COMP1013 Report - Ethan Rauv

Ethan-lee Rauv (20699967)

2023-05-23

Report Background:

Your company, Dawson Steward Analytics, a consultancy firm specialised in AI analytics is tasked by a retail chain in the northern hemisphere to analyse their sales data.

Question 1:

Write the code to compute the total revenue of each store at the end of each day. Is there a noted difference between the days? Write also the code to calculate the total revenue over the seven day period. Plot the latter on a graph.

To compute the total revenue of each store daily the following code was used :

```
# Set working directory to find CSV file
setwd("D:/Github Respositories/R-code-aut23/Report")

# Load required packages
library(dplyr)

# Read the sales data
sales_daily <- read.csv("sales_ug.csv")

# Compute total revenue by store and date
daily_revenue <- sales_daily %>%
  group_by(store_id, date) %>%
  summarise(total_revenue = sum(revenue))

# Print total revenue by store and date
print(daily_revenue)
```

and for plotting the weekly revenue the following was used:

```
#sets working directory to find file
setwd("D:/Github Respositories/R-code-aut23/Report")

#load the library
library(ggplot2)

# Read the data
sales_data <- read.csv("sales_ug.csv")

# Calculate total revenue for each store for a week
total_revenue <- aggregate(sales_data$revenue,
                  by=list(StoreID=sales_data$store_id),
                    sum)
```

```r
# Ploting the results
ggplot(total_revenue, aes(x=StoreID, y=x, fill=StoreID)) +
  geom_bar(stat="identity") +
  theme_minimal() +
  labs(title="Total Revenue Per Store Over Seven Days",
       x="Store ID",
       y="Total Revenue ($)") +
         theme(legend.position="none")
```

For the rationale between these two chunks of code:

For daily Revenue:

- **Choice of package (dplyr)**: The `dplyr` package is chosen because it provides efficient functions to help perform data manipulation tasks. It also offers functions that allow the code to express data transformation steps in a clear and readable manner.
- **Reading the sales data**: The `read.csv()` function is used to read the data from a CSV file named `sales_ug.csv`. This is then stored in the variable named `sales_data`.
- **Computing the daily revenue**: To compute the total revenue for each store at the end of each day, the previous data within the `sales_data` variable is used. the `group_by()` function is used to group the data by `store_id` and `date`. This function groups the data and calculates for each unique store id and date. Then, the `summarise()` function is used to reduce the multiple variables into single summaries and to improve readability, the `sum()` function is then applied to the revenue column, resulting in the `total_revenue` column that contains the sum of revenues for each store and date combination. The resulting `dataframe` daily_revenue contains the store ID, date, and corresponding total revenue.
- **Printing the daily revenue**: The `print()` function is used to display the `daily_revenue` dataframe, which contains the total revenue of each store at the end of each day. This will make it possible to examine the data produced by the calculations above.

For weekly Revenue:

- **Choice of ggplot2 package**: The `ggplot2` package is chosen because it provides functions to help visualise different tables and data.
- **Reading the sales data**: The `read.csv()` function is used to read the data from a CSV file named `sales_ug.csv`. This is then stored in the dataframe named `sales_data`.
- **Calculating the weekly revenue**: Using the `aggregate()` function, which calculates summary statistics from subsets of data, and the dataframe `sales_data` we are able to calculate the weekly revenue for each store. The `sales_data$revenue` specifies the dependent variable we need to aggregate, which is the revenue column from the `sales_data` dataframe, `by=list(StoreID=sales_data$store_id)` specifies the grouping variables for the aggregation. For this question used StoreID as the name of the grouping variable as we are trying to find revenues for each store, and `sales_data$store_id` is the actual column containing the store IDs. The sum provides the sum of the revenue for each store. The result of the `aggregate()` function is stored in `the total_revenue` dataframe, which will have two columns: StoreID (store ID) and x (weekly revenue).
- **Plotting the data**: Here to plot the data calculated from the code above, we make use of the `ggplot2` library. A bar graph was chosen due to its ability to display and easily compare the different stores. The `total_revenue` is the dataframe containing the data that we want to plot, `aes(x=StoreID, y=x, fill=StoreID)` defines the aesthetics mapping. For this we want,`x=StoreID` makes the StoreID the x-axis, **y=x** makes the x column (representing the total revenue) to the y-axis, and `fill=StoreID` determines the colour fill of the bars based on the store ID to help with readability of the graph. The `geom_bar(stat="identity")` adds the bars to the plot. The `stat="identity"` line indicates that the heights of the bars should directly correspond to the values in the y-axis variable (x

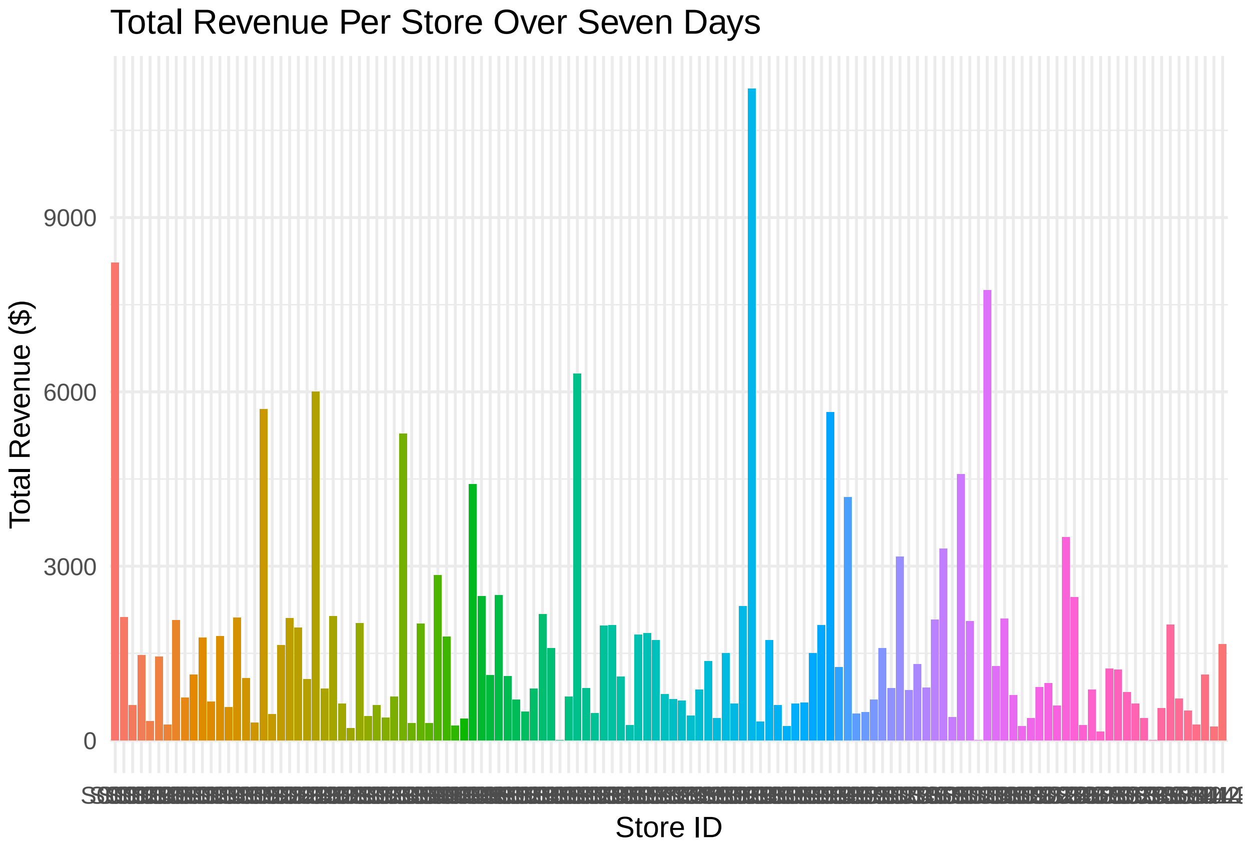
column). **labs(title="Total Revenue Per Store Over Seven Days", x="Store ID", y="Total Revenue")** sets the plot's title and axis labels. **theme(legend.position="none")** removes the legend from the plot to improve readability and as the stores are already labelled on the x-axis.

Results for daily revenue:

```
## # A tibble: 886 x 3
## # Groups:   store_id [128]
##    store_id date       total_revenue
##    <chr>    <chr>              <dbl>
##  1 S0001    2017-07-03          768.
##  2 S0001    2017-07-04         1296.
##  3 S0001    2017-07-05         1006.
##  4 S0001    2017-07-06          894.
##  5 S0001    2017-07-07         1248.
##  6 S0001    2017-07-08         1547.
##  7 S0001    2017-07-09         1465.
##  8 S0002    2017-07-03          347.
##  9 S0002    2017-07-04          226.
## 10 S0002    2017-07-05          175.
## # i 876 more rows
```

If we look at the days we can see that some days do indeed make more revenue for the stores than other days.

Results of plotting weekly revenue:



Question 2:

What's the most popular product type (hierarchy 1) sold in all stores over a week? How much revenue did the stores receive for that product during the week? How does that compare with the second most popular product? Provide a table that shows the product type ranked from most to least popular. For each product type provide: how many subtypes (hierarchy 2) are there, how many products are in this product type, what's the sales quantity, and the revenue generated.