\* Cài thư viện cần thiết

> install.packages("dplyr")

> install.packages("geosphere")

\* Gọi thư viện cần xài

> library(dplyr)

> library(tools)

// Q1: Xử lý dữ liệu file dirty\_data

\* Đọc dữ liệu

> dirty\_data = read.csv("./dirty\_data.csv")

> View(dirty\_data)

A screenshot of a computer

Description automatically generated

\* Làm sạch dữ liệu

Tạo một tệp con bao gồm một số biến cần phân tích:

> cleaning\_data = dirty\_data[,c("nearest\_warehouse","order\_price","delivery\_charges","customer\_lat","customer\_long","coupon\_discount","order\_total","season","is\_expedited\_delivery","distance\_to\_nearest\_warehouse","is\_happy\_customer")]

\* Kiểm tra dữ liệu khuyết

> apply(is.na(cleaning\_data),2,which)

integer(0)

-> Tệp không có dữ liệu khuyết

\* Làm sạch các dữ liệu có trong một số cột có kiểu dữ liệu character (chr): nearest\_warehouse, season (không làm sạch cột is\_expedited\_delivery và is\_happy\_customer bởi vì có logical mặc dù kiểu dữ liệu chr.)

> str(cleaning\_data)

A screenshot of a computer

Description automatically generated

\*\* nearest\_warehouse

> table(cleaning\_data$nearest\_warehouse, useNA = "always")

-> Không có dữ liệu khuyết nhưng có một vài lỗi lộn xộn



<-> Dùng hàm toTitleCase() để tái định nghĩa và đồng nhất cột

> cleaning\_data$nearest\_warehouse = toTitleCase(cleaning\_data$nearest\_warehouse)

> table(cleaning\_data$nearest\_warehouse, useNA = "always")



\*\* season

Tương tự như của cột nearest\_warehouse

> cleaning\_data$season = toTitleCase(cleaning\_data$season)

> table(cleaning\_data$season, useNA = "always")



\* Tạo tệp cleaned\_data sau khi xử lý xong dữ liệu và xóa cleaning\_data.

> cleaned\_data = cleaning\_data

> rm(cleaning\_data)

// Q2: Xử lý dữ liệu trong file missing\_data

\*

> missing\_data = read.csv("./missing\_data.csv")

\*\* date & season

> as.Date(missing\_data$date)

> missing\_data = missing\_data %>%

+ mutate(date\_onset = as.Date(date, format = "%Y/%m/%d"))

> fix\_season = function(date) {

+ month = as.POSIXlt(date)$mon +1

+ if (month %in% c(1, 2, 3)) {

+ return("Spring")

+ } else if (month %in% c(4, 5, 6)) {

+ return("Summer")

+ } else if (month %in% c(7, 8, 9)) {

+ return("Autumn")

+ } else {

+ return("Winter")

+ }

+ }

> missing\_data$season = sapply(missing\_data$date, fix\_season)

\*\* nearest\_warehouse

> warehouses = read.csv("./warehouses.csv")

> find\_nearest\_warehouse = function(customer, warehouses) {

+ distances = geosphere::distVincentySphere(

+ cbind(customer["customer\_long"], customer["customer\_lat"]), cbind(warehouses$lon, warehouses$lat)

+ )

+ nearest\_warehouse = warehouses[which.min(distances), ]

+ return(nearest\_warehouse$names)

+ }

> empty\_rows = missing\_data$nearest\_warehouse == ""

> missing\_data$nearest\_warehouse[empty\_rows] = apply(missing\_data[empty\_rows, c("customer\_long", "customer\_lat")], 1, function(row) find\_nearest\_warehouse(row, warehouses))

\*\* is\_happy\_customer

> get\_sentiment = function(review) {

+ positive\_keywords <- c("good", "excellent", "love", "like", "amazing", "happy", "recommend","great", "nice")

+ return(any(grepl(paste(positive\_keywords, collapse = "|"), tolower(review))))

+ }

> missing\_data$is\_happy\_customer = ifelse(missing\_data$is\_happy\_customer == "", sapply(missing\_data$latest\_customer\_review, get\_sentiment), missing\_data$is\_happy\_customer)

> missing\_data$is\_happy\_customer[missing\_data$is\_happy\_customer == 'TRUE'] = "True"

> missing\_data$is\_happy\_customer[missing\_data$is\_happy\_customer == 'FALSE'] = "False"

\*\* order\_price

> get\_price = function(total, discount, deli){

+ price= (total-deli)/(1-discount/100)

+ return(price)

+ }

> missing\_data$order\_price[is.na(missing\_data$order\_price)] = get\_price(missing\_data$order\_total[is.na(missing\_data$order\_price)],missing\_data$coupon\_discount[is.na(missing\_data$order\_price)],missing\_data$delivery\_charges[is.na(missing\_data$order\_price)])

\*\* order\_total

> get\_total = function(price, discount, deli){

+ total= price\*(1-discount/100)+deli

+ return(total)

+ }

> missing\_data$order\_total[is.na(missing\_data$order\_total)] = get\_total(missing\_data$order\_price[is.na(missing\_data$order\_total)],missing\_data$coupon\_discount[is.na(missing\_data$order\_total)],missing\_data$delivery\_charges[is.na(missing\_data$order\_total)])

\*\* distance\_to\_nearest\_warehouse

> get\_distance = function(customer, warehouses){

+ distances = geosphere::distVincentySphere(

+ cbind(customer["customer\_long"], customer["customer\_lat"]), cbind(warehouses$lon, warehouses$lat)

+ )

+ distance = min(distances)

+ return(distance/1000)

+ }

> missing\_data$distance\_to\_nearest\_warehouse[is.na(missing\_data$distance\_to\_nearest\_warehouse)] = apply(missing\_data[is.na(missing\_data$distance\_to\_nearest\_warehouse), c("customer\_long", "customer\_lat")], 1, function(row2) get\_distance(row2, warehouses))

\*\* NA Values

> na\_count = colSums(is.na(missing\_data))

> na\_count

A screenshot of a computer program

Description automatically generated

> missing\_data = missing\_data %>%

+ select(-date\_onset)

missing\_data = na.omit(missing\_data)

> na\_count <- colSums(is.na(missing\_data))

> na\_count

A screen shot of a computer code

Description automatically generated