Introduction to Tidyverse and Exploratory Data Visualization with ggplot2

1. Introduction

The **Tidyverse** is an essential collection of R packages designed for data science. It includes packages such as:

- **ggplot2** for data visualization
- **dplyr** for data manipulation
- **tidyr** for data tidying
- **readr** for reading data
- **tibble** for working with data frames
- **forcats** for handling categorical variables

This tutorial will guide you through using tidyverse for data manipulation and creating exploratory data visualizations using ggplot2.

---

2. Installing and Loading the Tidyverse

If you haven't installed tidyverse, install it using:

```
install.packages("tidyverse")
```

Then, load the package:

```
library(tidyverse)
```

---

3. Importing and Inspecting Data

We will use the **NHANES** dataset from the NHANES package, which contains health survey data.

First, install and load the package if needed:

```
install.packages("NHANES")
library(NHANES)
```

Load the dataset:

```
# Load dataset
health_data <- NHANES
```

```
# View first few rows
glimpse(health_data)
```

Key functions for inspecting data:

```
head(health_data)     # First few rows
dim(health_data)      # Dimensions of the dataset
summary(health_data)  # Summary statistics
colnames(health_data)  # Column names
```

---

4. Data Manipulation with dplyr

4.1 Selecting Columns

```
health_data %>% select(Age, Gender, BMI, BPSysAve, TotChol)
```

4.2 Filtering Rows

```
health_data %>% filter(Age > 50, BMI > 18.5) #18.5 is normal weight
```

4.3 Creating New Variables

```
health_data <- health_data %>% mutate(BMI_Category = ifelse(BMI > 25, "Overweight", "Normal"))
```

4.4 Summarizing Data

```
health_data %>% group_by(Gender) %>% summarise(avg_BMI = mean(BMI, na.rm = TRUE))
```

4.5 Identifying Missing Data, Cardinality, and Outliers

Identifying Missing Data

Missing data can impact analysis. We can check for missing values using:

```
sum(is.na(health_data))  # Total missing values
colSums(is.na(health_data))  # Missing values per column
```

To remove missing values:

```
health_data <- health_data %>% drop_na()
```

Imputing Missing Data

Instead of dropping missing values, we can impute them using different strategies:

- Fill with the mean:

```
health_data <- health_data %>% mutate(BMI = ifelse(is.na(BMI), mean(BMI, na.rm = TRUE), BMI))
```

- Fill with the median:

```
health_data <- health_data %>% mutate(BMI = ifelse(is.na(BMI), median(BMI, na.rm = TRUE), BMI))
```

- Fill using group-wise mean:

```
health_data <- health_data %>% group_by(Gender) %>% mutate(BMI = ifelse(is.na(BMI), mean(BMI, na.rm = TRUE), BMI)) %>% ungroup()
```

Checking Cardinality (Unique Values in Columns)

Cardinality refers to the number of unique values in a column:

```
sapply(health_data, function(x) length(unique(x)))
```

This helps identify categorical variables with too many unique values, which might not be useful for modeling.

Detecting Outliers

Boxplots help visualize outliers:

```
ggplot(health_data, aes(x = Gender, y = BMI)) +
  geom_boxplot() +
  labs(title = "BMI Outliers by Gender")
```

Another method is using the **interquartile range (IQR)**:

```
Q1 <- quantile(health_data$BMI, 0.25, na.rm = TRUE)
Q3 <- quantile(health_data$BMI, 0.75, na.rm = TRUE)
IQR <- Q3 - Q1
outliers <- health_data %>% filter(BMI < (Q1 - 1.5 * IQR) | BMI > (Q3 + 1.5 * IQR))
outliers
```

This identifies extreme BMI values that might need further investigation.

---

5. Exploratory Data Visualization with ggplot2

5.1 Histogram

A histogram is useful for visualizing the distribution of a single variable.

```
ggplot(health_data, aes(x = BMI)) +
  geom_histogram(binwidth = 2, fill = "blue", color = "black") +
  labs(title = "BMI Distribution", x = "BMI", y = "Count")
```

5.2 Scatterplot

```
ggplot(health_data, aes(x = Age, y = BPSysAve, color = Gender)) +
  geom_point(size = 3) +
  labs(title = "Age vs. Blood Pressure", x = "Age", y = "Blood Pressure")
```

5.3 Boxplot

```
ggplot(health_data, aes(x = Gender, y = BMI)) +
  geom_boxplot(fill = "lightblue") +
  labs(title = "BMI by Gender", x = "Gender", y = "BMI")
```

5.4 Density Plot

```
ggplot(health_data, aes(x = BMI, fill = Gender)) +
  geom_density(alpha = 0.5) +
  labs(title = "Density Plot of BMI by Gender", x = "BMI", y = "Density")
```

5.5 Bar Chart

```
ggplot(health_data, aes(x = BMI_Category, fill = Gender)) +
  geom_bar(position = "dodge") +
  labs(title = "Count of BMI Categories by Gender", x = "BMI Category", y = "Count")
```

---

6. Customizing ggplot2 Visualizations

6.1 Modifying Themes

```
ggplot(health_data, aes(x = Age, y = BPSysAve)) +
  geom_point(color = "blue") +
  theme_minimal() +
  labs(title = "Customized Scatterplot", x = "Age", y = "Blood Pressure")
```

Other available themes include:

theme_classic(), theme_light(), theme_dark(), theme_bw()

6.2 Customizing Axis Labels and Titles

```
ggplot(health_data, aes(x = Age, y = BMI)) +
  geom_point() +
  labs(title = "Age vs BMI", subtitle = "NHANES Dataset", x = "Age (years)", y = "Body Mass Index (BMI)")
```

6.3 Changing Legends and Colors

```
ggplot(health_data, aes(x = Age, y = BMI, color = Gender)) +
  geom_point() +
  scale_color_manual(values = c("Male" = "blue", "Female" = "red")) +
  labs(title = "Age vs BMI by Gender")
```

6.4 Customizing Grid Lines

```
ggplot(health_data, aes(x = Age, y = BPSysAve)) +
  geom_point() +
  theme(panel.grid.major = element_line(color = "grey", size = 0.5),
        panel.grid.minor = element_line(color = "lightgrey", size = 0.25))
```

6.5 Adding Annotations

```
ggplot(health_data, aes(x = Age, y = BMI)) +
  geom_point() +
  annotate("text", x = 80, y = 40, label = "Possible Outlier", color = "red", size = 5)
```

---

7. Saving Plots

To save a plot, use ggsave():

```
ggplot(health_data, aes(x = BMI)) +
  geom_histogram(binwidth = 2, fill = "blue", color = "black")

ggsave("bmi_histogram.png", width = 8, height = 6)
```
