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

Xác suất thống kê

Bài tập lớn

Phân tích tập dữ liệu về GPU

GVHD: Nguyễn Thị Hiên
SV thực hiện: Hoa Toàn Hạc – 2201917
Đặng Châu Anh – 2310063
Nguyễn Công Thành – 2313119
Nguyễn Chí Duy Khang – 2311436
Nguyễn Lê Khôi Nguyên – 2312366

TP. HỒ CHÍ MINH, THÁNG 11, 2024

1 Danh sách thành viên & Nhiệm vụ

STT	Họ và tên	MSSV	NHIỆM VỤ	PHẦN TRĂM
1	Hoa Toàn Hạc	2210917	Xử lý dữ liệu, hồi quy đa biến	20%
2	Nguyễn Công Thành	2313119	Kiểm định	20%
3	Nguyễn Chí Duy Khang	2311436	Hồi quy đơn biến	20%
4	Nguyễn Lê Khôi Nguyên	2312366	Giới thiệu & tiền xử lý dữ liệu	20%
5	Đặng Châu Anh	2310063	Phân tích suy diễn	20%

Mục lục

1 Danh sách thành viên & Nhiệm vụ **1**

2 Giới thiệu về dữ liệu **3**
2.1 Tổng quan về dữ liệu . . . 3
2.2 Tổng quan về biến . . . 3

3 Kiến thức nền **5**
3.1 Biểu đồ hộp (Boxplot) . . . 5
3.2 Biểu đồ Histogram . . . 6
3.3 Biểu đồ tương quan (Scatter Plot) . . . 6
3.4 Biểu đồ phân tán . . . 6
3.5 Hồi quy tuyến tính . . . 6

4 Tiền xử lý dữ liệu **8**
4.1 Đọc dữ liệu . . . 8
4.2 Kiểm tra và xử lý dữ liệu khuyết trong tập dữ liệu . . . 8
4.3 Lựa chọn biến . . . 9
4.4 Làm sạch dữ liệu . . . 10

5 Thống kê mô tả **11**
5.1 Tóm tắt dữ liệu đã được làm sạch . . . 11
5.2 Đồ thị histogram cho biến Memory_Speed . . . 12
5.3 Boxplot cho biến Memory_Speed theo Manufacturer, Dedicated, MemoryType . 12
5.4 Vẽ biểu đồ tương quan giữa các biến . . . 15
5.5 Đồ thị scatter plot cho biến Memory_Speed theo các biến "Memory_Bandwidth", "L2_Cache", "Memory_Bus", " Process", "Memory" . . . 16

6 Xử lý dữ liệu **17**
6.1 Xử lý biến Dedicated . . . 17
6.2 Xử lý biến Memory_Type . . . 18
6.3 Xử lý biến Architecture . . . 19
6.4 Xử lý biến Memory_Bus và Memory_Speed . . . 20
6.5 Xử lý biến Memory_Bandwidth . . . 20
6.6 Xử lý biến Memory . . . 20
6.7 Xử lý biến Process . . . 20
6.8 Kết luận . . . 21

7 Thống kê suy diễn **22**
7.1 Khoản tin cậy . . . 22
7.2 Kiểm định trung bình hai mẫu . . . 24
7.3 Hồi quy tuyến tính đơn . . . 25

8 Thảo luận và mở rộng **29**
8.1 Thảo luận . . . 29
8.2 Mở rộng: Hồi quy tuyến tính đa biến . . . 29
8.3 So sánh giữa mô hình hồi quy tuyến tính đơn biến và đa biến . . . 32

9 Code **34**

2 Giới thiệu về dữ liệu

2.1 Tổng quan về dữ liệu

Bộ dữ liệu này chứa thông số kỹ thuật chi tiết, ngày phát hành và giá phát hành của GPU. Đơn vị xử lý đồ họa (GPU) hiện nay là một thành phần quan trọng trong công nghệ máy tính, được sử dụng rộng rãi. GPU, nổi tiếng với khả năng xử lý song song, rất cần thiết trong nhiều ứng dụng liên quan đến xử lý đồ họa và video, và ngày càng được ứng dụng để tăng tốc các thuật toán AI và Machine Learning, giúp huấn luyện mô hình AI nhanh hơn, hiệu quả hơn.

Mục tiêu của bài báo cáo này là phân tích các đặc điểm cụ thể của GPU dựa trên tập dữ liệu được cung cấp. Thông tin về GPU được tóm tắt trong bảng dưới đây

Properties	**Information**
Nguồn dữ liệu	All_GPUs.csv
Đối tượng quan sát	Các đơn vị xử lý độ họa (GPU)
Số lượng quan sát	3406
Số lượng biến	34

2.2 Tổng quan về biến

Tên biến	**Phân loại biến**	**Đơn vị**	**Mô tả**
Architecture	Định tính		Phân loại mã định danh cho kiến trúc vi xử lý GPU.
Best Resolution	Định tính		Độ phân giải màn hình cao nhất.
Boost Clock	Định lượng	MHz	Tốc độ cao tạm thời mà GPU hoạt động khi nó chịu tải nặng.
Core Speed	Định lượng	MHz	Tốc độ hoạt động của bộ xử lý chính.
DVI Connection	Định tính		Số lượng cổng DVI dùng để truyền tín hiệu video số từ máy tính hoặc thiết bị khác đến màn hình hiển thị.
Dedicated	Định tính		Cho biết có phải là card đồ họa rời hay không. Card đồ họa rời có bộ nhớ và đơn vị xử lý riêng, tách biệt hoàn toàn với CPU.
Direct X	Định tính		Phiên bản DirectX mà GPU hỗ trợ. DirectX là một tập hợp các giao diện lập trình ứng dụng (APIs), được sử dụng để xử lý các tác vụ liên quan đến đa phương tiện.

DisplayPort Connection	Định tính		Cho biết GPU có hỗ trợ cổng kết nối DisplayPort hay không. DisplayPort là một tiêu chuẩn giao tiếp kỹ thuật số được sử dụng để truyền tín hiệu video và âm thanh từ GPU đến màn hình.
HDMI Connection	Định tính		Số lượng cổng HDMI có trên card đồ họa rời.
Integrated	Định tính		Card đồ họa được tích hợp vào CPU hoặc bo mạch chủ và chia sẻ tài nguyên hệ thống như bộ nhớ với CPU.
L2 Cache	Định lượng	KB	Một bộ nhớ đệm trên chip bổ sung để lưu giữ các bản sao của dữ liệu di chuyển qua lại giữa các đơn vị xử lý (SMs) và bộ nhớ chính.
Manufacturer	Định tính		Công ty thiết kế và sản xuất các đơn vị xử lý đồ họa (GPU).
Max Power	Định lượng	Watts	Công suất tiêu thụ tối đa của phần cứng hệ thống trong thực tế.
Memory	Định lượng	MB	Dung lượng bộ nhớ đồ họa (VRAM) mà GPU có.
Memory Bandwidth	Định lượng	GB/sec	Đo lượng dữ liệu có thể truyền tải giữa bộ nhớ và các lõi tính toán trong 1 giây.
Memory Bus	Định lượng	Bit	Lượng dữ liệu có thể được truyền giữa GPU và bộ nhớ của nó tại một thời điểm.
Memory Speed	Định lượng	MHz	Tốc độ mà bộ nhớ của GPU có thể đọc và ghi dữ liệu.
Memory Type	Định tính		Loại bộ nhớ đồ họa được dùng trong GPU.
Name	Định tính		Tên GPU.
Notebook GPU	Định tính		Card đồ họa dành cho notebook.
Open GL	Định lượng		Thư viện Đồ họa Mở - Giao diện lập trình ứng dụng (API) đa nền tảng và đa ngôn ngữ để kết xuất đồ họa 2D và 3D vector.
PSU	Định lượng	Watts & Amps	Phần của một PC chịu trách nhiệm chuyển đổi dòng điện xoay chiều (AC) từ ổ cắm điện thành dòng điện một chiều (DC) mà các thành phần máy tính có thể sử dụng.
Pixel Rate	Định lượng	GPixel/s	Số lượng pixel mà một GPU có thể vẽ (hiển thị) lên màn hình trong 1 giây.
Power Connector	Định tính	pin	Cáp kết nối card đồ họa với bộ cung cấp điện, cung cấp điện năng cần thiết để card hoạt động.

Process	Định lượng	nm	Kích thước của các transistor và các thành phần khác trong chip.
ROPs	Định tính		Render Output Unit - một thành phần cụ thể trên GPU chịu trách nhiệm xử lý các giá trị pixel cuối cùng trước khi vẽ chúng lên màn hình.
Release Date	Định lượng		Ngày GPU được công bố giới thiệu hoặc phát hành.
Release Price	Định lượng		Giá cụ thể bằng đơn vị tiền tệ mà GPU được giới thiệu chính thức.
Resolution WxH	Định tính		Kích thước chiều rộng và chiều cao (độ phân giải hiển thị) của một thiết bị hiển thị điện tử, được đo bằng pixel.
SLI Crossfire	Định tính		Phương pháp render nhiều GPU để tăng hiệu suất đồ họa.
Shader	Định tính		Một chương trình máy tính tính toán các mức độ ánh sáng, bóng tối và màu sắc phù hợp trong quá trình render cảnh 3D.
TMUs	Định lượng		Texture Mapping Unit - một thành phần cấp thấp của GPU hoạt động một cách độc lập, chuyên dụng cho việc xử lý bitmaps và lọc texture.
Texture Rate	Định lượng	GTexel/s	Số lượng pixel được đồ họa có thể render với texture trên màn hình mỗi giây.
VGA Connection	Định tính		Số lượng cổng giao diện hiển thị tiêu chuẩn được sử dụng để kết nối thiết bị xuất video với máy tính, máy chiếu, màn hình và TV.

3 Kiến thức nền

3.1 Biểu đồ hộp (Boxplot)

Trong thống kê mô tả, biểu đồ hộp hoặc biểu đồ hộp là một phương pháp để thể hiện đồ họa các nhóm vị trí, độ lan truyền và độ lệch của dữ liệu số thông qua các tứ phân vị của chúng. Ngoài hộp trên biểu đồ hộp, có thể có các đường (được gọi là râu) kéo dài từ hộp biểu thị sự thay đổi bên ngoài các tứ phân vị trên và dưới, do đó, biểu đồ cũng được gọi là biểu đồ hộp và râu và sơ đồ hộp và râu. Các giá trị ngoại lai khác biệt đáng kể so với phần còn lại của tập dữ liệu có thể được biểu diễn dưới dạng các điểm riêng lẻ bên ngoài râu trên biểu đồ hộp. Biểu đồ hộp không phải là tham số: chúng hiển thị sự thay đổi trong các mẫu của một quần thể thống kê mà không đưa ra bất kỳ giả định nào về phân phối thống kê cơ bản (mặc dù biểu đồ hộp của Tukey giả định tính đối xứng cho râu và tính chuẩn cho chiều dài của chúng). Khoảng cách trong mỗi tiểu phần của biểu đồ hộp biểu thị mức độ phân tán (độ lan truyền) và độ lệch của dữ liệu, thường được mô tả bằng cách sử dụng tóm tắt năm số. Ngoài ra, biểu đồ hộp cho phép

ước tính trực quan nhiều ước lượng L khác nhau , đặc biệt là phạm vi liên tứ phân vị, midhinge, range, mid-range và trimean. Biểu đồ hộp có thể được vẽ theo chiều ngang hoặc chiều dọc.

3.2 Biểu đồ Histogram

Là một biểu diễn trực quan về sự phân bố của dữ liệu định lượng. Để xây dựng biểu đồ histogram, bước đầu tiên là "phân loại" (hoặc "xô") phạm vi giá trị chia toàn bộ phạm vi giá trị thành một loạt các khoảng và sau đó đếm xem có bao nhiêu giá trị rơi vào mỗi khoảng. Các khoảng thường được chỉ định là các khoảng liên tiếp, không chồng lấn của một biến. Các khoảng (khoảng) nằm cạnh nhau và thường (nhưng không bắt buộc) có kích thước bằng nhau.

Biểu đồ histogram cung cấp cảm nhận sơ bộ về mật độ của phân phối dữ liệu cơ bản và thường dùng để ước tính mật độ : ước tính hàm mật độ xác suất của biến cơ bản. Tổng diện tích của biểu đồ histogram được sử dụng cho mật độ xác suất luôn được chuẩn hóa thành 1. Nếu độ dài của các khoảng trên trục x đều bằng 1, thì biểu đồ histogram giống hệt với biểu đồ tần suất tương đối .

3.3 Biểu đồ tương quan (Scatter Plot)

Ma trận tương quan cho phép phân tích mối quan hệ giữa từng cặp biến số của một tập dữ liệu. Mối quan hệ giữa từng cặp biến được trực quan hóa thông qua biểu đồ phân tán hoặc ký hiệu biểu diễn mối tương quan r_{xy} (bong bóng, đường thẳng, số..). Phạm vi của r_{xy}: $-1 \leq r_{xy} \leq 1$

- $-1 \leq r_{xy} \leq 0$ tương quan âm. r_{xy} gần hơn với -1 biểu thị tương quan âm mạnh hơn giữa X và Y
- $-1 \leq r_{xy} \leq 0$ tương quan dương. r_{xy} gần hơn với 1 biểu thị tương quan dương mạnh hơn giữa X và Y
- r_{xy} gần hơn với 0 biểu thị tương quan yếu giữa X và Y. $r_{xy} = 0$: biểu thị độc lập tuyến tính giữa X và Y

3.4 Biểu đồ phân tán

Là một loại biểu đồ hoặc sơ đồ toán học sử dụng tọa độ Descartes để hiển thị các giá trị cho hai biến thường gặp đối với một tập dữ liệu. Nếu các điểm được mã hóa (màu sắc/hình dạng/kích thước), một biến bổ sung có thể được hiển thị. Dữ liệu được hiển thị dưới dạng một tập hợp các điểm, mỗi điểm có giá trị của một biến xác định vị trí trên trục ngang và giá trị của biến khác xác định vị trí trên trục dọc.

3.5 Hồi quy tuyến tính

Hồi quy tuyến tính là một trong những phương pháp cơ bản và phổ biến nhất trong thống kê và học máy để mô hình hóa mối quan hệ giữa một biến phụ thuộc y và một hoặc nhiều biến độc lập $\mathbf{X}$. Mô hình hồi quy tuyến tính có dạng:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

Trong đó:

- y: Biến phụ thuộc (biến mục tiêu cần dự đoán).
- $x_1, x_2, \ldots, x_p$: Các biến độc lập (các yếu tố giải thích).

- $\beta_0, \beta_1, \ldots, \beta_p$: Các hệ số hồi quy, thể hiện mức độ ảnh hưởng của các biến độc lập lên biến phụ thuộc.
- ϵ: Sai số (phần dư), thể hiện sự khác biệt giữa giá trị thực tế và giá trị dự đoán.

Ưu điểm

- Mô hình đơn giản, dễ hiểu và dễ triển khai.
- Tính giải thích cao, có thể phân tích ý nghĩa thống kê của các biến độc lập.
- Hiệu quả khi dữ liệu có mối quan hệ tuyến tính giữa các biến.

Nhược điểm

- Không hiệu quả khi dữ liệu có mối quan hệ phi tuyến tính.
- Nhạy cảm với đa cộng tuyến giữa các biến độc lập.
- Phụ thuộc vào giả định rằng các phần dư tuân theo phân phối chuẩn và có phương sai đồng nhất.

Các giả định chính của hồi quy tuyến tính

- Mối quan hệ tuyến tính: Quan hệ giữa biến phụ thuộc và các biến độc lập là tuyến tính.
- Không có đa cộng tuyến: Các biến độc lập không có mối quan hệ tương quan cao.
- Phương sai đồng nhất: Sai số có phương sai không đổi (homoscedasticity).
- Sai số tuân theo phân phối chuẩn: Sai số phân phối chuẩn với trung bình bằng 0.
- Không có tự tương quan: Sai số không có mối quan hệ phụ thuộc lẫn nhau.

Đánh giá mô hình Hiệu quả của mô hình hồi quy tuyến tính thường được đánh giá qua các chỉ số sau:

- R^2 (Hệ số xác định): Thể hiện tỷ lệ phần trăm sự biến thiên của biến phụ thuộc được giải thích bởi các biến độc lập.
- RMSE (Sai số bình phương trung bình): Đo lường độ lệch giữa giá trị dự đoán và giá trị thực tế.
- Giá trị p (p-value): Kiểm tra mức ý nghĩa thống kê của từng biến độc lập.
- F-statistic: Kiểm tra mức ý nghĩa thống kê tổng thể của mô hình.

Ứng dụng Hồi quy tuyến tính được sử dụng rộng rãi trong các lĩnh vực như kinh tế, y học, khoa học xã hội, và khoa học dữ liệu để dự đoán hoặc giải thích mối quan hệ giữa các biến.

4 Tiền xử lý dữ liệu

4.1 Đọc dữ liệu

Đầu tiên, chúng ta đọc dữ liệu từ tệp tin All_GPUs.csv và in hiển thị một vài dòng đầu tiên. Việc này giúp chúng ta có cái nhìn tổng quan về dữ liệu, hiểu cấu trúc, định hướng trước khi đi sâu vào phân tích dữ liệu. Quá trình này được mô tả trong hình dưới đây:

```
All_GPUs <- read.csv("D:/BTL/All_GPUs.csv")
head(All_GPUs, 6)
```

4.2 Kiểm tra và xử lý dữ liệu khuyết trong tập dữ liệu

Ta sẽ thực hiện thống kê số lượng và tỷ lệ dữ liệu khuyết ở các biến, từ đó xử lý lựa chọn ra các biến chính để phân tích. Tuy nhiên, ta nhận thấy rằng các giá trị khuyết không cùng loại. Nó có thể là giá trị trống (hay rỗng) , gạch ngang "-" ,.. gây khó khăn cho việc thống kê dữ liệu khuyết . Vì thế, để thuận tiện ta chuyển đổi tất cả chúng thành giá trị NA.

Đầu tiên, tìm các dòng trống (hàng có toàn bộ các giá trị là rỗng hoặc "") trong dữ liệu và chuyển chúng thành giá trị NA:

```
All_GPUs[All_GPUs == ""] <- NA
```

Tiếp đó, tìm các giá trị trong dataframe có định dạng -(các giá trị bắt đầu bằng ký tự xuống dòng theo sau là dấu -) và chuyển chúng thành NA. Ta sử dụnglệnh gsub() để thay thế "-" bằng NA trong toàn bộ dataframe.

```
All_GPUs[] <- lapply(All_GPUs, function(x) gsub("^\\n- $", NA, x))
```

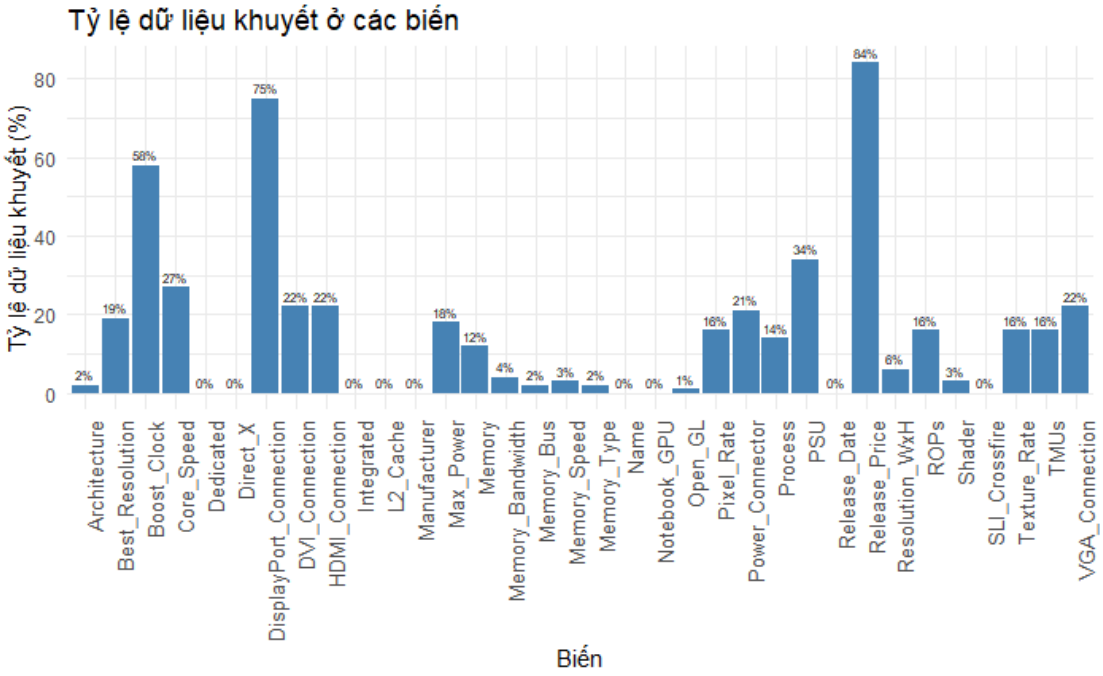
Cuối cùng là thay thế các chuỗi "NA" (N/A) thành NA.

```
All_GPUs[All_GPUs == "NA"] <- NA
```

Bây giờ các dữ liệu khuyết đã đồng nhất, ta tiến hành thống kê số lượng và tỷ lệ dữ liệu khuyết ở các biến bằng hàm freq.na() trong thư viện questionr.

```
# Su dung ham freq.na() de tim tan suat NA trong du lieu
na_summary <- freq.na(All_GPUs)
# Chuyen ket qua thanh mot data frame
na_summary_df <- as.data.frame(na_summary)
colnames(na_summary_df) <- c("NA_Count", "NA_Percentage")
kable(na_summary_df, format = "markdown", caption = "So luong va ti le
    du lieu khuyet o cac bien")
```

Dựa trên kết quả thu được ở trên, ta nhận thấy có nhiều biến chứa dữ liệu khuyết, để dễ dàng hơn, ta thực hiện vẽ đồ thị thể hiện tỷ lệ dữ liệu khuyết ở từng biến bằng cách sử dụng hàm ggplot2. Ta thu được đồ thị như hình dưới đây.



Hình 1: Đồ thị tỷ lệ dữ liệu bị khuyết

Với các biến có tỷ lệ dữ liệu khuyết lơn hơn 15% ta sẽ xóa đi để giảm nhiễu cho quá trình phân tích. Và đồng thời với các biến có tỷ lệ dữ liệu khuyết dưới 15%, ta sẽ xóa đi các quan sát chứa dữ liệu khuyết, với 3406 quan sát việc này vẫn đảm báo không làm mất đi tính chính xác và tầm quan trọng của biến.

```
# Giu lai cac bien co ty le du lieu khuyet duoi 15%
selected_cols <- rownames(na_summary_df)[na_summary_df$NA_Percentage <
    15]

# Giu lai cac bien thoa man dieu kien trong dataframe ban dau
new_All_GPUs <- All_GPUs[, selected_cols]

# Xoa cac quan sat chua du lieu khuyet
new_All_GPUs<-na.omit(new_All_GPUs)
```

4.3 Lựa chọn biến

Sau khi đã loại bỏ đi các biến có tỷ lệ khuyết cao, ta sẽ chọn lọc các biến quan trọng. Qua tìm hiểu, nhóm nhận thấy biến **Memory_Speed** là một nhân tố quan trọng ảnh hưởng đến hiệu suất của GPU. Tốc độ xung nhịp bộ nhớ là một yếu tố thiết yếu trong hiệu suất của GPU, đặc biệt đối với các tác vụ yêu cầu GPU truy cập nhiều dữ liệu. Tốc độ xung nhịp bộ nhớ cao hơn có nghĩa là GPU có thể truy cập bộ nhớ nhanh hơn, dẫn đến tăng hiệu suất. **Memory_Speed** không chỉ là về tốc độ xung nhịp, nó có liên quan và chịu ảnh hưởng trực tiếp bởi Memory_Bandwidth.

Và các nhân tố khác có ảnh hưởng với mức độ khác nhau đến **Memory_Speed** bao gồm: "**Memory_Type**","**Memory_Bus**","**L2_Cache**","**Architecture**","**Process**","**Memory**","**Dedicated**". Ta có thể lấy thêm biến "Manufacturer"để so sánh và nhận xét về **Memory_Speed** của các GPU ở các nhà sản xuất khác nhau.

Dây là một số nguồn tài liệu và cơ sở ta có thể tham khảo để tìm hiểu về các biến có ảnh hưởng đến **Memory_Speed** trong GPU.

- Link: https://vibox.co.uk/blog/gpu-memory-clock-speed-vs-gpu-core-clock-speed
- Link: https://www.hp.com/us-en/shop/tech-takes/does-ram-speed-matter

4.4 Làm sạch dữ liệu

Xử lý đơn vị Để thuận tiện hơn cho quá trình phân tích, ta sẽ bỏ đi đơn vị của các biến như "L2_Cache", "Memory","Memory_Bandwidth","Memory_Bus", "Memory_speed", "Process". Việc này giúp đồng nhất dữ liệu, hợp lý hóa phân tích về sau giúp việc so sánh và diễn giải dữ liệu trên các biến trở nên đơn giản hơn.

```
# Lua chon cac bien can xoa don vi
columns_to_clean <- c("L2_Cache","Memory","Memory_Bandwidth","
    Memory_Bus","Memory_Speed","Process")

# Tao ham thuc hien xoa don vi o cac bien
remove_units <- function(column) {
  # Su dung gsub de xoa tat ca cac ky tu khong phai so ke ca don vi
  cleaned_column <- gsub("[^0-9.]", "", column)
  # Chuyen doi ket qua ve kieu numeric
  cleaned_column <- as.numeric(cleaned_column)

  return(cleaned_column)
}

# Ap dung ham cho cac bien da chon
new_All_GPUs[columns_to_clean] <- lapply(new_All_GPUs[columns_to_clean
    ], remove_units)
```

Cuối cùng, kiểm tra lại số lượng giá trị bị khuyết để xác nhận rằng đây thực sự là tập dữ liệu cuối cùng sau khi làm sạch. Như minh họa trong hình dưới , không còn dữ liệu khuyết trong tập dữ liệu cuối cùng. Như vậy quá trình làm sạch dữ liệu đã thành công.

	missing	%
Process	0	0
Memory	0	0
Resolution_WxH	0	0
Memory_Bandwidth	0	0
Shader	0	0
Memory_Speed	0	0
Architecture	0	0
Memory_Bus	0	0
Memory_Type	0	0
Open_GL	0	0
Dedicated	0	0
Integrated	0	0
Direct_X	0	0
L2_Cache	0	0
Manufacturer	0	0
Name	0	0
Notebook_GPU	0	0
Release_Date	0	0
SLI_Crossfire	0	0

Hình 2: Tỷ lệ dữ liệu bị khuyết sau khi làm sạch

5 Thống kê mô tả

5.1 Tóm tắt dữ liệu đã được làm sạch

Để thực hiện tính các thống kê mô tả cho các biến định lượng, nhóm báo cáo sử dụng các hàm có sẵn trong R để tính trung bình, độ lệch chuẩn mẫu, các phân vị, trung vị, giá trị lớn nhất và giá trị nhỏ nhất. Sử dụng tập dữ liệu GPU_data để tính thống kê cho từng cột định lượng.

Đối với các biến định tính, nhóm lập bảng thống kê số lượng cho các phân loại.

	Mean	SD	Min	Q1.25%	Median	Q3.75%	Max
Memory_Speed	1280.09594	396.83295	275	950	1251.0	1553.0	2127
Memory	3140.95108	2829.52970	128	1024	2048.0	4096.0	32000
Memory_Bandwidth	162.36572	136.52718	4	64	130.6	224.4	1280
Memory_Bus	221.58210	234.99052	32	128	192.0	256.0	4096
L2_Cache	1482.62799	3408.42555	0	256	512.0	1024.0	30722
Process	32.02958	13.80621	14	28	28.0	40.0	150

Hình 3: Tóm tắt dữ liệu

DDR	DDR2	DDR3	eDRAM	GDDR2	GDDR3	GDDR4	GDDR5	GDDR5X	HBM-1	HBM-2
7	40	359	2	3	177	4	1956	78	6	5

Hình 4: Tóm tắt dữ liệu về Memory Type

No	Yes
3	2634

AMD	ATI	Intel	Nvidia
1004	82	3	1548

Hình 5: Tóm tắt dữ liệu về Dedicated và Manufacturer

5.2 Đồ thị histogram cho biến Memory_Speed

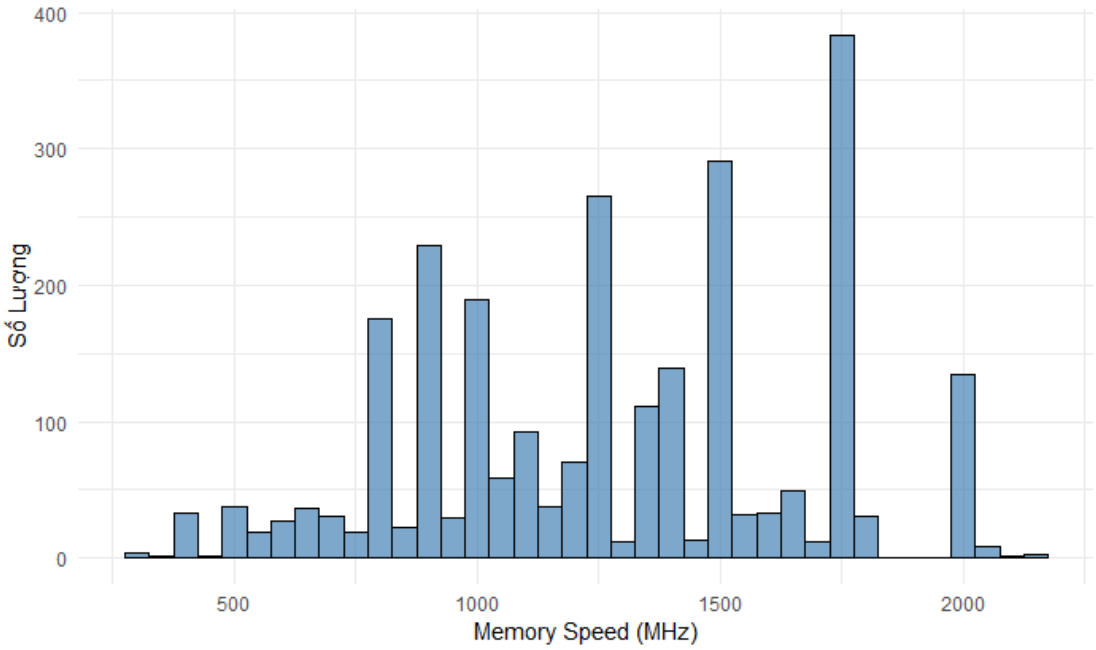
Nhóm rút ra được một số nhận xét về đồ thị histogram "Phân Phối Memory Speed": Đồ thị histogram thể hiện phân phối Memory Speed của các GPU trong tập dữ liệu. Phần lớn các giá trị tập trung trong khoảng từ 800 MHz đến 1600 MHz, cho thấy các GPU phổ biến thường hoạt động trong phạm vi tốc độ này. Đặc biệt, các bins xung quanh 1500 MHz có tần suất xuất hiện cao nhất, có thể phản ánh nhóm sản phẩm chủ lực hoặc tiêu chuẩn phổ biến trong ngành sản xuất GPU.

Ở hai đầu của phân phối, các giá trị dưới 500 MHz và trên 2000 MHz xuất hiện rất ít, biểu thị đây là các trường hợp hiếm. Điều này có thể liên quan đến các GPU cũ có hiệu năng thấp hoặc các GPU cao cấp dành cho những ứng dụng đặc thù.

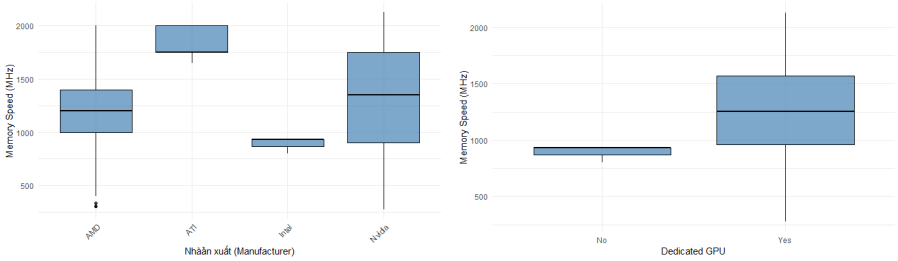
Phân phối không đồng đều này cho thấy sự đa dạng về hiệu năng của các GPU trong tập dữ liệu. Tuy nhiên, sự tập trung mạnh ở một số khoảng giá trị gợi ý rằng một vài dòng sản phẩm chiếm ưu thế lớn trên thị trường. Để hiểu rõ hơn về các đặc điểm chi tiết, việc điều chỉnh binwidth hoặc phân tích các thông số kỹ thuật tương ứng có thể hữu ích. Ngoài ra, cần kiểm tra các giá trị bất thường ở hai đầu phân phối để loại bỏ khả năng dữ liệu lỗi hoặc ngoại lệ không hợp lý.

5.3 Boxplot cho biến Memory_Speed theo Manufacturer, Dedicated, MemoryType

Dựa trên đồ thị, nhóm rút ra một số nhận xét về sự phân bố Memory Speed của GPU được phân loại theo nhà sản xuất, bao gồm AMD, ATI, Intel, và Nvidia: Các nhà sản xuất như AMD và Nvidia có phạm vi phân bố rộng, thể hiện danh mục sản phẩm đa dạng từ tầm thấp đến cao cấp. ATI tập trung vào các sản phẩm cao cấp với Memory Speed vượt trội. Intel có phân khúc



Hình 6: Biểu đồ Histogram cho Memory Speed



(a) Manufacturer vs Memory_Speed (b) Dedicated vs Memory_Speed

Hình 7: Box plot của Dedicated và Manufacturer

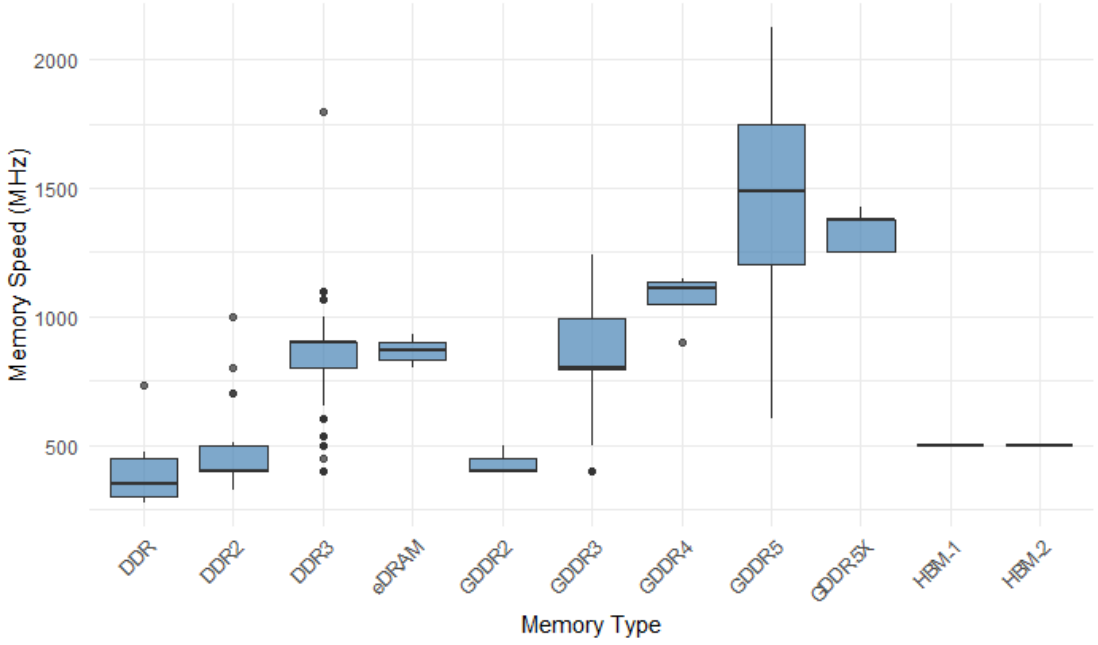
rõ ràng ở nhóm hiệu năng thấp, phù hợp với GPU tích hợp.

Non-Dedicated GPU (No): Phân bố Memory Speed của nhóm này rất hẹp, với giá trị chủ yếu nằm trong khoảng 1000 MHz. Điều này cho thấy GPU tích hợp có tốc độ bộ nhớ thấp và đồng nhất hơn, phù hợp với vai trò hỗ trợ cơ bản thay vì xử lý đồ họa chuyên sâu.

Dedicated GPU (Yes): Phân bố Memory Speed rộng hơn đáng kể, từ khoảng 500 MHz đến hơn 2000 MHz. Trung vị nằm ở mức 1500 MHz, cao hơn nhóm Non-Dedicated GPU, cho thấy GPU rời thường có hiệu năng cao hơn. Khoảng giá trị rộng thể hiện sự đa dạng trong phân khúc sản phẩm, từ các GPU tầm trung đến cao cấp.

GPU rời (Dedicated GPU) có tốc độ bộ nhớ cao hơn đáng kể so với GPU tích hợp, phù hợp với các nhu cầu xử lý đồ họa phức tạp và chuyên nghiệp. GPU tích hợp (Non-Dedicated GPU) có hiệu năng thấp hơn và ít biến động, phù hợp với các ứng dụng cơ bản như văn phòng, xem

phim, hoặc chơi game nhẹ.



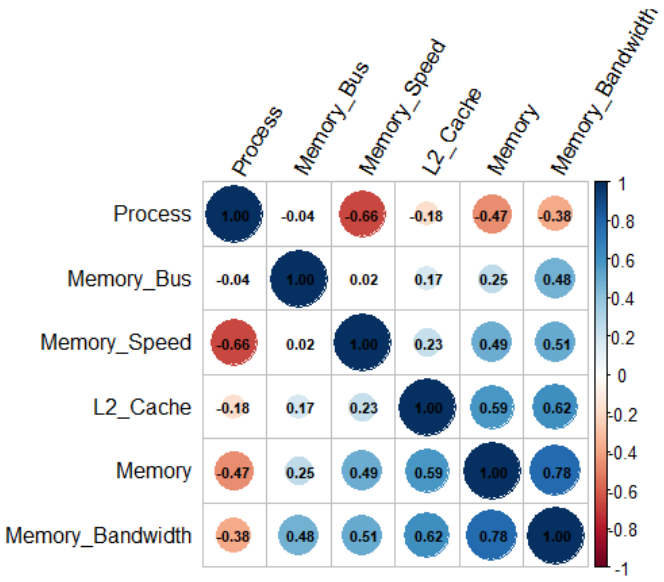
Hình 8: Boxplot của Memory Speed theo Memory Type

Dựa trên biểu đồ boxplot về Memory Speed (MHz) theo Memory Type, nhóm rút ra các nhận xét sau:

- Xu hướng tăng tốc độ theo thời gian: Các loại bộ nhớ đời sau (GDDR5, GDDR5X, HBM-1, HBM-2) có tốc độ cao hơn đáng kể so với các loại đời cũ (DDR, DDR2, DDR3). Điều này phản ánh sự tiến bộ trong công nghệ bộ nhớ.
- Phân tán dữ liệu: DDR, DDR2, DDR3: Có tốc độ thấp, dao động trong khoảng từ 500 MHz đến dưới 1000 MHz, với mức phân tán tương đối nhỏ.GDDR4, GDDR5: Có sự phân tán lớn hơn, đặc biệt GDDR5 với tốc độ trải dài từ khoảng 1000 MHz đến hơn 2000 MHz. GDDR5X: Tốc độ tập trung hơn ở mức cao, thể hiện hiệu suất ổn định.
- Ngoại lệ: Một số điểm dữ liệu ngoài hộp (outliers) xuất hiện ở các nhóm DDR2, DDR3, GDDR5, có thể đại diện cho các sản phẩm tối ưu hoặc không tiêu chuẩn.
- Bộ nhớ HBM (High Bandwidth Memory): HBM-1 và HBM-2 có tốc độ thấp hơn nhiều so với GDDR5X, nhưng HBM thường được tối ưu cho băng thông thay vì chỉ số MHz, vì vậy điều này không phản ánh trực tiếp hiệu năng.

Công nghệ bộ nhớ đã có sự cải tiến đáng kể theo thời gian, với tốc độ ngày càng cao ở các thế hệ mới. GDDR5 và GDDR5X hiện tại có ưu thế lớn về hiệu suất. Tuy nhiên, đặc điểm của từng loại bộ nhớ còn phụ thuộc vào mục tiêu sử dụng (băng thông, hiệu suất, hay tiết kiệm năng lượng).

5.4 Vẽ biểu đồ tương quan giữa các biến

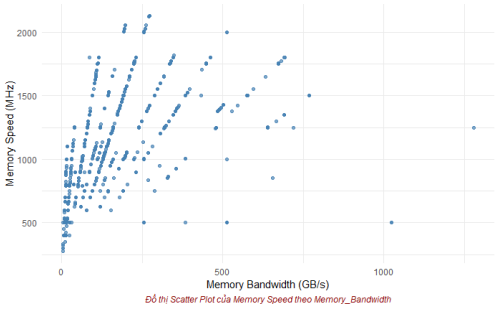


Hình 9: Biểu đồ tương quan

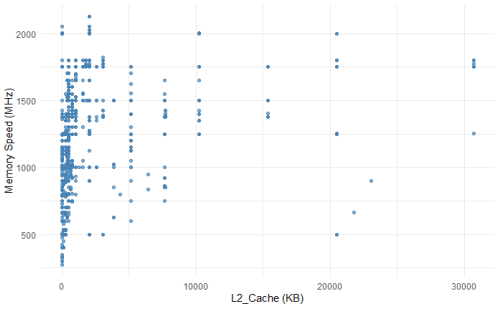
Dựa trên kết quả từ ma trận tương quan, ta có thể đưa ra nhận xét về mối quan hệ giữa Memory_Speed và các biến khác:

- Memory_Speed và Memory: Có mối tương quan dương trung bình với tương quan = 0.49. Điều này cho thấy rằng tốc độ bộ nhớ có sự phụ thuộc đáng kể vào dung lượng bộ nhớ. Điều này có nghĩa là khi dung lượng bộ nhớ tăng lên, tốc độ bộ nhớ cũng có thể tăng, và sự thay đổi này có thể được phản ánh trong các mô hình học máy hoặc phân tích dữ liệu.
- Memory_Speed và Memory_Bandwidth: Mối quan hệ giữa băng thông bộ nhớ và tốc độ bộ nhớ là rất quan trọng, vì tốc độ bộ nhớ và băng thông có liên quan mật thiết trong việc xử lý dữ liệu. Memory_Bandwidth có thể là yếu tố quan trọng ảnh hưởng đến Memory_Speed.
- Memory_Speed và Memory_Bus: Có mối tương quan rất yếu với tương quan = 0.02. Điều này cho thấy rằng độ rộng bus bộ nhớ không có ảnh hưởng đáng kể đến tốc độ bộ nhớ.
- Memory_Speed và L2_Cache: L2_Cache có thể ảnh hưởng đến tốc độ bộ nhớ, nhưng mối quan hệ này phụ thuộc vào nhiều yếu tố khác như kiến trúc và quy trình sản xuất. Nếu tương quan yếu hoặc không rõ ràng, có thể chỉ ra rằng bộ nhớ cache không phải yếu tố quyết định chính.
- Memory_Speed và Process:Mối quan hệ này có thể là nghịch, đặc biệt nếu quy trình sản xuất (Process) cũ hơn có thể làm giảm tốc độ bộ nhớ.

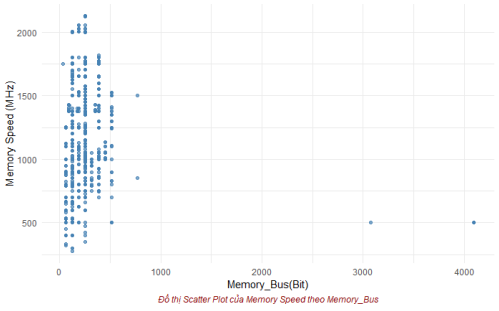
5.5 Đồ thị scatter plot cho biến Memory_Speed theo các biến "Memory_Bandwidth", "L2_Cache", "Memory_Bus", " Process", "Memory"



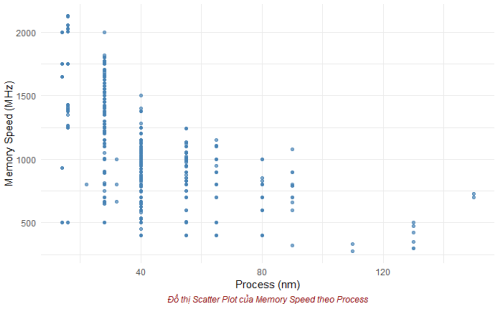
(a) Memory Speed theo Memory_Bandwidth



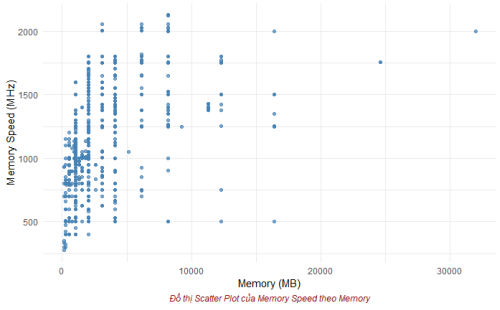
(b) Memory Speed theo L2_Cache



(c) Memory Speed theo Memory_Bus



(d) Memory Speed theo Process



(e) Memory Speed theo Memory

Hình 10: Các Scatter Plot

Dựa trên biểu đồ phân tán của biến Memory Speed theo các biến "Memory_Bandwidth", "L2_Cache", "Memory_Bus", "Process", "Memory", nhóm rút ra một số nhận xét:

Có một xu hướng chung chỉ ra rằng Memory Bandwidth cao hơn có liên quan đến Memory Speed cao hơn. Điều này có nghĩa là khi băng thông bộ nhớ tăng, tốc độ bộ nhớ cũng có xu hướng tăng.Có thể có một số giá trị ngoại lệ trong dữ liệu mà một số cấu hình bộ nhớ không tuân theo xu hướng chung. Việc xác định những giá trị này có thể cung cấp thông tin chi tiết về các trường hợp cụ thể mà hiệu suất bộ nhớ lệch khỏi chuẩn mực.

Có ít điểm dữ liệu hơn cho kích thước bộ nhớ đệm L2 lớn hơn. Các điểm này phân tán nhiều hơn về tốc độ bộ nhớ, cho thấy khi kích thước bộ nhớ đệm L2 tăng lên, tốc độ bộ nhớ thay đổi rộng hơn. Mặc dù biểu đồ cho thấy sự phân phối chung, nhưng nó không chỉ ra mối tương quan rõ ràng hoặc mạnh giữa kích thước bộ nhớ đệm L2 và tốc độ bộ nhớ. Tuy nhiên, nó cho thấy tồn tại nhiều cấu hình khác nhau với kích thước bộ nhớ đệm và tốc độ bộ nhớ khác nhau.

Mối quan hệ giữa tốc độ bộ nhớ và Bus bộ nhớ: Có một xu hướng chung cho thấy rằng các điểm dữ liệu tập trung ở các giá trị thấp của Memory Bus, tức là dưới 1000 Bits. Điều này cho thấy các hệ thống bộ nhớ có Bus bộ nhớ nhỏ hơn thường được sử dụng nhiều hơn.

Mối quan hệ giữa tốc độ bộ nhớ và kích thước quy trình: Có một xu hướng chung cho thấy rằng khi kích thước quy trình giảm (tức là quá trình sản xuất càng tiên tiến), tốc độ bộ nhớ có xu hướng tăng. Điều này cho thấy các công nghệ sản xuất tiên tiến hơn có thể mang lại hiệu suất bộ nhớ cao hơn. Phân bố dữ liệu: Các điểm dữ liệu phân bố khá đều trên đồ thị, cho thấy sự đa dạng trong kích thước quy trình.

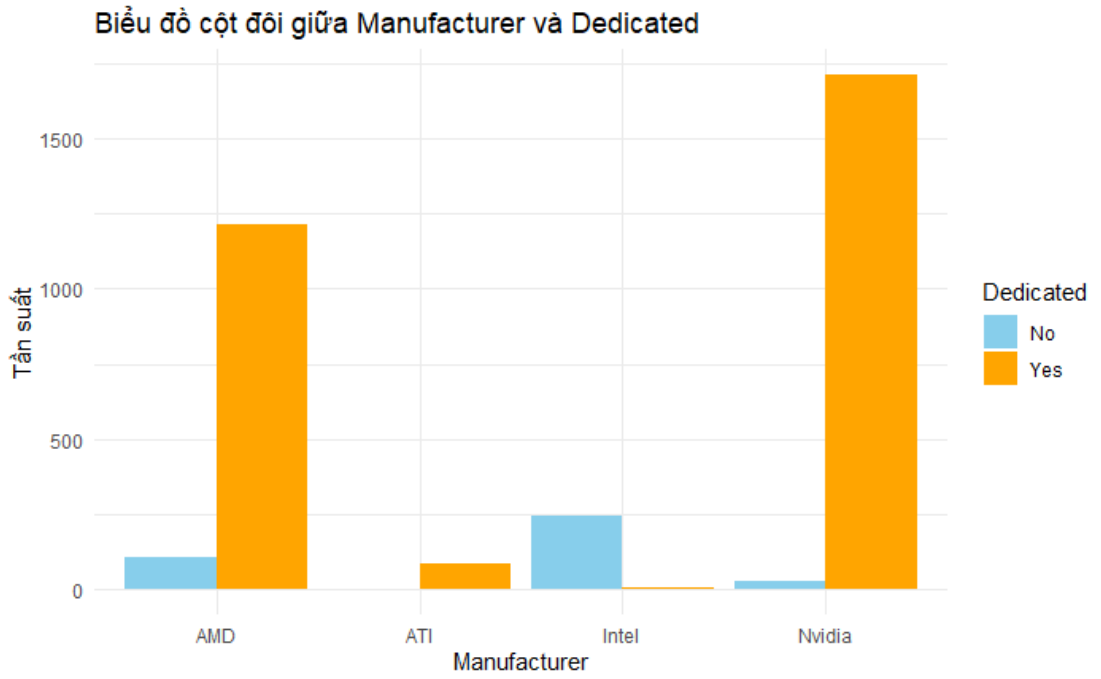
Đối với phần lớn dữ liệu, Memory Speed (MHz) không có xu hướng tăng rõ rệt khi Memory (MB) tăng. Vì vậy tốc độ bộ nhớ (Memory Speed) không tỷ lệ thuận với dung lượng bộ nhớ (Memory). Một số điểm có Memory lớn vẫn duy trì Memory Speed tương đối thấp (dưới 1,000 MHz). Memory Speed phân tán mạnh ở mức Memory nhỏ (dưới 10,000 MB), trong khi ở mức Memory cao, sự phân tán giảm và ít điểm dữ liệu hơn.

6 Xử lý dữ liệu

Nếu xử lý theo hướng xóa đi như ở Mục 1 ta sẽ gặp phải tình trạng mất dữ liệu khá nhiều vì nếu xóa ta chỉ còn 2637 records so với ban đầu là 3046 records nghĩa là ta đã mất đi gần 14% lượng dữ liệu. Điều này có thể dẫn đến việc Thống kê suy diễn thiếu chính xác cũng như lượng mất đi sẽ rất lãng phí. Vì thế nhóm chúng em đi đến quyết định dựa vào những nhận định trong phần Thống kê mô tả để có thể xử lý lượng dữ liệu bị khuyết, tránh mất mác. Đầu tiên ta sẽ xử lý sơ dữ liệu thô như mục trên đã làm.

6.1 Xử lý biến Dedicated

Khi quan sát dữ liệu ta sẽ thấy sự liên quan giữa Manufacturer và Dedicated vì thế ta sẽ tiến hành vẽ đồ thị tần suất của 2 biến.



Hình 11: Biểu đồ cột đôi giữa Manufacturer và Dedicated

Dựa vào biểu đồ trên ta nhận định rằng:

- Nếu Manufacturer là AMD, ATI, Nvidia thì Delicated sẽ là Yes
- Nếu Manufacturer là Intel thì Delicated sẽ là No Dựa vào nhận định trên ta sẽ tiến hành thay thế các dữ liệu na ở cột Delicated

```
processed_data$Dedicated[is.na(processed_data$Dedicated) &
    processed_data$Manufacturer %in% c("AMD", "ATI", "Nvidia")] <- "Yes"
processed_data$Dedicated[is.na(processed_data$Dedicated) &
    processed_data$Manufacturer == "Intel"] <- "No"
```

6.2 Xử lý biến Memory_Type

Ta nhận thấy rằng phần lớn bộ Memory_Type là DDR2 và DDR đều có L2_Cache = 0. SỐ lượng biến DDR và DDR2 cũng không chiếm qua nhiều (47 trong tổng số 3046) vì thế ta sẽ tiến hành gọp 2 biến đó thành một biến duy nhất là DDR1. Từ đây ta có thể nhận định rằng nếu L2_Cache = 0 thì Memory_Type sẽ là DDR1

Ngoài các loại có nhiều dữ liệu thì eDRAM, GDDR2, GDDR4, HBM-1 và HBM-2 có rất ít dữ liệu và chỉ còn khoản 10 records là còn na. Từ đây ta sẽ gọp các dữ liệu có ít và dữ liệu na thành một loại dữ liệu mới là Other.

Như ta thấy dữ liệu sau khi xử lý đã gọn hơn và gồm nhiều records hơn, điều này là rất quan trọng vì nó sẽ giúp cho các mô hình về sau thêm chính xác.

DDR	DDR2	DDR3	eDRAM	GDDR2	GDDR3	GDDR4	GDDR5	GDDR5X	HBM-1	HBM-2
7	40	359	2	3	177	4	1956	78	6	5

(a) Trước khi xử lý

DDR1	DDR3	DDR4	GDDR3	GDDR5	GDDR5X	Other
261	657	51	300	2020	79	38

(b) Sau khi xử lý

Hình 12: Dữ liệu trước và sau khi xử lý

6.3 Xử lý biến Architecture

Ở đây ta nhận thấy tên kiến trúc thường bắt đầu giống nhau và chỉ khác ở hậu tố ví dụ "Kepler GK107" và "Kepler GK104" vì thế ta sẽ tiến hành xóa đi hậu tố chỉ giữ lại tiền tố. Sau khi xóa đi hậu tố ta nhận thấy có rất nhiều kiến trúc chỉ có 1 từ và hậu tố được đính kèm theo ví dụ như RV380, RV410, ta sẽ tiến hành xử lý bằng cách xóa đi hậu tố và chỉ giữ lại chữ cái đầu tiên. Một số kiến trúc bắt đầu bằng 'R' cũng có rất nhiều records nên ta sẽ chỉ thay thế các kiến trúc có tần số bé hơn 10. Tiếp đến ta sẽ xử lý cụm kiến trúc bắt đầu bằng ký tự 'G', ở đây có các kiến trúc bắt đầu bằng 'G' nhưng có rất nhiều records nên ta sẽ chỉ chuyển đổi các records có tần số bé hơn 10. Ta sẽ tiến hành xử lý các kiến trúc còn lại như trên.

Tới đây cột dữ liệu về Architecture đã khá đẹp, ta sẽ tiến hành xử lý tới các giá trị na. Ta nhận thấy các giá trị na của cột Architecture đều có L2_Cache = 0 và dựa vào Manufacturer ta sẽ có nhận định như sau:

- Nếu Manufacturer là Intel: Arrandale
- Nếu Manufacturer là AMD: Evergreen
- Nếu Manufacturer là Nvidia: G

Sau khi đã xử lý các giá trị na ta sẽ đổi các giá trị có tần số thấp thành "Other"

Arrandale	Broadwell	Clarkdale	Evergreen	Fermi	G	G84M	G92
31	29	11	56	314	140	10	14
GCN	Haswell	Ivy	Kaveri	Kepler	M	Maxwell	N
783	22	22	18	480	34	350	85
Other	Pascal	R	R700	RV620	RV670	RV770	Sandy
119	233	136	34	12	10	10	50
Skylake	Terascale	Tesla					
56	252	95					

Hình 13: Kết quả sau khi xử lý biến Architecture

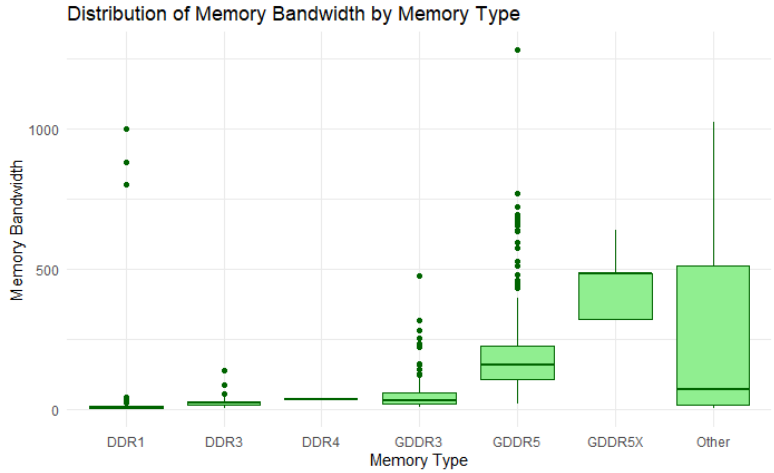
Kết quả này là rất đẹp khi so với tập dữ liệu gốc bởi tập dữ liệu gốc chứa rất rất nhiều các loại Architecture khác nhau, điều này sẽ làm giảm khả năng chính xác của các mô hình về sau.

6.4 Xử lý biến Memory_Bus và Memory_Speed

Ta nhận thấy rằng Memory Bus phụ thuộc vào Architecture vì thế ta sẽ lấy giá trị trung bình Memory Bus của Architecture để thay thế cho giá trị na. Nghĩa là ta sẽ lấy mean của Architecture A để thay vào Memory Bus đang có giá trị na và có Architecture là A. Tương tự như Memory Bus ta cũng sẽ tính toán Memory Speed thông qua trung bình của Memory_Type.

6.5 Xử lý biến Memory_Bandwidth

Biểu đồ hộp (Boxplot) để xem phân phối giữa Memory_Bandwidth và Memory_Type.



Hình 14: Phân bố của Memory Bandwidth bởi Memory Type

Dựa vào đây ta có thể nhận xét rằng giá trị giá trị của Memory Bandwith có thể biểu diễn cơ bản qua Memory Type nhưng ta sẽ bỏ qua các giá trị ngoại lai (outlier). Sau khi đã loại bỏ các giá trị ngoại lai ta làm tương tự như việc Xử lý biến Memory_Bus và Memory_Speed.

6.6 Xử lý biến Memory

Thay vì xử lý trên các biến khác thay thế bằng giá trị trung bình, ở biến Memory ta nhận thấy các giá trị memory tuân theo một số con số nhất định 128, 256, 512, ... Vì thế ở biến Memory ta sẽ thay thế bằng giá trị xuất hiện nhiều nhất ở Memory_Type tương ứng. Nhưng điều đặc biệt ở đây là toàn bộ nhóm Memory_Type là DDR4 đều không có dữ liệu Memory nên ta sẽ không thể tính toán Memory thông qua Memory_Type như chiến lược trên được. Tuy nhiên ta nhận thấy toàn bộ nhóm này đều có Memory_Bandwidth là 34.1 vì thế ta sẽ sử dụng giá trị mode của Memory_Bandwidth = 34.1 ứng với cột Memory để thay thế.

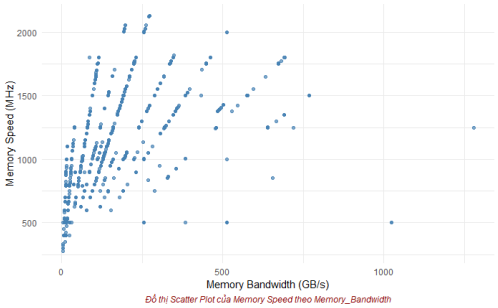
6.7 Xử lý biến Process

Tương tự như xử lý biến Memory, ta nhận thấy Process và Architecture có một mối liên hệ nào đó vì thế ta sẽ thay thế giá trị na của Process bằng mode của Architecture tương ứng. Tuy nhiên sẽ có một số giá trị vẫn chưa thể thay thế được tương tự như Memory ta cũng phải một thêm một giá trị khác nữa để có thể thay thế giá trị na. Ở đây ta cũng sử dụng biến

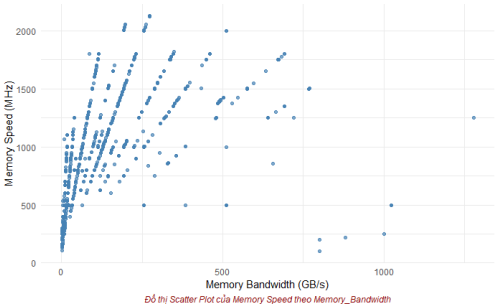
Memory_bandwidth bởi khi nhìn sang cột Memory_bandwidth ta sẽ thấy một số giá trị nhất định: 6.4, 7.0, 8.5, 10.7, 12.8, 17.1.

6.8 Kết luận

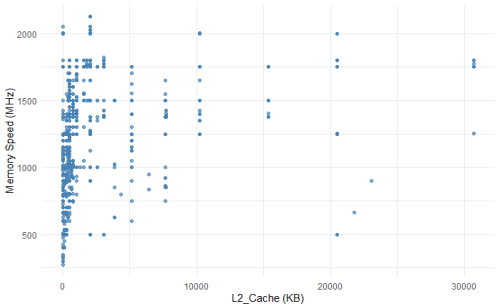
Sau khi đã thực hiện việc xử lý dữ liệu ta sẽ so sánh dữ liệu trước và sau khi xử lý.



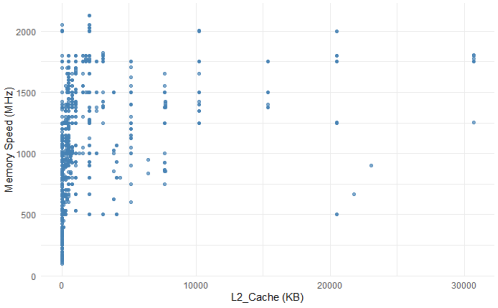
(a) Memory Speed theo Memory_Bandwidth trước xử lý



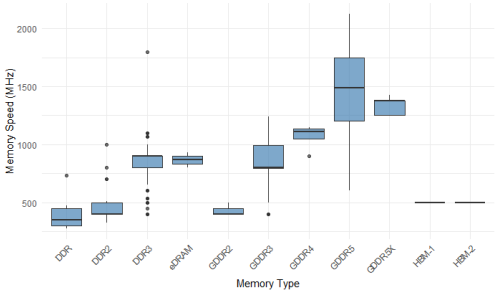
(b) Memory Speed theo Memory_Bandwidth sau xử lý



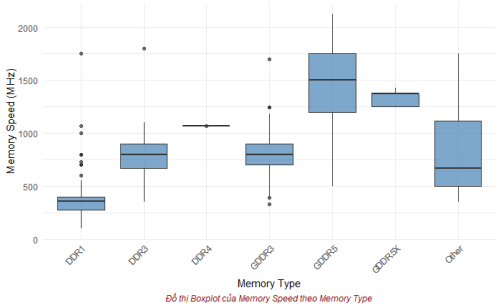
(c) Memory Speed theo L2_Cache



(d) Memory Speed theo L2_Cache sau khi xử lý



(e) Boxplot của Memory Speed theo Memory Type



(f) Boxplot của Memory Speed theo Memory Type sau khi xử lý

Hình 15: Một số biểu đồ so sánh sự khác nhau giữa 2 tập dữ liệu

Như ta thấy việc làm sạch dữ liệu lần nữa không mang nhiều ý nghĩa về mặc thống kê mô tả vì 2 tập dữ liệu phân dố gần như là tương đồng nhau. Tuy nhiên về mặc thống kê suy diễn

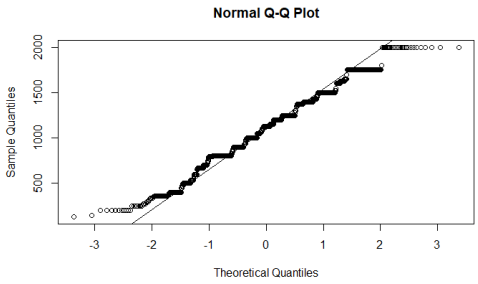
nó mang rất nhiều ý nghĩa vì tập dữ liệu đã nhiều hơn, các dữ liệu dễ gây nhiễu cho mô hình cũng đã được xử lý gọn gàn hơn tất cả những điều này sẽ giúp cho phần thống kê suy diễn có độ chính xác tăng lên không ít.

7 Thống kê suy diễn

7.1 Khoản tin cậy

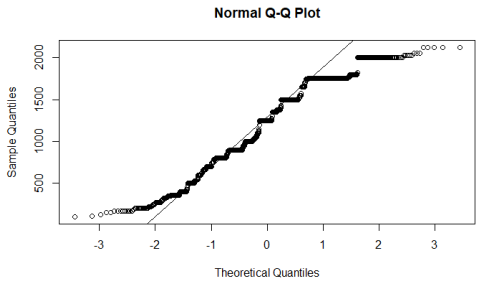
Tốc độ truy xuất bộ nhớ (Memory Speed) là một yếu tố quan trọng ảnh hưởng đến hiệu suất và độ ổn định của hệ thống máy tính. Việc kiểm tra tốc độ truy xuất bộ nhớ giúp đảm bảo rằng hệ thống không bị chậm hoặc gặp các vấn đề liên quan đến hiệu năng, điều này có thể ảnh hưởng đến trải nghiệm người dùng. Trong bài toán này, với độ tin cậy 95%, chúng ta sẽ thực hiện ước lượng trung bình giá trị của tốc độ truy xuất bộ nhớ trong một hệ thống máy tính. Điều này sẽ giúp xác định xem tốc độ bộ nhớ có đáp ứng các tiêu chuẩn về hiệu suất hay không, từ đó đảm bảo rằng hệ thống hoạt động mượt mà và hiệu quả. Với tình trạng hiện nay đa số các GPU đều đến từ 2 hãng là AMD và Nvidia vì thế ta sẽ tiến hành trên 2 hãng trên.

```r
AMD_data <- subset(processed_data ,processed_data$Manufacturer == "AMD")
qqnorm(AMD_data$Memory_Speed)
qqline(AMD_data$Memory_Speed)

Nvidia_data <- subset(processed_data ,processed_data$Manufacturer == "Nvidia")
qqnorm(Nvidia_data$Memory_Speed)
qqline(Nvidia_data$Memory_Speed)
```




```
Shapiro-Wilk normality test

data:  AMD_data$Memory_Speed
W = 0.98719, p-value = 2.331e-09
```

(a) AMD




```
Shapiro-Wilk normality test

data:  Nvidia_data$Memory_Speed
W = 0.9573, p-value < 2.2e-16
```

(b) Nvidia

Hình 16: Q-Q plot

Với độ tin cậy 95% và giả thiết:

- H_0: Memory_Speed của AMD/Nvidia tuân theo phân phối chuẩn
- H_1: Memory_Speed của AMD/Nvidia không tuân theo phân phối chuẩn.

Nhận xét

- Dựa vào đồ thị Q-Q Plot: ta thấy có nhiều điểm quan sát (records) lệch ra khỏi đường kỳ vọng phân phối chuẩn, nên ta dự đoán Memory_Speed của AMD và Nvidia đều không tuân theo phân phân phối chuẩn.
- Dựa vào kiểm định shapiro.test: Vì $p-value = 2.331e-09$ (AMD) và $p-value < 2.2e-16$ (Nvidia) nên bác bỏ H_0, chấp nhận H_1. Memory_Speed của AMD/Nvidia không tuân theo phân phối chuẩn.

Xác định khoản tin cậy Dựa vào kết quả trên đây là dạng bài ước lượng trung bình với giả thiết tổng thể có phân phối bất kỳ và cỡ mẫu lớn. Để xác định được khoản tin cậy ta cần tính các đặc trưng mẫu $z_{\alpha/2}$

```r
#AMD
n<-length(AMD_data$Memory_Speed)
xtb<-mean(AMD_data$Memory_Speed)
s<-sd(AMD_data$Memory_Speed)

z_critical <- qnorm(p=1-0.05/2,lower.tail=TRUE)
Epsilon <- qnorm(p=1-0.05/2)*s/sqrt(n)
Left_CI <- xtb-Epsilon
Right_CI <- xtb+Epsilon
data.frame(n,xtb,s,z_critical,Epsilon,Left_CI,Right_CI)

#Nvidia
n<-length(Nvidia_data$Memory_Speed)
xtb<-mean(Nvidia_data$Memory_Speed)
s<-sd(Nvidia_data$Memory_Speed)

z_critical <- qnorm(p=1-0.05/2,lower.tail=TRUE)
Epsilon <- qnorm(p=1-0.05/2)*s/sqrt(n)
Left_CI <- xtb-Epsilon
Right_CI <- xtb+Epsilon
data.frame(n,xtb,s,z_critical,Epsilon,Left_CI,Right_CI)
```

	n	xtb	s	z_critical	Epsilon	Left_CI	Right_CI
1	1317	1101.692	391.0011	1.959964	21.11705	1080.575	1122.81

(a) AMD

	n	xtb	s	z_critical	Epsilon	Left_CI	Right_CI
1	1743	1219.555	480.6176	1.959964	22.56312	1196.992	1242.118

(b) Nvidia

Hình 17: Khoản tin cậy

Nhận xét Dựa trên khoảng tin cậy 95% cho Memory_Speed trung bình của AMD/Nvidia lần lượt là: $(1080.575; 1122.81)$ cho AMD và $(1196.992; 1242.118)$ cho Nvidia. Điều nãy có nghĩa là với độ tin cậy 95% chúng ta có thể kết luận rằng với tốc độ trung bình này thua xa hiện nay là

khoảng $(2133; 2400)$ việc này chỉ ra rằng bộ dữ liệu của ta đã khá cũ so với thực tế và sẽ không đạt tiêu chuẩn hiệu năng cho việc vận hành các hệ thống máy tính hiện nay.

7.2 Kiểm định trung bình hai mẫu

Tốc độ truy xuất bộ nhớ (Memory Speed) là một chỉ số quan trọng phản ánh hiệu suất và khả năng xử lý dữ liệu của các hệ thống máy tính. Sự khác biệt về thiết kế phần cứng, công nghệ sản xuất và tối ưu hóa bộ nhớ giữa các hãng sản xuất có thể dẫn đến chênh lệch về tốc độ truy xuất bộ nhớ. Trong lĩnh vực này, hai thương hiệu lớn, AMD và Nvidia, thường được so sánh để đánh giá hiệu suất của bộ nhớ trong các hệ thống mà họ hỗ trợ.

Sự khác biệt về tốc độ truy xuất bộ nhớ không chỉ ảnh hưởng đến hiệu năng tổng thể của hệ thống mà còn đóng vai trò trong việc tối ưu hóa trải nghiệm người dùng, đặc biệt trong các tác vụ đòi hỏi hiệu suất cao như chơi game, thiết kế đồ họa, hoặc tính toán khoa học. Có những nghiên cứu cho rằng các sản phẩm của AMD và Nvidia có sự khác biệt đáng kể về tốc độ bộ nhớ, xuất phát từ các yếu tố như cấu trúc hệ thống, công nghệ hỗ trợ, và mục tiêu tối ưu hóa.

Với những tác động này, việc đánh giá và so sánh tốc độ truy xuất bộ nhớ giữa hai hãng AMD và Nvidia là rất cần thiết, nhằm làm rõ sự khác biệt về hiệu năng của các sản phẩm từ hai thương hiệu. Câu hỏi đặt ra là: Với mức ý nghĩa 5%, liệu tốc độ truy xuất bộ nhớ (Memory Speed) có sự khác biệt đáng kể giữa AMD và Nvidia hay không?

So sánh phương sai Với kết quả đã thực hiện ở phần trên ở 2 nhà sản xuất là AMD và Nvidia ta sẽ tiến hành so sánh phương sai tổng thể của biến Memory_Speed của 2 nhà sản xuất bằng kiểm định F-test với hàm var.test() với giả thuyết:

- H_0: $\sigma_1^2 = \sigma_2^2$
- H_0: $\sigma_1^2 \neq \sigma_2^2$

```r
var.test(AMD_data$Memory_Speed, Nvidia_data$Memory_Speed,alternative="greater")
```

```
        F test to compare two variances

data:  AMD_data$Memory_Speed and Nvidia_data$Memory_Speed
F = 0.66185, num df = 1316, denom df = 1742, p-value = 1
alternative hypothesis: true ratio of variances is greater than 1
95 percent confidence interval:
 0.6080988       Inf
sample estimates:
ratio of variances
         0.6618454
```

Hình 18: Kết quả F-test

Nhận xét Từ kết quả của F-test, ta thấy rằng giá trị $p - value = 1$ lớn hơn mức ý nghĩa 5% nên ta chưa thể bác bỏ H_0. Vậy phương sai về Memory Speed ở hai nhóm là bằng nhau.

Dựa vào các kết quả trên đây là bài toán kiểm định trung bình 2 mẫu trường hợp X_1, X_2 không có phân phối chuẩn với giả thiết $\sigma_1^2 = \sigma_2^2$. Ta sẽ tiến hành kiểm định trung bình với hàm t.test()

```r
t.test(AMD\_data$Memory_Speed, Nvidia_data$Memory\_Speed, var.equal=T)
```

```
        Two Sample t-test

data:  AMD_data$Memory_Speed and Nvidia_data$Memory_Speed
t = -7.2662, df = 3058, p-value = 4.671e-13
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -149.66740  -86.05836
sample estimates:
mean of x mean of y
 1101.692  1219.555
```

Hình 19: Kết quả kiểm định phương sai 2 mẫu

Vì $p - value = 4.67e - 13$ nhỏ hơn mức ý nghĩa 5% nên ta có thể bác bỏ giả thiết H_0. Vậy Memory Speed trung bình của hai nhà sản xuất là khác biệt đáng kể.

7.3 Hồi quy tuyến tính đơn

Hồi quy tuyến tính đơn biến là một phương pháp thống kê quan trọng được sử dụng để mô tả mối quan hệ giữa một biến phụ thuộc Y và một biến độc lập X. Trong bài toán này, dựa trên kết quả từ biểu đồ ma trận tương quan, chúng ta nhận thấy biến *process* có mối tương quan mạnh với biến phụ thuộc. Do đó, chúng ta sẽ lựa chọn biến *process* làm biến độc lập và biến *Memory_Speed* làm biến phụ thuộc để xây dựng mô hình hồi quy tuyến tính đơn.

Mô hình hồi quy tuyến tính đơn có dạng:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

trong đó:

- Y: Biến phụ thuộc (kết quả cần dự đoán).
- X: Biến độc lập (biến giải thích, ở đây là *process*).
- β_0: Hệ số chặn (intercept), biểu diễn giá trị trung bình của Y khi $X = 0$.
- β_1: Hệ số góc (slope), biểu thị mức thay đổi trung bình của Y khi X tăng lên 1 đơn vị.
- ϵ: Sai số ngẫu nhiên (random error), biểu diễn phần dao động của Y không giải thích được bởi X.

Mục tiêu của phương pháp này là ước lượng các tham số β_0 và β_1 sao cho đường hồi quy tối ưu nhất. Để đạt được điều này, phương pháp bình phương tối thiểu (Least Squares) được sử dụng, với hàm mục tiêu:

$$\text{Minimize } S = \sum_{i=1}^{n} (Y_i - (\beta_0 + \beta_1 X_i))^2$$

trong đó n là số quan sát.

Mô hình sẽ được kiểm định và đánh giá dựa trên các tiêu chí như:

- R^2: Hệ số xác định, đo lường mức độ phù hợp của mô hình.

- Kiểm định giả thuyết cho β_1: Để xác định xem biến độc lập có ảnh hưởng đáng kể đến biến phụ thuộc hay không.

```
one_linear_model <- lm(Memory_Speed ~ Process, data = processed_data)

summary(one_linear_model)
```

```
Call:
lm(formula = Memory_Speed ~ Process, data = processed_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1085.77  -259.93    -9.93   242.07  1489.82

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1548.1002    10.6347  145.57   <2e-16 ***
Process      -10.2917     0.2269  -45.37   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 360 on 3404 degrees of freedom
Multiple R-squared:  0.3768,    Adjusted R-squared:  0.3766
F-statistic:  2058 on 1 and 3404 DF,  p-value: < 2.2e-16
```

Hình 20: Kết quả hồi quy tuyến tính đơn

Ta sử dụng giả thiết không và giải thiết đối để kiểm định cho hệ số hồi quy β_1 từ đó xác định biến Process có ý nghĩa trong việc dự đoán Memory_Speed hay không.

- H_0: $\beta_1 = 0$
- H_1: $\beta_1 \neq 0$

Dựa vào kết quả cột Pr(>|t|) đại diện cho giá trị $p - value$ của biến Process $<$ 2.2e-16 bé hơn mức ý nghĩa 5% vì vậy ta có thể bác bỏ giả thuyết H_0 và có thể nói rằng hệ số β_1 có ý nghĩa trong việc dự đoán Memory_Speed. Dựa vào kết quả trên ta có thể viết lại rằng:

$$\textbf{Memory\_Speed} = \textbf{1548.1002} \times \textbf{Process} - 10.2917$$

Từ phương trình hồi quy trên, ta có thể ước lượng giá trị *Memory Speed* trong mẫu dữ liệu ứng với biến độc lập *Process*. Mỗi giá trị *Process* sẽ cho ra một giá trị ước lượng cho *Memory Speed*, đồng thời ta có thể tính được sai số của ước lượng này bằng công thức:

$$\epsilon_i = Y_i - \hat{Y}_i$$

Trong đó, Y_i là giá trị thực tế của *Memory Speed* tại điểm quan sát thứ i, và $\hat{Y}_i$ là giá trị ước lượng từ mô hình hồi quy.

Phương trình hồi quy là tiền đề để tính toán sai số và các thông số liên quan đến sai số như độ chính xác của mô hình. Sai số càng nhỏ thì mô hình hồi quy càng phù hợp với dữ liệu thực tế, ngược lại, sai số lớn sẽ chỉ ra rằng mô hình không mô phỏng chính xác mối quan hệ giữa các biến. Một yếu tố quan trọng trong việc đánh giá độ chính xác của mô hình là phân phối của sai số. Nếu sai số tuân theo phân phối chuẩn với kỳ vọng bằng 0 và phương sai không đổi, tức là:

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

thì mô hình hồi quy có thể được coi là chính xác và đáng tin cậy. Tuy nhiên, nếu sai số không tuân theo phân phối chuẩn, điều này có thể cho thấy rằng mô hình hồi quy không phù hợp với dữ liệu và cần phải điều chỉnh lại hoặc thử sử dụng các mô hình hồi quy khác để cải thiện kết quả.

Giá trị R^2 của mô hình là 0.3768, cho thấy rằng mô hình hồi quy này giải thích được khoảng 37.68% sự biến động của *Memory Speed*. Mặc dù giá trị này không quá cao, nhưng nó cũng cho thấy rằng *Process* đóng vai trò quan trọng trong việc dự đoán *Memory Speed*. Tuy nhiên, với giá trị này, một phần lớn sự biến động của *Memory Speed* vẫn chưa được giải thích, và có thể có những yếu tố khác ngoài *Process* ảnh hưởng đến *Memory Speed* mà mô hình chưa tính đến.

Giá trị Adjusted R^2 là 0.3766, tương đối gần với R^2, cho thấy rằng mô hình không bị ảnh hưởng quá nhiều bởi các biến vô nghĩa. Điều này cho thấy rằng các yếu tố trong mô hình có ý nghĩa và đóng góp vào việc giải thích biến động của *Memory Speed*.

Một chỉ số khác là Residual Standard Error là 360, cho thấy sai số của mô hình khá lớn so với giá trị trung bình của *Memory Speed*. Điều này có thể chỉ ra rằng mô hình chưa đủ chính xác để dự đoán chính xác *Memory Speed*, và có thể cần phải cải thiện thêm.

Kết luận Memory Speed có mối quan hệ tuyến tính với biến Process, nhưng sự ảnh hưởng của Process đến Memory Speed là khá thấp, với tỷ lệ giải thích chỉ khoảng 37.68%. Điều này cho thấy rằng ảnh hưởng của Process đến Memory Speed là có nhưng không mạnh, và có thể có những yếu tố khác ngoài Process ảnh hưởng đến Memory Speed mà mô hình chưa tính đến.

Để kiểm định tính thích hợp của mô hình hồi quy tuyến tính, ta thực hiện kiểm định giả thuyết như sau:

- $H_0 : \beta_1 = 0 \quad$ hoặc $\quad R^2 = 0 \quad$ (phương trình hồi quy không thích hợp)
- $H_1 : \beta_1 \neq 0 \quad$ hoặc $\quad R^2 \neq 0 \quad$ (phương trình hồi quy thích hợp)

Vì p-value $< 2.2 \times 10^{-16}$ ứng với kiểm định F, nhỏ hơn mức ý nghĩa 5%, nên ta bác bỏ giả thuyết H_0. Vậy phương trình hồi quy là thích hợp.

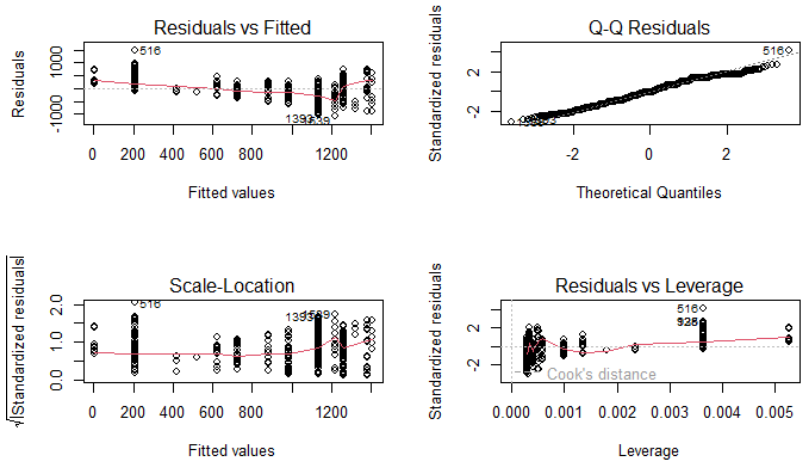
Các giả định của mô hình hồi quy tuyến tính đơn bao gồm:

- Mối quan hệ giữa biến độc lập X (Process) và biến phụ thuộc Y (Memory Speed) là tuyến tính, tức là:
$$E(Y|X = x) = \beta_0 + \beta_1 x$$
- Phương sai các sai số của biến phụ thuộc Y là hằng số với mọi giá trị của X.
- Các sai số $\epsilon_1, \epsilon_2, ..., \epsilon_n$ ứng với từng giá trị Y là độc lập với nhau.
- Với mỗi giá trị của biến độc lập X, phân phối có điều kiện của biến phụ thuộc Y là phân phối chuẩn, tức là:
$$Y|X = x \sim N(\beta_0 + \beta_1 x, \sigma^2)$$

Điều kiện này được xác minh qua việc kiểm tra phân phối của các sai số ϵ_i, và nếu các sai số tuân theo phân phối chuẩn $\epsilon_i \sim N(0, \sigma^2)$, mô hình sẽ được coi là hợp lý.

Phân tích thặng dư

```
par(mfrow = c(2, 2))
plot(one_linear_model)
```



Hình 21: Kết quả hồi quy tuyến tính đơn

Dựa trên đồ thị kiểm tra giả định, ta đưa ra được các nhận xét sau:

- **Đồ thị Residuals vs Fitted:** Đồ thị này vẽ các điểm sai số tương ứng với các giá trị dự báo. Mục đích của đồ thị này là kiểm tra giả định về mối quan hệ tuyến tính giữa biến phụ thuộc Y (Memory Speed) và biến độc lập X (Process), cũng như kiểm tra giả định phương sai của sai số là hằng số. Từ đồ thị này, ta thấy đường màu đỏ gần như là đường thẳng ngang, điều này chứng tỏ giả định mối quan hệ tuyến tính giữa các biến Y và X là thỏa mãn. Tuy nhiên, các sai số không phân tán ngẫu nhiên quanh đường màu đỏ, mà có sự tụ tập theo cụm, điều này chỉ ra rằng giả định phương sai sai số là hằng số bị vi phạm.
- **Đồ thị Q-Q Residuals:** Đồ thị này vẽ các sai số đã chuẩn hóa và dùng để kiểm tra giả định sai số có phân phối chuẩn. Dựa vào đồ thị Q-Q, ta thấy có một số điểm lệch rõ rệt khỏi đường thẳng kỳ vọng của phân phối chuẩn, đặc biệt là ở hai đầu đồ thị. Điều này cho thấy giả định sai số có phân phối chuẩn không được thỏa mãn.
- **Đồ thị Scale-Location:** Đồ thị này vẽ căn bậc hai các sai số chuẩn hóa và được dùng để kiểm tra giả định phương sai sai số là hằng số. Từ đồ thị này, ta nhận thấy các điểm sai số không phân tán đều dọc theo đường màu đỏ, mà thay vào đó, có sự phân tán không đồng đều. Do đó, ta kết luận rằng giả định phương sai sai số không phải là hằng số, điều này chỉ ra một sự vi phạm của giả định này.
- **Đồ thị Residuals vs Leverage:** Đồ thị này giúp chỉ ra các quan trắc có ảnh hưởng cao đến mô hình hồi quy, ví dụ như các quan trắc 5160 và 528. Đồ thị này không dùng để kiểm tra điều kiện giả định, mà chủ yếu là phát hiện các điểm ngoại lai. Dựa vào đồ thị, ta thấy có một số quan trắc vượt ra ngoài vùng Cook's distance, ví dụ như quan trắc 5160, điều này cho thấy các quan trắc này là ngoại lai và có thể ảnh hưởng lớn đến kết quả của mô hình hồi quy. Các quan trắc này cần phải được xem xét kỹ lưỡng và có thể phải loại bỏ khi phân tích.

8 Thảo luận và mở rộng

8.1 Thảo luận

Phân tích dữ liệu về GPU đã mang lại nhiều kết quả quan trọng, giúp làm rõ các yếu tố ảnh hưởng đến *Memory_ Speed* và hiệu năng của GPU. Dưới đây là các nhận xét chính:

- **Thống kê mô tả:**
  - Các GPU sử dụng bộ nhớ GDDR5 và GDDR5X cho thấy hiệu suất cao hơn đáng kể so với các loại bộ nhớ cũ như DDR hoặc DDR2.
  - Sự khác biệt rõ rệt về *Memory_ Speed* giữa các nhà sản xuất như AMD, Nvidia và Intel thể hiện sự đa dạng trong thiết kế và phân khúc thị trường.
  - Một số giá trị ngoại lai xuất hiện, đặc biệt ở các biến như *Memory_ Bandwidth* và *Memory_ Speed*, làm tăng nguy cơ sai lệch kết quả nếu không được xử lý cẩn thận.
- **Thống kê suy diễn:**
  - Kiểm định trung bình cho thấy sự khác biệt đáng kể giữa *Memory_ Speed* của AMD và Nvidia, phản ánh sự phân hóa về công nghệ giữa hai nhà sản xuất hàng đầu.
  - Khoảng tin cậy 95% của *Memory_ Speed* trung bình thấp hơn tiêu chuẩn hiện tại, chỉ ra rằng dữ liệu đã lỗi thời, chưa phản ánh được các GPU hiện đại.
- **Hồi quy tuyến tính:**
  - Mối quan hệ giữa *Memory_ Speed* và *Process* được xác định thông qua mô hình hồi quy tuyến tính đơn, với *Process* giải thích được 37.68% sự biến thiên. Tuy nhiên, còn nhiều yếu tố khác chưa được xem xét trong mô hình.
  - Kiểm tra thặng dư cho thấy sự vi phạm một số giả định, như phương sai không đồng nhất và sự tồn tại của điểm ngoại lai, cần cải thiện thêm để tăng độ chính xác.

Định hướng tương lai

- Cần thu thập và bổ sung các quan sát mới từ các GPU hiện đại để đảm bảo tính cập nhật của phân tích.
- Áp dụng các phương pháp hồi quy nâng cao như *Ridge Regression* hoặc *Lasso Regression* nhằm xử lý tốt hơn vấn đề ngoại lai và cải thiện khả năng giải thích của mô hình.
- Phân tích phi tuyến hoặc mô hình học máy có thể được xem xét để khai thác tối đa mối quan hệ phức tạp giữa các biến.

8.2 Mở rộng: Hồi quy tuyến tính đa biến

Hồi quy tuyến tính đa biến là một phương pháp thống kê được sử dụng để mô hình hóa mối quan hệ giữa một biến phụ thuộc (đầu ra) và nhiều biến độc lập (đầu vào). Phương pháp này mở rộng hồi quy tuyến tính đơn biến bằng cách sử dụng nhiều biến giải thích, cho phép phân tích sâu hơn về ảnh hưởng của các yếu tố khác nhau đến biến mục tiêu. Phương trình hồi quy tuyến tính đa biến có dạng:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Trong đó:

- y là biến phụ thuộc.
- $x_1, x_2, \ldots, x_n$ là các biến độc lập.
- $\beta_0, \beta_1, \ldots, \beta_n$ là các tham số cần ước lượng.
- ϵ là sai số ngẫu nhiên.

Hồi quy tuyến tính đa biến được ứng dụng rộng rãi trong các lĩnh vực như kinh tế, y học, và khoa học xã hội để dự đoán, phân tích mối quan hệ và đưa ra các quyết định dựa trên dữ liệu.

```r
multiple_linear_model <- lm(Memory_Speed ~ Memory + Memory_Bandwidth +
    Memory_Type + Memory_Bus + L2_Cache + Architecture + Process +
    Dedicated + Manufacturer, data = processed_data)

summary(multiple_linear_model)

par(mfrow = c(2, 2))
plot(multiple_linear_model)
```

```
Call:
lm(formula = Memory_Speed ~ Memory + Memory_Bandwidth + Memory_Type +
    Memory_Bus + L2_Cache + Architecture + Process + Dedicated +
    Manufacturer, data = processed_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1036.42   -95.16    -0.76    99.18  1237.80

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         5.325e+02  5.716e+01   9.316  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 165.3 on 3364 degrees of freedom
Multiple R-squared:  0.8701,    Adjusted R-squared:  0.8686
F-statistic: 549.8 on 41 and 3364 DF,  p-value: < 2.2e-16
```

Hình 22: Kết quả mô hình hồi quy đa biến

Bên cạnh kết quả trên còn rất nhiều kết quả của các hệ của β_i khác mà khó có thể thể hiện hết trong báo cáo, có thể xem qua tại code được đính kèm phía dưới.

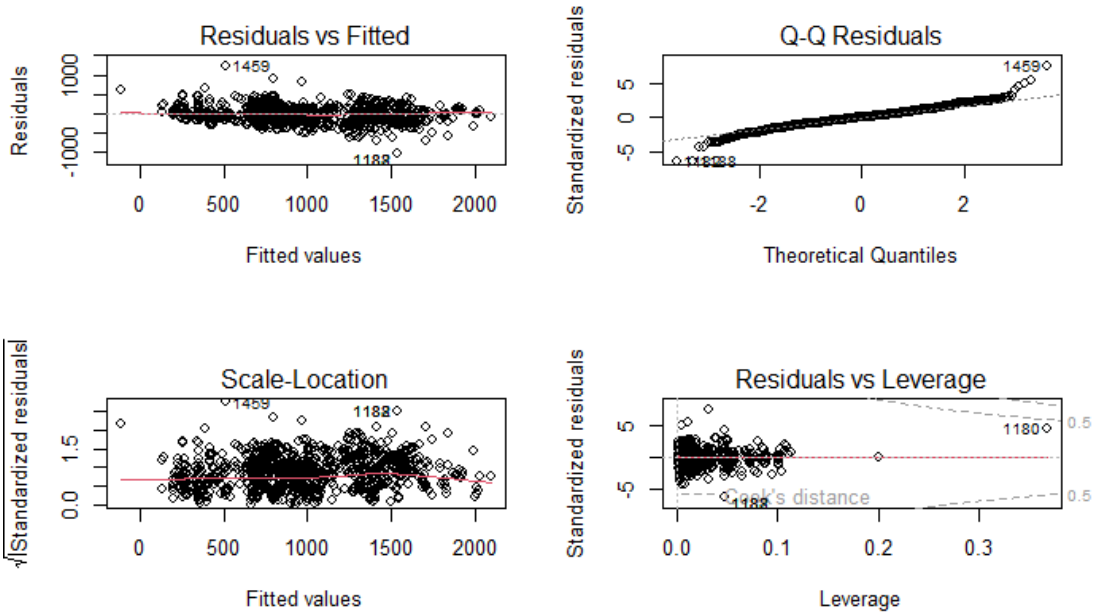
Nhận xét về mô hình hồi quy tuyến tính đa biến Mô hình hồi quy tuyến tính đa biến sử dụng nhiều biến độc lập để giải thích biến phụ thuộc **Memory_Speed**. Dưới đây là các nhận xét chi tiết:

- **Độ phù hợp của mô hình:**
  - Hệ số xác định $R^2 = 0.8701$ và hệ số xác định hiệu chỉnh $R^2_{adj} = 0.8686$, cho thấy mô hình giải thích được 87.01% sự biến động của tốc độ bộ nhớ (**Memory_Speed**). Đây là mức độ phù hợp rất cao.
  - Giá trị thống kê F ($F = 549.8$) với $p < 2.2 \times 10^{-16}$ cho thấy mô hình có ý nghĩa thống kê tổng thể rất cao.
- **Ý nghĩa của các biến trong mô hình:**
  - **Memory_Bandwidth:** Hệ số 1.384 ± 0.053, có ý nghĩa thống kê ($p < 2.2 \times 10^{-16}$), cho thấy băng thông bộ nhớ có tác động tích cực mạnh mẽ đến tốc độ bộ nhớ.
  - **Memory_Type:** Các loại **DDR3**, **DDR4**, **DDR5**, và **GDDR5X** đều có hệ số lớn và ý nghĩa thống kê cao ($p < 2.2 \times 10^{-16}$), khẳng định loại bộ nhớ ảnh hưởng đáng kể đến tốc độ.

  - **Memory_Bus:** Hệ số âm (-0.341 ± 0.016), có ý nghĩa thống kê ($p < 2.2 \times 10^{-16}$), cho thấy kích thước bus lớn có thể làm giảm tốc độ bộ nhớ.
  - Một số kiến trúc (**Maxwell**, **Pascal**, **Skylake**) và nhà sản xuất (**ATI**) cũng có tác động đáng kể đến tốc độ bộ nhớ.

- **Giá trị phần dư:**
  - Phần dư phân bố xung quanh giá trị trung bình là 0, nằm trong khoảng từ -1836.42 đến 1237.80.
  - Sai số chuẩn của phần dư là 165.3, cho thấy mức độ chênh lệch giữa giá trị thực và giá trị dự báo tương đối thấp.

- **Kết luận:**
  - Mô hình hồi quy tuyến tính đa biến giải thích tốt hơn so với mô hình đơn biến nhờ đưa vào nhiều yếu tố quan trọng.
  - Cần kiểm tra thêm các giả định hồi quy (như phân phối chuẩn của phần dư, phương sai đồng nhất) để đảm bảo mô hình đáng tin cậy.



Hình 23: Kết quả mô hình hồi quy đa biến

Phân tích thặng dư

1. Residuals vs Fitted (Phần dư so với giá trị dự đoán)

- **Ý nghĩa:** Biểu đồ này kiểm tra xem có mối quan hệ phi tuyến nào giữa các phần dư và giá trị dự đoán hay không.

- **Phân tích:**
  - Nếu các điểm phân bố ngẫu nhiên quanh đường 0, điều này gợi ý rằng mô hình phù hợp với dữ liệu.
  - Biểu đồ cho thấy có một số điểm nổi bật (`1459`, `1180`) và xu hướng cụm, điều này có thể cho thấy sự phi tuyến tính hoặc các biến bị bỏ sót.

2. Q-Q Plot (Phân phối chuẩn của phần dư)

- **Ý nghĩa:** Đánh giá xem phần dư có phân phối chuẩn hay không.
- **Phân tích:**
  - Đường thẳng trong biểu đồ biểu thị lý thuyết của phân phối chuẩn. Các điểm lệch ra khỏi đường thẳng ở các đầu (`1459`) cho thấy phần dư có thể không hoàn toàn chuẩn.
  - Tuy nhiên, phần lớn các điểm gần sát đường thẳng, cho thấy giả định về phân phối chuẩn khá hợp lý.

3. Scale-Location (Phần dư chuẩn hóa)

- **Ý nghĩa:** Kiểm tra tính đồng nhất phương sai (*homoscedasticity*).
- **Phân tích:**
  - Biểu đồ cho thấy độ phân tán không hoàn toàn đều, đặc biệt ở các giá trị dự đoán lớn, điều này có thể ám chỉ phương sai không đồng nhất.
  - Điều này gợi ý cần xem xét lại mô hình hoặc áp dụng các phương pháp sửa lỗi (ví dụ: chuyển đổi biến).

4. Residuals vs Leverage (Phần dư so với ảnh hưởng của điểm dữ liệu)

- **Ý nghĩa:** Xác định các điểm dữ liệu có ảnh hưởng mạnh đến mô hình.
- **Phân tích:**
  - Điểm `1180` và `1459` có leverage lớn, có thể là các điểm bất thường (*outliers*).
  - Nếu các điểm này có *Cook's distance* lớn, cần kiểm tra thêm vì chúng có thể ảnh hưởng nhiều đến kết quả hồi quy.

8.3 So sánh giữa mô hình hồi quy tuyến tính đơn biến và đa biến

Đánh giá chung của hai mô hình

- **Hồi quy đơn biến:**
  - Chỉ sử dụng một biến độc lập (**Process**) để giải thích biến phụ thuộc (**Memory_Speed**).
  - Hệ số xác định $R^2 = 0.3768$ cho thấy chỉ 37.68% sự biến động của **Memory_Speed** được giải thích bởi biến **Process**.
  - Mô hình chưa đủ tốt để phản ánh mối quan hệ đầy đủ giữa các biến.
- **Hồi quy đa biến:**

- Sử dụng nhiều biến độc lập (bao gồm **Memory**, **Memory_Bandwidth**, **Memory_Type**, **Process**, ...) để giải thích **Memory_Speed**.
- Hệ số xác định $R^2 = 0.8701$, cao hơn rất nhiều so với mô hình đơn biến, cho thấy mô hình phù hợp tốt với dữ liệu.

Kết luận: Mô hình đa biến cung cấp mức độ giải thích vượt trội hơn rất nhiều so với mô hình đơn biến.

So sánh ý nghĩa của biến Process

- **Trong mô hình đơn biến:**
  - Biến **Process** có ý nghĩa thống kê rất cao ($p < 2.2 \times 10^{-16}$).
  - Tuy nhiên, nó chỉ giải thích được một phần nhỏ của sự biến động trong tốc độ bộ nhớ ($R^2 = 0.3768$).
- **Trong mô hình đa biến:**
  - Biến **Process** vẫn có ý nghĩa thống kê ($p = 1.47 \times 10^{-14}$), nhưng hệ số -2.527×10^{-2} nhỏ hơn đáng kể.
  - Vai trò của **Process** giảm đi khi các biến khác (như **Memory_Bandwidth**, **Memory_Type**, ...) được thêm vào.

Kết luận: Mặc dù **Process** quan trọng, nó không đủ để mô tả toàn bộ mối quan hệ và cần được bổ sung bởi các biến khác trong mô hình đa biến.

Ý nghĩa của các biến khác trong mô hình đa biến

- Các biến như **Memory_Bandwidth**, **Memory_Type**, và **Memory_Bus** có ý nghĩa thống kê rất cao ($p < 2.2 \times 10^{-16}$):
  - **Memory_Bandwidth:** Hệ số 1.384, cho thấy băng thông bộ nhớ càng cao, tốc độ bộ nhớ càng tăng.
  - **Memory_Type:** Các loại **DDR3**, **DDR4**, và **DDR5** đều có tác động tích cực đáng kể.
  - **Memory_Bus:** Hệ số âm (-3.411×10^{-1}), chỉ ra rằng tăng kích thước bus có thể làm giảm tốc độ bộ nhớ.
- Một số kiến trúc (**Maxwell**, **Pascal**, **Skylake**) và nhà sản xuất (**ATI**) cũng có ảnh hưởng đáng kể.

Kết luận: Mô hình đa biến làm rõ tác động của các yếu tố khác ngoài **Process**, mang đến cái nhìn toàn diện và chính xác hơn.

So sánh tổng quát về chất lượng mô hình

Tiêu chí	**Mô hình đơn biến**	**Mô hình đa biến**
R^2	0.3768	0.8701
Số biến giải thích	1 (**Process**)	41
Độ phù hợp của mô hình	Thấp	Cao
Ý nghĩa thống kê tổng thể	Có ($p < 2.2 \times 10^{-16}$)	Có ($p < 2.2 \times 10^{-16}$)
Đóng góp của các biến	Chỉ từ **Process**	Từ nhiều yếu tố khác nhau

Kết luận tổng quát

- **Mô hình đơn biến:** Là bước khởi đầu tốt để kiểm tra mối quan hệ cơ bản giữa **Process** và **Memory_Speed**, nhưng chỉ cung cấp góc nhìn hạn chế.
- **Mô hình đa biến:** Cung cấp cái nhìn toàn diện hơn bằng cách đưa vào nhiều yếu tố kỹ thuật và kiến trúc liên quan, phù hợp hơn để phân tích và dự đoán tốc độ bộ nhớ.

9 Code

Code được đính kèm tại link: Github

Tài liệu tham khảo

[1] Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media.

[2] James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer.

[3] Hoerl, A. E., Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1), 55–67.

[4] Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), 267–288.

[5] Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer.

[6] NVIDIA. (2021). GPU Memory Technologies: An Overview of Memory Speed and Bandwidth.

[7] HP. (2020). Does RAM Speed Matter? Available at: https://www.hp.com/us-en/shop/tech-takes/does-ram-speed-matter.

[8] R Documentation. *glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models*. Available at: https://cran.r-project.org/web/packages/glmnet/index.html.