Progressive Education Society's
Modern College of Engineering, Shivajinagar, Pune-05.
(An Autonomous Institute Affiliated to Savitribai Phule Pune University)
Department of MCA

PRACTICAL SUBMISSION RECORD- A.Y. 2025-26

Class: SYMCA Division : A **Semester: III**	**Course Code:** MCA01604 **Course Name:** Data Science Laboratory	**Batch:** S2
Name: Pranav Raju Malwatkar	**Roll No:** 52037	
CO No: CO605.2	**Assignment No:** 10	

Title : Use the FP-Growth Algorithm to perform Market Basket Analysis on a retail dataset. You will extract frequent item sets and generate association rules to discover purchasing patterns and help improve sales strategies.

Code:

```
# FP-Growth Market Basket Analysis - full R script
# Requires: arules, rCBA (for fpgrowth), arulesViz (optional for plotting)
# Two usage modes: example with built-in Groceries, or custom CSV (TransactionID, Item)

# ------------------------
# 1) Install / load packages
# ------------------------
packages <- c("arules", "arulesViz", "rCBA")
to_install <- packages[!(packages %in% installed.packages()[, "Package"])]
if (length(to_install) > 0) {
  install.packages(to_install)
}
library(arules)
# arulesViz optional
suppressWarnings(suppressMessages(require(arulesViz)))
suppressWarnings(suppressMessages(require(rCBA)))

# ------------------------
# 2) Parameters (change as needed)
# ------------------------
min_support <- 0.001    # e.g. 0.1% of transactions
min_confidence <- 0.8   # 80%
max_len <- 5            # maximum itemset length for mining

# ------------------------
# 3) Load data (two options)
```

```
# ------------------------

# OPTION A: Use built-in Groceries dataset (quick demo)
use_builtin <- TRUE

if (use_builtin) {
  data("Groceries")       # from arules
  txns <- Groceries       # already a transactions object
  message("Using built-in Groceries dataset (", length(txns), " transactions).")
} else {
  # OPTION B: Load custom CSV: file must have columns TransactionID and Item
  # Example CSV format:
  # TransactionID,Item
  # 1,milk
  # 1,bread
  # 2,beer
  # 2,diapers
  csv_file <- "retail_transactions.csv"  # change path as needed
  if (!file.exists(csv_file)) stop("CSV file not found: ", csv_file)
  raw <- read.csv(csv_file, stringsAsFactors = FALSE)
  if (!all(c("TransactionID", "Item") %in% colnames(raw))) {
    stop("CSV must contain at least columns named TransactionID and Item")
  }
  # convert to transactions
  split_list <- split(raw$Item, raw$TransactionID)
  txns <- as(split_list, "transactions")
  message("Loaded custom CSV; transactions:", length(txns))
}

# Quick summary
summary(txns)

# ------------------------
# 4) FP-Growth with rCBA
# ------------------------
# rCBA::fpgrowth accepts arules::transactions objects directly.
# Note: rCBA's fpgrowth can optionally take a 'consequent' parameter if you
# want to mine rules with a target class (for associative classification),
# but for general association rules set consequent = NULL (default).

message("Running rCBA::fpgrowth (FP-Growth)...")
# rCBA::fpgrowth's 'confidence' param expects value in same scale as arules (0..1)
```

```
# If your dataset is very large / dense you may want to increase support.
# Wrapping in tryCatch: some installations / configurations might require Java/rJava.
fpgrowth_result <- NULL
try({
  fpgrowth_result <- rCBA::fpgrowth(txns,
                                    support = min_support,
                                    confidence = min_confidence,
                                    maxLength = max_len,
                                    verbose = TRUE,
                                    parallel = TRUE)
}, silent = TRUE)

# -------------------------
# 5) Inspect FP-Growth result (rCBA)
# -------------------------
if (!is.null(fpgrowth_result)) {
  message("FP-Growth (rCBA) completed. Inspecting result...")
  # Print basic object
  print(fpgrowth_result)
  # rCBA prints rules in its own format. We will try to convert to a readable data.frame.
  # The rCBA rules object is not identical to arules::rules, but you can capture the printed output:
  cat("\n--- Example rCBA FP-Growth rules (first 20 lines of print) ---\n")
  capture.output(print(fpgrowth_result), file = stdout(), append = TRUE)[1:20] -> NULL

  # If you want a tidy table, fall back to using arules::apriori on the same data
  # (apriori often returns similar rules—this provides support/confidence/lift easily).
} else {
  message("rCBA::fpgrowth failed or is unavailable. Falling back to arules::apriori (Apriori).")
}

# -------------------------
# 6) Fallback / Complement: Use arules::apriori to produce arules object
# -------------------------
# (This is useful so you get a rules object compatible with arules methods such as
# quality(), inspect(), plotting and saving.)
message("Running arules::apriori with same thresholds (as a fallback / complement)...")
apriori_rules <- apriori(txns,
                         parameter = list(supp = min_support, conf = min_confidence, maxlen = max_len, target =
"rules"))
message("Number of rules found by apriori:", length(apriori_rules))

# Create a data.frame of rules with support/confidence/lift sorted by confidence
```

```
if (length(apriori_rules) > 0) {
  apriori_rules_sorted <- sort(apriori_rules, by = "confidence", decreasing = TRUE)
  top_rules <- head(apriori_rules_sorted, n = 20)  # show top 20 by default
  # Convert to data.frame for easier viewing/export
  rules_df <- DATAFRAME(top_rules, setStart = "", itemSep = ", ", separate = TRUE)
  # Ensure support/confidence/lift columns exist
  if (!("support" %in% colnames(rules_df))) rules_df$support <- quality(top_rules)$support
  if (!("confidence" %in% colnames(rules_df))) rules_df$confidence <- quality(top_rules)$confidence
  if (!("lift" %in% colnames(rules_df))) rules_df$lift <- quality(top_rules)$lift

  message("\nTop rules (by confidence) — first 20:")
  print(rules_df[, c("LHS", "RHS", "support", "confidence", "lift")])
} else {
  message("No rules found by apriori with the given thresholds.")
}

# ------------------------
# 7) Save / export frequent itemsets (if desired)
# ------------------------
# Get frequent itemsets (equivalent of FP-Growth's output)
freq_itemsets <- eclat(txns, parameter = list(supp = min_support, maxlen = max_len))
message("Frequent itemsets found (eclat):", length(freq_itemsets))
# Inspect top frequent itemsets by support
freq_df <- DATAFRAME(sort(freq_itemsets, by = "support", decreasing = TRUE))
if (nrow(freq_df) > 0) {
  cat("\nTop frequent itemsets (by support):\n")
  print(head(freq_df, 20))
}

# ------------------------
# 8) Visualization (optional)
# ------------------------
if (length(apriori_rules) > 0 && requireNamespace("arulesViz", quietly = TRUE)) {
  # Plot top 20 rules
  plot(apriori_rules_sorted[1:min(20, length(apriori_rules_sorted))],
       method = "graph", control = list(type = "items"), main = "Top Association Rules (by confidence)")
}

# ------------------------
# 9) Interpreting results & Sales strategy ideas
# ------------------------
# Print a short summary to guide business interpretation
```

```
cat("\n--- Short business interpretation tips ---\n")
cat("1) Look for item pairs/triples with high support & high confidence (and lift > 1):\n")
cat("   - High support: items frequently bought (good candidates for cross-sell features)\n")
cat("   - High confidence: when A is bought, B is likely to be bought (good for product bundling)\n")
cat("   - High lift (>1): stronger-than-random association (prioritize these)\n\n")
cat("2) Actions to consider:\n")
cat("   - Bundle top pairs into promotions or combo discounts.\n")
cat("   - Place frequently co-purchased items close in store layout or recommend online.\n")
cat("   - Use rules for personalized recommendations and targeted promotions.\n\n")
cat("3) Next steps:\n")
cat("   - Validate candidate rules on a held-out period (time-slice) to avoid seasonal bias.\n")
cat("   - Use POS / time-of-day segmentation to find context-specific patterns.\n")
cat("   - A/B test bundles or recommendations based on rule-driven suggestions.\n\n")

# ------------------------
# 10) Save results to CSV (optional)
# ------------------------
# write top rules to CSV for reporting
if (exists("rules_df") && nrow(rules_df) > 0) {
  write.csv(rules_df, file = "top_rules_by_confidence.csv", row.names = FALSE)
  message("Top rules exported to top_rules_by_confidence.csv")
}

message("Done.")
```

Output :

File History Resize