theo phan phoi chuan hay khong
```

Sau khi chạy dòng lệnh, ta có được biểu đồ QQ-plot và giá trị p-value như hình dưới đây:

Result



Result

```
        Anderson-Darling normality test

data:  Nvidia$Memory_Bandwidth
A = 40.831, p-value < 2.2e-16
```

- Biểu đồ QQ-plot thể hiện phân phối của biến Bandwidth của hãng Nvidia.
- Từ biểu đồ, ta có thể thấy rằng nhiều giá trị quan sát không nằm trên đường thẳng kỳ vọng của phân phối chuẩn.

Sử dụng kiểm định Anderson-Darling, giá trị p-value nhỏ hơn rất nhiều so với mức ý nghĩa $\alpha = 0.05$. Do đó, ta bác bỏ giả thuyết H_0, đồng thời kết luận rằng biến `Memory_Bandwidth` của hãng Nvidia không tuân theo phân phối chuẩn.

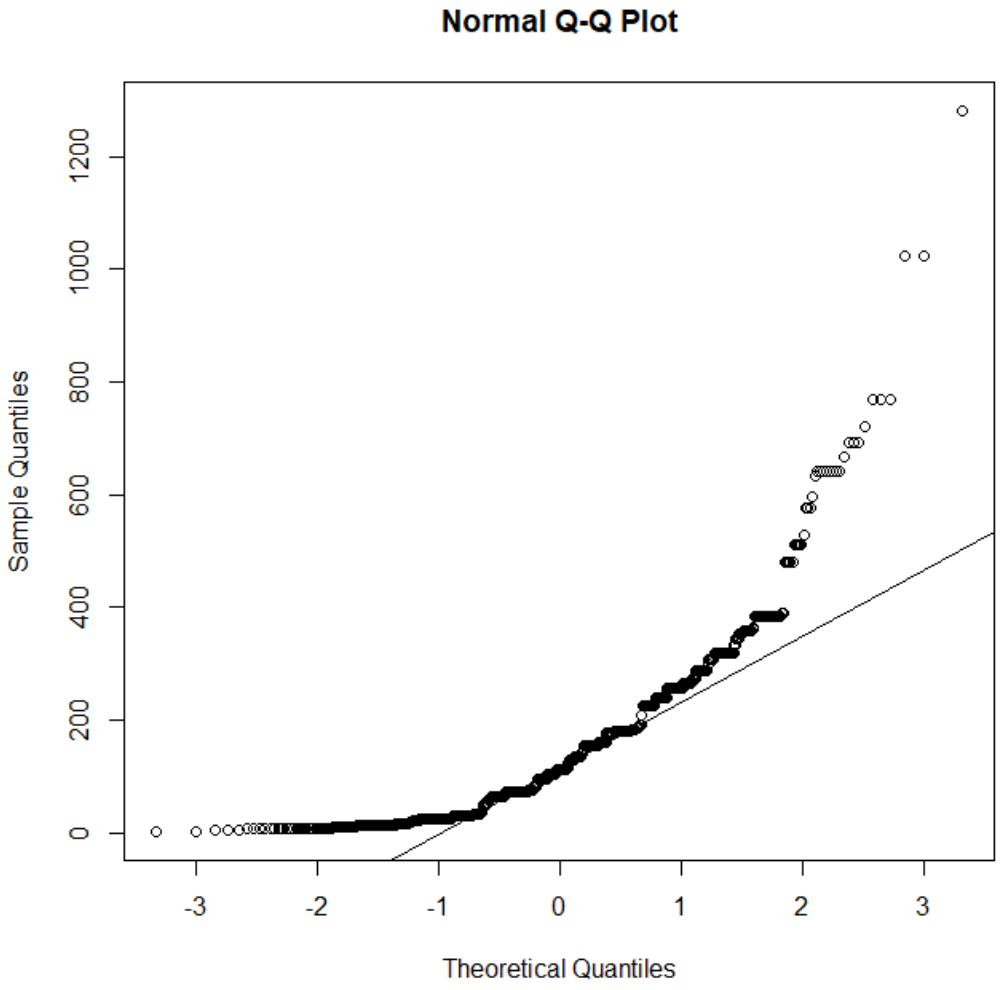
Cũng cách lập luận trên, ta thực hiện kiểm tra giả định của biến `Memory_Bandwidth` của các hãng AMD, Intel, ATI:

Code

```
#tuong tu su dung cho AMD, Intel, ATI
AMD = subset(selected_data, selected_data$Manufacturer == "AMD")
qqnorm(AMD$Memory_Bandwidth)
qqline(Nvidia$Memory_Bandwidth)
ad.test(Nvidia$Memory_Bandwidth)

Intel= subset(selected_data , selected_data$Manufacturer == "Intel")
qqnorm(Intel$Memory_Bandwidth)
qqline(Intel$Memory_Bandwidth)
ad.test(Intel$Memory_Bandwidth)

ATI= subset(selected_data , selected_data$Manufacturer == "ATI")
qqnorm(ATI$Memory_Bandwidth)
qqline(ATI$Memory_Bandwidth)
ad.test(ATI$Memory_Bandwidth)
```

Sau khi thực hiện các câu lệnh, ta có được lần lượt là QQ-plot và adtest của các hãng:

Result

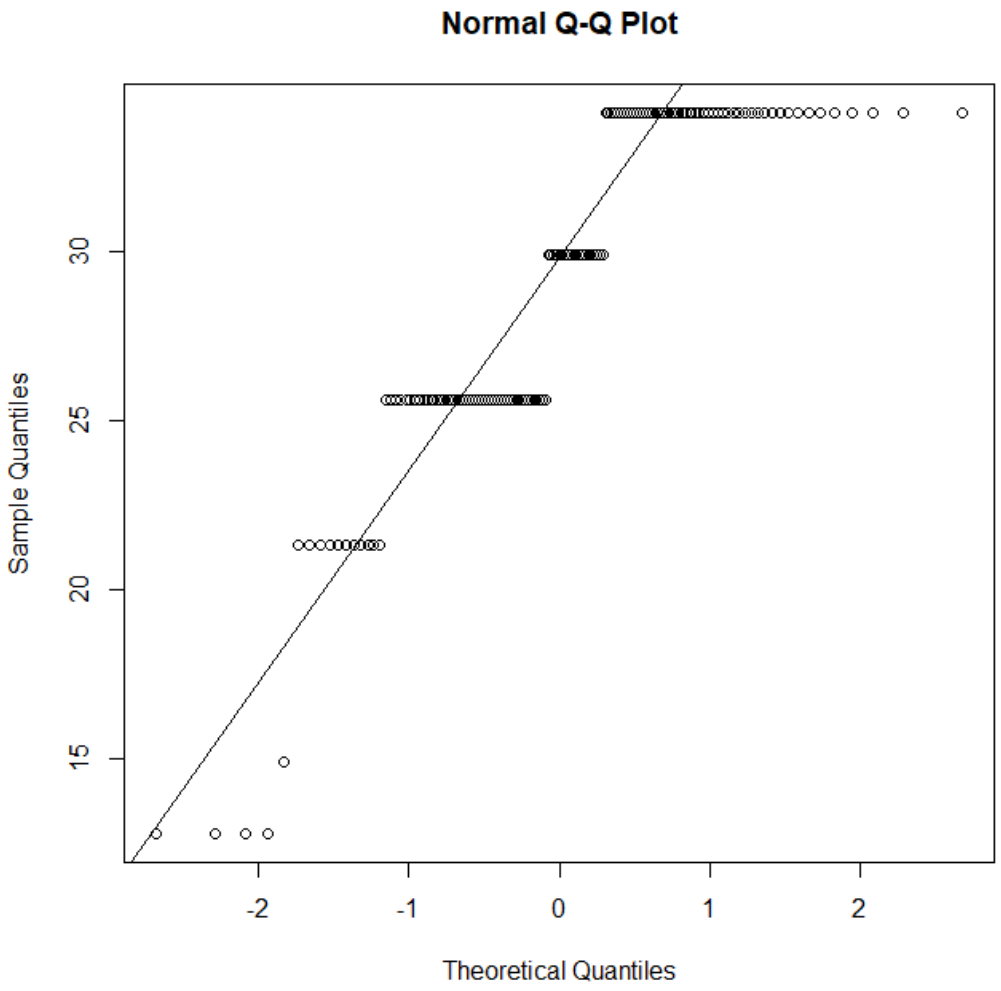


Result

```
        Anderson-Darling normality test

data:  AMD$Memory_Bandwidth
A = 39.664, p-value < 2.2e-16
```

QQ-plot và adtest của hãng AMD

Result

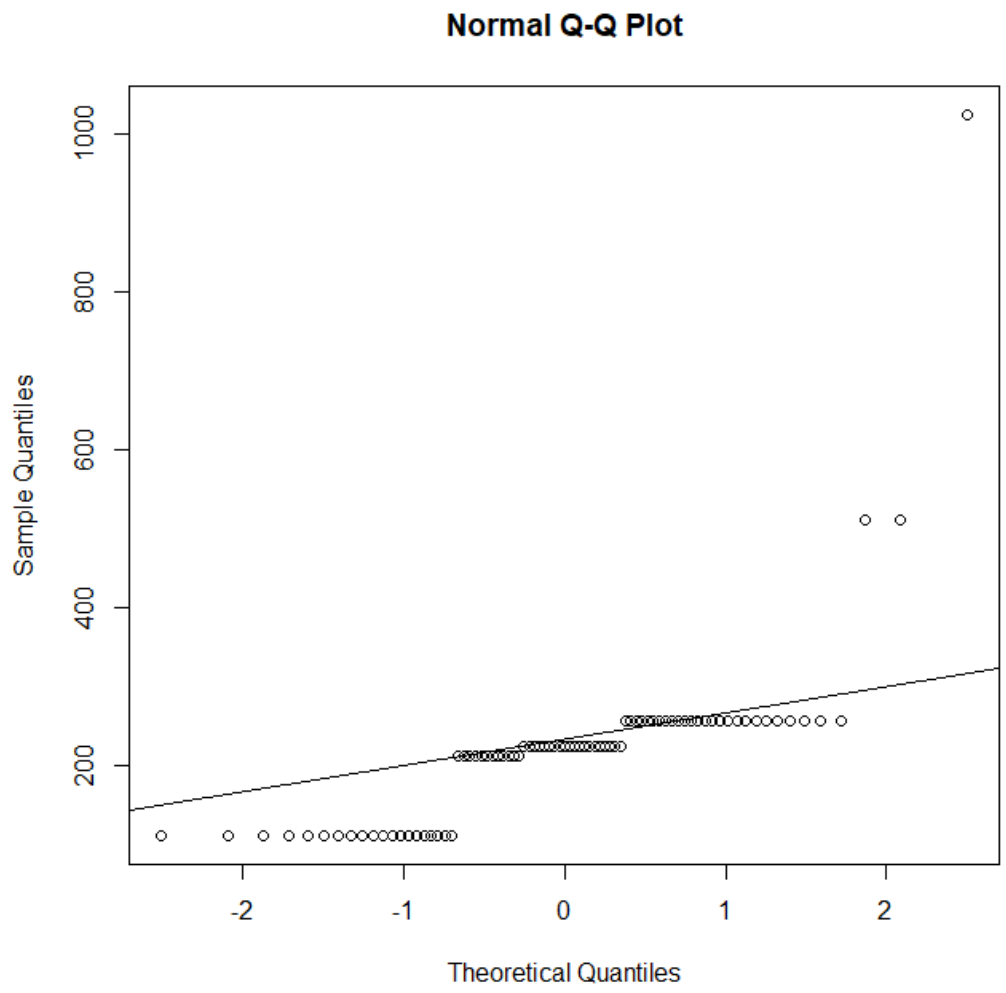


Result

```
        Anderson-Darling normality test

data:  Intel$Memory_Bandwidth
A = 8.0584, p-value < 2.2e-16
```

QQ-plot và adtest của hãng Intel

Result



Result

```
        Anderson-Darling normality test

data:  ATI$Memory_Bandwidth
A = 9.0217, p-value < 2.2e-16
```

QQ-plot và adtest của hãng ATI

Kết luận: Biến `Memory_Bandwidth` của các hãng Intel, AMD, Nvidia, ATI không tuân theo phân phối chuẩn.

- **Bước 2:** Đặt giả thuyết về tính đồng nhất của các phương sai.
  - **Giả thuyết H_0:** Phương sai giữa các nhãn hàng bằng nhau.
  - **Giả thuyết đối H_1:** Có ít nhất 2 nhãn hàng có phương sai khả năng truyền tải bộ nhớ của GPU khác nhau.

Trong RStudio, ta sử dụng phương pháp kiểm định Levene (Levene Test) để kiểm tra tính đồng nhất của phương sai.

Code

```
leveneTest(Memory_Bandwidth~as.factor(Manufacturer), data = selected_data) #levene test
```

Kết quả kiểm định Levene của biến `Memory_Bandwidth` được thể hiện dưới đây:

Result

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value    Pr(>F)
group   3  44.548 < 2.2e-16 ***
      2797
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nhận xét: Giá trị p-value trong kiểm định Levene nhỏ hơn rất nhiều so với mức ý nghĩa 0.05. Do đó, ta bác bỏ giả thuyết H_0 và kết luận rằng có ít nhất 2 nhãn hàng có phương sai khác nhau.

Kết luận: Từ việc chứng minh các giả định và giả thuyết kể trên, phương pháp ANOVA rõ ràng không phải là sự lựa chọn phù hợp đối với dữ liệu. Thay vào đó, ta có thể dùng các phương pháp kiểm định phi tham số. Trong số những phương pháp kiểm định phi tham số, phương pháp kiểm định Kruskal – Wallis phù hợp hơn cả.

5.3.2 Phương pháp Kruskal-Wallis

Cách thực hiện phương pháp Kruskal – Wallis trong Rstudio:

Code

```
kruskal.test(Memory_Bandwidth~Manufacturer, selected_data) #phuong phap Kruskal - Willis
```

Kết quả phương pháp Kruskal - Wallis:

Result

```
        Kruskal-Wallis rank sum test

data:  Memory_Bandwidth by Manufacturer
Kruskal-Wallis chi-squared = 245.06, df = 3, p-value < 2.2e-16
```

Nhận xét: Giá trị p-value nhỏ hơn rất nhiều so với mức ý nghĩa 0.05. Do đó, ta bác bỏ giả thuyết H_0 và khẳng định rằng có ít nhất 2 nhãn hàng có băng thông khác nhau.

Để chắc chắn rằng băng thông của các nhãn hàng đều khác nhau, ta sử dụng kiểm định Kruskal-Wallis nhiều cặp (multiple comparisons for Kruskal-Wallis).
Dưới đây là code và kết quả:

Code

```
#su dung multiple comparison test after Kruskal-Wallis
#de chac chan rang cac nhan hang co bang thong bo nho khac nhau
kruskalmc(selected_data$Memory_Bandwidth, selected_data$Manufacturer)
```

Result

```
Multiple comparison test after Kruskal-Wallis
alpha: 0.05
Comparisons
                obs.dif critical.dif stat.signif
AMD-ATI       639.1526    245.58152        TRUE
AMD-Intel     835.1995    195.79759        TRUE
AMD-Nvidia    185.4218     84.77375        TRUE
ATI-Intel    1474.3521    300.71542        TRUE
ATI-Nvidia    453.7308    243.47414        TRUE
Intel-Nvidia 1020.6213    193.14779        TRUE
```

Nhận xét: ta quan sát cột "stat.signif" có các phần tử đều là "TRUE" cho biết sự khác biệt về băng thông giữa các hãng có ý nghĩa thông kê. Do đó, băng thông của các nhãn hàng là khác nhau.

5.4 Hồi quy tuyến tính đa biến

Đặt bài toán: Với mức ý nghĩa 1%, khả năng truyền tải bộ nhớ Memory_Bandwidth có phụ thuộc theo các biến độc lập hay không? Dự đoán tác động của các biến đến Memory_Bandwidth.

- Biến phụ thuộc: Memory_Bandwidth
- Biến độc lập: Manufacturer, Memory_Bus, Texture_Rate, Year, Memory_Speed

Đầu tiên, để tạo được mô hình chính xác nhất, ta sẽ lấy bộ dữ liệu loại bỏ "NA" mà không dùng phương pháp thế, ta có data_hqtt.
Ta chia bộ dữ liệu thành 2 phần gồm: Train_data (90%) dùng để chạy thực thi mô hình hồi quy và Test_data (10%) thực hiện kiểm tra xem mô hình đã xây dựng:

Code

```
set.seed(100)
train_rows <- sample(rownames(data_hqtt), dim(data_hqtt)[1] * 0.9)
train_data <- data_hqtt[train_rows, ]
test_rows <- setdiff(rownames(data_hqtt), train_rows)
test_data <- data_hqtt[test_rows, ]

str(train_data)
str(test_data)
```

Kết quả:

Result

```
> str(train_data)
'data.frame':   2520 obs. of  6 variables:
 $ Manufacturer    : chr  "Nvidia" "Nvidia" "Nvidia" "Nvidia" ...
 $ Memory_Bandwidth: num  230.4 40.1 25.6 146.6 80 ...
 $ Memory_Bus      : num  256 64 128 192 128 256 128 128 384 128 ...
 $ Texture_Rate    : num  161 17 9 69 46 59 5 13 216 60 ...
 $ Memory_Speed    : num  900 1252 800 1527 1250 ...
 $ Year            : num  2012 2013 2010 2013 2012 ...
 - attr(*, "na.action")= 'omit' Named int [1:605] 10 13 15 17 18 21 22 27 70 71 ..
  ..- attr(*, "names")= chr [1:605] "10" "13" "15" "17" ...
> str(test_data)
'data.frame':   281 obs. of  6 variables:
 $ Manufacturer    : chr  "AMD" "Intel" "Intel" "Nvidia" ...
 $ Memory_Bandwidth: num  160 29.9 25.6 288.4 317.2 ...
 $ Memory_Bus      : num  256 128 128 384 384 128 384 128 128 128 ...
 $ Texture_Rate    : num  62 8 8 196 214 28 101 16 17 20 ...
 $ Memory_Speed    : num  1250 933 800 1502 1652 ...
```

Sau khi chia tập dữ liệu data, ta thấy train_data có 2520 quan sát và test_data có 281 quan sát.
Trên tập dữ liệu train_data, ta sẽ xây dựng mô hình hồi quy:

- Đầu tiên, ta sẽ in ra các thông số trên mô hình hồi quy:

Code

```
data_hqtt2 <- lm(formula = Memory_Bandwidth ~ ., train_data)
summary(data_hqtt2)
```

Kết quả:

Result

```
Residuals:
    Min      1Q  Median      3Q     Max
-311.81  -21.58   -3.02   18.44  410.78

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         1.458e+04  9.334e+02  15.617  < 2e-16 ***
ManufacturerATI    -3.145e+01  5.179e+00  -6.071 1.46e-09 ***
ManufacturerIntel   6.666e-01  4.248e+00   0.157    0.875
ManufacturerNvidia -2.252e+01  1.759e+00 -12.802  < 2e-16 ***
Memory_Bus          8.337e-02  3.113e-03  26.780  < 2e-16 ***
Texture_Rate        1.353e+00  1.217e-02 111.242  < 2e-16 ***
Memory_Speed        3.415e-02  3.131e-03  10.908  < 2e-16 ***
Year               -7.250e+00  4.649e-01 -15.597  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 40.81 on 2512 degrees of freedom
Multiple R-squared:  0.9145,    Adjusted R-squared:  0.9142
F-statistic:  3837 on 7 and 2512 DF,  p-value: < 2.2e-16
```

Trong phần Coefficients (hệ số), cột Estimate sẽ cho ta biết được hệ số của phương trình ước lượng:

$$y = 1.458 \times 10^4 - 3.145 \times 10 X_1 + 6.666 \times 10^{-1} \times X_2 - 2.252 \times 10 \times X_3 + 8.337 \times 10^{-2} \times X_4 + 1.353 X_5^{-7} + 3.415 \times 10^{-2} \times X_6 - 7.250 \times X_7$$

Hệ số xác định R^2 hiệu chỉnh là 0.9142 chứng tỏ rằng mô hình rất tốt. Hệ số này có ý nghĩa rằng: Trong những nguyên nhân gây ra sự biến động của khả năng truyền tải bộ nhớ thì chỉ có 91.42% nguyên nhân được giải thích là do các biến độc lập đang xét trong mô hình gây ra. Còn 8.58% khác là do những biến dữ liệu khác chưa đưa vào mô hình, chưa xét tới hoặc là do thành phần sai số gây ra.

- Để hiểu rõ hơn ta kiểm định hệ số hồi quy, xét p-value đối với từng biến:
  Ta đặt giả thuyết:
  H0: βi = 0 (hệ số đang xét không ảnh hưởng đến Memory_Bandwidth)
  H1 : βi $\neq$ 0 (hệ số đang xét ảnh hưởng đến Memory_Bandwidth)
  Khi p-value <= 1% thì ta bác bỏ giả thuyết H0, ở đây chỉ có biến ManufacturerIntel có p-value = 0.875 > 1%, ta chưa đủ cơ sở để bác bỏ H0, hay nói cách khác ManufacturerIntel không ảnh hưởng đến khả năng truyền tải bộ nhớ.

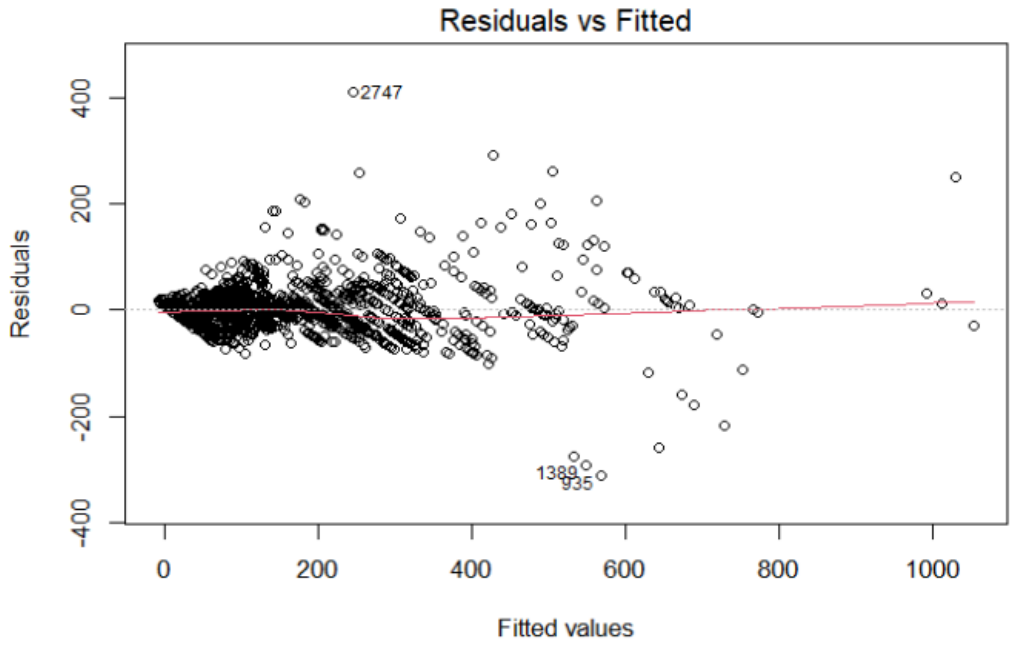
Tiếp theo, để đánh giá chất lượng của mô hình hồi quy tuyến tính, ta sẽ tạo các đồ thị chẩn đoán:

Code

```
par(mfrow = c(2, 2))
plot(data_hqtt2)
par(mfrow = c(1, 1))
```

Kết quả thu được là 4 hình:

Result

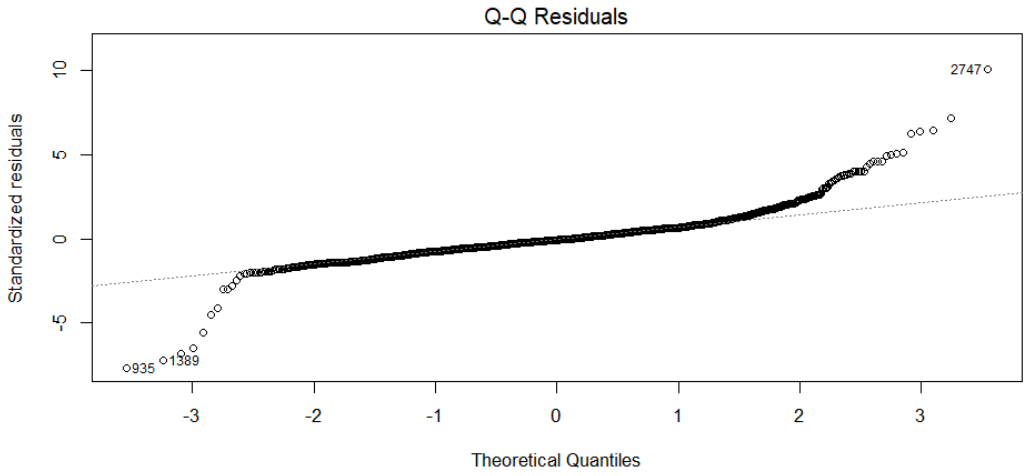


Dựa vào đồ thị "Residuals vs Fitted", ta có thể nhận xét các giả định của mô hình hồi quy như sau:

- **Quan hệ tuyến tính giữa Y và các biến độc lập X:** Đường màu đỏ trong đồ thị không hoàn toàn là đường thẳng nằm ngang, cho thấy giả định về mối quan hệ tuyến tính giữa biến phụ thuộc Y và các biến độc lập X chưa được thỏa mãn hoàn toàn. Điều này gợi ý rằng có thể tồn tại mối quan hệ phi tuyến giữa các biến.
- **Sai số có kỳ vọng bằng 0:** Đường màu đỏ nằm khá sát đường y = 0, điều này cho phép ta tạm chấp nhận rằng giả định sai số có kỳ vọng bằng 0 là thỏa mãn.

– **Phương sai các sai số là hằng số (Homoscedasticity):** Phần dư không phân tán ngẫu nhiên xung quanh đường màu đỏ, cho thấy giả định về phương sai các sai số là hằng số chưa được thỏa mãn. Điều này có thể ám chỉ rằng phương sai của sai số thay đổi theo giá trị dự đoán (heteroscedasticity), làm giảm tính ổn định của mô hình.

Result



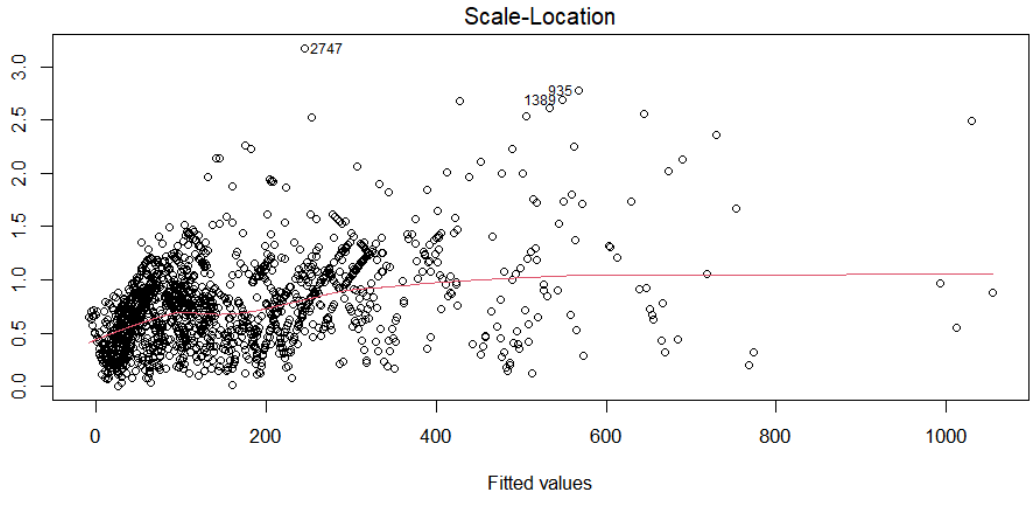
Dựa vào đồ thị Q-Q plot của phần dư, ta có thể đưa ra nhận xét như sau:

– **Giả định phân phối chuẩn của sai số:** Đồ thị Q-Q plot hiển thị sự so sánh giữa phần dư chuẩn hóa và phân phối chuẩn lý thuyết. Nếu phần dư tuân theo phân phối chuẩn, các điểm dữ liệu sẽ nằm trên đường chéo (đường lý thuyết). Tuy nhiên, trong đồ thị này, có một số điểm ở hai đầu (trái và phải) bị lệch xa khỏi đường chéo, đặc biệt là các điểm 0935, 1389, và 2747. Điều này cho thấy phần dư không hoàn toàn tuân theo phân phối chuẩn, và có thể có hiện tượng ngoại lệ hoặc phân phối đuôi dày.

– **Nhận xét về ngoại lệ:** Các điểm như 2747 nằm xa đường chéo có thể là các quan sát ngoại lệ, cho thấy rằng các giá trị này có phần dư lớn và không phù hợp với giả định phân phối chuẩn của mô hình.

Tóm lại, giả định rằng phần dư có phân phối chuẩn chưa được thỏa mãn hoàn toàn, và sự hiện diện của các ngoại lệ có thể ảnh hưởng đến tính chính xác và độ tin cậy của các kết quả hồi quy. Điều này có thể yêu cầu các phương pháp điều chỉnh như loại bỏ ngoại lệ hoặc áp dụng các phương pháp hồi quy mạnh mẽ hơn.

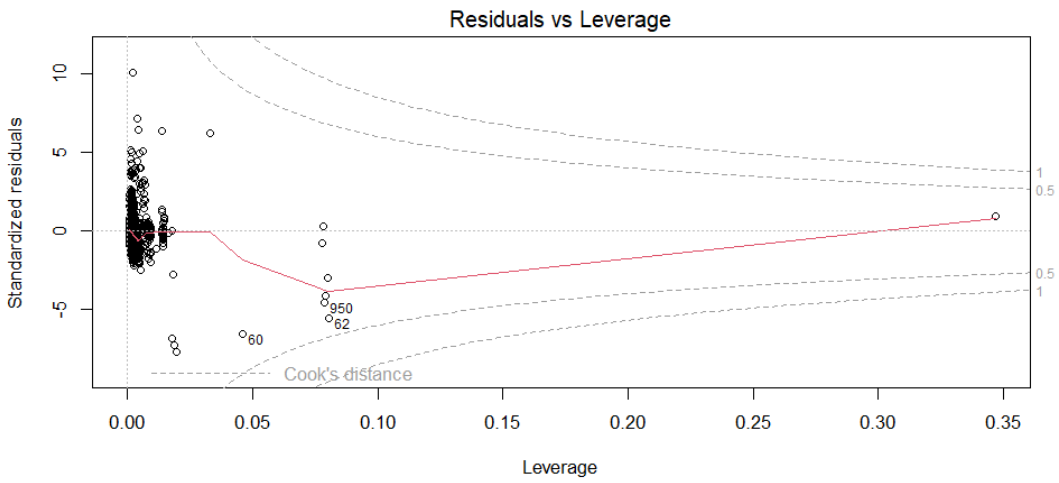
Result



Dựa vào đồ thị Scale-Location, ta có thể nhận xét như sau:

Giả định về phương sai đồng đều (Homoscedasticity): Đồ thị Scale-Location cho phép kiểm tra tính đồng đều của phương sai phần dư. Nếu giả định này được thỏa mãn, các điểm dữ liệu sẽ phân bố ngẫu nhiên quanh đường ngang (đường trung bình) và không tạo ra bất kỳ mẫu hình nào. Trong đồ thị này, phần dư có xu hướng phân tán không đều, đặc biệt là khi giá trị dự đoán tăng. Điều này cho thấy rằng phương sai phần dư có xu hướng thay đổi (có thể tăng lên) khi giá trị dự đoán lớn hơn, tức là phương sai không đồng đều (heteroscedasticity).

Result



Dựa vào đồ thị Residuals vs Leverage, ta có thể nhận xét như sau:

Đồ thị Residuals vs Leverage giúp xác định các điểm có ảnh hưởng lớn trong dữ liệu. Các điểm có ảnh hưởng cao có thể làm sai lệch kết quả phân tích hồi quy. Trong đồ thị này, các điểm như 60, 62, và 950 được xem là có tiềm năng ảnh hưởng đáng kể. Tuy nhiên, không có điểm nào vượt qua đường Cook's distance, điều này cho thấy không có điểm nào thực sự có ảnh hưởng quá mức đến mô hình hồi quy.

Sau đó, ta tiến hành kiểm tra mô hình với test_data, ta tiến hành thêm 1 biến là Predicted_Memory_Bandwidth là dữ liệu dự đoán cho mô hình:

Code

```
predicts <- predict(data_hqtt2, newdata = test_data)
test_data$Predicted_Memory_Bandwidth <- predicts
View(test_data)
```

Kết quả:

Result

	Manufacturer	Memory_Bandwidth	Memory_Bus	Texture_Rate	Memory_Speed	Year	Predicted_Memory_Bandwidth
9	AMD	160.0	256	62	1250	2014	122.882548
32	Intel	29.9	128	8	933	2015	21.722068
42	Intel	25.6	128	8	800	2015	17.179676
47	Nvidia	288.4	384	196	1502	2013	308.234262
54	Nvidia	317.2	384	214	1652	2013	337.716879
112	AMD	64.0	128	28	1000	2010	86.662136
117	AMD	264.0	384	101	1375	2014	190.602063
123	AMD	28.8	128	16	900	2012	52.506204
136	AMD	34.1	128	17	1067	2014	45.062267
146	AMD	34.1	128	20	1067	2013	56.372631
173	AMD	124.8	256	34	975	2009	111.849780
188	AMD	57.6	128	24	900	2009	85.083983
191	AMD	25.6	128	24	800	2009	81.668650
193	AMD	32.1	128	24	1004	2008	95.886355
195	AMD	22.4	128	19	700	2008	78.737184
244	AMD	8.5	64	3	533	2012	17.043288
246	AMD	25.6	128	22	800	2010	71.711599

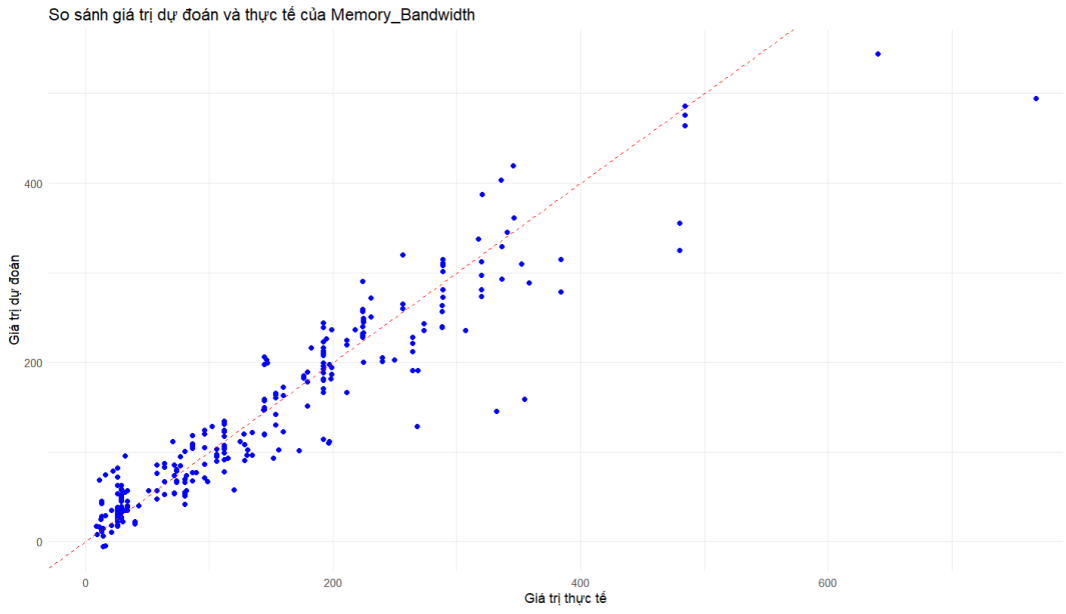
Để biết được giá trị dự đoán có chính xác hay không, ta vẽ đồ thị sau:

Code

```
ggplot(test_data, aes(x = Memory_Bandwidth, y = Predicted_Memory_Bandwidth)) +
  geom_point(color = "blue") +  # Vẽ các điểm phân tán
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +  # Thêm đường chéo
  labs(x = "Giá trị thực tế", y = "Giá trị dự đoán",
       title = "So sánh giá trị dự đoán và thực tế của Memory_Bandwidth") +
  theme_minimal()
```

Kết quả thu được:

Result



Nhận xét:

Phân bố dữ liệu và đường dự đoán:

- Đường đỏ (đường y = x) đại diện cho trường hợp lý tưởng khi giá trị dự đoán bằng với giá trị thực tế.
- Các điểm màu xanh biểu thị giá trị dự đoán so với giá trị thực. Phần lớn các điểm nằm gần đường lý tưởng, cho thấy mô hình hoạt động khá tốt.

Hiệu quả dự đoán:

- Ở vùng giá trị thấp (khoảng từ 0 đến 200), mô hình có vẻ dự đoán khá chính xác vì các điểm tập trung gần đường y = x.
- Ở các giá trị cao hơn (trên 400), có một số điểm nằm xa đường y = x, điều này cho thấy mô hình có thể bị sai lệch nhiều trong việc dự đoán giá trị lớn.

Cuối cùng, từ mô hình trên ta có thể dự đoán khả năng truyền tải bộ nhớ trung bình năm 2025 của các hãng "Intel", "ATI", "Nvidia" và “AMD”:

- Đầu tiên, ta sẽ tính các giá trị trung bình với các biến còn lại:

Code

```
average_values <- colMeans(data_hqtt[, c("Memory_Bus", "Texture_Rate", "Memory_Speed")], na.rm = TRUE)
```

- Tiếp theo, tạo bộ dữ liệu new_data có chứa cột dự báo:

Code

```
new_data <- data.frame(
  Manufacturer = c("Intel", "ATI", "Nvidia","AMD"),
  Year = 2025,
  Memory_Bus = average_values["Memory_Bus"],
  Texture_Rate = average_values["Texture_Rate"],
  Memory_Speed = average_values["Memory_Speed"]
)

# Dự báo giá trị Memory_Bandwidth cho năm 2025
predictions_2025 <- predict(data_hqtt2, newdata = new_data)

# Thêm cột dự báo vào data frame mới
new_data$Predicted_Memory_Bandwidth <- predictions_2025
```

Kết quả:

Result

	Manufacturer	Year	Memory_Bus	Texture_Rate	Memory_Speed	Predicted_Memory_Bandwidth
1	Intel	2025	217.3938	90.91896	1260.166	80.05946
2	ATI	2025	217.3938	90.91896	1260.166	47.94769
3	Nvidia	2025	217.3938	90.91896	1260.166	56.87253
4	AMD	2025	217.3938	90.91896	1260.166	79.39288

Khả năng truyền tải bộ nhớ trung bình với các hãng "Intel", "ATI", "Nvidia" và “AMD” năm 2025 lần lượt là 80.05946, 47.94769, 56.87253 và 79.39288 (GB/sec).

6 Thảo luận và mở rộng

Trong các mô hình hồi quy, khi dữ liệu có sự phân phối lệch hoặc các giá trị có sự chênh lệch lớn, việc sử dụng phép biến đổi log (logarithmic transformation) có thể giúp cải thiện chất lượng mô hình bằng cách làm giảm độ lệch của phân phối dữ liệu, ổn định phương sai và tạo ra mối quan hệ tuyến tính hơn giữa các biến.
Ở phần thống kê mô tả, ta đã biết được đồ thị histogram của biến phụ thuộc "Memory_Bandwidth" có dạng lệch phải và không có phân phối chuẩn, ta sẽ thử với bộ dữ liệu khi qua hàm log() như sau:

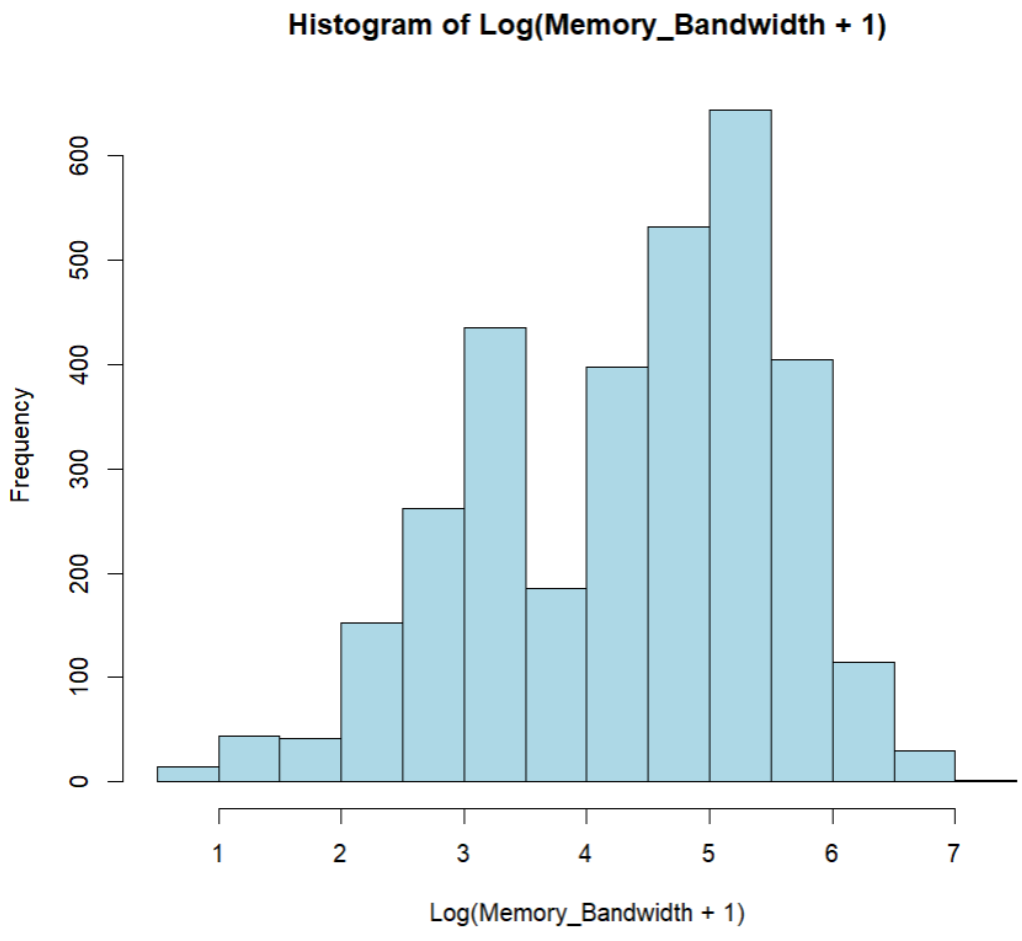
- Đầu tiên, ta vẽ đồ thị histogram sau khi qua log:

Code

```
# Biến đổi biến Memory_Bandwidth bằng log(x + 1)
data$Memory_Bandwidth <- log(data$Memory_Bandwidth + 1)

# Vẽ histogram cho biến Memory_Bandwidth sau khi biến đổi
hist(data$Memory_Bandwidth,
     main = "Histogram of Log(Memory_Bandwidth + 1)",
     xlab = "Log(Memory_Bandwidth + 1)",
     col = "lightblue",
     border = "black")
```

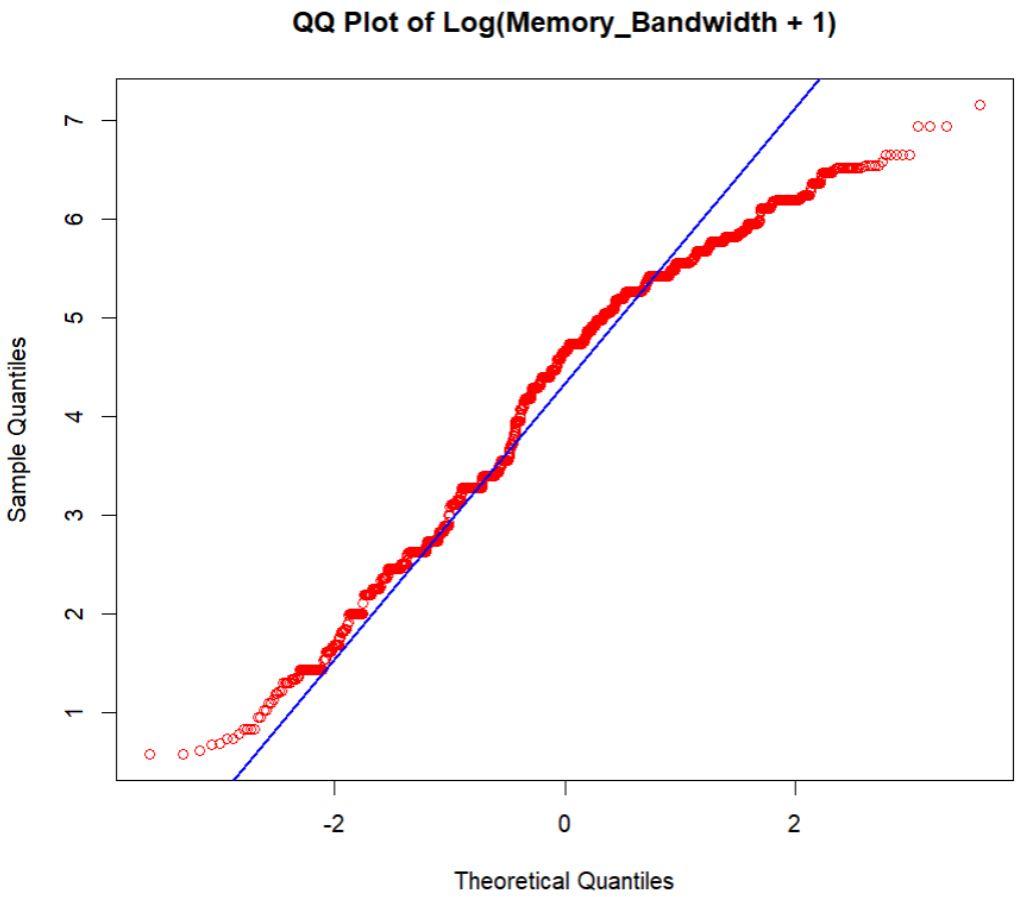
Việc sử dụng thêm hằng số +1 để tránh trường hợp log(0) trong bộ dữ liệu. Kết quả:

Result



Đồ thị có dạng phân phối chuẩn, để rõ hơn, ta vẽ Q-Q plot:

Result



Có thể thấy các giá trị nằm gần so với đường màu xanh, chỉ bị lệch khi tăng giá trị, dễ dàng hơn khi xây dựng mô hình hồi quy tuyến tính.

- Ta chuyển hết dữ liệu của train_data về dạng log(x+1) và xây dựng lại mô hình hồi quy:

Code

```
train_data$Memory_Bandwidth <- log(train_data$Memory_Bandwidth + 1)
train_data$Memory_Bus <- log(train_data$Memory_Bus + 1)
train_data$Texture_Rate <- log(train_data$Texture_Rate + 1)
train_data$Memory_Speed <- log(train_data$Memory_Speed + 1)

# Xây dựng lại mô hình hồi quy với dữ liệu đã biến đổi
data_hqtt_log_model <- lm(Memory_Bandwidth ~ ., data = train_data)
summary(data_hqtt_log_model)
```

Kết quả:

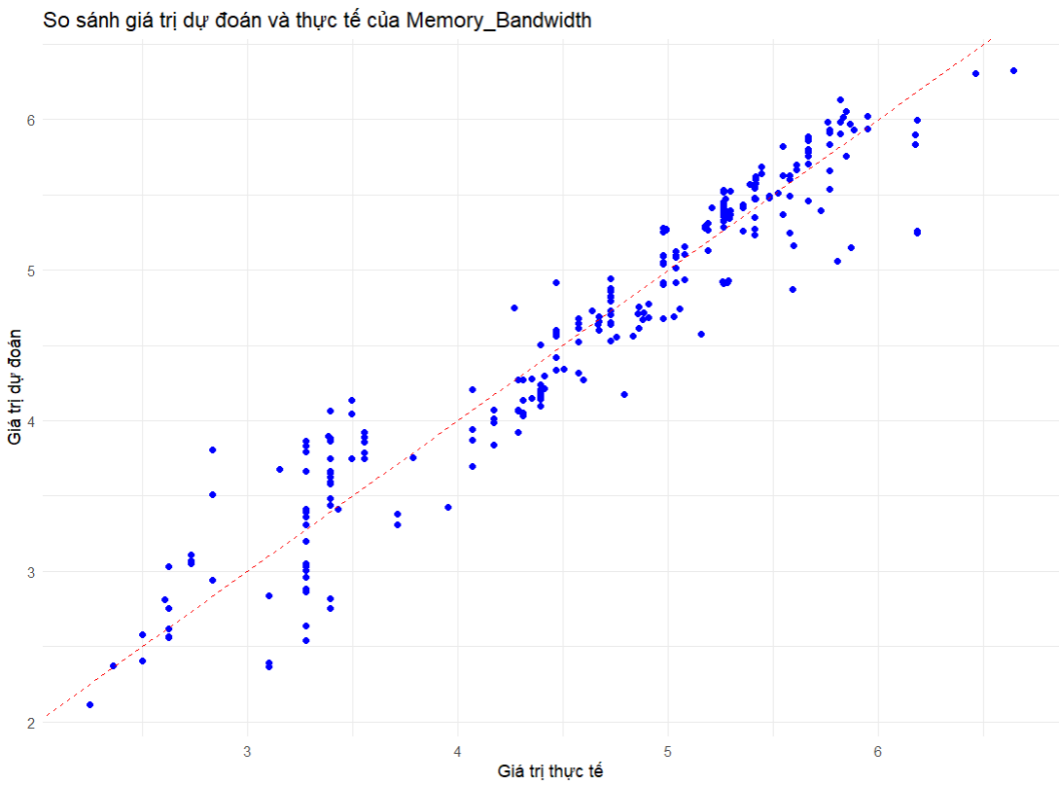
Result

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.94149 -0.14877 -0.02951  0.15703  1.25864

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       45.884493   6.908826   6.641 3.79e-11 ***
ManufacturerATI   -0.168502   0.033943  -4.964 7.36e-07 ***
ManufacturerIntel  0.170286   0.031277   5.444 5.70e-08 ***
ManufacturerNvidia -0.070294   0.011586  -6.067 1.50e-09 ***
Memory_Bus         0.546486   0.015859  34.460  < 2e-16 ***
Texture_Rate       0.561661   0.011760  47.758  < 2e-16 ***
Memory_Speed       0.670500   0.027261  24.595  < 2e-16 ***
Year              -0.025368   0.003447  -7.360 2.47e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2699 on 2512 degrees of freedom
Multiple R-squared:  0.9371,    Adjusted R-squared:  0.937
F-statistic:  5349 on 7 and 2512 DF,  p-value: < 2.2e-16
```

Hệ số xác định R^2 hiệu chỉnh là 0.937, lớn hơn đáng kể so với trước đó (0.9142), tăng tính chính xác khi dự đoán mô hình.

- Tương tự, ta cũng thiết lập dự báo cho test_data bằng mô hình hồi quy và vẽ đồ thị so sánh:

Result



Phần lớn các điểm nằm gần đường lý tưởng, cho thấy mô hình hoạt động khá tốt, ngay cả khi giá trị tăng lên.

Kết luận: Việc sử dụng phép biến đổi log(x + 1) giúp cải thiện mô hình hồi quy, đặc biệt khi dữ liệu có sự phân phối lệch, tức là khi có sự chênh lệch lớn giữa các giá trị nhỏ và lớn của các biến.

7 Nguồn dữ liệu và nguồn code

- Nguồn dữ liệu (link truy cap) : Kaggle
- Nguồn code (link truy cap): Code R

8 Tài liệu tham khảo

[1] D. C. Montgomery, *Applied Statistics And Probability For Engineers - 7th Edition*, Wiley Abridged Print Comp, 2017.

[2] Nguyễn Đình Huy, *GIÁO TRÌNH XÁC SUẤT VÀ THỐNG KÊ*, NXB Đại Học Quốc Gia TP.HCM, 2019.

[3] P. Dalgaard, *Introductory statistics with R*, Springer-Verlag New York, 2008.