```r
# Load required libraries
library(tidyverse)      # For data manipulation and visualization
library(psych)          # For factor analysis functions
library(corrplot)       # For correlation visualization
library(GPArotation)    # For factor rotation methods
library(factoextra)     # For factor visualization
library(ggplot2)        # For advanced plotting
library(gridExtra)      # For arranging multiple plots
library(mice)           # For imputation of missing values
library(MVN)            # For multivariate normality testing

# Set seed for reproducibility
set.seed(123)


# Select variables for factor analysis
# We'll focus on spending patterns, shopping behavior, and promotional response
factor_vars <- c("Spend_Wine", "Spend_OrganicFood", "Spend_Meat",
                 "Spend_WellnessProducts", "Spend_Treats", "Spend_LuxuryGoods",
                 "Purchases_Online", "Purchases_Catalog", "Purchases_Store",
                 "Visits_OnlineLastMonth", "Promo_Purchases",
                 "Online_Preference", "Promo_Response_Rate")

# Create a subset with only the variables for factor analysis
factor_data <- smartfresh_data[, factor_vars]

# Scale the data (standardize to mean 0, sd 1)
factor_data_scaled <- scale(factor_data)

# Compute correlation matrix
corr_matrix <- cor(factor_data_scaled)

# Visualize correlation matrix
corrplot(corr_matrix, method = "color",
         tl.col = "black", tl.srt = 45,
         title = "Correlation Matrix of Shopping Variables")

# Parallel analysis
parallel_result <- fa.parallel(factor_data_scaled, fm = "ml", fa = "fa")

# Based on parallel analysis
num_factors_parallel <- parallel_result$nfact
cat("Number of factors suggested by parallel analysis:", num_factors_parallel, "\n")

# Let's determine the optimal number of factors based on these methods
# For this analysis, we'll use the number suggested by parallel analysis
num_factors <- num_factors_parallel

# Maximum Likelihood Factor Analysis
ml_result <- fa(factor_data_scaled, nfactors = num_factors, rotate = "none", fm = "ml")
print(ml_result)


Orthogonal rotation (Varimax) - assumes factors are uncorrelated
varimax_result <- fa(factor_data_scaled, nfactors = num_factors, rotate = "varimax", fm =
"ml")
print(varimax_result)

# Oblique rotation (Promax) - allows factors to be correlated
promax_result <- fa(factor_data_scaled, nfactors = num_factors, rotate = "promax", fm =
"ml")
print(promax_result)

# Compare factor correlation matrix from oblique rotation
# If correlations are substantial, oblique rotation is preferred
print(promax_result$Phi)

# Determine which rotation to use based on factor correlations
# If factor correlations are > 0.3, use oblique rotation (Promax)
# Otherwise, use orthogonal rotation (Varimax)
use_oblique <- any(abs(promax_result$Phi[upper.tri(promax_result$Phi)]) > 0.3)

if(use_oblique) {
  final_rotation <- "promax"
  final_result <- promax_result
  cat("Using oblique rotation (Promax) due to correlated factors\n")
} else {
  final_rotation <- "varimax"
  final_result <- varimax_result
  cat("Using orthogonal rotation (Varimax) due to uncorrelated factors\n")
}

# Create a heatmap of factor loadings
loadings_matrix <- as.data.frame(unclass(final_result$loadings))
colnames(loadings_matrix) <- paste0("Factor_", 1:num_factors)
loadings_matrix$Variable <- rownames(loadings_matrix)
loadings_long <- pivot_longer(loadings_matrix,
                              cols = starts_with("Factor_"),
                              names_to = "Factor",
                              values_to = "Loading")

# Plot the heatmap
ggplot(loadings_long, aes(x = Factor, y = Variable, fill = Loading)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", mid = "white", high = "red", midpoint = 0) +
  theme_minimal() +
  labs(title = "Factor Loadings Heatmap", x = "", y = "") +
  theme(axis.text.x = element_text(angle = 0, hjust = 0.5),
        axis.text.y = element_text(hjust = 1))
```