3. Tiền xử lý dữ liệu

3.1. Đọc dữ liệu

Đọc dữ liệu từ file Intel_CPUs.csv bằng hàm *read.csv()* và gán các giá trị chuỗi rỗng hay "N/A" thành NA bằng cách sử dụng tham số na.string(). Đồng thời chọn các cột cần sử dụng và in ra các thống kê sơ bộ.

```
# ĐỌC SỐ LIỆU####################
# đọc dữ liệu và gán các giá trị chuỗi rỗng và "N/A" thành NA
Intel_CPUs <- read.csv("Intel_CPUs.csv",na.strings = c("", "N/A"))
# Lựa chọn các cột cần sử dụng
CPUs_data <- Intel_CPUs[,c("Product_Collection","Vertical_Segment","Status","Launch_Date",
                           "Lithography","Recommended_Customer_Price","nb_of_Cores","nb_of_Threads",
                           "Processor_Base_Frequency","Cache","Instruction_Set","TDP","Max_Memory_Size",
                           "Max_nb_of_Memory_Channels","Max_Memory_Bandwidth")]
# in ra bảng thống kê sơ bộ của dữ liệu
summary(CPUs_data)
```

```
 Product_Collection Vertical_Segment     Status          Launch_Date        Lithography
 Length:2283        Length:2283       Length:2283        Length:2283        Length:2283
 Class :character   Class :character  Class :character   Class :character   Class :character
 Mode  :character   Mode  :character  Mode  :character   Mode  :character   Mode  :character


 Recommended_Customer_Price  nb_of_Cores     nb_of_Threads    Processor_Base_Frequency    Cache
 Length:2283                Min.   : 1.000   Min.   : 1.000   Length:2283              Length:2283
 Class :character           1st Qu.: 1.000   1st Qu.: 4.000   Class :character         Class :character
 Mode  :character           Median : 2.000   Median : 4.000   Mode  :character         Mode  :character
                            Mean   : 4.067   Mean   : 8.728
                            3rd Qu.: 4.000   3rd Qu.: 8.000
                            Max.   :72.000   Max.   :56.000
                                             NA's   :856
 Instruction_Set        TDP            Max_Memory_Size    Max_nb_of_Memory_Channels Max_Memory_Bandwidth
 Length:2283        Length:2283        Length:2283        Min.   : 1.000            Length:2283
 Class :character   Class :character   Class :character   1st Qu.: 2.000            Class :character
 Mode  :character   Mode  :character   Mode  :character   Median : 2.000            Mode  :character
                                                           Mean   : 2.615
                                                           3rd Qu.: 3.000
                                                           Max.   :16.000
                                                           NA's   :869
```

3.2. Xử lý dữ liệu khuyết

```
apply(is.na(CPUs_data),2,sum)   # kiem tra so luong nhung du lieu khuyet
      Product_Collection          Vertical_Segment                   Status              Launch_Date
                       0                         0                        0                      412
            Lithography Recommended_Customer_Price             nb_of_Cores            nb_of_Threads
                      71                       982                        0                      856
Processor_Base_Frequency                     Cache          Instruction_Set                      TDP
                      18                        12                      141                       67
        Max_Memory_Size  Max_nb_of_Memory_Channels     Max_Memory_Bandwidth
                     880                       869                     1136
```

Nhận thấy nhiều cột bị khuyết dữ liệu, như Launch_Date, Lithography, Recommended Customer Price, Processor Base Frequency, Instruction Set, Max_nb_of_Memory, Max_Memory_Bandwidth. Ta có thể lấp vào đó nhưng dữ liệu mới bằng cách sự dụng mean (giá trị trung bình), median (trung vị) hoặc tìm hiểu các mối quan hệ để điền vào.

- Launch_Data

Sử dụng hàm *substr()* để trích xuất chuỗi con từ chuỗi các ký tự, nchar() để tích độ dài của chuỗi. Ở đây ta trích xuất hai ký tự cuối cùng trong dãy. Chuyển chúng về năm, dữ liệu được thu thập vào năm 2022 nên mọi giá trị lớn hơn 22 sẽ được xem là được phát triển vào thập niên 20

```
# LAUNCH DATE ##################
# bỏ những dòng không có dữ liệu
CPUs_data <- CPUs_data[complete.cases(CPUs_data$Launch_Date), ]

# tách bỏ quý
CPUs_data$Launch_Date <- substr(CPUs_data$Launch_Date,nchar(CPUs_data$Launch_Date)-1,nchar(CPUs_data$Launch_Date))

CPUs_data$Launch_Date <- as.integer(CPUs_data$Launch_Date)
# biến đổi về năm (dữ liệu được thu thập vào năm 2022)
CPUs_data$Launch_Date <- ifelse(CPUs_data$Launch_Date>22,1900+CPUs_data$Launch_Date,2000+CPUs_data$Launch_Date)
# sắp xếp lại dataframe theo trật tự ưu tiên sau: năm, loại CPU, loại phân khúc
CPUs_data <- CPUs_data[order(CPUs_data$Launch_Date,CPUs_data$Product_Collection,CPUs_data$Vertical_Segment), ]
```

- Lithography

Lithography giảm dần theo từng năm cho các loại CPU hay phân khúc vì sự tiến bộ của công nghệ nên có thể điền dữ liệu thiếu bằng dữ liệu của năm trước. Sử dụng hàm *na.locf()* để điền vào những ô có giá trị còn thiếu bằng giá trị quan sát được ngay trước nó. Chuyển giá trị của Lithography về dạng định lượng của số thực và cắt đi phần đơn vị.

```
# LITHOGRAPHY ####################
CPUs_data$Lithography<- na.locf(CPUs_data$Lithography)
# Last Observation Carried Forward (filling in any missing values in that column with the last observed value.)
CPUs_data$Lithography <-as.double( gsub(" nm$", "", CPUs_data$Lithography)) # chuyển định dạng thành số thực
```

- Max memory size

Dữ liệu vào những năm nhỏ hơn 2009 mất đồng loạt nên việc lấp đầy là không khả thi. Ta bỏ tất cả những ô dữ liệu bị khuyết. Chuyển đổi giá trị của các ô có đơn vị TB thành GB bằng cách nhân cho 1024. Sau đó bỏ đơn vị và chuyển các gái trị của cột về dạng định lượng

```
# MAX MEMORY SIZE ####################
CPUs_data <- CPUs_data[complete.cases(CPUs_data$Max_Memory_Size), ] # loại bỏ ô có dữ liệu bị khuyết
Mem_size_func <- function(size){
  if(grepl('G',size)){
    return ( as.double(gsub(" GB","",size)) )
  }
  return ( as.double(gsub(" TB","",size)) * 1024 )
}
# Xử lý từng ô dữ liệu: chuyển TB thành GB và đổi về cùng dạng định lượng
CPUs_data$Max_Memory_Size <- sapply(CPUs_data$Max_Memory_Size,Mem_size_func)
```

- Number of threads

```
ratio <- as.double(CPUs_data$nb_of_Threads/CPUs_data$nb_of_Cores)
summary(ratio)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   1.00    1.00    2.00    1.69    2.00    2.00      17
```

Dựa vào dữ liệu thu được, ta có trung bị của tỉ lệ giữa số lượng nhân và luồng là 2 có nghĩa rằng một nhân thường có xác suất cao sẽ có 2 luồng đi với nó.

```
> (correlation <- cor(CPUs_data$nb_of_Threads, CPUs_data$nb_of_Cores, method = "pearson"))
[1] 0.9926145
```

Sử dụng phương pháp thống kê như hệ số tương quan Pearson ta nhận được hệ số tương quan cao.

Vì dữ liệu cột nb_of_Cores không bị mất và do có độ tương quang cao nên có thể dễ dàng điền vào nb_of_Threads bằng cách nhân 2 lần số nb_of_Cores.

```
# NUMBER OF THREAD  ####################
ratio <- as.double(CPUs_data$nb_of_Threads/CPUs_data$nb_of_Cores)
summary(ratio)
CPUs_data$nb_of_Threads <- ifelse(is.na(CPUs_data$nb_of_Threads), CPUs_data$nb_of_Cores * 2, CPUs_data$nb_of_Threads)
```

- Max memory bandwidth

```
> (correlation <- cor(CPUs_data$Max_Memory_Bandwidth, CPUs_data$Max_nb_of_Memory_Channels, method = "pearson"))
[1] 0.9675164
```

Sử dụng phương pháp thống kê như hệ số tương quan Pearson ta nhận được hệ số tương quan cao. Thế nênk hi tăng số lượng kênh tối đa sẽ dẫn đến tăng bandwitdth, vì vậy ta sẽ điền dữ liệu theo median của Max_Memory_Bandwidth theo Max_nb_of_Memory_Channels

```
## ---MAX MEMORY BANDWIDTH----------------------/
Mem_bandwidth_func<- function(mem){
  return ( as.double(strsplit(mem," ")[[1]][1]) )
}
CPUs_data$Max_Memory_Bandwidth <- sapply(CPUs_data$Max_Memory_Bandwidth,Mem_bandwidth_func)
for( i in unique(CPUs_data$Max_nb_of_Memory_Channels) ){
  fill_value =  median(CPUs_data[CPUs_data$Max_nb_of_Memory_Channels==i,'Max_Memory_Bandwidth'],na.rm = TRUE)
  subset <- CPUs_data[CPUs_data$Max_nb_of_Memory_Channels==i,'Max_Memory_Bandwidth']
  CPUs_data[CPUs_data$Max_nb_of_Memory_Channels==i,'Max_Memory_Bandwidth'] = na.fill(subset,fill_value)
}
```

- Product collection

Các dòng CPU có thể được chia thành các nhóm: Legacy, Celeron, Pentinum, Quark, Atom, Itanium, Xeon, Core

```
# PRODUCT COLLECTION ####################
product_collect <- c('Legacy', 'Celeron', 'Pentium', 'Quark', 'Atom', 'Itanium', 'Xeon','Core')

for (i in product_collect) {
  CPUs_data$Product_Collection <- ifelse(grepl(i, CPUs_data$Product_Collection), i, CPUs_data$Product_Collection)
}
```

- TDP (Thermal design power)

Xoá đi đơn vị và đưa về định dạng số thực.

Dữ liệu chỉ bị mất ở nhóm mobile nên ta điền dữ liệu thiếu theo TDP mobile của năm trước năm trước bằng kỹ thuật Last Observation Carried Forward.

```
# TDP  ####################
# Đưa về dạng định lượng
CPUs_data$TDP <-as.double( gsub(" W", "", CPUs_data$TDP))

# vì dữ liệu mất là các mobile nên sẽ fill theo nhóm mobile bằng kỹ thuật Last Observation Carried Forward
CPUs_data[CPUs_data$Vertical_Segment == "Mobile", "TDP"] <- na.locf(CPUs_data[CPUs_data$Vertical_Segment == "Mobile", "TDP"])
```

- Cache

Vì cache có 2 thành phần là loại cache và size của nó nên ta sẽ tách thành 2 cột: Cache_Type và Cache_Size.

Đối với Cache_Size ta đưa về dạng số bằng cách xóa đi đơn vị của nó và chuyển sang định dạng số thực.

Đối với Cache_Type, có một loại biến không chứa cache cụ thể (như: SmartCache, L2) nên ta thay thế nó bằng normal.

```
# CACHE   ####################
Cache_Size_Clean <- function(size){  # default: MB
  if(grepl('M',size)){
    return (as.double(gsub(" M","",size)))
  }
  else{
    return (as.double(gsub(" K","",size)) /1024)
  }
}
# Tách dữ liệu => type & cache
CPUs_data <- separate(CPUs_data,Cache,into = c("Cache_Size","Cache_Type"),sep="B")

# Thêm loại normal và biến rỗng
CPUs_data$Cache_Type <- ifelse(CPUs_data$Cache_Type == "", "Normal", sub(" ","",CPUs_data$Cache_Type))

# xử lý chuỗi và đưa về kiểu số thực
CPUs_data$Cache_Size <- sapply(CPUs_data$Cache_Size,Cache_Size_Clean)
summary(CPUs_data$Cache_Size)
```

- Instruction Set

Theo bản số liệu thì instruction set chỉ có 2 loại dữ liệu là 32-bit và 64-bit.

Thực hiện loại bỏ nhưng biến bị khuyết, ta lưu dữ liệu tạm thời vào SubData đồng thời chuyển dữ liệu về dạng số thực bằng cách bỏ đi đơn vị bit.

```
# INSTRUCTION SET ####################
(SubData <- CPUs_data[complete.cases(CPUs_data$Instruction_Set), ])
(temp <- as.double( gsub("-bit", "", temp$Instruction_Set)))
summary(temp)
```

```
summary(tempCol)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  32.00   64.00   64.00   63.23   64.00   64.00
```

Theo kết quả thu được, ba giá trị của khoảng tứ phân vị đều đạt giá trị 64 nên Mode của cột dữ liệu này sẽ là 64.

Vậy ta sẽ điền các dữ liệu còn thiếu là 64-bit với lý do tập 64-bit được ưu chuộng hơn.

```
CPUs_data$Instruction_Set <- na.fill(CPUs_data$Instruction_Set,"64-bit")
```

- Max_Memory_Banwidth

Tất cả dữ liệu đều ở dạng GB/s, ta xử lý chuỗi và đưa dữ liệu về dạng số thực bằng cách bỏ đi đơn vị

```
bandwidth_clean <- function(mem){
  return ( as.double(strsplit(mem," ")[[1]][1]) ) # truy cập đầu đến khoảng trắng đầu tiên trong 'mem'
}

CPUs_data$Max_Memory_Bandwidth <- sapply(CPUs_data$Max_Memory_Bandwidth,bandwidth_clean)
```

Max memory bandwidth có tương quan cao với max number memory channels, khi tăng số lượng kênh tối đa sẽ dẫn đến tăng bandwitdth.

```
> (correlation <- cor(CPUs_data$Max_Memory_Bandwidth, CPUs_data$Max_nb_of_Memory_Channels, method = "pearson"))
[1] 0.9675164
```

Vậy nên ta sẽ điền dữ liệu bị khuyết theo median của số Max Memory Banwidth theo từng nhóm của Max_nb_of_Memory_Channels.

```
for( i in unique(CPUs_data$Max_nb_of_Memory_Channels) ){
  subset <- CPUs_data[CPUs_data$Max_nb_of_Memory_Channels==i,'Max_Memory_Bandwidth']
  fillValue =  median(CPUs_data[CPUs_data$Max_nb_of_Memory_Channels==i,'Max_Memory_Bandwidth'],na.rm = TRUE)
  CPUs_data[CPUs_data$Max_nb_of_Memory_Channels==i,'Max_Memory_Bandwidth'] = na.fill(subset,fillValue)
}
```

- Recommended Customer Price

Xử lý định dạng chuỗi: xoá đi các kí tự '$' và dấu phân chia hàng nghìn ','.

```
# RECOMMEND CUSTOMER PRICE ####################
# sửa định dạng chuỗi
# vì $ là ký tự đặt biệt -> \\$
CPUs_data$Recommended_Customer_Price <- gsub("\\$", "", CPUs_data$Recommended_Customer_Price)
CPUs_data$Recommended_Customer_Price <- gsub(",", "", CPUs_data$Recommended_Customer_Price)
```

Một số dữ liệu ở dạng khoảng giá thì lấy giá trị trung bình 2 đầu của khoảng.

```
recommend_price <- function(price_range) {
  if(grepl('-', price_range)) {
    range <- strsplit(price_range, "-")[[1]]
    return((as.double(range[1]) + as.double(range[2])) / 2)
  }
  return (price_range)
}
# apply hàm để xử lý số liệu
CPUs_data$Recommended_Customer_Price <- sapply(CPUs_data$Recommended_Customer_Price, recommend_price)
CPUs_data$Recommended_Customer_Price <- as.double(CPUs_data$Recommended_Customer_Price) # đưa phần còn lại về dạng định lượng
CPUs_data <- CPUs_data %>%
group_by(Product_Collection) %>%
```

Giá tiền đề xuất được chia rõ ràng theo thời gian với các dòng CPU cùng loại, vì vậy ta áp dụng điền dữ liệu với từng loại CPU theo năm trước đó rồi theo năm sau đó

```
CPUs_data <- CPUs_data %>%  # piping
  group_by(Product_Collection) %>%  # nhóm theo production collection
    fill(Recommended_Customer_Price, .direction = "updown") # ưu tiên điền the forward carried rồi backward
```

- Processor Base Frequency

Đưa từ dạng chuỗi về cùng một đơn vị GHz và bỏ đi đơn vị của tần số để đưa về
dạng số thực.

```
# PROCESSOR BASE FREQUENCY ####################
frequency_clean <- function(f){
  if (grepl(' GHz',f)) {
    return (as.double(gsub(" GHz","",f)))
  }
  return (as.double(gsub(" MHz","",f)) /1000)
}
CPUs_data$Processor_Base_Frequency <- as.double( sapply(CPUs_data$Processor_Base_Frequency,frequency_clean))
```

Dữ liệu bị thiếu ở loại mobile vì vậy ta điền dữ liệu thiếu theo tần số của mobile năm trước.

```
subset <- CPUs_data[CPUs_data$Vertical_Segment == "Mobile", "Processor_Base_Frequency"]
CPUs_data[CPUs_data$Vertical_Segment == "Mobile", "Processor_Base_Frequency"] <- na.locf(subset)
```

3.3. Kiểm tra lại dữ liệu

Dữ liệu đã không còn thiếu

```
apply(is.na(CPUs_data),2,sum)
      Product_Collection           Vertical_Segment                    Status                 Launch_Date
                       0                          0                         0                           0
             Lithography Recommended_Customer_Price                nb_of_Cores               nb_of_Threads
                       0                          0                         0                           0
Processor_Base_Frequency                 Cache_Size                Cache_Type             Instruction_Set
                       0                          0                         0                           0
                     TDP             Max_Memory_Size  Max_nb_of_Memory_Channels       Max_Memory_Bandwidth
                       0                          0                         0                           0
```

Dữ liệu đã được đưa về đúng định dạng

```
> str(CPUs_data)
gropd_df [1,402 × 16] (S3: grouped_df/tbl_df/tbl/data.frame)
 $ Product_Collection        : chr [1:1402] "Legacy" "Legacy" "Legacy" "Legacy" ...
 $ Vertical_Segment          : chr [1:1402] "Desktop" "Desktop" "Desktop" "Desktop" ...
 $ Status                    : chr [1:1402] "End of Interactive Support" "End of Interactive Support" "End of Interactive
Support" "End of Interactive Support" ...
 $ Launch_Date               : num [1:1402] 2008 2008 2008 2009 2009 ...
 $ Lithography               : num [1:1402] 45 45 45 45 45 45 45 45 45 45 ...
 $ Recommended_Customer_Price: num [1:1402] 555 305 990 203 305 ...
 $ nb_of_Cores               : int [1:1402] 4 4 4 4 4 4 4 4 4 4 ...
 $ nb_of_Threads             : num [1:1402] 8 8 8 4 8 8 8 8 8 8 ...
 $ Processor_Base_Frequency  : num [1:1402] 2.93 2.66 3.2 2.66 2.93 3.06 3.2 2.8 3.33 1.6 ...
 $ Cache_Size                : Named num [1:1402] 8 8 8 8 8 8 8 8 8 6 ...
  ..- attr(*, "names")= chr [1:1402] "8 M" "8 M" "8 M" "8 M" ...
 $ Cache_Type                : chr [1:1402] "SmartCache" "SmartCache" "SmartCache" "SmartCache" ...
 $ Instruction_Set           : chr [1:1402] "64-bit" "64-bit" "64-bit" "64-bit" ...
 $ TDP                       : num [1:1402] 130 130 130 95 95 130 130 95 130 45 ...
 $ Max_Memory_Size           : num [1:1402] 24 24 24 16 16 24 24 16 24 8 ...
 $ Max_nb_of_Memory_Channels : int [1:1402] 3 3 3 2 2 3 3 2 3 2 ...
 $ Max_Memory_Bandwidth      : num [1:1402] 25.6 25.6 25.6 21 21 25.6 25.6 21 25.6 21 ...
```