Intro Data Analysis in R

Lateef

University of Gloucoustershire

Learning objectives:

- Understand and apply fundamental R tools to **read, clean** and **explore** datasets.
- **Create** and **interpret** basic data visualization using R tools.

Contents:

- Data Preprocessing
- Data Manipulation
- Data Visualization

Preliminaries:

⚠️ **Assumptions**

✔️ R and RStudio (or an alternative R-friendly interface) is installed on each student's computer.

➡️ Students have a basic understanding of R, including familiarity with simple commands and data structures.

💡 Nevertheless, all R commands will be introduced in a clear and simplified manner to enhance the understanding of all learners.

Starter

What is R? Packages installation:

- R needs packages to work. There are many available packages to use in R. E.g, *tidyverse*, *rio*, *lubridate*, *dplyr*, etc.
- To install a package use install.package(“*package_name*”)

Libraries:

- Use to load **packages** into working file. E.g. library(*package_name*)

```
library(tidyverse)   # Load tidyverse into work session
library(gt)
library(readxl)
```

Data Preprocessing

In-built data

- There are several data available in R. This can be access via:

```
data()    # Show all inbuilt data in R
```

Data loading

There are several ways of loading data in R, depending on the type of dataset.

```
data1 <- read_csv("dt_exampl.csv")       # load csv file
data2 <- read_excel("dt_example.xlsx")   # load excel file
data3 <- data(mpg)                        # load in-built data
```

Data exploring

After loading a dataset, it's important to explore and examine the dataset to gain a better understanding before proceeding with analysis.

```r
str(data1)       # display the structure of the data
```

```
spc_tbl_ [1,000 × 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ TransactionID : chr [1:1000] "TX00001" "TX00002" "TX00003" "TX00004" ...
 $ BuyerName     : chr [1:1000] "Total Nigeria" "Eterna Plc" "Oando Plc" "Eterna Plc" ...
 $ FuelType      : chr [1:1000] "PMS" "AGO" "DPK" "DPK" ...
 $ SalesLiters   : num [1:1000] 43485 29355 22492 36538 24847 ...
 $ PricePerLitre : num [1:1000] 645 700 645 610 610 700 530 700 530 645 ...
 $ SalesDate     : Date[1:1000], format: "2025-02-28" "2025-01-09" ...
 $ Terminal      : chr [1:1000] "Port Harcourt Terminal" "Lagos Jetty" "Lekki Port" "Lagos Jetty" ...
 $ Operator      : chr [1:1000] "Shift C" "Shift B" "Shift B" "Shift B" ...
 $ VehicleType   : chr [1:1000] "Trailer" "Trailer" "Container" "Container" ...
 $ PaymentMethod : chr [1:1000] "Bank Transfer" "Cash" "Bank Transfer" "Bank Transfer" ...
 $ DeliveryStatus: chr [1:1000] "Cancelled" "Pending" "Cancelled" "Pending" ...
 $ Region        : chr [1:1000] "South-West" "North-East" "South-West" "South-East" ...
 $ InvoiceNumber : chr [1:1000] "INV000001" "INV000002" "INV000003" "INV000004" ...
 $ BatchID       : chr [1:1000] "BATCH918" "BATCH345" "BATCH826" "BATCH262" ...
 $ TotalSales    : num [1:1000] 28047528 20548570 14507443 22287954 15156536 ...
 - attr(*, "spec")=
  .. cols(
  ..   TransactionID = col_character(),
  ..   BuyerName = col_character(),
```

Data exploring

```r
dim(data1)  # display the dimension of the data
```

```
[1] 1000   15
```

```r
head(data1)  # display the top 6 row of the data
```

```
# A tibble: 6 × 15
  TransactionID BuyerName FuelType SalesLiters PricePerLitre SalesDate  Terminal
  <chr>         <chr>     <chr>          <dbl>         <dbl> <date>     <chr>
1 TX00001       Total Ni… PMS           43485.           645 2025-02-28 Port Ha…
2 TX00002       Eterna P… AGO           29355.           700 2025-01-09 Lagos J…
3 TX00003       Oando Plc DPK           22492.           645 2025-03-22 Lekki P…
4 TX00004       Eterna P… DPK           36538.           610 2025-04-13 Lagos J…
5 TX00005       Forte Oil DPK           24847.           610 2025-03-10 Lekki P…
6 TX00006       Forte Oil LPG           43798.           700 2025-01-17 Lekki P…
# ℹ 8 more variables: Operator <chr>, VehicleType <chr>, PaymentMethod <chr>,
#   DeliveryStatus <chr>, Region <chr>, InvoiceNumber <chr>, BatchID <chr>,
#   TotalSales <dbl>
```

Data exploring

```r
glimpse(data1)  # display structural overview of the data
```

```
Rows: 1,000
Columns: 15
$ TransactionID  <chr> "TX00001", "TX00002", "TX00003", "TX00004", "TX00005", …
$ BuyerName      <chr> "Total Nigeria", "Eterna Plc", "Oando Plc", "Eterna Plc…
$ FuelType       <chr> "PMS", "AGO", "DPK", "DPK", "DPK", "LPG", "DPK", "AGO",…
$ SalesLiters    <dbl> 43484.54, 29355.10, 22492.16, 36537.63, 24846.78, 43797…
$ PricePerLitre  <dbl> 645, 700, 645, 610, 610, 700, 530, 700, 530, 645, 700, …
$ SalesDate      <date> 2025-02-28, 2025-01-09, 2025-03-22, 2025-04-13, 2025-0…
$ Terminal       <chr> "Port Harcourt Terminal", "Lagos Jetty", "Lekki Port", …
$ Operator       <chr> "Shift C", "Shift B", "Shift B", "Shift B", "Shift D", …
$ VehicleType    <chr> "Trailer", "Trailer", "Container", "Container", "Contai…
$ PaymentMethod  <chr> "Bank Transfer", "Cash", "Bank Transfer", "Bank Transfe…
$ DeliveryStatus <chr> "Cancelled", "Pending", "Cancelled", "Pending", "Delive…
$ Region         <chr> "South-West", "North-East", "South-West", "South-East",…
$ InvoiceNumber  <chr> "INV000001", "INV000002", "INV000003", "INV000004", "IN…
$ BatchID        <chr> "BATCH918", "BATCH345", "BATCH826", "BATCH262", "BATCH4…
$ TotalSales     <dbl> 28047528, 20548570, 14507443, 22287954, 15156536, 30658…
```

```r
View(data1)  # open the entire dataset in a new spreadsheet
```

Data Manipulation

Data subseting

To filter our data, we first load the necessary package for data manipulation (dplyr)

```
filtered_data <- data1 %>% filter(SalesLiters > 30000)
filtered_data
```

```
# A tibble: 420 × 15
   TransactionID BuyerName     FuelType SalesLiters PricePerLitre SalesDate
   <chr>         <chr>         <chr>          <dbl>         <dbl> <date>
 1 TX00001       Total Nigeria PMS           43485.           645 2025-02-28
 2 TX00004       Eterna Plc    DPK           36538.           610 2025-04-13
 3 TX00006       Forte Oil     LPG           43798.           700 2025-01-17
 4 TX00007       NNPC Retail   DPK           45134.           530 2025-02-24
 5 TX00010       Eterna Plc    AGO           30025.           645 2025-02-24
 6 TX00011       Ardova Plc    PMS           45706.           700 2025-06-17
 7 TX00013       Oando Plc     PMS           31525.           645 2025-04-12
 8 TX00014       NNPC Retail   PMS           31946.           610 2025-06-14
 9 TX00015       Eterna Plc    AGO           36923.           645 2025-04-28
10 TX00017       Conoil        LPG           36075.           530 2025-03-09
# ℹ 410 more rows
# ℹ 9 more variables: Terminal <chr>, Operator <chr>, VehicleType <chr>,
#   PaymentMethod <chr>, DeliveryStatus <chr>, Region <chr>,
#   InvoiceNumber <chr>, BatchID <chr>, TotalSales <dbl>
```

Data group by

Observe that the filtered data contains multiple categories of fuel type. To gain deeper insights, we will group the data by fuel type and then summarize key statistics for each group.

```
filtered_data %>%
  group_by(FuelType) %>%
   summarise( mean(SalesLiters),
              median(SalesLiters),
              sd(SalesLiters))
```

```
# A tibble: 4 × 4
  FuelType `mean(SalesLiters)` `median(SalesLiters)` `sd(SalesLiters)`
  <chr>                  <dbl>                 <dbl>             <dbl>
1 AGO                   39717.                40330.             5854.
2 DPK                   40141.                40057.             5710.
3 LPG                   39535.                39642.             6034.
4 PMS                   38842.                38066.             5927.
```

Data cleaning

When preparing data for analysis, it is often necessary to modify the structure of your dataset by adding new columns, removing irrelevant ones, or renaming existing columns for clarity. To add a column, we use the **mutate()** function in *dplyr* as follows:

```r
data1_more <- data1 %>%
  mutate(Expenditure = SalesLiters * PricePerLitre) # Creating a new column

data1_hl <- data1_more %>%
  mutate(Price_Category = ifelse(PricePerLitre > 600, "High", "Low"))   # Creating another column
```

```r
data1_hl <- data1 %>%
  mutate(Expenditure = SalesLiters * PricePerLitre) %>%
  mutate(Price_Category = ifelse(PricePerLitre > 600, "High", "Low"))

dim(data1_hl)
```

```
[1] 1000   17
```

Removing column

Similarly, columns can be removed by using **select()** function from *dplyr*.

```
data1_less <- data1 %>%
  select(-c(Region, InvoiceNumber))  # Removing column 'Region' and 'InvoiceNumber' from the data

dim(data1_less)
```

```
[1] 1000   13
```

Checking for missing values (NA)

Real-world datasets often contain missing values, which can negatively impact the quality of our analysis. It is good practice to identify and handle them before proceeding. In R, the **is.na()** function is commonly used to detect missing values.

```r
x <- c(1, 2, NA, 4)
is.na(x)
```

```
[1] FALSE FALSE  TRUE FALSE
```

...

```r
sum(is.na(data1))  # Counting total missing values
```

```
[1] 0
```

```r
colSums(is.na(data1)) # Finding missing values by column
```

```
 TransactionID      BuyerName       FuelType    SalesLiters  PricePerLitre 
             0              0              0              0              0 
     SalesDate       Terminal       Operator    VehicleType  PaymentMethod 
             0              0              0              0              0 
DeliveryStatus         Region  InvoiceNumber        BatchID     TotalSales 
             0              0              0              0              0 
```

```r
data1_clean <- na.omit(data1)   # Removing NA in data
```

Replacing NA

Missing values can also be replace with **mean**, **median**, **mode** or **a fixed value** in the column for completeness.

```r
data1 <- data1 %>%
  mutate(SalesLiters = ifelse(is.na(SalesLiters),
                              mean(SalesLiters, na.rm = TRUE),
                              SalesLiters))
```

Summary of Filtered Data

```
filtered_data %>%
  group_by(Region) %>%
  summarise(
    Total_Liters = sum(SalesLiters),
    Total_Revenue = sum(TotalSales)
  )
```

```
# A tibble: 6 × 3
  Region        Total_Liters Total_Revenue
  <chr>                <dbl>         <dbl>
1 North-Central     2689212.   1676389685.
2 North-East        2462199.   1516967741.
3 North-West        3186724.   1975223068.
4 South-East        2576748.   1587752157.
5 South-South       3097156.   1929140745.
6 South-West        2599955    1608376159.
```

Data visualization

ggplot

ggplot is a powerful and flexible R package for data visualization, based on the **Grammar of Graphics**. It allows users to create complex, multi-layered plots in a structured and consistent way. It's basic syntax is

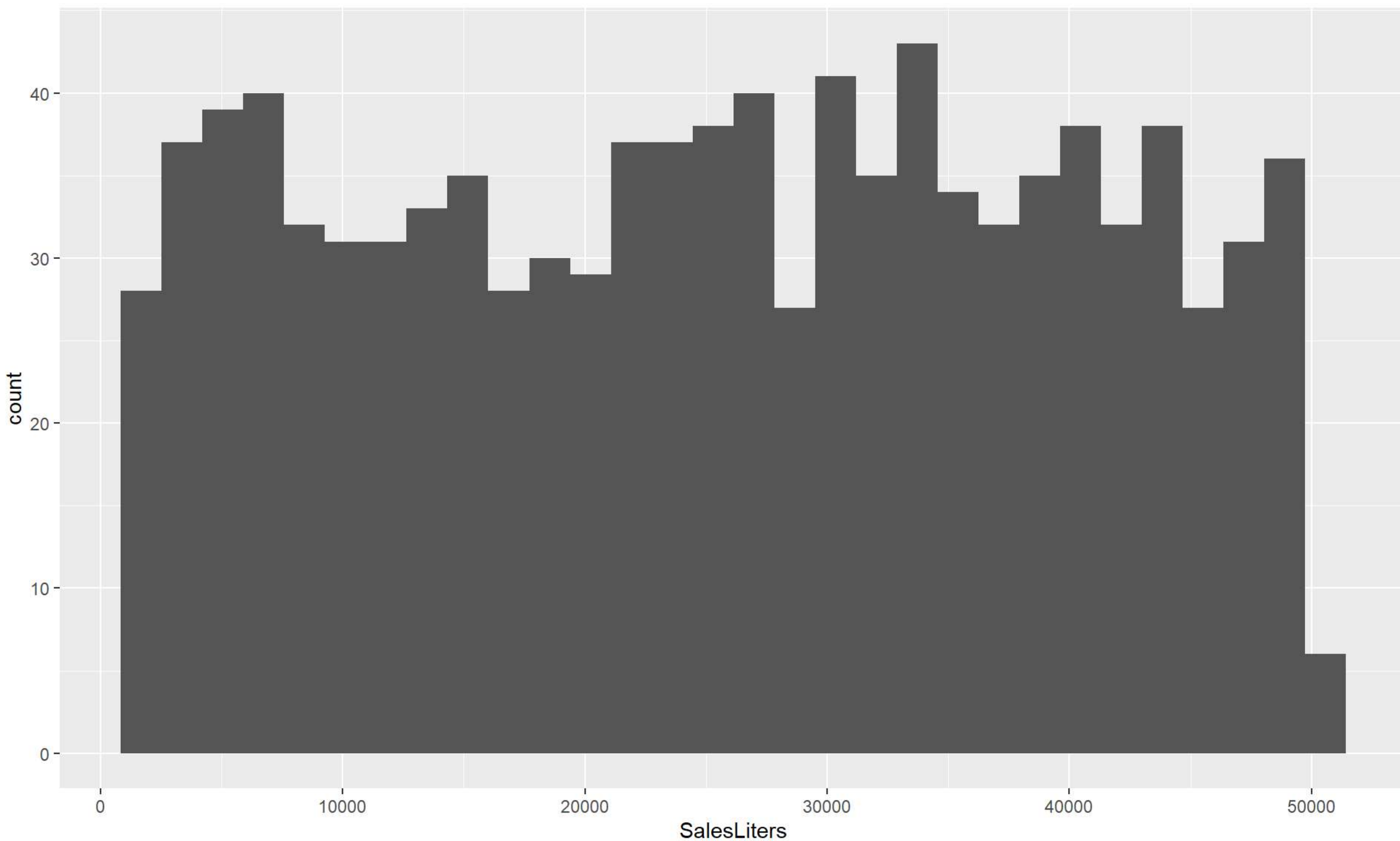
ggplot(data, aes(x = variable1, y = variable2)) + geom_style()

```
library(ggplot2)    # loading the necessary package
```

Histogram

A histogram is a graphical representation of the distribution of a numeric variable. It divides the data into intervals (bins) and shows the count (or frequency) of values within each bin. For instance, Understanding the shape of the data (e.g., normal, skewed), Detecting outliers, spread, and central tendency.

```r
ggplot(data1, aes(x = SalesLiters)) +
  geom_histogram()            #Plotting a histogram
```

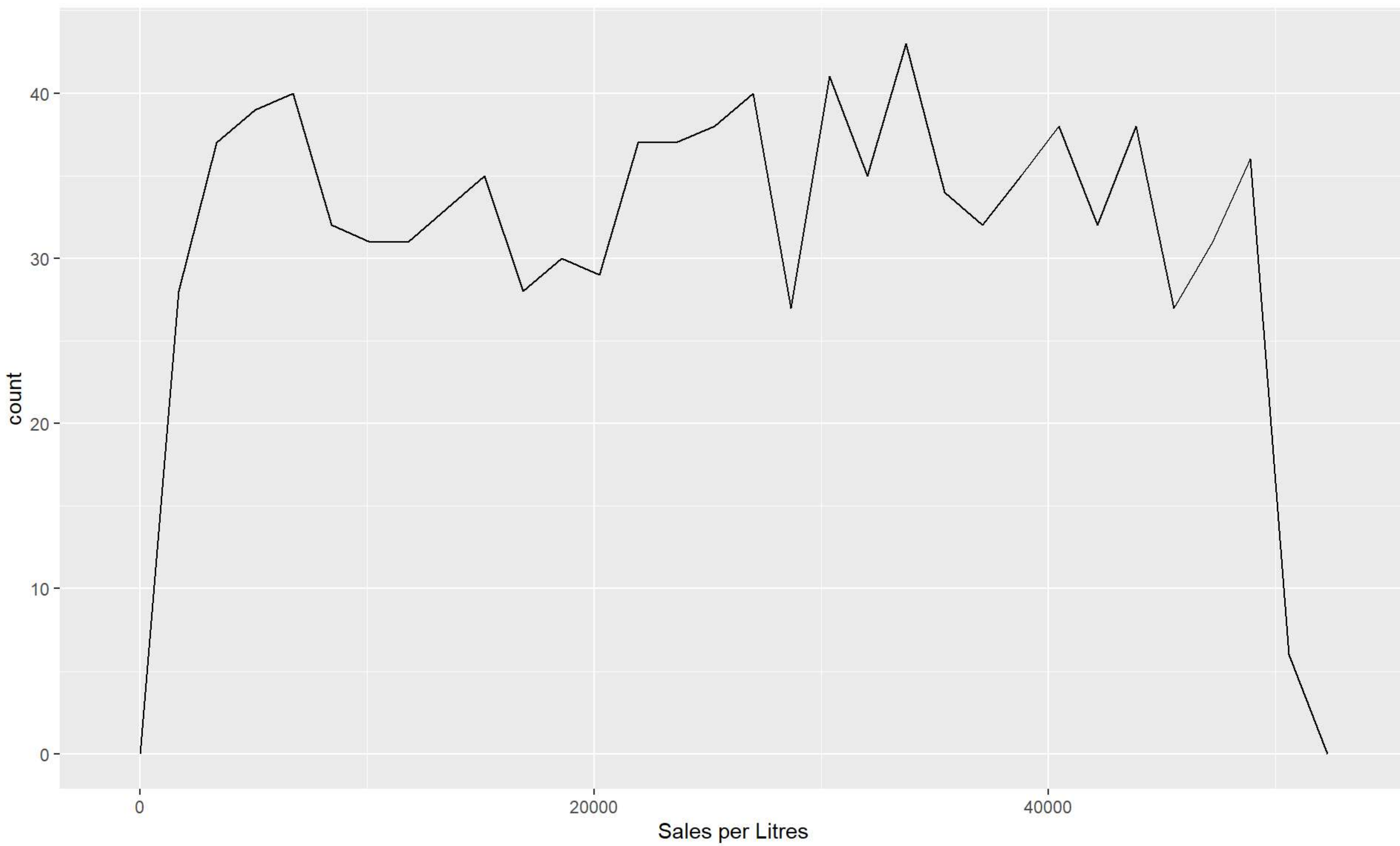


count
40
30
20
10
0
0
10000
20000
30000
40000
50000
SalesLiters

Frequency Polygon

A frequency polygon is a line graph that connects the midpoints of the top of each histogram bin. It shows the distribution shape more smoothly and is useful for comparing multiple distributions. It shows trends or patterns more clearly than histograms.

```r
ggplot(data1, aes(x = SalesLiters)) +
  geom_freqpoly() +
  labs(x = "Sales per Litres")    # Plotting a frequency polygon
```

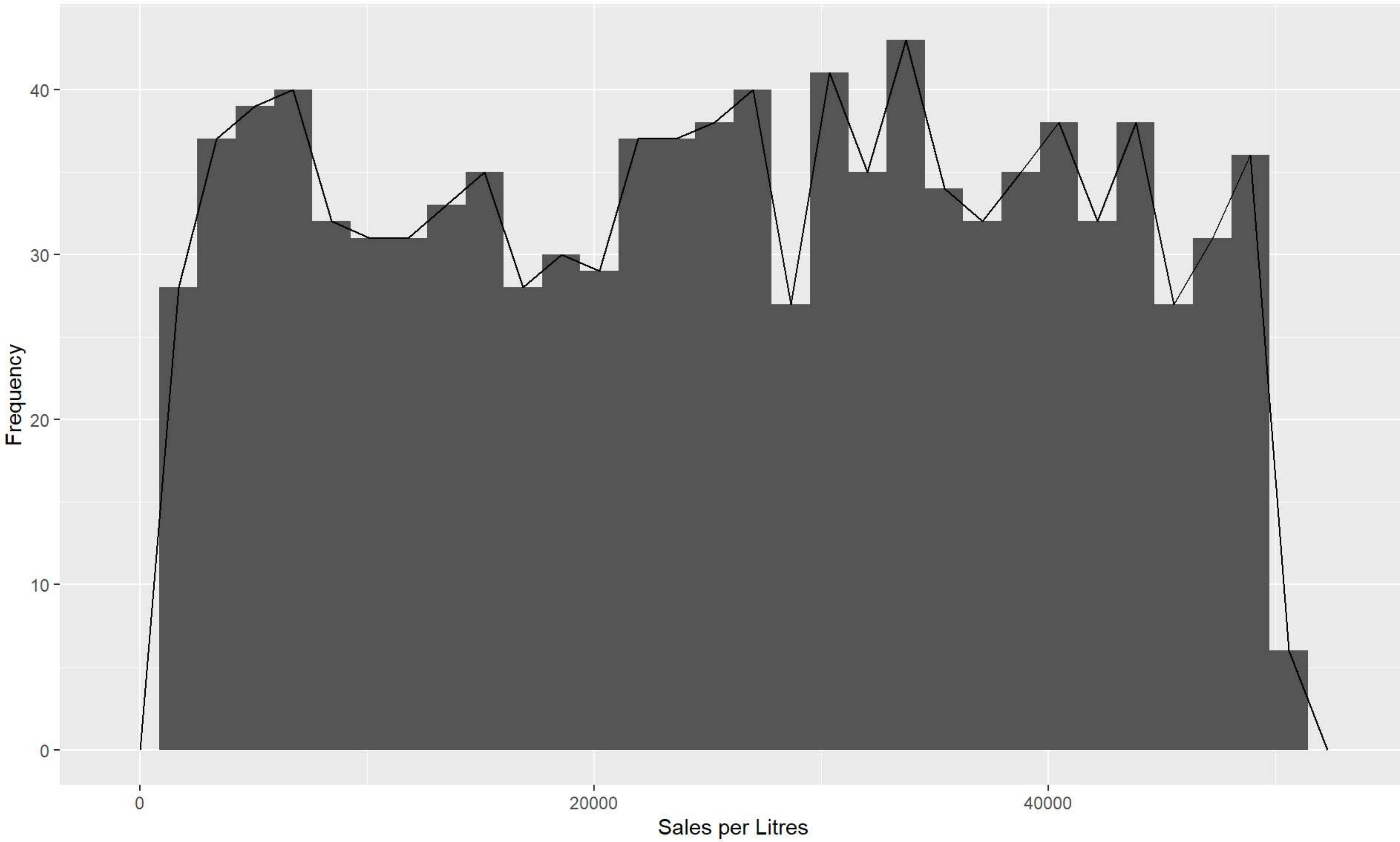


count
40
30
20
10
0
0
20000
40000
Sales per Litres

Histogram + Frequency Polygon

Typically, it is good to combine both a histogram and a frequency polygon to provide a more comprehensive view of the data distribution. Using the histogram to show actual counts per bin and the frequency polygon to highlight the overall shape or trend of the distribution.

```
ggplot(data1, aes(x = SalesLiters)) +
  geom_histogram() +
  geom_freqpoly() +
  labs(x = "Sales per Litres") +
  labs(y = "Frequency")
```


Frequency
40
30
20
10
0
0
20000
40000
Sales per Litres

Bar Plots

A bar plot is used to display the frequency or value of categorical variables. Each bar represents a category, and the height of the bar corresponds to its value or count. It is typical used for:

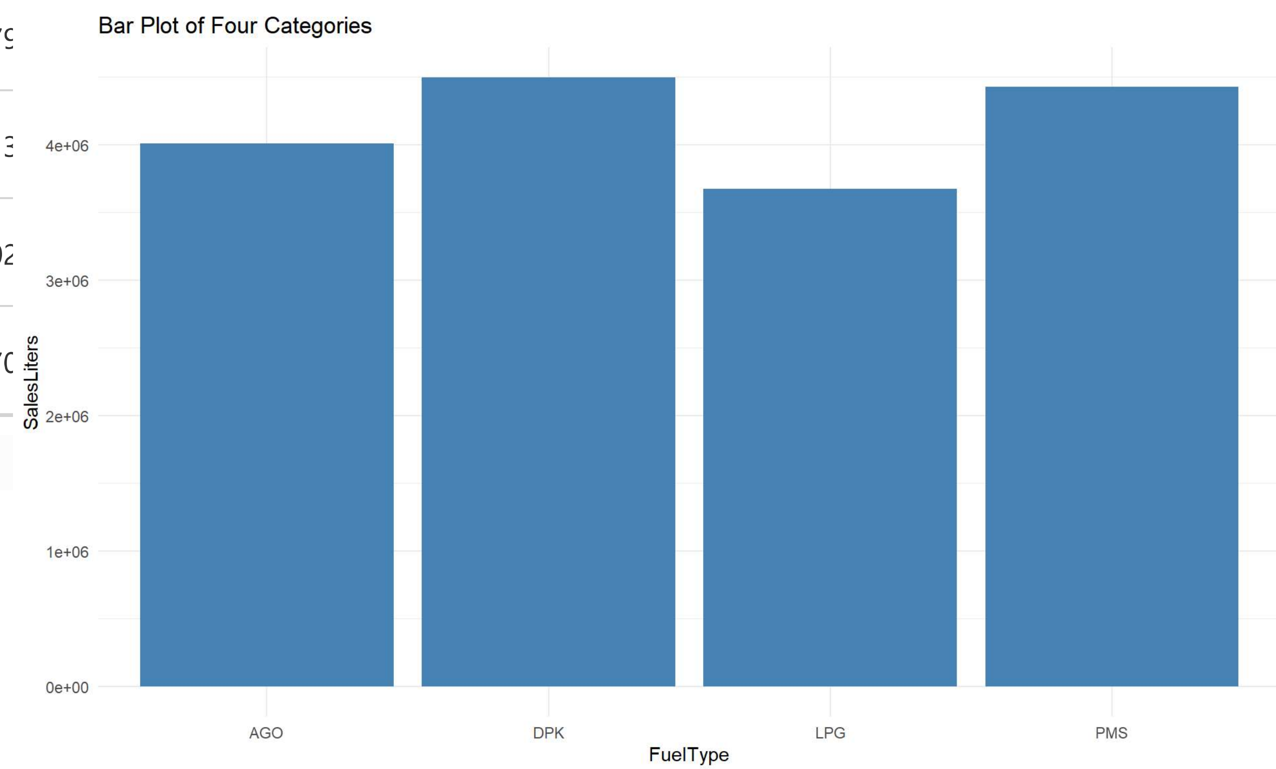
- Compare values of categories.
- Show the number of observations per category (e.g., fuel types, regions)
- Visualize grouped data.

Syntax : ggplot(data, aes(x = Category, y = Value)) + geom_bar(stat = “identity”) # for pre-summarized values

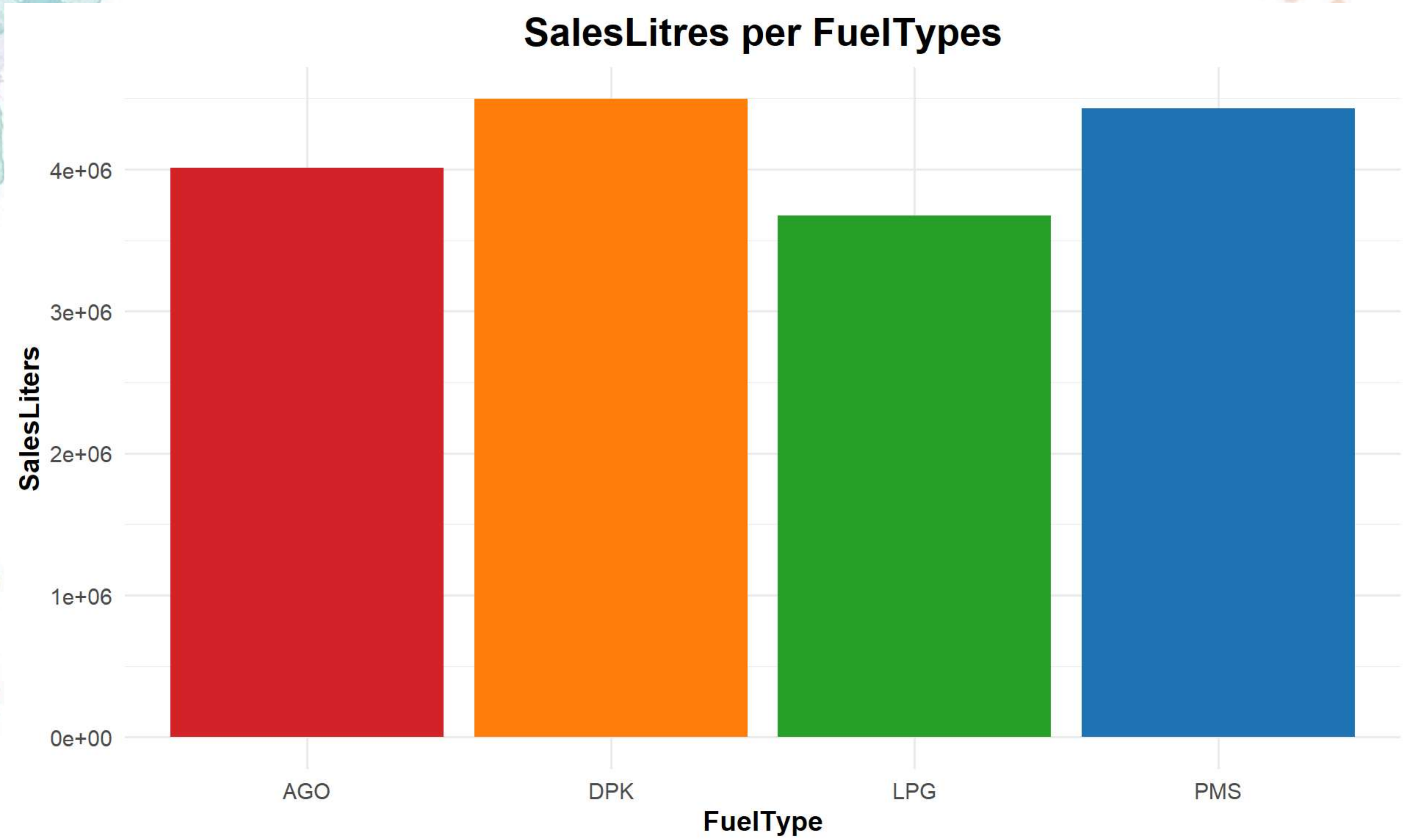
Filtered data

TransactionID	BuyerName	FuelType	SalesL
TX00001	Total Nigeria	PMS	4348
TX00004	Eterna Plc	DPK	3653
TX00006	Forte Oil	LPG	4379
TX00007	NNPC Retail	DPK	4513
TX00010	Eterna Plc	AGO	3002
TX00011	Ardova Plc	PMS	4570

```r
ggplot(filtered_data, aes(x = FuelType, y = SalesLiters)
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "Bar Plot of Four Categories",
       x = "FuelType",
       y = "SalesLiters") +
  theme_minimal()
```



```r
ggplot(filtered_data, aes(x = FuelType, y = SalesLiters, fill = FuelType)) +
  geom_bar(stat = "identity") +
  labs(
       title = "SalesLitres per FuelTypes",
       x = "FuelType",
       y = "SalesLiters") +
  scale_fill_manual(values = c("PMS" = "#1f77b4",   # Blue
                               "DPK" = "#ff7f0e",   # Orange
                               "LPG" = "#2ca02c",   # Green
                               "AGO" = "#d62728")) +  # Red
theme_minimal(base_size = 14) +
  theme(
    plot.title = element_text(face = "bold", size = 20, hjust = 0.5),
    axis.title = element_text(face = "bold"),
    legend.position = "none"
  )
```

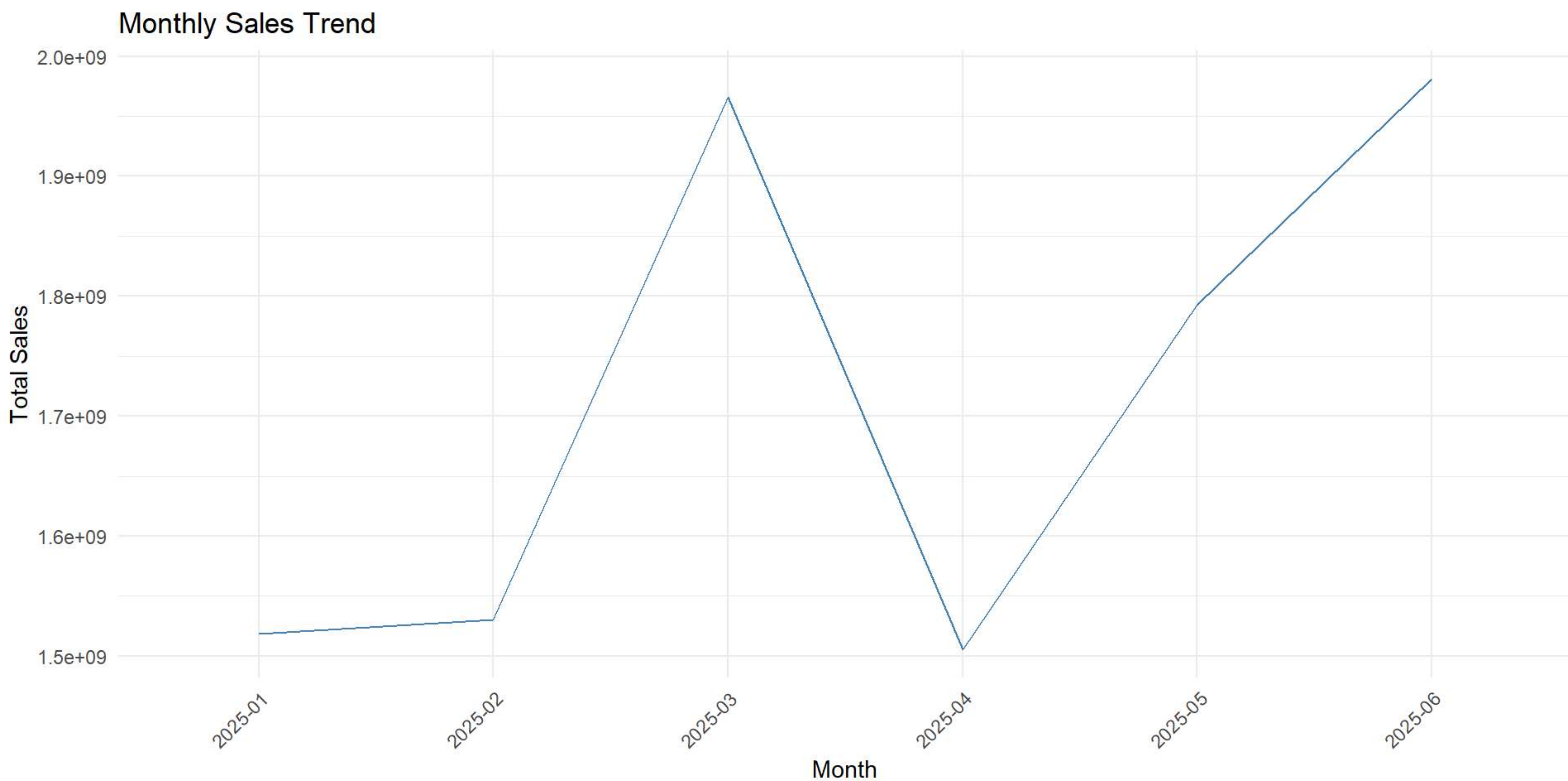

SalesLitres per FuelTypes
SalesLiters
4e+06
3e+06
2e+06
1e+06
0e+00
AGO
DPK
LPG
PMS
FuelType

Line Plots

Line Plots

A line plot displays data points connected by straight lines. It is commonly used to visualize trends over time or continuous variables. E.g, Tracking monthly or daily sales, Visualizing temperature or price changes over time.

```
filtered_data %>%
  mutate(Month = format(SalesDate, "%Y-%m")) %>%
  group_by(Month) %>%
  summarise(Monthly_Sales = sum(TotalSales)) %>%
  ggplot(aes(x = Month, y = Monthly_Sales)) +
  geom_line(group = 1, color = "steelblue") +
  theme_minimal() +
  labs(title = "Monthly Sales Trend", x = "Month", y = "Total Sales") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

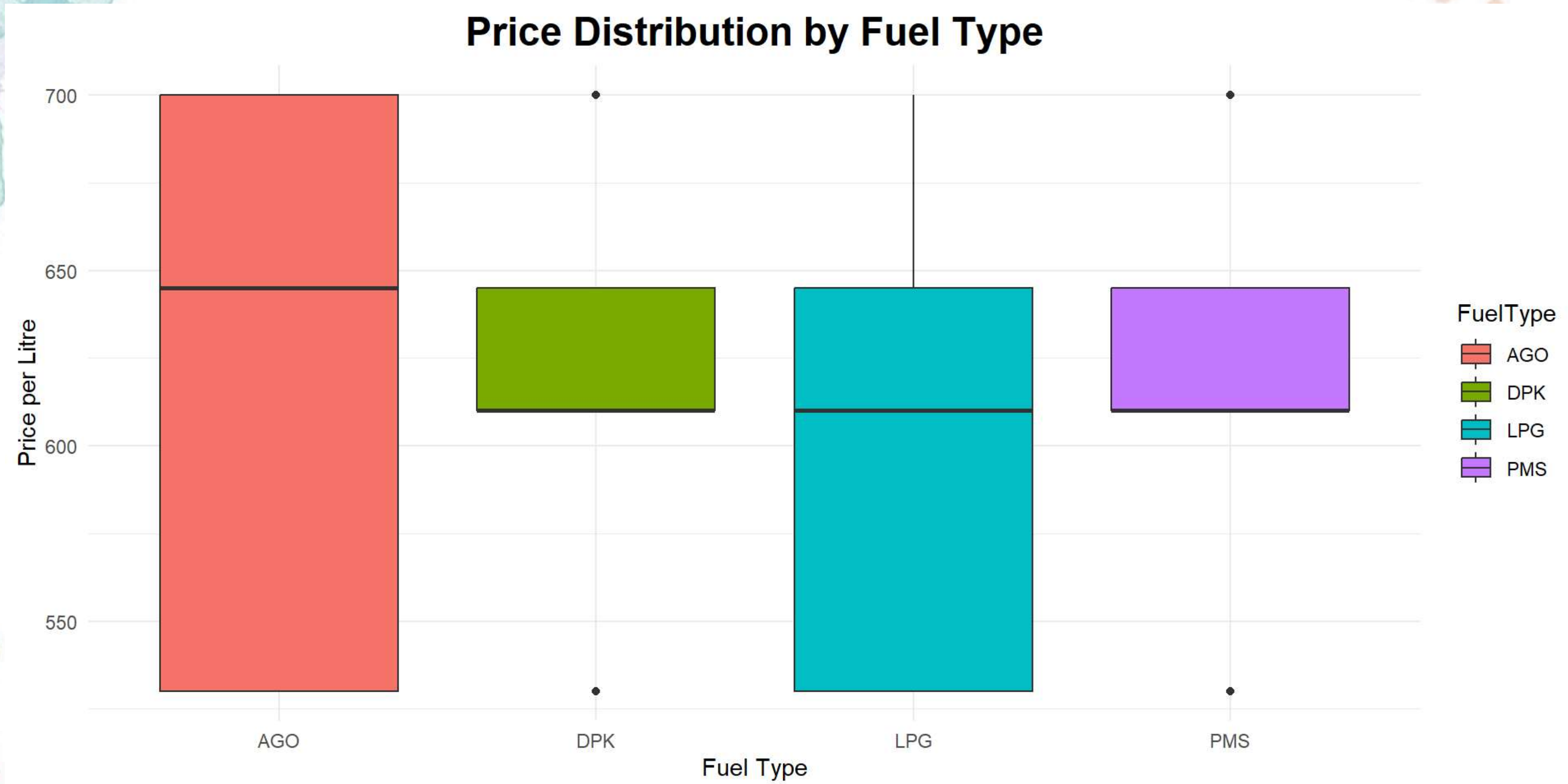

Monthly Sales Trend
2.0e+09
1.9e+09
1.8e+09
1.7e+09
1.6e+09
1.5e+09
Total Sales
2025-01
2025-02
2025-03
2025-04
2025-05
2025-06
Month

Box Plots

Box Plots

A box plot (also called a box-and-whisker plot) summarizes the distribution of a numeric variable using five statistics: Minimum, 1st quartile (Q1), Median, 3rd quartile (Q3), and Maximum.

```r
ggplot(filtered_data, aes(x = FuelType, y = PricePerLitre, fill = FuelType)) +
  geom_boxplot() +
  labs(title = "Price Distribution by Fuel Type", x = "Fuel Type", y = "Price per
  theme_minimal() +
  theme(plot.title = element_text(face = "bold", size = 18, hjust = 0.5))
```


Price Distribution by Fuel Type
700
650
600
550
Price per Litre
AGO
DPK
LPG
PMS
Fuel Type
FuelType
AGO
DPK
LPG
PMS

Summary

Important

In this class, we have covered the following topics:

- Basic of data preprocessing: str(), head(), dim(), glimpse(), View()
- Basic of data manipulation using *dplyr()*: filter(), group_by(), summarise(), mutate(), select(), is.na()
- Basic of data visualization: histogram(), freqploy(), barplot(), lineplot(), boxplot(), customize plot.
- Rmarkdown: Use to prepare reports. **This note is prepared using Rmarkdown**