MDPI

Article

ProyectoTD2025

1. 1. Introducción

En la actualidad, la digitalización de los tickets de compra se ha convertido en una práctica común en grandes cadenas de supermercados. Estos tickets electrónicos, enviados en formato PDF al correo del cliente, no solo reducen el uso de papel, sino que también generan datos valiosos que pueden ser analizados para obtener información relevante sobre los hábitos de consumo, la evolución de precios y las preferencias de los compradores.

Este proyecto tiene como objetivo desarrollar un sistema de análisis automatizado que permita extraer, procesar y visualizar la información contenida en los tickets de Mercadona. Mediante técnicas de tratamiento de datos en R, exploraremos patrones de compra, identificaremos los productos más vendidos, analizaremos la evolución temporal de los precios y determinaremos tendencias en función de la ubicación de la tienda y el momento de la compra.

2. 2. Material y Métodos

Para llevar a cabo este proyecto se han seleccionando un conjunto de librerías específicas que respondieran a los distintos requerimientos del análisis.

Para la manipulación de datos se emplearon los paquetes tidyverse (incluyendo dplyr y stringr), que permitieron realizar operaciones de filtrado, transformación y procesamiento de texto de manera eficiente. La extracción del contenido textual desde los archivos PDF se realizó mediante el paquete pdftools, capaz de preservar la estructura original de los documentos.

Las visualizaciones se generaron utilizando ggplot2, seleccionado por su versatilidad para crear gráficos de alta calidad. Para la presentación de resultados en formatos reproducibles se implementó knitr, facilitando la integración de código, resultados y texto explicativo.

Utilizaremos dos data frames para manejar los datos de manera más eficiente. El primer data frame contendrá la información general del ticket, como la dirección del supermercado, la fecha y hora de la compra, el monto total, entre otros. En este caso, todos los productos registrados en el ticket se almacenarán como una sola cadena de texto en una única variable. El segundo data frame desglosará los productos en variables separadas

Ambos data frames estarán vinculados a través de la variable fs (factura simplificada).

3. 3. Importación de los datos

3.1. 3.1 Carga de ficheros

Para evitar errores durante el procesamiento posterior, se realizó un cambio en los nombres de los archivos PDF originales. Los archivos fueron renombrados de forma secuencial con un formato estándar. Esta acción se llevó a cabo una única vez, y por ello el código correspondiente fue comentado en el script, con el fin de prevenir que los archivos se sobrescriban accidentalmente al ejecutar el programa más de una vez.

Received:
Revised:
Accepted:
Published:

Citation: . ProyectoTD2025. *Journal Not Specified* **2023**, *1*, 0.
https://doi.org/

Copyright: © 2025 by the author. Submitted to *Journal Not Specified* for possible open access publication under the terms and conditions of the Creative Commons Attri- bution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

Table 1. Descripción de variables

Variable	Descripción
comercio	nombre del comercio
empresa	tipo y código de empresa
direccion	dirección del comercio
cp	código postal
telefono	teléfono del comercio
fecha	fecha de la compra (día-mes-año)
hora	hora de la compra (horas y minutos)
op	número del código de la operación
fs	código de la factura simplificada
productos	lista con los productos comprados
total	dinero total de la compra
forma_pago	forma de pago (tarjeta o en efectivo)
base_imp	base imponible (IVA)
cuota	cuota del IVA

Se procedió a cargar automáticamente todos los archivos PDF contenidos en la carpeta de trabajo designada. Para ello, se empleó una función que permite listar únicamente los archivos con extensión .pdf, garantizando así que solo se consideren los documentos relevantes para el análisis. Esta carga automatizada facilita el procesamiento por lotes y evita la necesidad de seleccionar manualmente cada archivo.

Para el proyecto se han importado un total de 303 archivos.

3.2. 3.2 Carga de datos

Se construyó un data frame a partir de los datos extraídos, realizando las transformaciones necesarias para asegurar que cada variable tuviera el formato y tipo de dato adecuados.

Para la lectura de los archivos PDF se utilizaron funciones de la librería pdftools, mientras que la limpieza y manipulación de las cadenas de texto se llevó a cabo con funciones de la librería stringr.

Durante el procesamiento, se asignó el tipo de dato Date a la variable de fecha y se convirtieron en valores numéricos las variables decimales como el total de la compra, la base imponible y la cuota de IVA. Cabe señalar que muchos de los datos no pueden incorporarse directamente al data frame, ya que en el formato original aparecen combinados en una misma línea. Es el caso de la fecha, la hora y el número de operación, que debieron separarse y asignarse a variables distintas.

Por motivos de espacio y legibilidad, no se incluye la visualización del data frame generado a partir de los tickets.

```
type <- c("comercio","empresa", "direccion", "cp", "telefono", "fecha", "hora", "o
knitr::kable(rticles::string_to_table(type, 2),align = 'c',
             col.names = c("Variable", "Descripción"),
             format = "latex", booktab = TRUE,
             caption = "Descripción de variables")
```

Las variables finales obtenidas se presentan en la Tabla @ref(tab:tabla-variables).

```
##           archivo aparcamiento
## 1   ./data/M1.pdf           NA
## 2  ./data/M10.pdf           NA
```

```
## 3 ./data/M100.pdf           NA
## 4 ./data/M101.pdf           NA
## 5 ./data/M102.pdf           NA
## 6 ./data/M103.pdf           NA
```

4. 4. Analizamos los productos

La información relativa a los productos comprados se encontraba inicialmente agrupada dentro de una única columna del data frame principal. Para facilitar su análisis, se extrajo esta columna a un nuevo data frame, separando los productos que venían concatenados en una misma celda.

4.1. 4.1 Productos pescateria

El procesamiento de los productos se realizó en varias etapas, según el tipo de producto y la forma en que aparecían en el ticket. En primer lugar, se identificaron los productos de pescadería, que siguen un formato particular: aparecen siempre precedidos por una línea con la palabra "PESCADO", seguida del nombre del producto en la línea siguiente y de los detalles de compra (peso, precio por kilo y total) en una tercera línea. A partir de esta estructura, se extrajeron los datos relevantes y se almacenaron en un nuevo data frame específico para pescado.

4.2. 4.2 Productos vendidos por peso

Después, se eliminaron las filas correspondientes a productos de pescadería para poder trabajar exclusivamente con los productos que también se venden por peso, como frutas y verduras. Estos artículos generalmente constan de dos líneas: la primera contiene el nombre del producto y la segunda incluye el peso, el precio por kilogramo y el importe total. A partir de esta estructura se construyó un segundo data frame con las frutas y verduras, extrayendo y transformando la información necesaria.

4.3. 4.2 Productos vendidos por unidad

Una vez separados los productos por peso, se procedió a procesar el resto de productos, es decir, aquellos que se venden por unidades. En este caso, se extrajeron datos como la cantidad, el nombre del producto, el precio unitario y el importe total. Se aplicaron técnicas de procesamiento de texto para limpiar y estructurar la información, ya que algunos productos con una sola unidad incluían el precio directamente dentro del nombre del producto.

5. 4.4 productos analizados

Finalmente, los tres grupos de productos —pescado, frutas y verduras, y productos por unidades— se combinaron en un único data frame unificado. A este conjunto se le añadió una columna adicional que indicaba si el ticket incluía un servicio de aparcamiento, en caso de que esa información estuviera disponible. El resultado fue un data frame final, estructurado y homogéneo, con todos los productos organizados por tipo, cantidad, precio, importe y número de ticket, listo para su análisis posterior.

```r
df_producto <- df %>% select(c(num_ticket, productos)) %>%
  separate_rows(productos, sep = ";")
```

```r
# Procesamiento de pescado por kg

# Identificar filas con "PESCADO"
filas_pescado <- which(df_producto$productos == "PESCADO")
```

```r
# Inicializar vectores para almacenar datos
num_ticket_vec <- character()
nombre_producto_vec <- character()
peso_kg_vec <- numeric()
precio_kg_vec <- numeric()
precio_total_vec <- numeric()

for (i in seq_along(filas_pescado)) {
  idx <- filas_pescado[i]

  # Extraer información básica
  num_ticket <- df_producto$num_ticket[idx]
  nombre <- df_producto$productos[idx + 1]  # Nombre en la siguiente fila

  # Procesar la fila de detalles
  detalles <- df_producto$productos[idx + 2]

  # Limpiar y dividir la cadena de detalles
  detalles_limpio <- gsub(",", ".", detalles)  # Reemplazar comas por puntos
  detalles_split <- strsplit(trimws(detalles_limpio), "\\s+")[[1]]

  # Extraer valores (asumiendo orden: peso, unidad, precio_kg, moneda, precio_tota
  if (length(detalles_split) >= 5) {
    peso_kg <- as.numeric(detalles_split[1])
    precio_kg <- as.numeric(detalles_split[3])
    importe <- as.numeric(detalles_split[5])

    # Almacenar en vectores
    num_ticket_vec <- c(num_ticket_vec, num_ticket)
    nombre_producto_vec <- c(nombre_producto_vec, nombre)
    peso_kg_vec <- c(peso_kg_vec, peso_kg)
    precio_kg_vec <- c(precio_kg_vec, precio_kg)
    precio_total_vec <- c(precio_total_vec, importe)
  }
}

df_pescado <- data.frame(
  num_ticket = num_ticket_vec,
  nombre = nombre_producto_vec,
  peso_kg = peso_kg_vec,
  precio_kg = precio_kg_vec,
  importe = precio_total_vec,
  stringsAsFactors = FALSE
)
```

```r
# Procesamiento fruta y la verdura

#Identificar TODOS los bloques de pescado (3 filas cada uno)
bloques_pescado <- which(df_producto$productos == "PESCADO")
```

```r
# Crear vector con TODAS las filas a eliminar (cada bloque son 3 filas)
filas_a_eliminar <- unlist(lapply(bloques_pescado, function(x) x:(x+2)))

# Eliminar todos los bloques
df_sin_pescado <- df_producto[-filas_a_eliminar, ]

ind_detalles_kg <- grep("kg.*€/kg", df_sin_pescado$productos, value = FALSE)

df_fruta_verdura <- data.frame(
  num_ticket = df_sin_pescado$num_ticket[ind_detalles_kg],
  nombre = df_sin_pescado$productos[ind_detalles_kg - 1],
  detalles = df_sin_pescado$productos[ind_detalles_kg],
  stringsAsFactors = FALSE
) %>%
mutate(
  # Limpiar el nombre (eliminar números iniciales)
  nombre = gsub("^\\d+\\s*", "", nombre),

  # Extraer peso (kg) - primer número en la línea
  peso_kg = as.numeric(gsub(",", ".", str_extract(detalles, "^[0-9,]+"))),

  # Extraer precio por kg - método mejorado
  precio_kg = as.numeric(gsub(",", ".",
    str_extract(detalles, "[0-9,]+(?=\\s*€/kg)"))),

  # Extraer importe total - último número en la línea
  importe = as.numeric(gsub(",", ".",
    str_extract(detalles, "[0-9,]+$")))
) %>%
select(-detalles)
```

```r
# Procesamiento resto de productos sin kg

# Identificar las filas de detalles (kg y €/kg)
ind_detalles_kg <- grep("kg.*€/kg", df_sin_pescado$productos, value = FALSE)

# Las filas de nombres están justo antes de los detalles
ind_nombres_kg <- ind_detalles_kg - 1

# Combinar todos los índices a eliminar
filas_fruta_verdura <- sort(unique(c(ind_nombres_kg, ind_detalles_kg)))

# Eliminar filas ya procesadas
df_resto <- df_sin_pescado[-filas_fruta_verdura, ]

df_productos_unidades <- df_resto %>%
  mutate(
    # 1. Extraer cantidad (siempre es el primer número)
    cantidad = as.numeric(str_extract(productos, "^\\d+")),
```

```
    # 2. Extraer posible precio en el nombre (para productos de 1 unidad)
    precio_en_nombre = ifelse(cantidad == 1,
                              as.numeric(gsub(",", ".", str_extract(productos, "\\d+
                              NA_real_),

    # 3. Procesamiento vectorizado de componentes
    componentes = strsplit(productos, "\\s+"),

    # 4. Extraer importe normal (para productos con múltiples unidades)
    importe_normal = sapply(componentes, function(x) {
      if(length(x) >= 3) as.numeric(gsub(",", ".", x[length(x)])) else NA_real_
    }),

    # 5. Determinar el importe final
    importe = ifelse(!is.na(precio_en_nombre), precio_en_nombre, importe_normal),

    # 6. Extraer descripción limpia (MODIFICACIÓN CLAVE)
    nombre = mapply(function(comp, prod, cant, precio_nombre) {
      # Primero eliminar la cantidad inicial (si existe)
      nombre_limpio <- gsub("^\\d+\\s*", "", prod)

      if(length(comp) <= 2) return(nombre_limpio)  # Caso simple

      if(!is.na(precio_nombre)) {
        # Para productos de 1 unidad: eliminar precio final
        gsub("\\s+\\d+,\\d+$", "", nombre_limpio)
      } else {
        # Para múltiples unidades: eliminar elementos numéricos finales
        paste(comp[2:(length(comp)-2)], collapse=" ")
      }
    }, componentes, productos, cantidad, precio_en_nombre, SIMPLIFY = TRUE) %>%
      str_trim()  # Eliminar espacios sobrantes
  ) %>%
  mutate(
    # 7. Calcular precio unitario
    precio_unitario = importe / cantidad
  ) %>%
  select(num_ticket, nombre, cantidad, precio_unitario, importe)
```

```
## Warning: There were 33 warnings in `mutate()`.
## The first warning was:
## i In argument: `importe_normal = sapply(...)`.
## Caused by warning in `FUN()`:
## ! NAs introducidos por coerción
## i Run `dplyr::last_dplyr_warnings()` to see the 32 remaining warnings.
```

```
# dataframe final con todos los productos analizados:

# Añadir columna 'tipo' a cada dataframe
df_pescado <- df_pescado %>% mutate(tipo = "pescado")
```

```r
df_fruta_verdura <- df_fruta_verdura %>% mutate(tipo = "fruta_verdura")
df_productos_unidades <- df_productos_unidades %>% mutate(tipo = "unidades")

# Unificar columnas para combinar
df_final <- bind_rows(
  df_pescado %>% select(num_ticket, nombre, cantidad = peso_kg, precio = precio_kg
  df_fruta_verdura %>% select(num_ticket, nombre, cantidad = peso_kg, precio = pre
  df_productos_unidades %>% select(num_ticket, nombre = nombre, cantidad, precio =
)

# Crear columna tiene_aparcamiento
if ("aparcamiento" %in% colnames(df_aparcamiento)) {
  df_aparcamiento <- df_aparcamiento %>%
    mutate(nombre_archivo = basename(archivo)) %>%
    mutate(num_ticket = str_extract(nombre_archivo, "\\d+")) %>%
    mutate(tiene_aparcamiento = !is.na(aparcamiento)) %>%
    select(num_ticket, tiene_aparcamiento)

  # Convertir num_ticket a carácter en ambos dataframes
  df_final <- df_final %>% mutate(num_ticket = as.character(num_ticket))
  df_aparcamiento <- df_aparcamiento %>% mutate(num_ticket = as.character(num_tick

  # Combinar los datos
  df_final <- left_join(df_final, df_aparcamiento, by = "num_ticket")
} else {
  df_final$tiene_aparcamiento <- NA  # columna vacía si no existe 'aparcamiento'
}

# Resultado final ordenado
df_final <- df_final %>% arrange(num_ticket)

# Verificación final
cat("\nResumen del DataFrame final:\n")
```

```
##
## Resumen del DataFrame final:
```

```r
cat("- Pescado:", sum(df_final$tipo == "pescado"), "registros\n")
```

```
## - Pescado: 28 registros
```

```r
cat("- Fruta/Verdura:", sum(df_final$tipo == "fruta_verdura"), "registros\n")
```

```
## - Fruta/Verdura: 395 registros
```

```r
cat("- Unidades:", sum(df_final$tipo == "unidades"), "registros\n")
```

```
## - Unidades: 4702 registros
```

```r
cat("- Total:", nrow(df_final), "registros\n")
```

```
## - Total: 5125 registros
```

```r
print(head(df_final, 10))
```

```
##    num_ticket              nombre cantidad precio importe          tipo
## 1      007267    PERA CONFERENCIA    0.854   2.55    2.18 fruta_verdura
## 2      007267    MANZ. ROJA DULCE    1.474   2.90    4.27 fruta_verdura
## 3      007267      PIZZA BOLOÑESA    1.000   2.60    2.60      unidades
## 4      007267    MIX FRUTOS ROJOS    1.000   1.66    1.66      unidades
## 5      007267   PISTO DE VERDURAS    1.000   2.30    2.30      unidades
## 6      007267    FILETE DE TRUCHA    1.000   3.92    3.92      unidades
## 7      007267        PAN SEMILLAS    1.000   1.60    1.60      unidades
## 8      007267         SNACK PIPAS    1.000   1.40    1.40      unidades
## 9      007267 +PROT NATILLA VAINI    1.000   1.75    1.75      unidades
## 10     007267     SNACK CHOCOLATE    1.000   1.50    1.50      unidades
##    tiene_aparcamiento
## 1                  NA
## 2                  NA
## 3                  NA
## 4                  NA
## 5                  NA
## 6                  NA
## 7                  NA
## 8                  NA
## 9                  NA
## 10                 NA
```

#Preguntas

¿Cuáles son los productos menos vendidos por unidades? ¿Y por kilos?

¿Qué productos han generado mayor ingreso total (precio x cantidad)?

¿Cuáles productos han aumentado o disminuido su venta a lo largo del tiempo?

¿Qué productos se compran habitualmente juntos?

¿Existen diferencias de precios para el mismo producto en diferentes tiendas o ubicaciones?

¿Qué días de la semana hay más ventas? ¿Y a qué horas?

¿En qué meses se venden más frutas/verduras, o pescados?

¿Desde qué ciudades se emiten más tickets?

¿Existen diferencias de consumo por ciudad?

¿Cuánto se gasta semanal o mensualmente en un supermercado?

¿Influye la disponibilidad de aparcamiento en el importe total de la compra?

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.