

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH
KHOA KHOA HỌC ỨNG DỤNG
BỘ MÔN TOÁN ỨNG DỤNG



BÀI TẬP LỚN
XÁC SUẤT THỐNG KÊ (MT2013)

Computer Parts (CPUs and GPUs)

Giảng viên hướng dẫn: Huỳnh Thái Duy Phương
Nhóm: DL05, Nhóm 12

| STT | Họ và tên | MSSV | Lớp | Ngành học |
|-----|--------------------|---------|------|-------------------|
| 1 | Nguyễn Chí Thiết | 2213242 | DL05 | Khoa học máy tính |
| 2 | Mai Trần Kim Chi | 2210369 | DL05 | Cơ khí |
| 3 | Nguyễn Quan | 1732029 | DL05 | Cơ khí |
| 4 | Võ Huỳnh Thực Khuê | 2311717 | DL05 | Xây dựng |

Thành phố Hồ Chí Minh, tháng 7 năm 2024

Mục lục

| | | |
|----------|---|-----------|
| 1 | Tổng quan dữ liệu | 1 |
| 2 | Kiến thức nền | 2 |
| 2.1 | Lý thuyết mẫu | 2 |
| 2.1.1 | Các đặc trưng | 2 |
| 2.2 | Lí thuyết ước lượng | 2 |
| 2.2.1 | Các khái niệm | 2 |
| 2.2.2 | Bài toán tìm khoảng tin cậy đối xứng cho tỉ lệ trong tổng thể | 3 |
| 2.3 | Lí thuyết kiểm định | 3 |
| 2.3.1 | Bài toán kiểm định so sánh hai giá trị trung bình | 4 |
| 2.4 | Phân tích phương sai | 4 |
| 2.5 | Hồi quy tuyến tính bội | 6 |
| 2.5.1 | Khái niệm | 6 |
| 2.5.2 | Hệ số xác định | 6 |
| 3 | Tiền xử lý số liệu | 7 |
| 3.1 | Đọc dữ liệu | 7 |
| 3.2 | Chọn lọc dữ liệu | 7 |
| 3.3 | Xử lý định dạng dữ liệu | 8 |
| 3.4 | Xử lý dữ liệu khuyết | 11 |
| 4 | Thống kê mô tả | 13 |
| 4.1 | Các giá trị đặc trưng của mẫu | 13 |
| 4.2 | Các loại đồ thị | 14 |
| 4.3 | Mối liên hệ giữa các biến | 17 |
| 5 | Thống kê suy diễn | 19 |
| 5.1 | Tìm khoảng tin cậy một mẫu | 19 |
| 5.1.1 | Mục tiêu | 19 |
| 5.1.2 | Bài toán | 19 |
| 5.1.3 | Các lệnh R sử dụng | 19 |
| 5.1.4 | Tiền xử lý bài toán | 19 |
| 5.1.5 | Giải bài toán trên R | 19 |
| 5.1.6 | Nhận xét | 20 |
| 5.1.7 | Kết luận | 21 |
| 5.2 | Kiểm định hai mẫu | 22 |
| 5.2.1 | Mục tiêu | 22 |
| 5.2.2 | Bài toán | 22 |
| 5.2.3 | Các lệnh R sử dụng | 22 |
| 5.2.4 | Tiền xử lý bài toán | 22 |
| 5.2.5 | Giải quyết bài toán trên R | 24 |
| 5.2.6 | Nhận xét | 25 |
| 5.2.7 | Kết luận | 25 |
| 5.3 | Phân tích phương sai một yếu tố | 26 |
| 5.3.1 | Mục tiêu | 26 |
| 5.3.2 | Bài toán | 26 |
| 5.3.3 | Điều kiện để thực hiện bài toán Anova | 26 |
| 5.3.4 | Phân tích phương sai | 30 |

| | | |
|----------|---|-----------|
| 5.3.5 | Kết luận | 32 |
| 5.4 | Hồi quy tuyến tính | 33 |
| 5.4.1 | Mục tiêu | 33 |
| 5.4.2 | Bài toán | 33 |
| 5.4.3 | Xây dựng mô hình hồi quy tuyến tính | 33 |
| 5.4.4 | Kết luận | 39 |
| 6 | Thảo luận và mở rộng | 40 |
| 7 | Nguồn dữ liệu và nguồn code | 40 |
| | Tài liệu tham khảo | 41 |

1 Tổng quan dữ liệu

Tập tin **All_GPUs.csv** là một tập dữ liệu của hơn 3400 GPU (Graphic Processing Unit - một linh kiện chuyên dụng được thiết kế để xử lý các tác vụ liên quan đến đồ họa trên máy tính) Trong **hơn 30 thông số** về GPU được mô tả trong tập tin, dưới đây là một vài thông số đáng chú ý:

- **Best Resolution:** Độ phân giải tối ưu nhất mà GPU có thể xử lý để hiển thị hình ảnh chất lượng cao trên màn hình. Độ phân giải được đo bằng số điểm ảnh trên chiều rộng và chiều cao của màn hình, ví dụ: 1920x1080 (Full HD) hoặc 3840x2160 (4K Ultra HD). Độ phân giải cao hơn ngoài giới hạn này có thể gây ra hiện tượng giật, chậm hoặc không ổn định.
- **Core Speed:** Đây là tốc độ xử lý của GPU, được đo bằng MHz hoặc GHz, thể hiện tốc độ làm việc của các nhân xử lý trung tâm (core) trong GPU. Tốc độ này ảnh hưởng trực tiếp đến hiệu năng xử lý đồ họa của GPU.
- **Max Power:** Đây là công suất tối đa mà GPU có thể tiêu thụ khi hoạt động ở hiệu năng cao nhất. Đơn vị được sử dụng là watt (W). Thông thường, các GPU mạnh hơn thường tiêu thụ nhiều năng lượng hơn.
- **Memory:** Đây là bộ nhớ (VRAM - Video Random Access Memory) được sử dụng bởi GPU để lưu trữ dữ liệu đồ họa và thông tin cần thiết cho quá trình xử lý đồ họa. Bộ nhớ này có thể là GDDR6, GDDR6X hoặc các loại khác, và được đo bằng đơn vị gigabyte (GB). Độ lớn của bộ nhớ này ảnh hưởng đến khả năng GPU xử lý các tác vụ đồ họa phức tạp.
- **Memory Bandwidth:** Băng thông bộ nhớ, thể hiện tốc độ tối đa mà GPU có thể truyền qua VRAM trong một đơn vị thời gian. Đơn vị được sử dụng là gigabyte/giây (GB/s). Băng thông bộ nhớ càng cao, GPU càng có khả năng truyền dữ liệu nhanh hơn và cải thiện hiệu suất xử lý đồ họa.
- **Memory Bus:** Đây là kênh truyền dẫn dữ liệu giữa GPU và bộ nhớ VRAM. Độ rộng của Memory Bus ảnh hưởng đến khả năng truyền dữ liệu giữa GPU và bộ nhớ VRAM. Memory Bus thường được đo bằng bit, ví dụ 256-bit hoặc 384-bit.
- **Memory Speed:** Đây là tốc độ hoạt động mà GPU có thể truy xuất hoặc đọc dữ liệu từ bộ nhớ VRAM, được đo bằng MHz hoặc GHz.
- **Manufacturer:** Các nhà sản xuất GPU lớn như Nvidia, AMD, Intel,... Mỗi nhà sản xuất có sản phẩm GPU riêng của họ với các đặc điểm và tính năng riêng biệt.
- **Release Date:** Ngày phát hành mẫu GPU, vì là sản phẩm công nghệ, thông số này không thể bỏ qua khi nghiên cứu về GPU.

Bên cạnh đó, tập tin còn cung cấp các thông tin khác về GPU như **Pixel Rate** (Số lượng điểm ảnh), **Texture Rate** (Tốc độ làm đầy), **Architecture** (Kiến trúc của GPU) hay giá bán chính thức khi phát hành lần đầu (**Release price**), cũng như số lượng các cổng xuất hình ảnh như **Displayport**, **HDMI**, **VGA**,...

2 Kiến thức nền

2.1 Lý thuyết mẫu

Trong bài tập lớn này, nhóm chúng em trình bày lý thuyết mẫu như một công cụ làm tiền để phân tích các mô hình thống kê. Lý thuyết mẫu giúp nhận diện các khuôn mẫu lặp lại, từ đó suy rộng ra cho một số thông tin mô tả tổng thể.

Khi khảo sát tổng thể theo một dấu hiệu nghiên cứu nào đó, người ta mô hình hóa nó bởi một biến ngẫu nhiên X , gọi là **biến ngẫu nhiên gốc**.

Mẫu ngẫu nhiên một chiều gồm n biến ngẫu nhiên độc lập X_1, X_2, \dots, X_n được chọn từ biến ngẫu nhiên gốc, có cùng quy luật phân phối xác suất với X .

Tuy nhiên khi ta thực hiện một phép thử đối với các biến X_i , ta chỉ thu được các giá trị x_1, x_2, \dots, x_n . Các giá trị này tạo thành một **mẫu cụ thể**.

Ta chỉ có thể nghiên cứu trên các mẫu cụ thể này, rút ra các đặc trưng rồi từ đó suy rộng cho các đặc trưng của tổng thể. Vì vậy có sự phân biệt giữa các đặc trưng của mẫu và đặc trưng của tổng thể. Đặc trưng của tổng thể là những giá trị ta không thể đo được trực tiếp trên thực tế.

2.1.1 Các đặc trưng

Bảng sau tóm tắt các đặc trưng ta thường quan tâm khi nghiên cứu về tổng thể thông qua các mẫu đã thu thập được.

| Đặc trưng | Mẫu | Tổng thể |
|------------------------|--|-----------------------------------|
| Trung bình | $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ | μ |
| Phương sai | $s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$ | σ^2 |
| Độ lệch | $s = \sqrt{s^2}$ | σ |
| Tỉ lệ (Biến định tính) | $f = \frac{m}{n}$ | p |
| Sai số chuẩn | $SE(\bar{x}) = \frac{s}{\sqrt{n}}$ | $SE(X) = \frac{\sigma}{\sqrt{n}}$ |

Bảng 2.1. Bảng các đặc trưng của mẫu và tổng thể

Ngoài ra, mẫu còn có một số đặc trưng khác như Trung vị, Yếu vị, Tứ phân vị cũng được nhóm sử dụng trong bài tập lớn này.

2.2 Lí thuyết ước lượng

2.2.1 Các khái niệm

Lý thuyết ước lượng cung cấp công cụ liên quan đến việc xác định giá trị cho các đặc trưng của một tổng thể dựa trên mẫu rút ra. Một số đặc trưng nhóm sẽ ứng dụng lý thuyết ước lượng như: trung bình tổng thể, tỉ lệ các phần tử mang cùng một tính chất.

Có hai phương pháp ước lượng gồm ước lượng điểm và ước lượng khoảng. Nội dung chủ yếu của đề tài này sẽ tập trung vào phương pháp ước lượng khoảng.

Ước lượng khoảng: tìm ra khoảng ước lượng $(G_1; G_2)$ cho tham số θ trong tổng thể sao cho ứng với độ tin cậy (confidence) bằng $1 - \alpha$ cho trước, $\mathbb{P}(G_1 < \theta < G_2) = 1 - \alpha$. Cụ thể:

- α là khả năng mắc sai lầm của phương pháp, có giá trị nhỏ.
- $1 - \alpha$ là độ tin cậy của khoảng ước lượng tìm được.

Trong khuôn khổ bài tập lớn này, nhóm khảo sát khoảng tin cậy đối xứng cho tỉ lệ các thành phần cùng một tính chất trong tổng thể, dựa trên một mẫu. Đây thường được gọi là bài toán tìm khoảng ước lượng cho tỉ lệ một mẫu.

2.2.2 Bài toán tìm khoảng tin cậy đối xứng cho tỉ lệ trong tổng thể

Khoảng ước lượng cần tìm cho tỉ lệ p các thành phần cùng tính chất trong tổng thể có dạng:

$$(f - \varepsilon; f + \varepsilon)$$

Giá trị f là giá trị đo được trên một mẫu cụ thể. Giá trị ε được tính như sau:

$$\varepsilon = \frac{z_{\alpha/2} \times \sqrt{f(1-f)}}{\sqrt{n}}$$

Trong đó, $z_{\alpha/2}$ là giá trị trong phân phối chuẩn, mang ý nghĩa $\mathbb{P}(z_{\alpha/2} < X < \infty) = \alpha/2$

2.3 Lí thuyết kiểm định

Kiểm định giả thiết thống kê là phương pháp để đánh giá giả thiết về dữ liệu để xác định xem có đủ bằng chứng để chấp nhận hay bác bỏ giả thiết đó. Mục tiêu nhằm đưa ra kết luận về tổng thể dựa trên dữ liệu mẫu có sẵn.

Đây là một trong những nội dung trọng tâm nhóm sẽ sử dụng trong bài tập lớn.

Quy trình kiểm định bao gồm các bước chính:

Đặt các giả thiết

- **Giả thiết không H_0** : giả thiết về yếu tố cần kiểm định của tổng thể ở trạng thái bình thường, không chịu tác động từ hiện tượng liên quan.
- **Giả thiết thay thế H_1** : giả thuyết mà chúng ta muốn chấp nhận nếu có đủ bằng chứng để bác bỏ giả thuyết H_0 , mâu thuẫn với H_0 và thể hiện xu hướng cần kiểm định.

Đưa ra tiêu chuẩn kiểm định

Đưa ra hàm cụ thể $G = G(X_1, X_2, \dots, X_n, \theta_0)$ xây dựng trên mẫu ngẫu nhiên $W = (X_1, X_2, \dots, X_n)$ và tham số θ_0 liên quan đến H_0 . Nếu H_0 đúng thì quy luật phân phối xác suất của G hoàn toàn xác định.

Tìm miền bác bỏ RR

Miền số thực thỏa $\mathbb{P}(G \in \text{RR} \mid H_0 \text{ đúng}) = \alpha$, tức với điều kiện H_0 đúng thì xác suất G thuộc vào đó là α .

Số α thường dưới 10%, được gọi là **mức ý nghĩa** của kiểm định.

Tính giá trị kiểm định và kết luận

Tính giá trị cụ thể từ mẫu thực nghiệm, $g_{qs} = G(X_1, X_2, \dots, X_n, \theta_0)$. Nếu

- $g_{qs} \in \text{RR}$: bác bỏ giả thuyết H_0 , chấp nhận H_1 .

- $g_{qs} \notin \text{RR}$: chưa đủ bằng chứng kết luận H_0 sai, do đó chưa thể chấp nhận H_1 .

Dựa vào cơ sở lý thuyết trên, nhóm ứng dụng vào bài toán kiểm định so sánh giá trị trung bình cho hai tổng thể độc lập dựa trên hai mẫu độc lập.

2.3.1 Bài toán kiểm định so sánh hai giá trị trung bình

Bài toán này có nhiều trường hợp, do đó cần xác định trường hợp cụ thể để giả quyết đúng. Dưới đây là bảng tóm tắt ứng với các trường hợp **hai mẫu độc lập** mà nhóm sẽ ứng dụng ở nội dung phía dưới.

| Phân phối | H_1 | Miền bác bỏ H_0 | Tiêu chuẩn |
|---|--------------------|---|---|
| X_1, X_2 phân phối chuẩn Đã biết σ_1^2, σ_2^2 | $\mu_1 \neq \mu_2$ | $(-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; +\infty)$ | $Z_{qs} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$ |
| | $\mu_1 < \mu_2$ | $(-\infty; -z_{\alpha})$ | |
| | $\mu_1 > \mu_2$ | $(z_{\alpha}; +\infty)$ | |
| X_1, X_2 phân phối chuẩn Chưa biết σ_1^2, σ_2^2 $s_1/s_2 \in [1/2; 2]$ | $\mu_1 \neq \mu_2$ | $(-\infty; -t_{\alpha/2; (n_1+n_2-2)}) \cup (t_{\alpha/2; (n_1+n_2-2)}; +\infty)$ | $Z_{qs} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_p^2}{n_1} + \frac{s_p^2}{n_2}}}$ |
| | $\mu_1 < \mu_2$ | $(-\infty; -t_{\alpha; (n_1+n_2-2)})$ | |
| | $\mu_1 > \mu_2$ | $(t_{\alpha; (n_1+n_2-2)}; +\infty)$ | |
| X_1, X_2 phân phối chuẩn Chưa biết σ_1^2, σ_2^2 $s_1/s_2 \notin [1/2; 2]$ | $\mu_1 \neq \mu_2$ | $(-\infty; -t_{\alpha/2; (v)}) \cup (t_{\alpha/2; (v)}; +\infty)$ | $Z_{qs} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$ |
| | $\mu_1 < \mu_2$ | $(-\infty; -t_{\alpha; (v)})$ | |
| | $\mu_1 > \mu_2$ | $(t_{\alpha; (v)}; +\infty)$ | |
| X_1, X_2 phân phối tùy ý Kích thước hai mẫu lớn $n_1, n_2 \geq 30$ Biết hoặc chưa biết σ_1^2, σ_2^2 | $\mu_1 \neq \mu_2$ | $(-\infty; -z_{\alpha/2}) \cup (z_{\alpha/2}; +\infty)$ | $Z_{qs} = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$ |
| | $\mu_1 < \mu_2$ | $(-\infty; -z_{\alpha})$ | |
| | $\mu_1 > \mu_2$ | $(z_{\alpha}; +\infty)$ | |

Bảng 2.2. Bảng phân dạng kiểm định trung bình hai mẫu độc lập

Trong đó, các giá trị v, s_p có giá trị

- Bậc tự do $v = \frac{(A+B)^2}{\frac{A^2}{n_1-1} + \frac{B^2}{n_2-1}}$ với $A = \frac{s_1^2}{n_1}$ và $B = \frac{s_2^2}{n_2}$.
- Mẫu gộp $s_p^2 = \frac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1 + n_2 - 2}$

2.4 Phân tích phương sai

Analysis of Variance hay ANOVA, là một kỹ thuật thống kê được sử dụng để so sánh sự khác biệt giữa trung bình của hai hoặc nhiều nhóm dựa trên trung bình của các mẫu quan sát được.

Nhóm tập trung trình bày về phương pháp phân tích phương sai một yếu tố, tức so sánh trung bình giữa nhiều nhóm dựa trên một yếu tố độc lập.

Bài toán phân tích phương sai một yếu tố yêu cầu các điều kiện sau:

- Các tổng thể có phân phối chuẩn $\mathcal{N}(\mu, \sigma^2)$

- Phương sai các tổng thể bằng nhau $\sigma_1^2 = \sigma_2^2 = \dots = \sigma_k^2$.
- Các mẫu từ các tổng thể được lấy độc lập.

Đặt giả thiết kiểm định

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

Tính các giá trị kiểm định

Tính toán các giá trị kiểm định đặc trưng cho các mẫu ở từng nhóm (hay tổng thể).

| Đặc trưng | Nhóm 1 | Nhóm 2 | ... | Nhóm k |
|--------------------|--|-------------|-----|-------------|
| Kích thước mẫu | n_1 | n_2 | | n_k |
| Trung bình 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 | $\bar{x} = \frac{1}{N} \sum_{i=1}^k \sum_{j=1}^{n_i} x_{ij}$ | | | |

| Đại lượng | Miêu tả | Công thức |
|-----------|--------------------------------|--|
| SSB | Sự biến thiên giữa các mẫu | $\sum_{i=1}^k n_i (\bar{x}_i - \bar{x})^2$ |
| SSW | Tổng biến thiên trong từng mẫu | $\sum_{i=1}^k \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)^2$ |
| SST | Tổng bình phương chênh lệch | $\sum_{i=1}^k \sum_{j=1}^{n_i} (x_{ij} - \bar{x})^2 = SSW + SSB$ |

Bảng 2.3. Bảng các đặc trưng của mẫu từ các nhóm

Về mặt ý nghĩa, các đại lượng trên mô tả:

- **SSB**: Biến thiên của biến nghiên cứu do yếu tố xem xét tạo ra
- **SSW**: Biến thiên của biến nghiên cứu do các yếu tố khác không đề cập
- **SST**: Tổng biến thiên của biến do tất cả các yếu tố

Tìm miền bác bỏ

Miền bác bỏ có dạng: $RR = (f_\alpha(k-1, N-k), +\infty)$ với f là giá trị trong phân phối Fisher cho α .

Tính giá trị kiểm định $F = \frac{MSB}{MSW}$

Với $MSB = \frac{SSB}{k-1}$ và $MSW = \frac{SSW}{N-k}$

Kết luận

$F \in RR$, chấp nhận H_1 , ngược lại, vẫn chấp nhận H_0 , nghĩa là chưa khẳng định được H_0 sai.

Hệ số xác định R^2 : $R^2 = \frac{SSB}{SST} \times 100\%$

Hệ số xác định 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 . R^2 càng lớn, mô hình càng thích hợp.

2.5 Hồi quy tuyến tính bội

2.5.1 Khái niệm

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.

Nhóm tập trung phân tích và áp dụng mô hình hồi quy tuyến tính bội trong đề tài.

Mô hình hồi quy tuyến tính bội giả định rằng có một mối quan hệ tuyến tính giữa biến phụ thuộc và các biến độc lập, và mục tiêu là tìm ra mối quan hệ đó một cách tốt nhất.

Phương trình hồi quy tuyến tính bội có dạng:

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

Trong đó:

- Y : biến phụ thuộc.
- X_i : các biến độc lập, dùng để mô tả Y .
- β_i : các hệ số hồi quy, mức độ biến thiên của Y khi X_i thay đổi.
- Hệ số β_0 : hệ số chặn, giá trị dự đoán của biến phụ thuộc khi tất cả các biến độc lập đều bằng 0.
- ε : sai số ngẫu nhiên, thành phần không thể giải thích bằng các biến độc lập trong mô hình.

2.5.2 Hệ số xác định

Hệ số xác định R^2 giúp đánh giá mức độ phù hợp mô hình hồi quy. Giá trị này càng cao, mô hình càng gọi là thích hợp.

Hệ số xác định R^2 được tính:

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

Trong đó:

- **SSR**: Đo lường sai số do khác biệt giữa đường hồi quy mẫu và trung bình của Y .
- **SST**: Đo lường tổng biến động các giá trị quan sát y_i xung quanh giá trị trung bình của mẫu.

3 Tiền xử lý số liệu

3.1 Đọc dữ liệu

Để đọc dữ liệu trong tập tin **All_GPUs.csv**, sử dụng lệnh `read.csv("All_GPUs.csv")`, lưu vào biến `dat`. Lệnh `head(dat)` để hiển thị một số dòng đầu tiên của dữ liệu, mặc định là 6 dòng để xem tổng quát bảng dữ liệu:

```
1 dat<-read.csv(All_GPUs.csv)
2 head(dat)
```

Source

Console Terminal Background Jobs

R 4.3.3 · ~/

> head(dat)

| | Architecture | Best_Resolution | Boost_Clock | Core_Speed | DVI_Connection | Dedicated | Direct_X | DisplayPort_Connection | HDMI_Connection | Integrated | L2_Cache | Manufacturer | Max_Power | Memory | Memory_Bandwidth | Memory_Bus | Memory_Speed | Memory_Type | Name | Notebook_GPU |
|---|--------------|--------------------|-------------|-----------------|----------------|-----------|---------------|------------------------|-----------------|---------------|-------------|--------------|----------------|---------|------------------|------------|--------------|-------------|-----------------------------------|--------------|
| 1 | Tesla G92b | | | 738 MHz | 2 | Yes | DX 10.0 | | | | | Nvidia | 141 Watts | 1024 MB | | | | | GeForce GTS 150 | No |
| 2 | R600 XT | 1366 x 768 | | \n- | 2 | Yes | DX 10 | | | | | AMD | 215 Watts | 512 MB | | | | | Radeon HD 2900 XT | No |
| 3 | R600 PRO | 1366 x 768 | | \n- | 2 | Yes | DX 10 | | | | | AMD | 200 Watts | 512 MB | | | | | Radeon HD 2900 Pro | No |
| 4 | RV630 | 1024 x 768 | | \n- | 2 | Yes | DX 10 | | | | | AMD | | 256 MB | | | | | Radeon HD 2600 XT Diamond Edition | No |
| 5 | RV630 | 1024 x 768 | | \n- | 2 | Yes | DX 10 | | | | | AMD | 45 Watts | 256 MB | | | | | Radeon HD 2600 XT | No |
| 6 | RV630 | 1024 x 768 | | \n- | 2 | Yes | DX 10 | | | | | AMD | 50 Watts | 256 MB | | | | | Radeon HD 2600 XT 256MB GDDR4 | No |
| | Open_GL | PSU | Pixel_Rate | Power_Connector | Process | ROPs | Release_Date | Release_Price | Resolution_WxH | SLI_Crossfire | Shader_TMUs | Texture_Rate | VGA_Connection | | | | | | | |
| 1 | 3.3 | 450 Watt & 38 Amps | 12 GPixel/s | None | 55nm | 16 | \n01-Mar-2009 | | 2560x1600 | Yes | 4 | 64 | 47 GTexel/s | 0 | | | | | | |
| 2 | 3.1 | 550 Watt & 35 Amps | 12 GPixel/s | None | 80nm | 16 | \n14-May-2007 | | 2560x1600 | Yes | 4 | 16 | 12 GTexel/s | 0 | | | | | | |
| 3 | 3.1 | 550 Watt & 35 Amps | 10 GPixel/s | None | 80nm | 16 | \n01-Dec-2007 | | 2560x1600 | Yes | 4 | 16 | 10 GTexel/s | 0 | | | | | | |
| 4 | 3.3 | | 3 GPixel/s | None | 65nm | 4 | \n01-Jul-2007 | | 2560x1600 | Yes | 4 | 8 | 7 GTexel/s | 0 | | | | | | |
| 5 | 3.1 | 400 Watt & 25 Amps | 3 GPixel/s | None | 65nm | 4 | \n28-Jun-2007 | | 2560x1600 | Yes | 4 | 8 | 6 GTexel/s | 0 | | | | | | |
| 6 | 3.3 | 400 Watt & 26 Amps | 3 GPixel/s | None | 65nm | 4 | \n26-Jun-2007 | | 2560x1600 | Yes | 4 | 8 | 6 GTexel/s | 0 | | | | | | |

> |

Hình 3.1. Một vài dòng đầu tiên của dữ liệu

3.2 Chọn lọc dữ liệu

Chọn ra 10 thuộc tính cần thiết từ 34 thuộc tính trong dữ liệu GPUs ban đầu: **Name**, **Best_Resolution**, **Core_Speed**, **Max_Power**, **Memory**, **Memory_Bandwidth**, **Memory_Speed**, **Manufacturer**, **Release_Date**, **Release_Price**.

```
1 dat<-dat[,c("Name", "Best_Resolution", "Core_Speed", "Max_Power", "Memory", "Memory_Bandwidth",
, "Memory_Speed", "Manufacturer", "Release_Date", "Release_Price")]
2 View(dat)
```

| Name | Best_Resolution | Core_Speed | Max_Power | Memory | Memory_Bandwidth | Memory_Speed | Manufacturer | Release_Date | Release_Price |
|---|-----------------|------------|-----------|----------|------------------|--------------|--------------|--------------|---------------|
| 1 GeForce GTS 150 | | 738 MHz | 141 Watts | 1024 MB | 64GB/sec | 1000 MHz | Nvidia | 01-Mar-2009 | |
| 2 Radeon HD 2900 XT 512MB | 1366 x 768 | - | 215 Watts | 512 MB | 106GB/sec | 828 MHz | AMD | 14-May-2007 | |
| 3 Radeon HD 2900 Pro | 1366 x 768 | - | 200 Watts | 512 MB | 51.2GB/sec | 800 MHz | AMD | 07-Dec-2007 | |
| 4 Radeon HD 2600 XT Diamond Edition | 1024 x 768 | - | | 256 MB | 36.8GB/sec | 1150 MHz | AMD | 01-Jul-2007 | |
| 5 Radeon HD 2600 XT | 1024 x 768 | - | 45 Watts | 256 MB | 22.4GB/sec | 700 MHz | AMD | 28-Jun-2007 | |
| 6 Radeon HD 2600 XT 256MB GDDR4 | 1024 x 768 | - | 50 Watts | 256 MB | 35.2GB/sec | 1100 MHz | AMD | 26-Jun-2007 | |
| 7 Radeon HD 4890 Sapphire Vapor-X OC 2GB Edition | 1920 x 1080 | 870 MHz | 190 Watts | 2048 MB | 134.4GB/sec | 1050 MHz | AMD | 13-Jul-2009 | |
| 8 Radeon HD 2900 GT | 1024 x 768 | - | 150 Watts | 256 MB | 51.2GB/sec | 800 MHz | AMD | 06-Nov-2007 | |
| 9 FirePro D300 | 1920 x 1080 | - | 150 Watts | 2048 MB | 160GB/sec | 1250 MHz | AMD | 18-Jan-2014 | |
| 10 Radeon 7000 64mb | | - | 32 Watts | 64 MB | 2.9GB/sec | 366 MHz | AMD | 02-Jan-2001 | |
| 11 Quadro4 980 XGL | | - | | 128 MB | 5.2GB/sec | 325 MHz | Nvidia | 01-Nov-2002 | |
| 12 Tesla M2090 | 1920 x 1080 | 650 MHz | 250 Watts | 6144 MB | 177.6GB/sec | 925 MHz | Nvidia | 25-Jul-2011 | |
| 13 Tesla K20 | | 705 MHz | 225 Watts | 5120 MB | 168GB/sec | 1050 MHz | Nvidia | 01-Nov-2012 | |
| 14 Tesla K40c | 2560 x 1600 | 706 MHz | 245 Watts | 12288 MB | 288.4GB/sec | 1502 MHz | Nvidia | 12-Nov-2013 | |
| 15 All-in-Wonder Radeon 7500 | | - | | 64 MB | 5.8GB/sec | 360 MHz | AMD | 22-Jan-2002 | |
| 16 Radeon R7 250 v2 MSI OC 2GB + Radeon R7 7870K Dual | 1600 x 900 | 1050 MHz | 150 Watts | 3072 MB | 57.6GB/sec | 900 MHz | AMD | 28-May-2015 | |
| 17 Tesla K10 | | - | 300 Watts | 8192 MB | 320GB/sec | 1250 MHz | Nvidia | 15-May-2012 | |
| 18 Tesla K20X | | 732 MHz | 235 Watts | 6144 MB | 249.6GB/sec | 1300 MHz | Nvidia | 12-Nov-2012 | |
| 19 Iris i3 6167U | 1366 x 768 | 300 MHz | 28 Watts | | 34.1GB/sec | 1067 MHz | Intel | 01-Sep-2015 | |
| 20 Radeon 9800 XT | 1024 x 768 | - | 60 Watts | 256 MB | 23.4GB/sec | 730 MHz | AMD | 01-Oct-2003 | |
| 21 Tesla C2070 | | 575 MHz | 238 Watts | 6144 MB | 144GB/sec | 750 MHz | Nvidia | 01-Sep-2010 | |
| 22 Tesla C2075 | | 575 MHz | 247 Watts | 6144 MB | 144GB/sec | 750 MHz | Nvidia | 01-Jul-2011 | |
| 23 Quadro4 750 XGL | | - | | 128 MB | 3.6GB/sec | 225 MHz | Nvidia | 19-Feb-2002 | |
| 24 Quadro4 900 XGL | | - | | 128 MB | 5.2GB/sec | 325 MHz | Nvidia | 01-Nov-2002 | |
| 25 Iris i5 6360U | 1366 x 768 | 300 MHz | 15 Watts | | 34.1GB/sec | 1067 MHz | Intel | 01-Sep-2015 | |
| 26 Quadro4 780 XGL | | - | | 128 MB | 4.4GB/sec | 275 MHz | Nvidia | 01-Nov-2002 | |

Hình 3.2. Vài phần tử của dữ liệu sau khi chọn lọc

3.3 Xử lý định dạng dữ liệu

Sử dụng thư viện `janitor` và `tidyr` của R để thực hiện định dạng lại tên biến, xóa các hàng và cột trống (nếu có), tính toán sơ bộ.

```

1 library(tidyr)
2 library(janitor)
3 # Định dạng lại tên biến
4 dat2 <- clean_names(dat)
5 head(dat2)
6 # Xóa hàng và cột trống
7 dat3 <- remove_empty(dat2, which = c("rows", "cols"), quiet=FALSE)
8 head(dat3)

> head(dat3)
  name best_resolution core_speed max_power memory memory_bandwidth memory_speed manufacturer release_date release_price
1   GeForce GTS 150      1366 x 768      738 MHz 141 Watts 1024 MB      64GB/sec      1000 MHz      Nvidia \n01-Mar-2009
2   Radeon HD 2900 XT 512MB 1366 x 768      \n-    215 Watts 512 MB      106GB/sec      828 MHz      AMD \n14-May-2007
3   Radeon HD 2900 Pro      1366 x 768      \n-    200 Watts 512 MB      51.2GB/sec      800 MHz      AMD \n07-Dec-2007
4   Radeon HD 2600 XT Diamond Edition 1024 x 768      \n-    256 MB      36.8GB/sec      1150 MHz      AMD \n01-Jul-2007
5   Radeon HD 2600 XT      1024 x 768      \n-    45 Watts 256 MB      22.4GB/sec      700 MHz      AMD \n28-Jun-2007
6   Radeon HD 2600 XT 256MB GDDR4 1024 x 768      \n-    50 Watts 256 MB      35.2GB/sec      1100 MHz      AMD \n26-Jun-2007

```

Hình 3.3. Vài phần tử đầu của dữ liệu sau khi định dạng

Mẫu `best_resolution` đề cập đến độ phân giải tối ưu mà GPU hoặc thiết bị có thể hỗ trợ, bao gồm số lượng pixel trên mỗi chiều (chiều rộng x chiều cao). Do đó ta đổi tên mẫu này thành `number_of_pixels` và tính toán số điểm ảnh (pixels).

```

1 # Đổi tên best_resolution thành number_of_pixels
2 names(dat3)[ names (dat3) == "best_resolution" ] <- "number_of_pixels"
3 # Thực hiện phép nhân số pixel hai chiều ngang và dọc
4 dat3$number_of_pixels <- sapply (strsplit(dat3$number_of_pixels, "x"), function(x) as.numeric (
  x [1])*as.numeric (x[2]))

```

Đối với các mẫu `core_speed`, `max_power`, `memory`, `memory_bandwidth`, `memory_speed` để dễ dàng thao tác tính toán, cần tách mỗi mẫu thành hai cột gồm phần **giá trị** và phần **đơn vị**:

```
1 # Loại bỏ giá trị khuyết biểu diễn bởi "-"
2 dat3$core_speed[grepl("-", dat3$core_speed)] <- ""
3 # Chia thành 2 cột core_speed_value và core_speed_unit
4 dat3 <- separate(dat3,col = core_speed, into = c("core_speed_value",
5         "core_speed_unit"),sep=" ", fill="right")
6 # Đổi giá trị thành dạng số
7 dat3$core_speed_value <- as.numeric(dat3$core_speed_value)
8 # Thực hiện tương tự cho các cột còn lại
9 dat3 <- separate(dat3, col = max_power, into = c("max_power_value",
10         "max_power_unit"), sep=" ", fill = "right")
11 dat3$max_power_value <- as.numeric(dat3$max_power_value)
12
13 dat3 <- separate(dat3, col = memory, into = c("memory_value",
14         "memory_unit"), sep=" ", fill = "right", extra = "drop")
15 dat3$memory_value <- as.numeric(dat3$memory_value)
16
17 dat3 <- separate(dat3, col = memory_bandwidth,
18         into = c("memory_bandwidth_value", "memory_bandwidth_unit"),
19         sep = "(?<=\\d)(?=[A-Za-z])", fill = "right")
20 dat3$memory_bandwidth_value <- as.numeric(dat3$memory_bandwidth_value)
21
22 dat3 <- separate(dat3, col = memory_speed ,
23         into = c("memory_speed_value", "memory_speed_unit"),
24         sep = " ", fill = "right")
25 dat3$memory_speed_value <- as.numeric (dat3$memory_speed_value)
```

Trong đó:

- function `separate()` của thư viện `tidyr`
- `separate()` chứa `col` là tên cột cần phân tách, `into` là vector chứa tên của cột mới, `sep` kí tự dùng để phân tách cột dữ liệu, `fill="right"` điền giá trị mới vào bên phải của kết quả

| name | number_of_pixels | core_speed_value | core_speed_unit | max_power_value | max_power_unit | memory_value | memory_unit | memory_bandwidth_value | memory_bandwidth_unit |
|---|------------------|------------------|-----------------|-----------------|----------------|--------------|-------------|------------------------|-----------------------|
| 1 GeForce GTX 150 | N/A | 738 | MHz | 141 | Watts | 1024 | MB | 64.0 | GB/sec |
| 2 Radeon HD 2900 XT 512MB | 1049088 | N/A | N/A | 215 | Watts | 512 | MB | 106.0 | GB/sec |
| 3 Radeon HD 2900 Pro | 1049088 | N/A | N/A | 200 | Watts | 512 | MB | 51.2 | GB/sec |
| 4 Radeon HD 2600 XT Diamond Edition | 786432 | N/A | N/A | N/A | N/A | 256 | MB | 36.8 | GB/sec |
| 5 Radeon HD 2600 XT | 786432 | N/A | N/A | 45 | Watts | 256 | MB | 22.4 | GB/sec |
| 6 Radeon HD 2600 XT 256MB GDDR4 | 786432 | N/A | N/A | 50 | Watts | 256 | MB | 35.2 | GB/sec |
| 7 Radeon HD 4890 Sapphire Vapor-X OC 2GB Edition | 2073600 | 870 | MHz | 190 | Watts | 2048 | MB | 134.4 | GB/sec |
| 8 Radeon HD 2900 GT | 786432 | N/A | N/A | 150 | Watts | 256 | MB | 51.2 | GB/sec |
| 9 FirePro D300 | 2073600 | N/A | N/A | 150 | Watts | 2048 | MB | 160.0 | GB/sec |
| 10 Radeon 7000 64mb | N/A | N/A | N/A | 32 | Watts | 64 | MB | 2.9 | GB/sec |
| 11 Quadro4 980 XGL | N/A | N/A | N/A | N/A | N/A | 128 | MB | 5.2 | GB/sec |
| 12 Tesla M2090 | 2073600 | 650 | MHz | 250 | Watts | 6144 | MB | 177.6 | GB/sec |
| 13 Tesla K20 | N/A | 705 | MHz | 225 | Watts | 5120 | MB | 168.0 | GB/sec |
| 14 Tesla K40c | 4096000 | 706 | MHz | 245 | Watts | 12288 | MB | 288.4 | GB/sec |
| 15 All-in-Wonder Radeon 7500 | N/A | N/A | N/A | N/A | N/A | 64 | MB | 5.8 | GB/sec |
| 16 Radeon R7 250 v2 MSI OC 2GB + Radeon R7 7870K Dual | 1440000 | 1050 | MHz | 150 | Watts | 3072 | MB | 57.6 | GB/sec |
| 17 Tesla K10 | N/A | N/A | N/A | 300 | Watts | 8192 | MB | 320.0 | GB/sec |
| 18 Tesla K20X | N/A | 732 | MHz | 235 | Watts | 6144 | MB | 249.6 | GB/sec |
| 19 Iris i3 6167U | 1049088 | 300 | MHz | 28 | Watts | N/A | N/A | 34.1 | GB/sec |
| 20 Radeon 9800 XT | 786432 | N/A | N/A | 60 | Watts | 256 | MB | 23.4 | GB/sec |
| 21 Tesla C2070 | N/A | 575 | MHz | 238 | Watts | 6144 | MB | 144.0 | GB/sec |
| 22 Tesla C2075 | N/A | 575 | MHz | 247 | Watts | 6144 | MB | 144.0 | GB/sec |
| 23 Quadro4 750 XGL | N/A | N/A | N/A | N/A | N/A | 128 | MB | 3.6 | GB/sec |
| 24 Quadro4 900 XGL | N/A | N/A | N/A | N/A | N/A | 128 | MB | 5.2 | GB/sec |
| 25 Iris i5 6360U | 1049088 | 300 | MHz | 15 | Watts | N/A | N/A | 34.1 | GB/sec |

Hình 3.4. Kết quả sau khi tách cột

Kiểm tra số lượng đơn vị của một thuộc tính bằng lệnh `table()`:

```
1 table(dat3$core_speed_unit)
```

```

2 table(dat3$max_power_unit)
3 table(dat3$memory_unit)
4 table(dat3$memory_bandwidth_unit)
5 table(dat3$memory_speed_unit)

> table(dat3$core_speed_unit)
MHz
2470
> table(dat3$max_power_unit)
Watts
2781
> table(dat3$memory_unit)
MB
2986
> table(dat3$memory_bandwidth_unit)
GB/sec MB/sec
3281 4
> table(dat3$memory_speed_unit)
MHz
3301
>

```

Hình 3.5. Số lượng các đơn vị

Trong những thuộc tính trên, chỉ có **memory_bandwidth_unit** là chứa hai đơn vị $[MB/sec]$ và $[GB/sec]$. Do đó ta cần đồng nhất dữ liệu bằng cách đưa thuộc tính này về cùng một đơn vị là $[GB/sec]$ bằng hàm `ifelse(test,yes,no)` trong R.

```

1 dat3$memory_bandwidth_value <- ifelse(dat3$memory_bandwidth_unit == "MB/sec", dat3$
  memory_bandwidth_value/1024, dat3$memory_bandwidth_value)
2 dat3$memory_bandwidth_unit <- ifelse(dat3$memory_bandwidth_unit == "MB/sec", "GB/sec", dat3$
  memory_bandwidth_unit)

```

```

> dat3$memory_bandwidth_value <- ifelse(dat3$memory_bandwidth_unit == "MB/sec",
  dat3$memory_bandwidth_value/1024, dat3$memory_bandwidth_value)
> dat3$memory_bandwidth_unit <- ifelse(dat3$memory_bandwidth_unit == "MB/sec",
  "GB/sec", "GB/sec")
> table(dat3$memory_bandwidth_unit)
GB/sec
3285
> |

```

Hình 3.6. Kết quả sau khi đồng nhất đơn vị

Cột dữ liệu **release_date** chuyển từ định dạng chuỗi về định dạng ngày tháng bằng lệnh `as.Date()`. Sau đó chuyển thành kiểu dữ liệu số và tính toán đưa về đơn vị $[năm]$

```

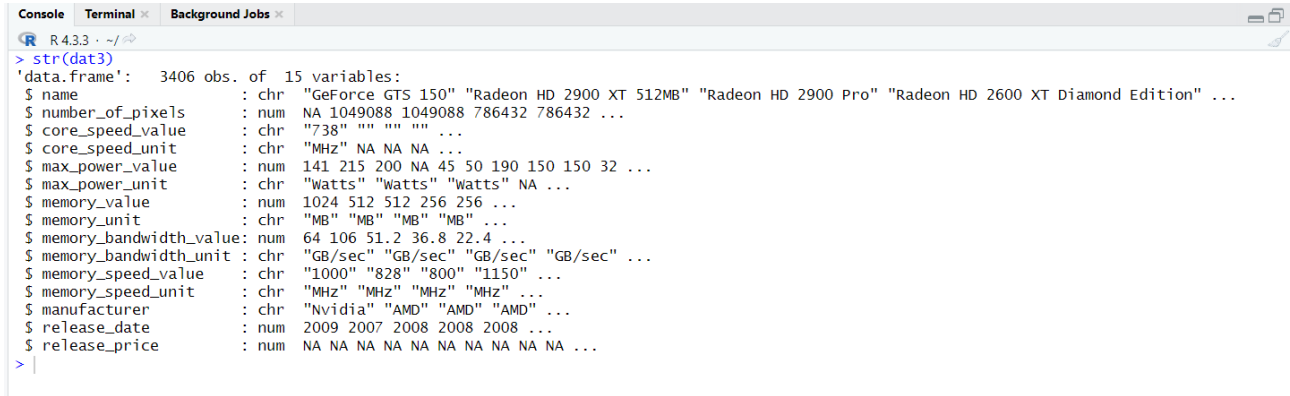
1 # Từ định dạng chuỗi -> định dạng ngày tháng
2 dat3$release_date <- as.Date(sub("\\s*\\n", "", dat3$release_date),
  format = "%d-%b-%Y")
3
4 # Tính giá trị của cột theo đơn vị "năm"
5 dat3$release_date <- as.numeric(format(dat3$release_date, "%Y")) +
6   as.numeric(format(dat3$release_date, "%m")) / 12

```

Với cột dữ liệu **release_price**, xóa bỏ ký tự **\$** và chuyển thành định dạng số để dễ dàng thực hiện tính toán

```
1 dat3$release_price <- gsub("\\$", "", dat3$release_price)
2 dat3$release_price <- as.numeric(dat3$release_price)
```

Cuối cùng, kiểm tra lại định dạng của dữ liệu bằng lệnh **str()**



```
> str(dat3)
'data.frame': 3406 obs. of 15 variables:
 $ name          : chr "GeForce GTS 150" "Radeon HD 2900 XT 512MB" "Radeon HD 2900 Pro" "Radeon HD 2600 XT Diamond Edition" ...
 $ number_of_pixels : num NA 1049088 1049088 786432 786432 ...
 $ core_speed_value : chr "738" "" "" "" "" ...
 $ core_speed_unit  : chr "MHz" NA NA NA NA ...
 $ max_power_value  : num 141 215 200 NA 45 50 190 150 150 32 ...
 $ max_power_unit   : chr "watts" "watts" "watts" NA ...
 $ memory_value     : num 1024 512 512 256 256 ...
 $ memory_unit      : chr "MB" "MB" "MB" "MB" ...
 $ memory_bandwidth_value : num 64 106 51.2 36.8 22.4 ...
 $ memory_bandwidth_unit : chr "GB/sec" "GB/sec" "GB/sec" "GB/sec" ...
 $ memory_speed_value : chr "1000" "828" "800" "1150" ...
 $ memory_speed_unit : chr "MHz" "MHz" "MHz" "MHz" ...
 $ manufacturer     : chr "Nvidia" "AMD" "AMD" "AMD" ...
 $ release_date     : num 2009 2007 2008 2008 2008 ...
 $ release_price    : num NA NA NA NA NA NA NA NA NA ...
```

Hình 3.7. Kết quả kiểm tra dữ liệu

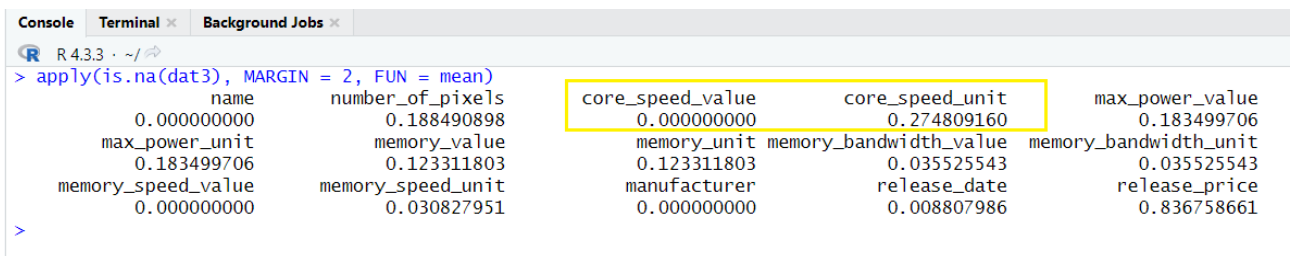
3.4 Xử lý dữ liệu khuyết

Trong R, các giá trị "N/A" thường xuất hiện khi có dữ liệu bị thiếu trong data frame. Việc này xảy ra khi dữ liệu đó không có sẵn hoặc bị thiếu. Có vài cách để xử lý dữ liệu khuyết:

- Loại bỏ dữ liệu khuyết
- Thay thế dữ liệu khuyết bằng giá trị trung bình hoặc trung vị của các phần tử không khuyết
- Thay thế bằng một giá trị cố định khác

Đầu tiên, ta cần kiểm tra tỉ lệ dữ liệu bị khuyết của các thuộc tính bằng lệnh **apply()**. Bởi vì ban đầu ở mục trên ta đã loại bỏ những giá trị khuyết "-" của **core_speed_value** nên tỉ lệ khuyết của thuộc tính này bằng với tỉ lệ khuyết của **core_speed_unit**.

```
1 apply(is.na(dat3), MARGIN = 2, FUN = mean)
```



```
> apply(is.na(dat3), MARGIN = 2, FUN = mean)
      name          number_of_pixels core_speed_value core_speed_unit max_power_value
0.000000000 0.188490898 0.000000000 0.274809160 0.183499706
max_power_unit memory_value      memory_unit memory_bandwidth_value memory_bandwidth_unit
0.183499706 0.123311803 0.123311803 0.035525543 0.035525543
memory_speed_value memory_speed_unit manufacturer      release_date      release_price
0.000000000 0.030827951 0.000000000 0.008807986 0.836758661
```

Hình 3.8. Tỉ lệ bị khuyết của dữ liệu

Sau đó, ta tiến hành xử lý dữ liệu khuyết bằng cách loại bỏ dữ liệu thông qua hàm **na.omit()**, riêng thuộc tính **release_price** không loại bỏ khuyết vì tỉ lệ khuyết quá lớn.

```
1 # Tạo Data frame "temp" từ dat3 trừ release_price
2 temp <- dat3[, !(names(dat3) %in% "release_price")]
3 # Loại bỏ dữ liệu khuyết
4 temp <- na.omit(temp)
5 # Gộp 2 data frame lại
```

```
6 dat3 <- dat3[rownames(temp), ]
```

Sau khi xử lý các giá trị khuyết, kiểm tra lại dữ liệu bằng hàm `apply()`

```
##      name      number_of_pixels      core_speed_value      core_speed_unit      max_power_value
## 0.0000000      0.0000000      0.0000000      0.0000000      0.0000000
## max_power_unit      memory_value      memory_unit      memory_bandwidth_value      memory_bandwidth_unit
## 0.0000000      0.0000000      0.0000000      0.0000000      0.0000000
## memory_speed_value      memory_speed_unit      manufacturer      release_date      release_price
## 0.0000000      0.0000000      0.0000000      0.0000000      0.7675483
> |
```

Hình 3.9. Kết quả kiểm tra khuyết

4 Thống kê mô tả

4.1 Các giá trị đặc trưng của mẫu

Ta dùng các hàm `sapply()`, `mean()`, `var()` và `fivenum()` để tính(hầu hết) các giá trị đặc trưng của mẫu định lượng từ file số liệu đã xử lý:

```
1 data <- read_csv("my_data.csv");
2 qtt_data <- data[, c("number_of_pixels", "core_speed_value", "max_power_value", "memory_value",
3   "memory_bandwidth_value", "memory_speed_value", "release_date")]
4
5 # Tìm giá trị trung bình
6 avg = sapply(qtt_data, mean)
7
8 # Tìm phương sai
9 s2 = sapply(qtt_data, var)
10
11 # Tìm 5 thông số lần lượt, bao gồm
12 # Min, Tứ phân vị 0.25, Trung vị, Tứ phân vị 0.75, Max
13 fivenum = sapply(qtt_data, fivenum)
14
15 # Tạo bảng chứa các giá trị đặc trưng
16 # và làm tròn đến 2 chữ số thập phân
17 fivenum = t(fivenum)
18 t = split(fivenum, rep(1:ncol(fivenum), each = nrow(fivenum)))
19 names(t) = c("Min", "Quar1", "Quar2", "Quar3", "max")
20 table = round(data.frame(avg, s2, t), 2)
21 show(table)
```

Description: df [7 x 7]

| | Avg <dbl> | s2 <dbl> | Min <dbl> | Quar1 <dbl> | Quar2 <dbl> | Quar3 <dbl> | Max <dbl> |
|------------------------|--------------|--------------|--------------|----------------|----------------|----------------|--------------|
| number_of_pixels | 2313406.78 | 2.580256e+12 | 480000.00 | 1440000.00 | 2073600.00 | 2073600.00 | 9216000.00 |
| core_speed_value | 980.88 | 6.413929e+04 | 200.00 | 800.00 | 1000.00 | 1106.00 | 1784.00 |
| max_power_value | 144.88 | 1.086828e+04 | 1.00 | 65.00 | 130.00 | 190.00 | 780.00 |
| memory_value | 3308.73 | 7.700892e+06 | 128.00 | 1280.00 | 2048.00 | 4096.00 | 24576.00 |
| memory_bandwidth_value | 166.13 | 1.922599e+04 | 4.00 | 72.00 | 128.30 | 224.40 | 1280.00 |
| memory_speed_value | 1307.28 | 1.466370e+05 | 400.00 | 1000.00 | 1350.00 | 1650.00 | 2127.00 |
| release_date | 2013.68 | 4.690000e+00 | 2006.92 | 2012.25 | 2013.83 | 2015.42 | 2017.33 |

7 rows

Hình 4.1. Các giá trị đặc trưng thu được

Riêng thông số `Release_Price` do có nhiều giá trị khuyết nên ta sẽ sử dụng hàm `summary()` để xác định:

```
1 summary(data$release_price)
```

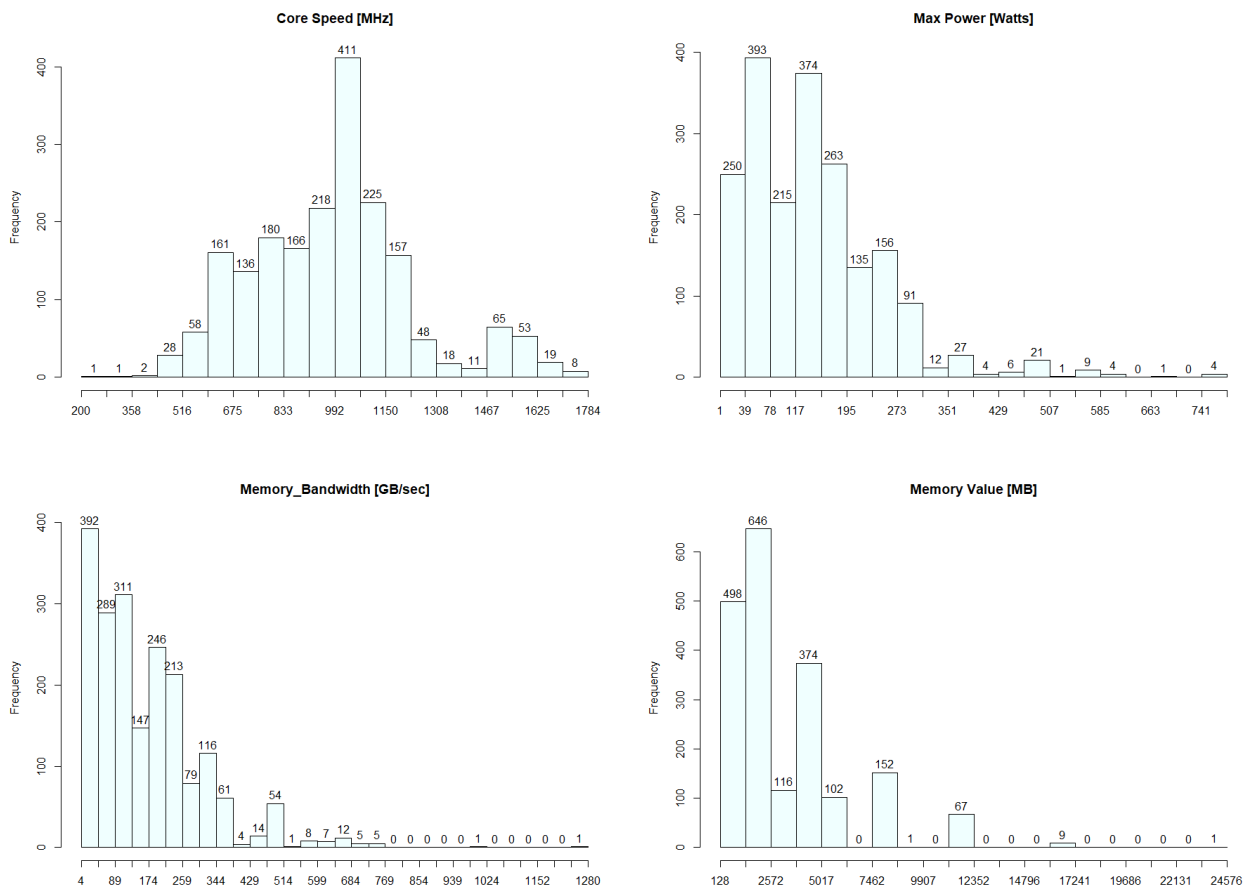
```
Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
23.0 150.0 249.0 385.8 449.0 14999.0 1509
```

Hình 4.2. Các giá trị đặc trưng của `Release_Price`

4.2 Các loại đồ thị

Dùng hàm `hist()`, ta có thể quan sát được một cách trực quan sự phân phối của một số mẫu định lượng và đưa ra nhận xét:

```
1 #Hàm tự định nghĩa để vẽ biểu đồ Histogram
2 plotHist <- function(sample, name, num_of_breaks){
3   sample_x_axis = floor(seq(floor(min(sample)),
4     max(sample), length.out = num_of_breaks+1))
5   hist(sample, sample_x_axis, freq = TRUE, xlab = name,
6     labels = TRUE, xaxt="n", col = "azure")
7   axis(1, at = sample_x_axis)
8 }
9
10 plotHist(data$core_speed_value, "Core Speed [MHz]", 20)
11 plotHist(data$max_power_value, "Max Power [Watts]", 20)
12 plotHist(data$memory_bandwidth_value, "Memory_Bandwidth [GB/sec]", 30)
13 plotHist(data$memory_value, "Memory Value [MB]", 20)
```

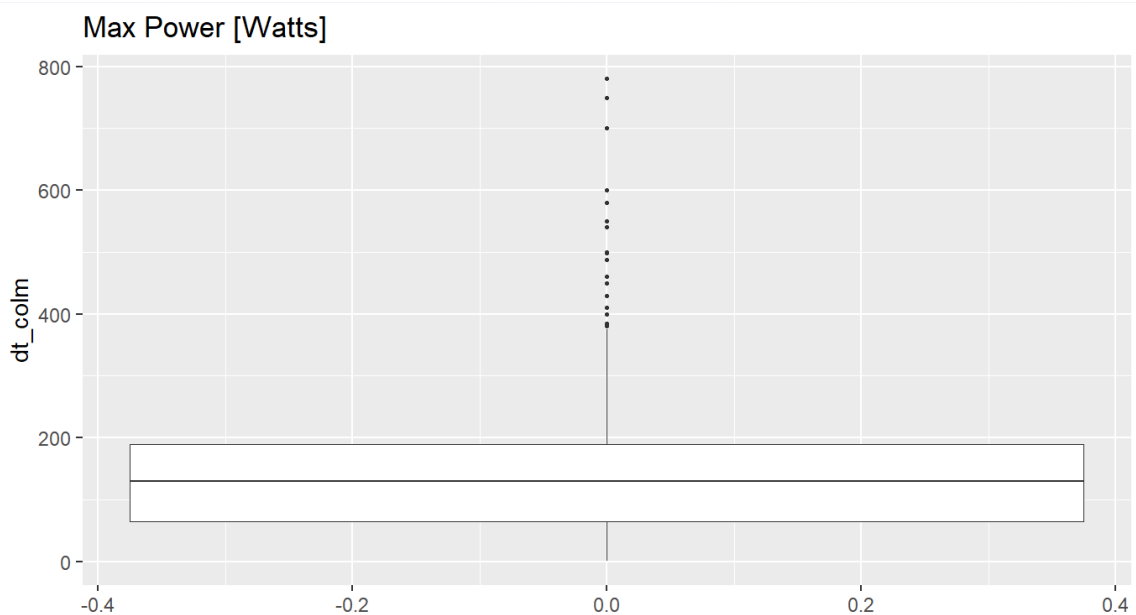
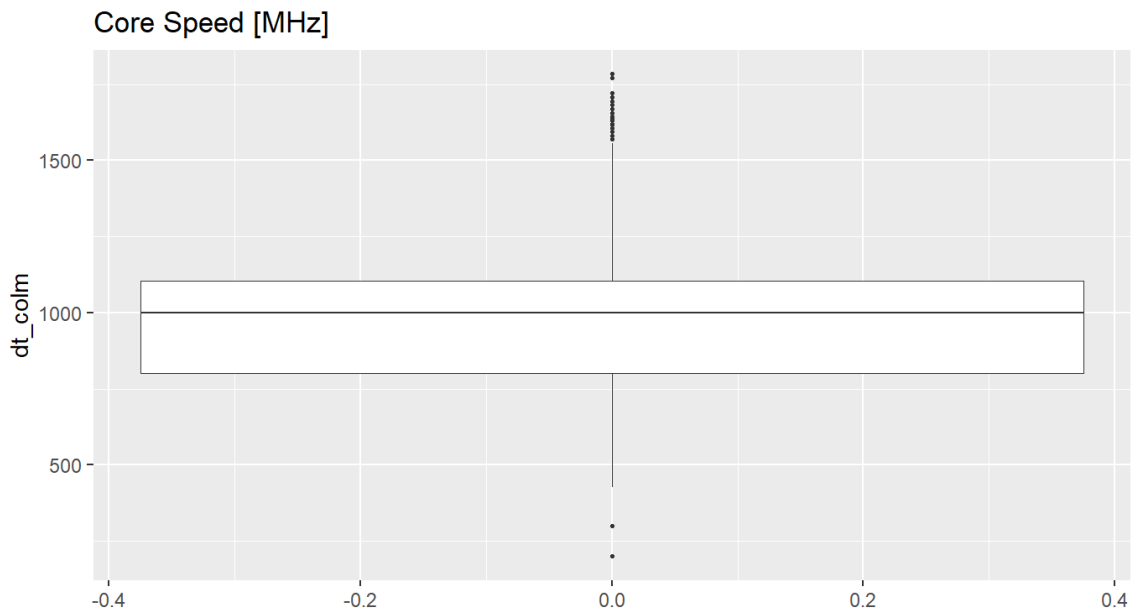


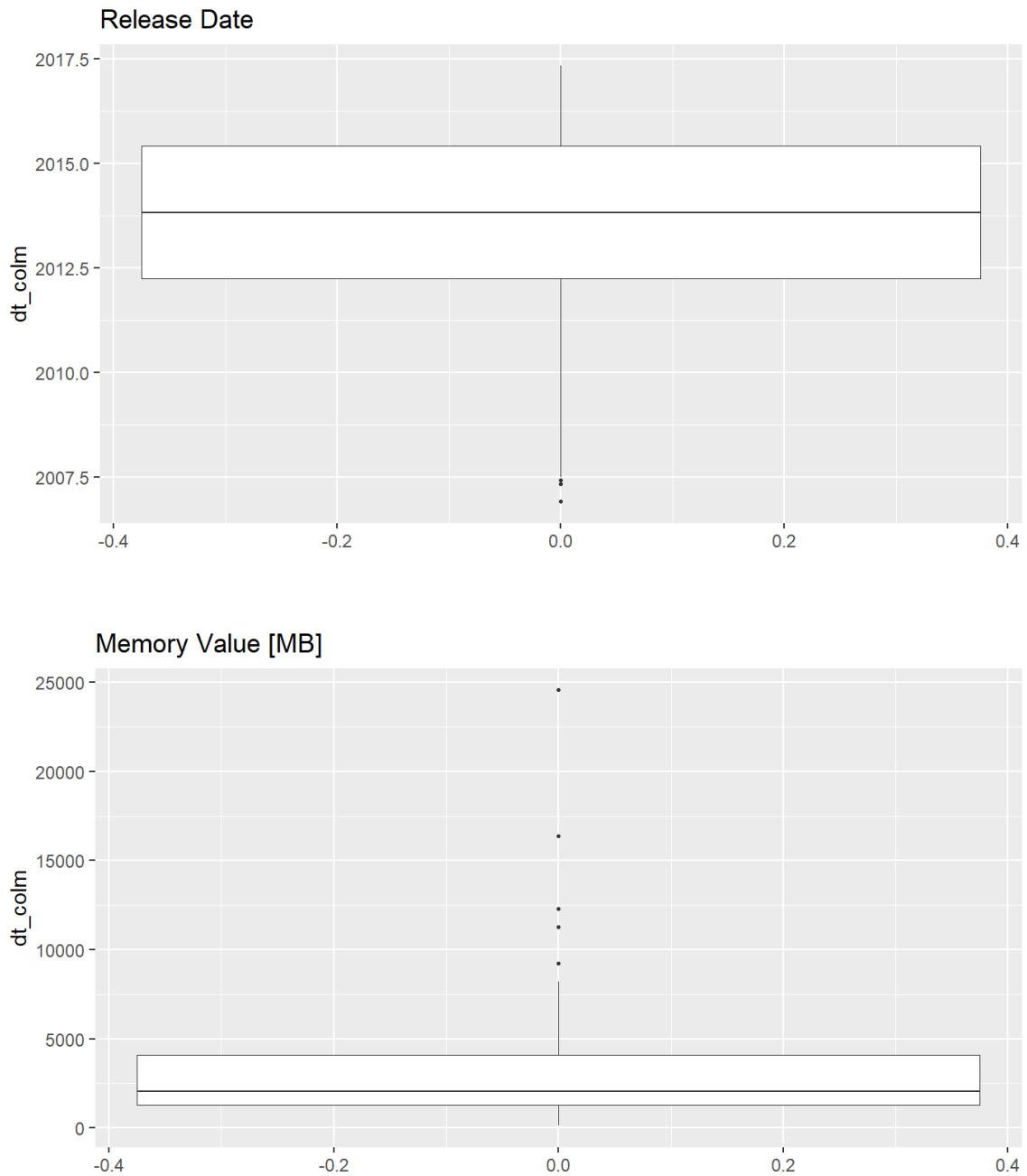
Hình 4.3. Biểu đồ phân phối tần số một số thông số của mẫu

Nhận xét: Hầu hết các thông số có phân phối bị lệch trái như `max_power_value`, `memory_power_value` và `memory_value` với phần lớn các giá trị nằm trong khoảng < 300 Watts đối với `max_power_value`, < 400 GB/sec đối với `memory_power_value` và < 4096 MB đối với `memory_value`, riêng `core_speed_value` lại có phân phối tương đối chuẩn tập trung trong khoảng từ 900 MHz đến 1000 MHz.

Tương tự ta thực hiện vẽ đồ thị boxplot bằng hàm **ggplot** trong thư viện **ggplot2**:

```
1 #Tạo hàm tự định nghĩa để vẽ biểu đồ boxplot
2 library(ggplot2)
3 my_boxplot <- function(dt, dt_colm, name){
4   ggplot(dt, aes(x = dt_colm)) +
5     labs(
6       title = name)+
7     geom_boxplot() +
8     coord_flip() +
9     theme_grey(base_size = 22)
10 }
11
12 my_boxplot(data, data$core_speed_value, "Core Speed [MHz]")
13 my_boxplot(data, data$max_power_value, "Max Power [Watts]")
14 my_boxplot(data, data$release_date, "Release Date")
15 my_boxplot(data, data$memory_value, "Memory Value [MB]")
```





Hình 4.4. Biểu đồ boxplot một số thông số của mẫu

Nhận xét:

- **core_speed_value** có đồ thị tương đối đối xứng với trung vị nằm ở đoạn giữa của khoảng tứ phân vị và khoảng tứ phân vị cũng nằm ở giữa hai râu.
- **max_power_value** và **release_date** có trung vị nằm ở giữa khoảng tứ phân vị nhưng lại không đối xứng giữa hai râu.
- Các thông số có rất nhiều điểm dị biệt nằm ở rất xa so với trung vị, đặc biệt là **core_speed_value** và **max_power_value**.

4.3 Mối liên hệ giữa các biến

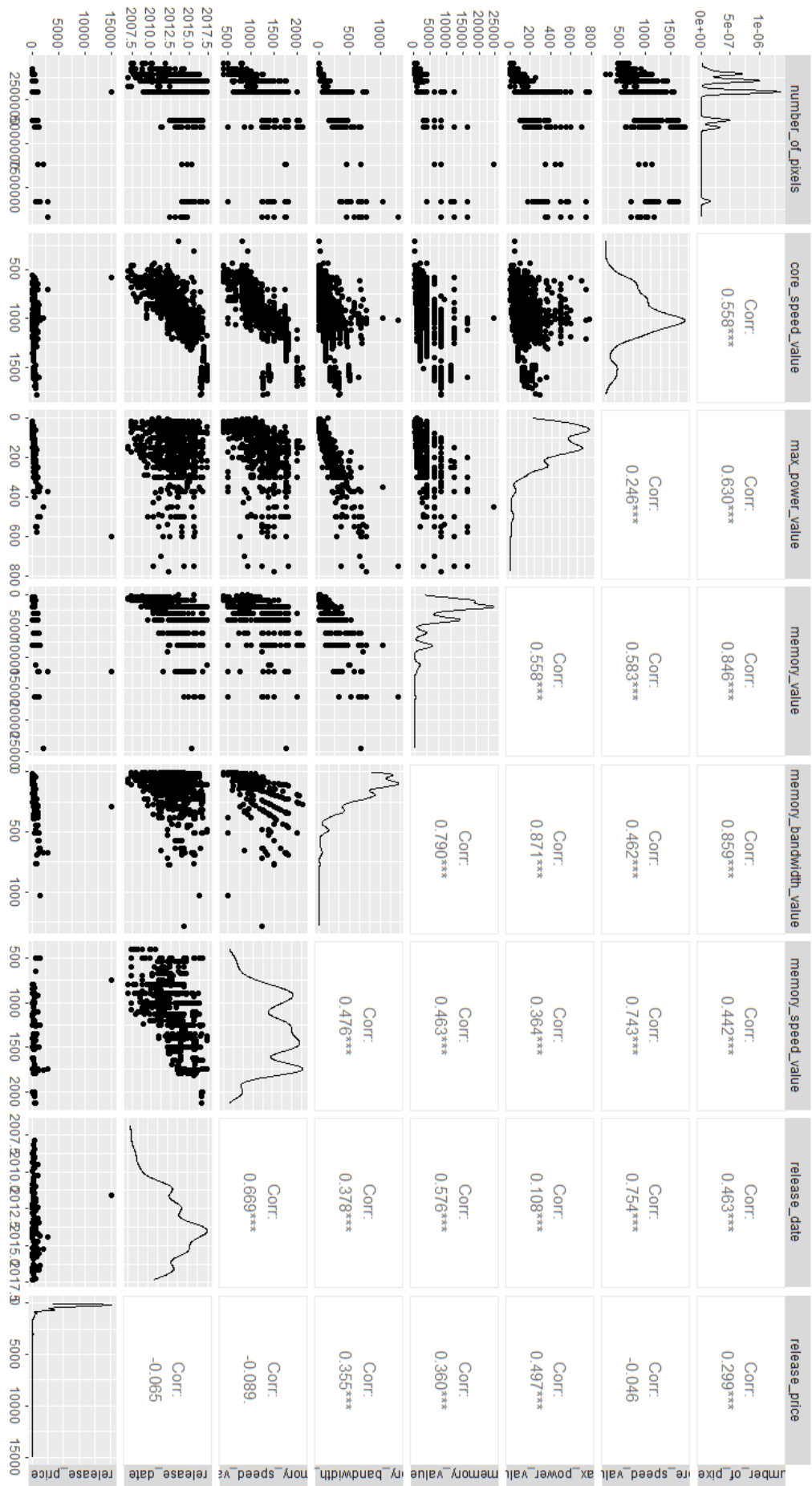
Ta sử dụng hàm **ggpairs()** trong thư viện **GGally** để tạo ma trận biểu đồ tán xạ tổng hợp thể hiện mối liên hệ giữa các thông số:

```
1 library(GGally)
2 qtt_data <- data[, c("number_of_pixels", "core_speed_value", "max_power_value", "memory_value",
3   "memory_bandwidth_value", "memory_speed_value", "release_date", "release_price")]
3 ggpairs(qtt_data)
```

Giới thiệu về **ggpairs**: Hàm **ggpairs()** mặc định sẽ trả về một ma trận gồm ba thành phần chính là bảng giá trị phía trên đường chéo chính mang hệ số tương quan của từng cặp thông số, đồ thị nằm trên đường chéo chính là biểu đồ đường của từng hàm số và nằm dưới đường chéo chính là biểu đồ tán xạ(scatter plot) của từng cặp thông số.

Nhận xét:

- Ba cặp thông số có tương quan mạnh với nhau là ($r > 0.8$):
 - **memory_bandwidth_value** và **number_of_pixels**:
Có thể khả năng xử lý nhiều điểm ảnh cao đồng nghĩa cần băng thông lớn để đáp ứng lượng dữ liệu.
 - **memory_value** và **number_of_pixels**:
Có thể là do bộ nhớ có băng thông cao sẽ yêu cầu nhiều điện năng hơn.
 - **memory_bandwidth_value** và **max_power_value**:
Có thể là do bộ nhớ có băng thông cao sẽ yêu cầu nhiều điện năng hơn.
- Năm cặp thông số có tương quan khá mạnh với nhau là ($0.8 > r > 0.6$):
 - **max_power_value** và **number_of_pixels**:
Có thể để xử lý nhiều điểm ảnh thì cần cung cấp nhiều điện năng hơn
 - **memory_value** và **memory_bandwidth_value**
 - **memory_speed_value** và **core_speed_value**
 - **core_speed_value** và **release_date**:
 - **memory_speed_value** và **release_date**:
Có thể là do GPU ra đời gần đây sẽ có tốc độ nhanh hơn
- Thông số **release_price** ít tương quan với những thông số còn lại:
Có thể là do giá của một GPU phụ thuộc vào nhiều thông số song ta chỉ so nó với từng thông số một nên dẫn đến sự tương quan của nó với từng thông số là nhỏ.



Hình 4.5. Matri trận tán xạ tổng hợp thể hiện mối liên hệ giữa các mẫu

5 Thống kê suy diễn

5.1 Tìm khoảng tin cậy một mẫu

5.1.1 Mục tiêu

- Nắm được các khái niệm liên quan đến lý thuyết ước lượng và bài toán tìm khoảng ước lượng tin cậy của một tỉ lệ trong tổng thể.
- Biết phương pháp giải quyết bài toán tìm khoảng tin cậy, cụ thể cho một mẫu.
- Nắm được căn bản cách ứng dụng ngôn ngữ R vào bài toán.

5.1.2 Bài toán

Bài toán 1

Xét mẫu **Memory_Value** từ tập dữ liệu đã được tiền xử lý. Tìm khoảng ước lượng hai phía cho tỉ lệ những GPU có giá trị **Memory_Value** từ 4096 MB trở lên trong tổng thể với độ tin cậy 95%.

5.1.3 Các lệnh R sử dụng

Các lệnh nhóm sử dụng để giải quyết bài toán được liệt kê trong bảng sau:

| Thư viện và lệnh R | Chức năng |
|--------------------------|--|
| <code>prop.test()</code> | Kiểm định cho một tỉ lệ bao gồm khoảng ước lượng |
| <code>mean()</code> | Tìm trung bình của mẫu |
| <code>sd()</code> | Tìm độ lệch chuẩn của mẫu |
| <code>qnorm()</code> | Tìm giá trị z-score ứng với độ tin cậy |
| <code>cat</code> | Định dạng lại kết quả đầu ra |

Bảng 5.1. Các lệnh R sử dụng trong bài toán

5.1.4 Tiền xử lý bài toán

Vì tỉ lệ các GPU của mẫu có giá trị **Memory_Value** trên 4096 MB cũng là biến ngẫu nhiên tuân theo phân phối nhị thức, mà kích thước mẫu lớn nên sử dụng định lý giới hạn trung tâm, biến ngẫu nhiên này có thể được xấp xỉ về phân phối chuẩn.

Đây là bài toán tìm khoảng tin cậy hai phía, sử dụng cơ sở lý thuyết, nhóm đưa ra lời giải trên R. Ngoài ra, nhóm sẽ sử dụng hàm được cung cấp sẵn bởi R là `prop.test()` để so sánh kết quả.

5.1.5 Giải bài toán trên R

Trước hết, ta tính toán các giá trị cần thiết như sai số chuẩn, giá trị `z_score` với độ tin cậy tương ứng.

```
1 # Đọc dữ liệu đã được xử lý và lấy mẫu của giá trị memory_value
2 df <- read.csv("my_data.csv")
3 mem <- df$memory_value
4
5 # Lấy ra các mẫu có giá trị memory_value >= 4096 MB
6 mem_larger <- mem[mem >= 4096]
7
8 # Tính kích thước mẫu và tỉ lệ các GPU có memory_value từ 4096 MB trở lên
9 size_total <- length(mem)
10 size_larger <- length(mem_larger)
```

```
11
12 f <- size_larger / size_total
13
14 # Tính giá trị z_score ứng với alpha/2
15 confidence <- 0.95
16 alpha <- 1 - confidence
17
18 z_score <- qnorm(1 - alpha / 2)
```

Trình bày kết quả tính toán được

```
1 # Tính khoảng tin cậy và trình bày kết quả
2 SE <- sqrt(f * (1 - f) / size_total)
3 epsilon <- z_score * SE
4
5 lower_bound <- f - epsilon
6 upper_bound <- f + epsilon
7 cat("The 95% confidence interval of p is (",
8     lower_bound, ", ", upper_bound, ")", sep = "")
```

Kết quả mà nhóm thu được với khoảng tin cậy cho tỉ lệ tổng thể các GPU có giá trị **Memory_Value** từ 4096 GB trở lên là:

```
1 The 95% confidence interval of p is (0.3378987, 0.3803109)
```

Sử dụng thêm hàm `prop.test()` được cung cấp sẵn bởi R, nhóm so sánh với khoảng tin cậy tỉ lệ vừa tìm được.

```
1 prop.test(
2   x = size_larger,
3   n = size_total,
4   alternative = "two.sided",
5   conf.level = confidence
6 )
```

Kết quả thu được:

```
1 1-sample proportions test with continuity correction
2
3 data:  size_larger out of size_total, null probability 0.5
4 X-squared = 155.55, df = 1, p-value < 2.2e-16
5 alternative hypothesis: true p is not equal to 0.5
6 95 percent confidence interval:
7  0.3379418 0.3808238
8 sample estimates:
9           p
10 0.3591048
```

5.1.6 Nhận xét

Dữ liệu từ dòng 95 percent confidence interval cho thấy khoảng tin cậy (0.3379418;0.3808238) tương đương với kết quả đã tìm được bằng cơ sở lý thuyết, với sai lệch rất nhỏ.

Kết quả sự sai khác nhỏ này là do ngay từ ban đầu, ta đang sử dụng định lý giới hạn trung tâm để xấp xỉ cho phân phối nhị thức về phân phối chuẩn.

5.1.7 Kết luận

Đối với bài toán này, nhóm đã được thực hành các lệnh với ngôn ngữ thống kê R, tìm ra được khoảng ước lượng cho trung bình tổng thể, áp dụng từ cơ sở lý thuyết về khoảng tin cậy đã được học.

5.2 Kiểm định hai mẫu

5.2.1 Mục tiêu

- Nắm và hiểu được khái niệm liên quan đến lý thuyết kiểm định giả thuyết thống kê.
- Biết phương pháp giải quyết bài toán kiểm định trung bình, tỉ lệ một mẫu, hai mẫu.
- Nắm được căn bản cách ứng dụng ngôn ngữ R vào bài toán

5.2.2 Bài toán

Bài toán 2

Xét mẫu dựa trên giá trị **Memory_Bandwidth_Value**, phân loại ra hai mẫu con dựa theo **Manufacturer** (nhà sản xuất GPU) theo thứ tự là **Nvidia** và **AMD**. Tính toán trên mẫu cho thấy giá trị trung bình về giá trị Memory Bandwidth của mẫu thứ nhất lớn hơn của mẫu thứ hai. Với mức ý nghĩa $\alpha = 5\%$, có thể coi giá trị trung bình về Memory Bandwidth của tổng thể các GPU do hãng sản xuất Nvidia lớn hơn hãng AMD không.

5.2.3 Các lệnh R sử dụng

Các lệnh nhóm sử dụng để giải quyết bài toán được liệt kê trong bảng sau:

| Thư viện và lệnh R | Chức năng |
|--|--|
| BSDA | Thư viện cung cấp lệnh <code>z.test()</code> |
| <code>z.test()</code> | Tìm khoảng tin cậy của trung bình tổng thể |
| <code>mean()</code> | Tìm trung bình của mẫu |
| <code>sd()</code> | Tìm độ lệch chuẩn của mẫu |
| <code>qnorm()</code> | Tìm giá trị z-score ứng với độ tin cậy |
| <code>length()</code> | Tìm kích thước vector mẫu |
| <code>print()</code> và <code>paste()</code> | Hiển thị kết quả ra console |

Bảng 5.2. Các lệnh R sử dụng trong bài toán

5.2.4 Tiền xử lý bài toán

Kiểm tra phân phối của mẫu

Trước hết, nhóm kiểm tra các giá trị **Memory_Bandwidth_Value** của GPU sản xuất từ hãng Nvidia và AMD có tuân theo phân phối chuẩn hay không.

Có nhiều phương pháp để kiểm tra phân phối chuẩn, nhóm sẽ sử dụng hai phương pháp:

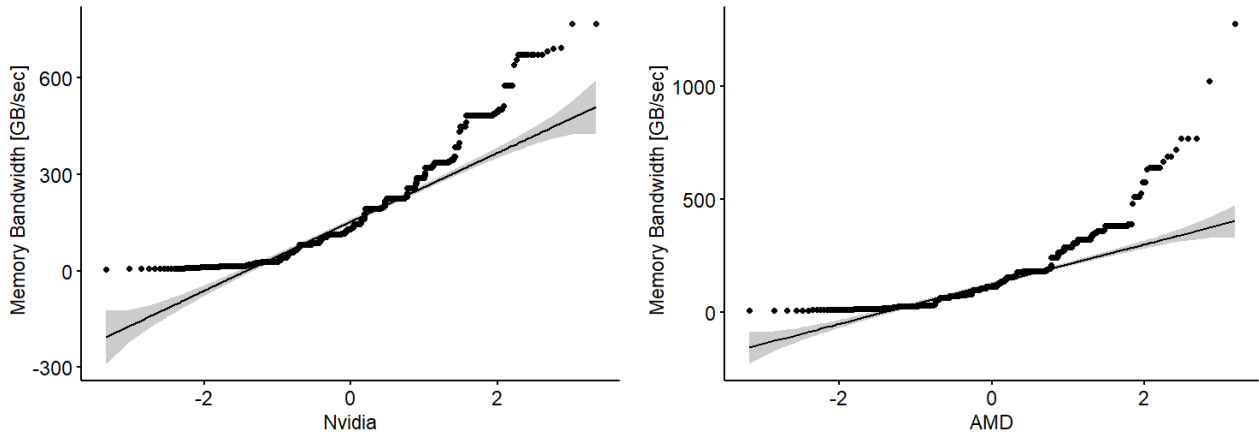
- Phương pháp đồ thị: Sử dụng Biểu đồ Q-Q (Quantile-Quantile)
- Phương pháp thống kê: Sử dụng Kiểm định Shapiro-Wilk

Sử dụng Biểu đồ Q-Q

```

1 # Trích dữ liệu memory_bandwidth_value theo hãng Nvidia và AMD
2 df <- read.csv("my_data.csv")
3 nvidia <- df$memory_bandwidth_value[df$manufacturer == "Nvidia"]
4 amd <- df$memory_bandwidth_value[df$manufacturer == "AMD"]
5
6 library(ggpubr)
7 ggqqplot(nvidia, ylab = "Memory Bandwidth [GB/sec]")
8 ggqqplot(amd, ylab = "Memory Bandwidth [GB/sec]")

```



Hình 5.1. Biểu đồ Q-Q-plot cho mẫu của Nvidia và AMD

Ta thấy các giá trị tứ phân vị nằm khá xa lệch đường thẳng kì vọng của phân phối chuẩn nên những giá trị **Memory_Bandwidth** của GPU từ hai hãng Nvidia và AMD có khả năng cao không theo phân phối chuẩn.

Để kết luận chắc chắn hơn, nhóm sử dụng thêm kiểm định Shapiro-Wilk.

Kiểm định Shapiro-Wilk

Giả thuyết kiểm định:

- Giả thuyết H_0 : Biến kiểm định tuân theo phân phối chuẩn
- Giả thuyết H_1 : Biến kiểm định không tuân theo phân phối chuẩn

```
1 shapiro.test(nvidia)
2 shapiro.test(amd)

Shapiro-Wilk normality test
data:  nvidia
W = 0.88913, p-value < 2.2e-16

Shapiro-Wilk normality test
data:  amd
W = 0.80454, p-value < 2.2e-16
```

Kết quả cho thấy giá trị $p_value = 2.2e - 16 \ll 0.05$. Do đó có thể bác bỏ giả thuyết H_0 và chấp nhận giả thuyết H_1 rằng các biến ngẫu nhiên đang khảo sát đến từ hãng Nvidia và AMD không tuân theo phân phối chuẩn.

Chọn phương án giải quyết

Bài toán kiểm định trung bình hai mẫu trên có những đặc điểm sau:

- Hai mẫu độc lập, do mỗi GPU chỉ đến từ một nhãn hàng)
- Giá trị **Memory_Bandwidth_Value** có phân phối tùy ý khác phân phối chuẩn.
- Kích thước hai mẫu lớn.
- Chưa biết phương sai của hai tổng thể.

Do đó, áp dụng kết quả của định lý giới hạn trung bình, chênh lệch giữa trung bình hai tổng thể xấp xỉ phân phối chuẩn, nhóm sẽ sử dụng tiêu chí kiểm định `z.test` cho bài toán. Ngoài ra vì không biết phương sai tổng thể, nên sẽ sử dụng phương sai của mẫu để thay thế.

Nhóm sẽ giải quyết bài toán theo hai hướng, theo cơ sở lý thuyết đã được học và sử dụng hàm `z.test()` được hỗ trợ bởi R để kiểm tra, so sánh kết quả.

5.2.5 Giải quyết bài toán trên R

Vì xu hướng kiểm định của ta là kiểm tra xem giá trị trung bình **Memory_Bandwidth_Value** trên tổng thể của mẫu thứ nhất (hãng Nvidia) có lớn hơn tổng thể của mẫu đại diện thứ hai (hãng AMD) hay không, do đó, giả thuyết nhóm đưa ra là:

- Giả thuyết $H_0: \mu_1 \leq \mu_2$
- Giả thuyết $H_1: \mu_1 > \mu_2$

Xác định giá trị `z_score` ứng với $\alpha = 0.05$:

```
1 # Giá trị z_score ứng với alpha = 0.05
2 alpha <- 0.05
3 z_score <- qnorm(1 - alpha)
```

Do đó, miền bác bỏ là: $RR = (1.6449; \infty)$.

Kế tiếp, tính toán các giá trị trung bình, phương sai của hai mẫu.

```
1 # Tính kích thước, giá trị trung bình và phương sai của hai mẫu
2 size_nvidia <- length(nvidia)
3 mean_nvidia <- mean(nvidia)
4 sd_nvidia <- sd(nvidia)
5
6 size_amd <- length(amd)
7 mean_amd <- mean(amd)
8 sd_amd <- sd(amd)
```

Cuối cùng từ các số liệu đã có, nhóm tính giá trị quan sát để rút ra kết luận.

```
1 # Tính giá trị quan sát Z_qs
2 SE <- sqrt(sd_nvidia^2 / size_nvidia + sd_amd^2 / size_amd)
3 Z_stat <- (mean_nvidia - mean_amd) / SE
4 print(paste("Z_stat = ", Z_qs))
```

```
1 Z_stat = 2.56752041896274
```

Từ kết quả thu được, ta thấy giá trị quan sát $Z_stat \in (1.6449; \infty)$.

Do đó, có thể bác bỏ giả thuyết H_0 , chấp nhận giả thuyết H_1 , rằng trên tổng thể, giá trị trung bình về thông số GPU Memory_Bandwidth_Value do hãng Nvidia sản xuất lớn hơn do hãng AMD sản xuất.

Sử dụng thêm công cụ `z.test()` được cung cấp bởi R, ta kiểm tra kết quả đã tính toán.

```
1 library(BSDA)
2 z.test(
3   x = nvidia,
4   y = amd,
5   mu = 0,
6   sigma.x = sd_nvidia,
```

```
7  sigma.y = sd_amd,
8  alternative = "greater",
9  conf.level = 1 - alpha
10 )
```

```
1  Two-sample z-Test
2
3  data:  nvidia and amd
4  z = 2.5675, p-value = 0.005121
5  alternative hypothesis: true difference in means is greater than 0
6  95 percent confidence interval:
7    6.252307      NA
8  sample estimates:
9  mean of x mean of y
10 170.8046 153.4062
```

5.2.6 Nhận xét

Quan sát kết quả từ lệnh, ta thấy giá trị $p_value = 0.005121 < \alpha = 0.05$ hay tương đương với $z = 2.5675 > z_stat = 1.6449$. Có thể bác bỏ giả thuyết H_0 , chấp nhận giả thuyết H_1 .

Kết quả hoàn toàn trùng khớp với các giá trị đã tính toán.

5.2.7 Kết luận

Nhóm có cơ hội thực hành, tìm hiểu các lệnh của R để thực hiện việc tính toán, giải quyết bài toán kiểm định trung bình tổng thể hai mẫu độc lập, qua đó có cái nhìn sâu sắc hơn về cơ sở lý thuyết đi cùng với thực hành.

5.3 Phân tích phương sai một yếu tố

5.3.1 Mục tiêu

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

5.3.2 Bài toán

Bài toán 3

Dùng mô hình ANOVA kiểm định trung bình của yếu tố **Memory_Speed** giữa các nhóm có **manufacturer** (hãng) khác nhau với mức ý nghĩa $\alpha = 5\%$.

5.3.3 Điều kiện để thực hiện bài toán Anova

Để thực hiện được mô hình Anova thì phải thỏa mãn 3 yếu tố sau:

- Trong các mẫu có phân phối chuẩn. Có thể sử dụng đồ thị để kiểm định phân phối chuẩn. Ngoài ra còn có thể thực hiện kiểm định bằng hàm **shapiro.test()** để kiểm định giả thuyết.
- Phương sai tổng thể giữa các của các mẫu phải bằng nhau. Có thể sử dụng hàm **levenTest()** để kiểm tra các phương sai của các mẫu có bằng nhau hay không.
- Các mẫu được lấy phải độc lập với nhau.

5.3.3.1 Kiểm tra - xử lý mẫu phân loại:

Dùng hàm **table()** để thống kê số lượng quan sát tương ứng với từng tên trong mẫu **manufacturer**

```
1 # Lưu kết quả của hàm table () vào biến và in kết quả đó ra
2 print(manufacturer_table <- table(my_data$manufacturer))
```

| | AMD | ATI | Intel | Nvidia |
|---|-----|-----|-------|--------|
| 1 | 690 | 65 | 2 | 1209 |

Ta thấy, có một vài loại GPU có số lượng quan sát khá thấp. Do đó, để mô hình ANOVA chính xác hơn, ta sẽ lọc bỏ các quan sát có số lượng không lớn hơn 10.

```
1 # Trích xuất dữ liệu với số lượng mỗi quan sát theo tên nhiều hơn 10
2 my_data_filtered <- my_data[my_data$manufacturer %in% names( manufacturer_table [
  manufacturer_table > 10]), ]
3 # Kiểm tra lại thống kê các hãng
4 table(my_data_filtered $manufacturer )
```

| | AMD | ATI | Nvidia |
|---|-----|-----|--------|
| 1 | 690 | 65 | 1209 |

Sau khi thực thi đoạn chương trình trên, ta còn lại ba hãng sản xuất. Đây sẽ là ba hãng sản xuất cho đầu vào của mô hình ANOVA.

5.3.3.2 Kiểm tra điều kiện 1: Các mẫu phải có phân phối chuẩn.

Để kiểm tra các mẫu có phân phối chuẩn hay không, ta sử dụng phương pháp là sử dụng đồ thị nhưng phương pháp sử dụng đồ thị chỉ mang tính cảm quan nên để chắc chắn nhóm đã đề xuất sử dụng thêm hàm **shapiro.test()** để kiểm định giả thuyết.

- Với phương pháp sử dụng đồ thị, để nhận biết mẫu có phân phối chuẩn thì các điểm quan trắc phải nằm xung quanh đường thẳng thì có thể kết luận mẫu có phân phối chuẩn.
- Với phương pháp sử dụng hàm `shapiro.test()`, để nhận biết mẫu có phân phối chuẩn thì ta có thể đặt các giả thiết để kiểm tra phân phối chuẩn với mức ý nghĩa $\alpha = 5\%$:

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

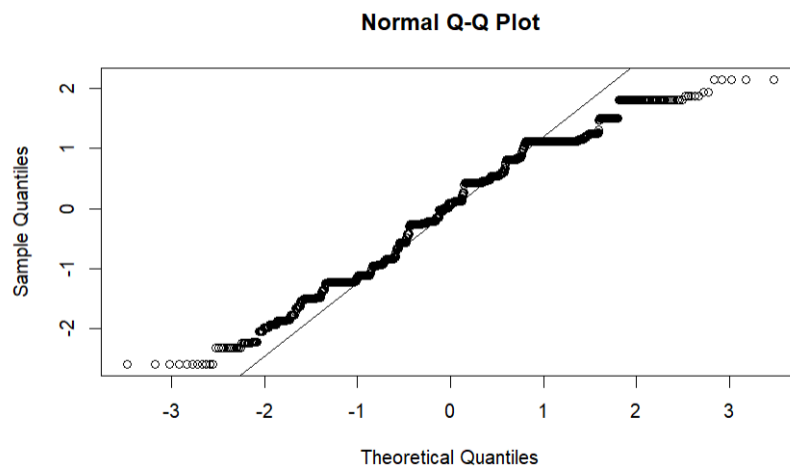
Sau đó cần kiểm tra giá trị **p-value** mà hàm trả về

- **p-value** $\leq \alpha$: ta bác bỏ giả thuyết H_0 , tức thừa nhận mẫu không tuân theo phân phối chuẩn.
- **p-value** $> \alpha$: ta chưa thể bác bỏ giả thuyết H_0 , tức chưa thể kết luận X không tuân theo phân phối chuẩn. Lúc này t có thể coi như X tuân theo phân phối chuẩn

1. Kiểm tra mẫu AMD

Sử dụng đồ thị:

```
1 #Vẽ biểu đồ phân phối chuẩn cho AMD
2 AMD_data <- subset(my_data_filtered, my_data_filtered$manufacturer=="AMD")
3 qqnorm(AMD_data$memory_speed_value)
4 qqline(AMD_data$memory_speed_value)
```



Hình 5.2. Sử dụng đồ thị để kiểm tra phân phối đều của hãng AMD

Sử dụng hàm `shapiro.test()`:

```
1 #Kiểm tra phân phối chuẩn cho AMD bằng hàm shapiro.test
2 shapiro.test(AMD_data$memory_speed_value)

1 Shapiro-Wilk normality test
2 data: AMD_data$memory_speed_value
3 W = 0.97096, p-value = 1.838e-10
```

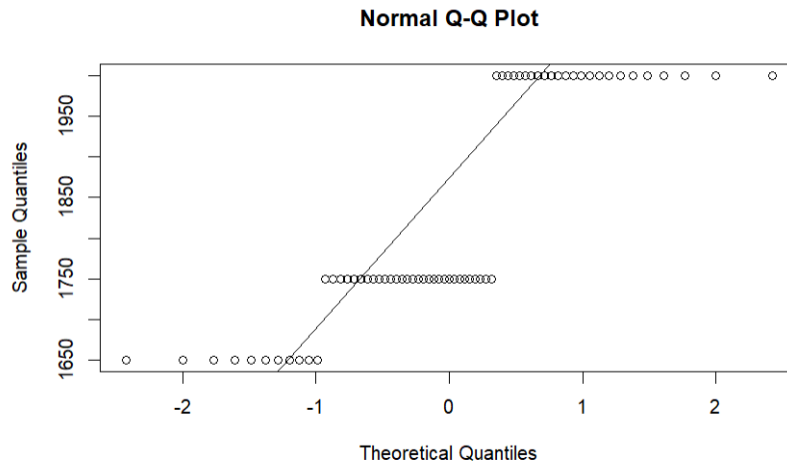
Nhận xét: Nhìn vào đồ thị kết hợp với giá trị **p-value** của hàm `shapiro.test()` trả về với **p-value** $< 5\%$ nên có thể kết luận mẫu không có phân phối chuẩn

2. Kiểm tra mẫu ATI

Sử dụng đồ thị:

```
1 #Vẽ biểu đồ phân phối chuẩn cho ATI
```

```
2 ATI_data <- subset(my_data_filtered, my_data_filtered$manufacturer=="ATI")
3 qqnorm(ATI_data$memory_speed_value)
4 qqline(ATI_data$memory_speed_value)
```



Hình 5.3. Sử dụng đồ thị để kiểm tra phân phối đều của hãng ATI

Sử dụng hàm `shapiro.test()`:

```
1 #Kiểm tra phân phối chuẩn cho ATI bằng hàm shapiro.test
2 shapiro.test(ATI_data$memory_speed_value)
```

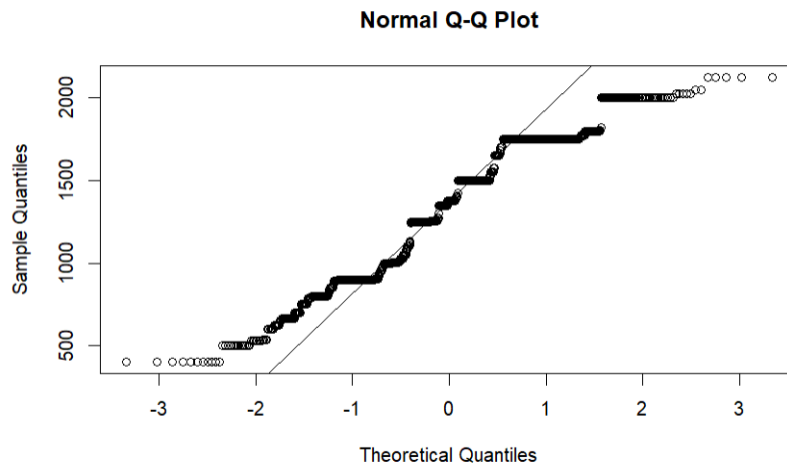
```
1 Shapiro-Wilk normality test
2 data: ATI_data$memory_speed_value
3 W = 0.74895, p-value = 3.343e-09
```

Nhận xét: Nhìn vào đồ thị kết hợp với giá trị **p-value** của hàm `shapiro.test()` trả về với **p-value** < 5% nên có thể kết luận mẫu không có phân phối chuẩn

3. Kiểm tra mẫu Nvidia

Sử dụng đồ thị:

```
1 #Vẽ biểu đồ phân phối chuẩn cho Nvidia
2 Nvidia_data <- subset(my_data_filtered, my_data_filtered$manufacturer=="Nvidia")
3 qqnorm(Nvidia_data$memory_speed_value)
4 qqline(Nvidia_data$memory_speed_value)
```



Hình 5.4. Sử dụng đồ thị để kiểm tra phân phối đều của hãng Nvidia

Sử dụng hàm `shapiro.test()`:

```
1 #Kiểm tra phân phối chuẩn cho Nvidia bằng hàm shapiro.test
2 shapiro.test(Nvidia_data$memory_speed_value)

1 Shapiro-Wilk normality test
2 data:  Nvidia_data$memory_speed_value
3 W = 0.95021, p-value < 2.2e-16
```

Nhận xét: Nhìn vào đồ thị kết hợp với giá trị **p-value** của hàm `shapiro.test()` trả về với **p-value** < 5% nên có thể kết luận mẫu không có phân phối chuẩn. Trên lý thuyết, mẫu này không thỏa giả thiết có thể áp dụng mô hình ANOVA, tuy nhiên với giới hạn về mặt dữ liệu, ta giả sử mẫu thỏa giả thiết và tiếp tục thực hiện mô hình.

5.3.3.3 Kiểm tra điều kiện 2: Các mẫu có phương sai đồng thời bằng nhau

Ngoài điều kiện các mẫu có phân phối chuẩn thì chúng cũng cần phải đáp ứng được yêu cầu các phương sai của các mẫu bằng nhau. Do đó, ta cần kiểm định thêm về sự tương đồng của phương sai các nhóm thông qua hàm `leveneTest()`. Với giả thuyết:

$$\begin{cases} H_0 : \text{phương sai của các mẫu đồng thời bằng nhau} \\ H_1 : \text{tồn tại một phương sai khác với các phương sai còn lại} \end{cases}$$

Sau đó cần kiểm tra giá trị **p-value** mà hàm trả về

- **p-value** $\leq \alpha$: ta bác bỏ giả thuyết H_0 , tức thừa nhận có một phương sai khác với các phương sai còn lại.
- **p-value** $> \alpha$: ta chưa thể bác bỏ giả thuyết H_0 , tức chưa thể kết luận tồn tại một phương sai khác với các phương sai còn lại. Lúc này t có thể coi như các phương sai bằng nhau.

```
1 # sử dụng hàm leveneTest() để thực hiện kiểm tra điều kiện phương sai bằng nhau giữa các mẫu
2 library("car") # input thư viện car để sử dụng hàm leveneTest()
3 leveneTest(memory_speed_value~as.factor(manufacturer),my_data_filtered)

1 Levene's Test for Homogeneity of Variance (center = median)
2      Df F value    Pr(>F)
3 group  2  81.888 < 2.2e-16 ***
4      1961
5 ----
```



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

Nhận xét: Nhận xét: p-value của kiểm định rất nhỏ so với mức ý nghĩa $\alpha = 5\%$. Điều này có nghĩa ta bác bỏ giả thuyết về sự bằng nhau giữa phương sai các nhóm. Nói cách khác, có sự khác biệt về phương sai giữa các hãng sản xuất. Trên lý thuyết, mẫu này không thoả giả thiết có thể áp dụng mô hình ANOVA, tuy nhiên với giới hạn về mặt dữ liệu, ta giả sử mẫu thoả giả thiết và tiếp tục thực hiện mô hình.

5.3.4 Phân tích phương sai

5.3.4.1 Tiến hành phân tích phương sai

Dùng hàm `aov()` kết hợp `summary()` với những tham số phù hợp để tiến hành áp dụng mô hình phân tích phương sai một yếu tố cho mẫu `memory_speed_value`, phân loại theo `manufacturer` đã lọc. Đặt giả thuyết

$$\begin{cases} H_0 : \text{Giá trị trung bình memory_speed_value ở các hãng bằng nhau} \\ H_1 : \text{Có ít nhất hai hãng có giá trị trung bình memory_speed_value khác nhau} \end{cases}$$

```
1 # Lưu kết quả của mô hình ANOVA vào biến (để sử dụng lại cho sau này)
2 anova_result <- aov (memory_speed_value ~ manufacturer, data = my_data_filtered)
3 # Hiển thị kết quả chi tiết của mô hình ANOVA
4 summary(anova_result)
```

```
1              Df    Sum Sq Mean Sq F value Pr(>F)
2 manufacturer    2  25846920 12923460   96.77 <2e-16 ***
3 Residuals    1961 261896870  133553
4 ---
5 Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Từ kết quả thu được ở trên, ta thống kê các giá trị đặc trưng của mô hình ANOVA:

- Bậc tự do $DfB = k - 1 = 2$
- Bậc tự do $DfW = N - k = 1961$
- $SSB = 25846920$
- $SSW = 261896870$
- $MSB = 12923460$
- $MSW = 133553$
- Kiểm định thống kê $F = 96.77$
- p-value $< 2.10^{-16}$

```
1 #Tính giá trị F(0.05,2,1961)
2 qf(p=0.05,df1 = 2, df2 = 1961, lower.tail = FALSE)
```

```
1 #Kết quả thu được
2 [1] 3.000321
```

Nhận xét:

Giá trị thống kê kiểm định F biểu thị tỷ lệ bình phương trung bình giữa các nhóm với bình phương trung bình trong mỗi nhóm. Giá trị F thuộc miền bác bỏ $RR = (3.000321; +\infty)$, cho thấy trung bình của nhóm không phải tất cả đều bằng nhau.

Hơn nữa, **p-value** thấp hơn rất nhiều so với mức ý nghĩa $\alpha = 5\%$, cho thấy rõ ràng sự khác biệt về trung bình giữa các nhóm được quan sát là có ý nghĩa thống kê.

Tóm lại, kết quả mô hình ANOVA cho bài toán này dẫn đến việc bác bỏ giả thuyết không H_0 – giả thuyết cho rằng giá trị **memory_Speed_value** trung bình của các nhóm đều bằng nhau. Nói cách khác, ta có thể kết luận giá trị trung bình **memory_speed_value** của các hãng khác nhau không cùng bằng nhau.

5.3.4.2 Phân tích sâu

Ta chỉ thực hiện phân tích sâu khi có sự khác biệt về trung bình của bài toán ANOVA. Do giả thuyết không H_0 đã được bác bỏ, ta có nhu cầu kiểm định giả thuyết về trung bình **memory_speed_value** giữa từng nhóm.

Sử dụng hàm **TukeyHSD()** để thực hiện phân tích sâu giữa các nhóm. Với n nhóm thì ta sẽ có C_n^2 cặp so sánh bội. Đặt giả thuyết:

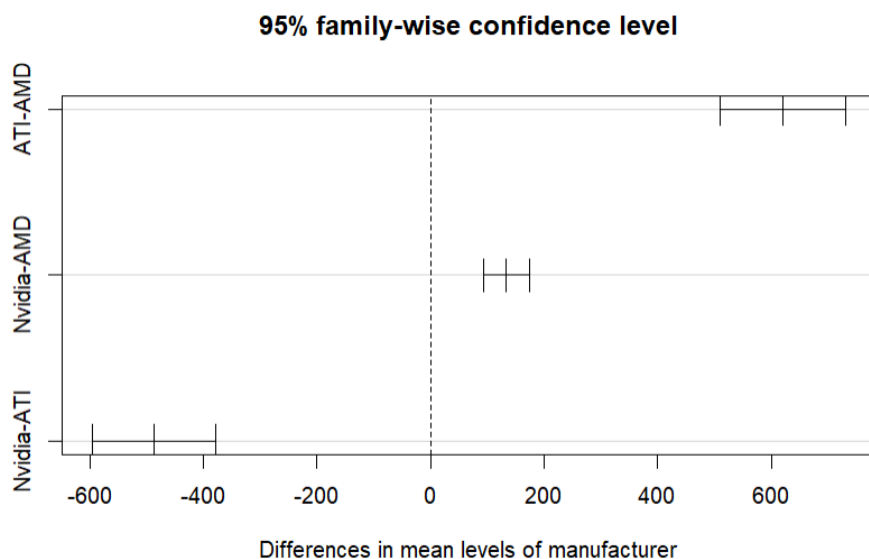
$$\begin{cases} H_0 : \text{trung bình cặp so sánh bội bằng nhau} \\ H_1 : \text{trung bình cặp so sánh bội nhóm khác nhau} \end{cases}$$

```
1 # Dùng hàm TukeyHSD() để thực hiện so sánh bội giữa các nhóm
2 TukeyHSD(anova_result)

#> Tukey multiple comparisons of means
#> 95% family-wise confidence level
#>
#> Fit: aov(formula = memory_speed_value ~ manufacturer, data = my_data_filtered)
#> $manufacturer
#>      diff      lwr      upr p adj
#> ATI-AMD  620.3890  509.17706  731.6009    0
#> Nvidia-AMD 133.5304   92.63412  174.4267    0
#> Nvidia-ATI -486.8586 -595.99604 -377.7211    0
```

Đồ thị mô tả phân tích sâu:

```
1 # vẽ biểu đồ cho việc so sánh bội giữa các nhóm
2 plot(TukeyHSD(anova_result))
```



Hình 5.5. Đồ thị mô tả phân tích sâu

Nhận xét: Vì ta có 3 nhóm nên sẽ có C_3^2 cặp cần so sánh:

- Xét cặp so sánh bội ATI và AMD thì ta thấy **p-value** $\ll 5\%$ nên ta có thể bác bỏ H_0 từ đó có thể kết luận $\mu_1 \neq \mu_2$ và vì $diff > 0(\bar{x}_1 > \bar{x}_2)$ nên $\mu_1 > \mu_2$
- Xét cặp so sánh bội Nvidia và AMD thì ta thấy **p-value** $\ll 5\%$ nên ta có thể bác bỏ H_0 từ đó có thể kết luận $\mu_1 \neq \mu_2$ và vì $diff > 0(\bar{x}_1 > \bar{x}_2)$ nên $\mu_1 > \mu_2$
- Xét cặp so sánh bội Nvidia và ATI thì ta thấy **p-value** $\ll 5\%$ nên ta có thể bác bỏ H_0 từ đó có thể kết luận $\mu_1 \neq \mu_2$ và vì $diff < 0(\bar{x}_1 < \bar{x}_2)$ nên $\mu_1 < \mu_2$

\Rightarrow Có thể rút ra kết luận tổng về sự khác nhau giữa ba hãng sản xuất về tốc độ của GPUs với tốc độ trung bình được xếp theo thứ tự $AMD < Nvidia < ATI$

5.3.5 Kết luận

Trong bài toán này, nhóm đã thành công trong việc nghiên cứu các lý thuyết về phân tích phương sai ANOVA, tìm hiểu các lệnh của **R** để thực hiện việc tính toán, áp dụng mô hình ANOVA cho một yếu tố cụ thể, đồng thời so sánh bội được giả thuyết về trung bình giữa các nhóm yếu tố, đưa ra được kết luận cuối cùng trong bài toán.

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

- Ảnh hưởng của việc loại bỏ các quan sát khuyết trong quá trình tiền xử lý số liệu.
- Tổng thể chưa thỏa giả thiết của mô hình: phân phối chưa thật sự chuẩn, phương sai của các nhóm không bằng nhau.
- Sự chênh lệch lớn về số lượng quan sát giữa các nhóm.

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

5.4 Hồi quy tuyến tính

5.4.1 Mục tiêu

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

5.4.2 Bài toán

Bài toán 4

Xây dựng mô hình hồi quy để đánh giá các yếu tố ảnh hưởng đến **release_price** (giá phát hành) với mức ý nghĩa $\alpha = 5\%$

Xây dựng mô hình hồi quy trong đó:

- Biến phụ thuộc: **release_price**
- Biến độc lập: **number_of_pixels**, **core_speed_value**, **memory_value**, **memory_bandwidth_value**, **memory_speed_value**, **manufacturer**, **max_power_value**

5.4.3 Xây dựng mô hình hồi quy tuyến tính

5.4.3.1 Xử lý dữ liệu

Trích ra các mẫu định lượng trong tập dữ liệu, đồng thời lọc lấy những quan sát có yếu tố **release_price** trong tập dữ liệu.

```
1 # Trích các biến phụ thuộc và các biến độc lập
2 samples1 <- my_data[,c("release_price", "number_of_pixels",
3 "core_speed_value", "memory_value", "memory_bandwidth_value",
4 "memory_speed_value", "manufacturer", "max_power_value")]
5 head(samples1, 10) #In ra 10 dòng đầu của bộ data samples
```

| | release_price | number_of_pixels | core_speed_value | memory_value |
|----|------------------------|--------------------|------------------|-----------------|
| 1 | 371.5624 | 2073600 | 870 | 2048 |
| 2 | 371.5624 | 2073600 | 650 | 6144 |
| 3 | 371.5624 | 4096000 | 706 | 12288 |
| 4 | 371.5624 | 1440000 | 1050 | 3072 |
| 5 | 371.5624 | 3686400 | 837 | 6144 |
| 6 | 2999.0000 | 9216000 | 705 | 12288 |
| 7 | 2999.0000 | 8294400 | 705 | 12288 |
| 8 | 1099.0000 | 8294400 | 1140 | 12288 |
| 9 | 1998.0000 | 6220800 | 1000 | 24576 |
| 10 | 999.0000 | 8294400 | 1000 | 12288 |
| | memory_bandwidth_value | memory_speed_value | manufacturer | max_power_value |
| 11 | 134.4 | 1050 | AMD | 190 |
| 12 | 177.6 | 925 | Nvidia | 250 |
| 13 | 288.4 | 1502 | Nvidia | 245 |
| 14 | 57.6 | 900 | AMD | 150 |
| 15 | 288.4 | 1502 | Nvidia | 250 |
| 16 | 672.0 | 1750 | Nvidia | 375 |
| 17 | 672.0 | 1750 | Nvidia | 375 |
| 18 | 336.6 | 1753 | Nvidia | 250 |
| 19 | 673.2 | 1753 | Nvidia | 450 |
| 20 | 336.6 | 1753 | Nvidia | 250 |

5.4.3.2 Tổng quan mô hình

Ta có mô hình tổng quát theo những dữ liệu đã được trích xuất ở trên theo cấu trúc sau:

Gọi $Y, X_1, X_2, X_3, X_4, X_5, X_6, X_7$ lần lượt là các biến `release_price`, `number_of_pixels`, `core_speed_value`, `memory_value`, `memory_bandwidth_value`, `memory_speed_value`, `manufacturer`, `max_power_value`

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \epsilon$$

Sau khi thực hiện mô hình hồi quy tuyến tính ta sẽ được phương trình ước lượng:

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_3 + \hat{\beta}_4 X_4 + \hat{\beta}_5 X_5 + \hat{\beta}_6 X_6 + \hat{\beta}_7 X_7$$

5.4.3.3 Xây dựng mô hình

Dùng hàm `lm()` để tiến hành áp dụng mô hình hồi quy tuyến tính cho bộ dữ liệu đã xử lý, với `release_price` là biến phụ thuộc, các biến còn lại là các biến độc lập. Với giả thuyết:

$$\begin{cases} H_0 : \text{biến độc lập không có ý nghĩa cho mô hình} \\ H_1 : \text{biến độc lập có ý nghĩa cho mô hình} \end{cases}$$

Ta xét **p-value** và so sánh với mức ý nghĩa $\alpha = 5\%$ để kết luận giả thuyết cho từng biến.

```
1 # Dấu . thể hiện các biến còn lại trong samples
2 lm_model <- lm( release_price ~ release_price+number_of_pixels+core_speed_value+memory_value+
  memory_bandwidth_value+memory_speed_value+manufacturer+max_power_value, samples1 )
3 summary(lm_model)
```

```
1 Call:
2 lm(formula = release_price ~ release_price + number_of_pixels +
3   core_speed_value + memory_value + memory_bandwidth_value +
4   memory_speed_value + manufacturer + max_power_value, data = samples1)
5
6 Residuals:
7     Min       1Q   Median       3Q      Max
8  -710.0   -76.3     5.7    67.6  13481.2
9
10 Coefficients:
11             Estimate Std. Error t value Pr(>|t|)
12 (Intercept)    4.246e+02   3.674e+01  11.560 < 2e-16 ***
13 number_of_pixels -5.390e-06   1.249e-05  -0.432  0.666067
14 core_speed_value -1.870e-01   5.582e-02  -3.350  0.000824 ***
15 memory_value     5.782e-02   5.833e-03   9.912 < 2e-16 ***
16 memory_bandwidth_value -1.021e+00   2.170e-01  -4.703  2.74e-06 ***
17 memory_speed_value -7.046e-02   3.402e-02  -2.071  0.038504 *
18 manufacturerATI  -1.553e+02   4.940e+01  -3.143  0.001695 **
19 manufacturerIntel  1.186e+01   2.440e+02   0.049  0.961243
20 manufacturerNvidia  4.502e+01   1.737e+01   2.592  0.009626 **
21 max_power_value   1.338e+00   1.865e-01   7.176  1.02e-12 ***
22 ---
23 Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
24
25 Residual standard error: 342.3 on 1956 degrees of freedom
26 Multiple R-squared:  0.1221, Adjusted R-squared:  0.1181
27 F-statistic: 30.24 on 9 and 1956 DF, p-value: < 2.2e-16
```

Đánh giá p-value

- `number_of_pixels`: $p\text{-value} = 0.666067 > 5\% \Rightarrow$ Chấp nhận H_0 - Biến không có ý nghĩa với mô hình.
- `core_speed_value`: $p\text{-value} = 0.000824 < 5\% \Rightarrow$ Bác bỏ H_0 - Thừa nhận biến có ý nghĩa với mô hình.

- **memory_value**: **p-value** = $2 * 10^{-16} < 5\% \Rightarrow$ Bác bỏ H_0 - Thừa nhận biến có ý nghĩa với mô hình.
- **memory_bandwidth_value**: **p-value** = $2.74 * 10^{-6} < 5\% \Rightarrow$ Bác bỏ H_0 - Thừa nhận biến có ý nghĩa với mô hình.
- **memory_speed_value**: **p-value** = $0.038504 < 5\% \Rightarrow$ Bác bỏ H_0 - Thừa nhận biến có ý nghĩa với mô hình.
- **manufacturerATI**: **p-value** = $0.001695 < 5\% \Rightarrow$ Bác bỏ H_0 - Thừa nhận biến có ý nghĩa với mô hình.
- **manufacturerIntel**: **p-value** = $0.961243 > 5\% \Rightarrow$ Chấp nhận H_0 - Biến không có ý nghĩa với mô hình.
- **manufacturerNvidia**: **p-value** = $0.009626 < 5\% \Rightarrow$ Bác bỏ H_0 - Thừa nhận biến có ý nghĩa với mô hình.
- **max_power_value**: **p-value** = $1.02 * 10^{-12} < 5\% \Rightarrow$ Bác bỏ H_0 - Thừa nhận biến có ý nghĩa với mô hình.

Nhận xét

Dựa vào cột **Estimate**, ta có được các hệ số a, b_1, b_2, \dots của mô hình, từ đó xác định được phương trình hồi quy tuyến tính mẫu:

Lúc này, phương trình hồi quy đa biến có dạng:

$$\hat{Y} = 424.6 - 5.39 * 10^{-6} X_1 - 0.187 X_2 + 0.05782 X_3 - 1.021 X_4 - 0.07046 X_5 - 155.3 X_6 ATI + 11.86 X_6 Intel + 45.02 X_6 Nvidia + 1.338 X_7$$

Hệ số hiệu chỉnh: $R^2 = 11.81\%$

Xây dựng mô hình tối ưu

Do các biến **number_of_pixels** và **manufacturerIntel** không có ý nghĩa cho mô hình, ta sẽ loại bỏ các biến này, đồng thời giữ lại những biến còn lại.

```
1 # Dấu . thể hiện các biến còn lại trong samples
2 lm_model2 <- lm( release_price ~ core_speed_value+memory_value+memory_bandwidth_value+
   memory_speed_value+manufacturer+max_power_value, samples2 )
3 summary(lm_model2)
```

```
1 Call:
2 lm(formula = release_price ~ core_speed_value + memory_value +
3   memory_bandwidth_value + memory_speed_value + manufacturer +
4   max_power_value, data = samples2)
5
6 Residuals:
7     Min       1Q   Median       3Q      Max
8  -717.7   -75.8     6.1    68.4  13486.2
9
10 Coefficients:
11             Estimate Std. Error t value Pr(>|t|)
12 (Intercept)   4.226e+02  3.642e+01  11.602  < 2e-16 ***
13 core_speed_value -1.897e-01  5.547e-02  -3.420  0.000638 ***
14 memory_value     5.689e-02  5.416e-03  10.504  < 2e-16 ***
15 memory_bandwidth_value -1.076e+00  1.753e-01  -6.137  1.02e-09 ***
16 memory_speed_value -6.914e-02  3.389e-02  -2.040  0.041447 *
17 manufacturerATI -1.524e+02  4.895e+01  -3.113  0.001876 **
18 manufacturerNvidia  4.336e+01  1.694e+01   2.559  0.010562 *
19 max_power_value  1.364e+00  1.767e-01   7.718  1.87e-14 ***
20 ---
21 Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
22
23 Residual standard error: 342.3 on 1956 degrees of freedom
24 Multiple R-squared: 0.1221, Adjusted R-squared: 0.1189
25 F-statistic: 38.85 on 7 and 1956 DF, p-value: < 2.2e-16
```

Lúc này, phương trình hồi quy đa biến tối ưu có dạng:

$$\hat{Y} = 422.6 - 0.1897X_2 + 0.05609X_3 - 1.076X_4 - 0.06914X_5 - 152.4X_6ATI + 43.36X_6Nvidia + 1.364X_7$$

Hệ số hiệu chỉnh: $R^2 = 11.89\%$

Đánh giá: Đây là một tỉ lệ tương đối thấp, cho thấy các biến độc lập đưa vào phân tích hồi quy giải thích được 11.89% sự biến thiên của biến phụ thuộc, phần còn lại có thể được giải thích bởi phần dư gồm các biến độc lập ngoài mô hình và sai số ngẫu nhiên.

Ngoài ra, sau khi thực hiện loại bỏ các biến không ảnh hưởng đến phương trình hồi quy tuyến tính thì ta thấy hệ số hiệu chỉnh tăng lên, từ đó có thể giải thích được phương trình càng tối ưu thì hệ số hiệu chỉnh càng cao. Từ đó hiệu quả của mô hình càng lớn, để kiểm chứng cho nhận định trên ta thực hiện việc so sánh hiệu quả 2 mô hình.

So sánh hiệu quả hai mô hình

Để so sánh hiệu quả hai mô hình ta sử dụng hàm **anova()** để kiểm định hiệu quả của hai mô hình. Với giả thuyết:

$$\begin{cases} H_0 : \text{mô hình 2 hiệu quả hơn mô hình 1} \\ H_1 : \text{mô hình 1 hiệu quả hơn mô hình 2} \end{cases}$$

```
1 # thực hiện so sánh hai mô hình
2 anova(lm_model2, lm_model)
```

```
1 Analysis of Variance Table
2
3 Model 1: release_price ~ release_price | core_speed_value | number_of_pixels | memory_value |
  memory_bandwidth_value | memory_speed_value | manufacturer | max_power_value
4 Model 1: release_price ~ release_price | core_speed_value | memory_value |
  memory_bandwidth_value | memory_speed_value | manufacturer | max_power_value
5 Res.Df    RSS Df Sum of Sq    F Pr(>F)
6 1    1957  229153659
7 2    1956  229131836 1    21823  0.1863    0.6661
```

Nhận xét:

Vì **p-value** = 0.6661 > 5% nên chưa thể bác bỏ H_0 , đồng nghĩa với việc chấp nhận giả thuyết mô hình 2 hiệu quả hơn so với mô hình 1.

5.4.3.4 Ước lượng khoảng tin cậy cho các hệ số

Dùng hàm **confint()** để tìm các khoảng ước lượng cho các hệ số $\beta_1, \beta_2, \beta_3, \dots, \beta_n$ của mô hình hồi quy tuyến tính cho giá trị **release_price** của tổng thể các GPU với độ tin cậy 95%.

```
1 #Tìm khoảng ước lượng cho các hệ số
2 confint(lm_model2, level = 0.95)
```

```
1              2.5 %      97.5 %
2 (Intercept)  351.16176722 494.031950772
3 core_speed_value    -0.29849606 -0.080938002
4 memory_value       0.04626534  0.067508102
5 memory_bandwidth_value -1.41966468 -0.732018756
```

```
6 memory_speed_value      -0.13560139  -0.002683769
7 manufacturerATI         -248.41795830 -56.406081557
8 manufacturerNvidia      10.13500724  76.592957768
9 max_power_value         1.01740684   1.710641569
```

Từ kết quả trên, ta xác định được các khoảng tin cậy cho hệ số chặn β_0 cũng như các hệ số của các biến độc lập trong mô hình tổng thể với độ tin cậy 95%:

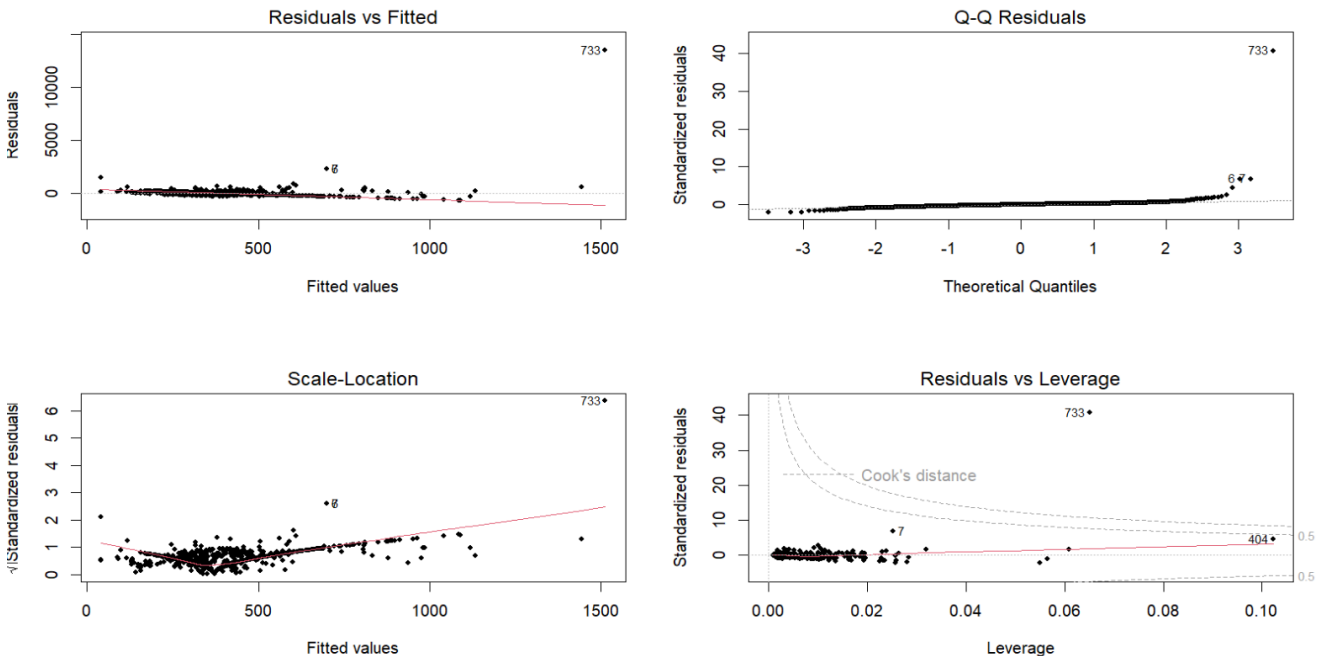
5.4.3.5 Kiểm tra giả thuyết

Từ kết quả có được về mô hình hồi quy tuyến tính đã xây dựng, ta cần kiểm định lại các giả thiết của mô hình:

1. Tính tuyến tính của dữ liệu.
2. Phần dư có phân phối chuẩn.
3. Phần dư có trung bình bằng 0.
4. Phần dư có phương sai không đổi.
5. Sai số ngẫu nhiên có phân phối chuẩn.
6. Sai số ngẫu nhiên có kỳ vọng tại mỗi giá trị bằng 0.

Dùng hàm `plot()` vẽ biểu đồ phần dư để kiểm tra các giả thiết về phần dư trên cho mô hình đã xây dựng:

```
1 par ( mfrow = c(2, 2))
2 plot(lm_model2, pch=20)
```



Hình 5.6. Kết quả thực thi đoạn chương trình trên.

Ý nghĩa của các biểu đồ:

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

càng chứng tỏ tính tuyến tính của dữ liệu càng cao. Mặc khác, giả thiết phần dư có trung bình bằng 0 thỏa mãn nếu đường màu đỏ gần với đường nét đứt nằm ngang (ứng với phần dư bằng 0) trên biểu đồ.

- **Biểu đồ Normal Q-Q:** Dùng để kiểm tra giả thiết phần dư có phân phối chuẩn. Nếu các điểm thẳng dư có xu hướng phân bố trên một đường thẳng thì điều kiện về phân phối chuẩn của phần dư được thỏa.
- **Biểu đồ Scale - Location:** Dùng để kiểm định giả thiết phương sai của phần dư là không đổi. Trục tung là căn bậc hai của phần dư (đã được chuẩn hóa), trục hoành là giá trị tiên lượng \hat{y}_i của các biến phụ thuộc. Nếu đường màu đỏ trên đồ thị là đường thẳng nằm ngang và các điểm thẳng dư phân tán đều xung quanh đường thẳng này thì giả thiết về phương sai của phần dư được thỏa.
- **Biểu đồ Residuals vs Leverage:** mặc dù không dùng để kiểm định giả thiết cho mô hình, tuy nhiên biểu đồ này giúp xác định những điểm outliers (các phần tử bất thường có thể là nguyên nhân gây ra sự vi phạm các giả thiết hay làm sai lệch kết quả dự báo của mô hình). Những điểm outliers sẽ cách xa đường màu đỏ trên biểu đồ.

Nhận xét:

- **Biểu đồ Residuals vs Fitted** cho thấy giả thiết về tính tuyến tính của dữ liệu là có thể chấp nhận được. Tuy nhiên giả thiết về trung bình của phần dư có thể coi là thỏa mãn.
- **Biểu đồ Normal Q-Q** cho thấy giả thiết về phần dư có phân phối chuẩn được thỏa mãn.
- **Biểu đồ Scale - Location** đường màu đỏ có độ dốc, không thẳng, các điểm thẳng dư phân tán không đều xung quanh đường thẳng này. Do đó, giả thiết về tính đồng nhất của phương sai đối với mô hình này bị vi phạm.
- **Biểu đồ Residuals vs Leverage** chỉ ra quan sát thứ 733 có thể là điểm outliers, gây ảnh hưởng đến mô hình đã xây dựng. Vì vậy cần loại bỏ đi để mô hình chuẩn hơn.

5.4.3.6 Dự đoán

Vì lý do hệ số hiệu chỉnh R^2 chỉ có 11.89% nên việc dự đoán là không hiệu quả. Do đó, cần phải cân nhắc để thực hiện dự đoán cho mô hình hồi quy tuyến tính này. Giả sử ta có hệ số hiệu chỉnh cao và thực hiện quá trình dự đoán cho mô hình hồi quy tuyến tính đã xây dựng ở trước.

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

```
1 # Khởi tạo các giá trị cần dự đoán cho các biến độc lập
2 core_speed_value <- c(870 , 935 , 1011 , 853)
3 memory_value <- c(16000 , 8190 , 2048 , 1024)
4 memory_bandwidth_value <- c(220 , 70 , 110 , 150)
5 memory_speed_value <- c(1055 , 1024 , 1048 , 1100)
6 manufacturer <- c("AMD", "ATI", "Nvidia", "ATI")
7 max_power_value <- c(145, 124, 95, 191)
8 new <- data.frame(core_speed_value , memory_value , memory_bandwidth_value ,
9                   memory_speed_value, manufacturer, max_power_value)
10 # Dự đoán giá trị Release Price tương ứng với giá trị các biến độc lập đã tạo
11 predict (lm_model2 , new , interval = "confidence")
```

| | fit | lwr | upr |
|---|-----------|-----------|-----------|
| 1 | 1055.8835 | 924.97090 | 1186.7961 |
| 2 | 581.7297 | 467.23807 | 696.2214 |
| 3 | 329.4392 | 301.26119 | 357.6172 |
| 4 | 189.7037 | 87.84011 | 291.5674 |

Nhận xét: Kết quả cho ta các dự đoán và khoảng ước lượng của dự đoán về giá ra mắt GPU tương ứng với các thông số độc lập trong mô hình.

Ví dụ, với một mẫu GPU có `core_speed_value` là $870[MHz]$, `memory_value` là $16000[MB]$, `memory_bandwidth_value` là $220[GB/sec]$, `memory_speed_value` là $1055[MHz]$ thì giá ra mắt dự đoán cho mẫu GPU này là khoảng \$1055.8835 và khoảng ước lượng cho giá trị ra mắt thật sự là $(924.97090; 1186.7961)$.

5.4.4 Kết luận

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

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

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

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

6 Thảo luận và mở rộng

Bài tập lớn này đã giúp nhóm có cơ hội được áp dụng được những kiến thức lý thuyết đã học từ môn Xác suất thống kê ở học kỳ này vào ứng dụng thực tế; học hỏi và nghiên cứu các nội dung cơ bản trong lĩnh vực thống kê. Cùng với đó, nhóm cũng đã tìm tòi và biết cách áp dụng ngôn ngữ R cho việc xử lý thống kê, từ đó thấy được sức mạnh của công cụ này trong việc hỗ trợ tính toán.

Do giới hạn về mặt kiến thức, thời gian và khả năng làm việc nhóm, bài tập này không thể tránh khỏi những sai sót. Nhóm sẽ ghi nhận, khắc phục và lấy đó làm kinh nghiệm cho những bài tập sau này.

Các nội dung nhóm đã trình bày trong bài tập này chỉ mới là một phần nhỏ trong lĩnh vực thống kê. Ngoài ra, nhóm hi vọng sẽ có cơ hội được tiếp tục tham khảo, thực hiện các nghiên cứu các vấn đề khác trong lĩnh vực này, chẳng hạn mô hình logistics, ứng dụng trí tuệ nhân tạo trong thống kê, thuật toán KNN,...

7 Nguồn dữ liệu và nguồn code

- Nguồn file `All_GPUs .csv` : <https://www.kaggle.com/datasets/iliassekkaf/computerparts/>
- Nguồn code : <https://github.com/thiet28/code-k233.git>

Tài liệu

- [1] Huỳnh Thái Duy Phương, Slides bài giảng Xác suất thống kê.
- [2] Nguyễn Đình Huy, Giáo trình Xác suất thống kê, NXB ĐHQG TP.HCM.
- [3] Seal, H. L. (1967). Studies in the History of Probability and Statistics. XV The historical development of the Gauss linear model. *Biometrika*, 54(1-2), 1-24.
- [4] González-Estrada, E., & Cosmes, W. (2019). Shapiro–Wilk test for skew normal distributions based on data transformations. *Journal of Statistical Computation and Simulation*, 89(17), 3258-3272.
- [5] De Iorio, M., Müller, P., Rosner, G. L., & MacEachern, S. N. (2004). An ANOVA model for dependent random measures. *Journal of the American Statistical Association*, 99(465), 205-215.
- [6] Mokhtari, A., & Frey, H. C. (2005). Sensitivity analysis of a two-dimensional probabilistic risk assessment model using analysis of variance. *Risk Analysis: An International Journal*, 25(6), 1511-1529.