Assignment 1 - Chapter 1-7 Summary LK

Lynnstacy Kegeshi

2024-10-27

Contents

This is a summary of Chapter 1-7 of the book **R for Data Science**. We shall be using be data set *Purchase Prediction* to perform our analysis based on the content from Chapter 1 - 7.

We shall be using the following libraries

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
```

```
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(dplyr)
library(modelr)
```

Chapter 1

The chapter focuses on ggplot2, part of the tidyverse collection of R packages.To get started on this, we shall load our dataset to R Studio and visualize it.

```
purchaseprediction<-read.csv("Purchase Prediction.csv")
head(purchaseprediction)
```

```
##   CUSTOMER_ID ProductChoice MembershipPoints ModeOfPayment ResidentCity
## 1           1             2                6   MoneyWallet      Madurai
## 2           2             3                2    CreditCard      Kolkata
## 3           3             2                4   MoneyWallet   Vijayawada
## 4           4             3                2   MoneyWallet       Meerut
## 5           5             2                6   MoneyWallet      Madurai
## 6           6             3                6     DebitCard      Kolkata
##   PurchaseTenure Channel IncomeClass CustomerPropensity CustomerAge
## 1              4  Online           4             Medium          55
## 2              4  Online           7           VeryHigh          75
## 3             10  Online           5            Unknown          34
## 4              6  Online           4                Low          26
## 5              3  Online           7           VeryHigh          38
## 6              3  Online           4               High          71
##   MartialStatus LastPurchaseDuration
## 1             0                   4
## 2             0                  15
## 3             0                  15
## 4             0                   6
## 5             1                   6
## 6             0                  10
```

A scatter plot is a chart type that is normally used to observe and visually display the relationship between variables. In this case, we used it to explore the relationship between customer age and last purchase duration. We also incorporated the channel variable to gain insights into the mode of shopping and its distribution. This approach helps us understand how age affects purchasing behavior, providing a clearer picture of customer engagement over time.

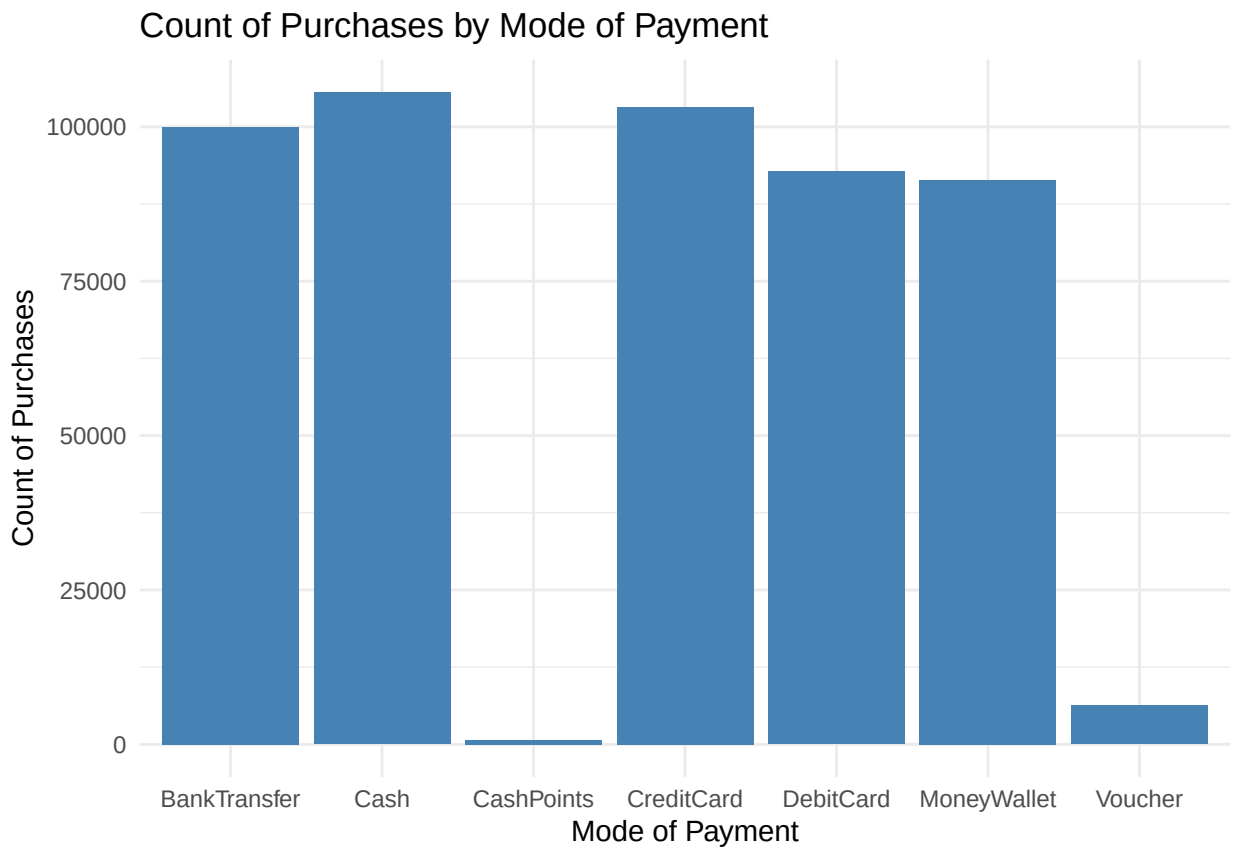
By adding facets based on customer propensity, we can further break down the data and see how different customer segments behave, making it easier to identify patterns and trends in their purchasing habits.

```
ggplot(data = purchaseprediction, mapping = aes(x = CustomerAge, y = LastPurchaseDuration)) +
  geom_point(aes(colour = Channel)) +
  labs(title = "Relationship Between Customer Age and Last Purchase Duration",
       x = "Customer Age",
       y = "Last Purchase Duration") +
  theme_minimal() +
  facet_wrap(~ CustomerPropensity)
```

```
## Warning: Removed 14056 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

We also created a bar graph. By plotting the count of purchases for each ModeOfPayment, we can quickly identify which payment methods are most popular among customers and understand their preferences better.

```
# Create a bar graph of counts
ggplot(data = purchaseprediction, aes(x = ModeOfPayment)) +
  geom_bar(fill = "steelblue") +
  labs(title = "Count of Purchases by Mode of Payment",
       x = "Mode of Payment",
       y = "Count of Purchases") +
  theme_minimal()
```

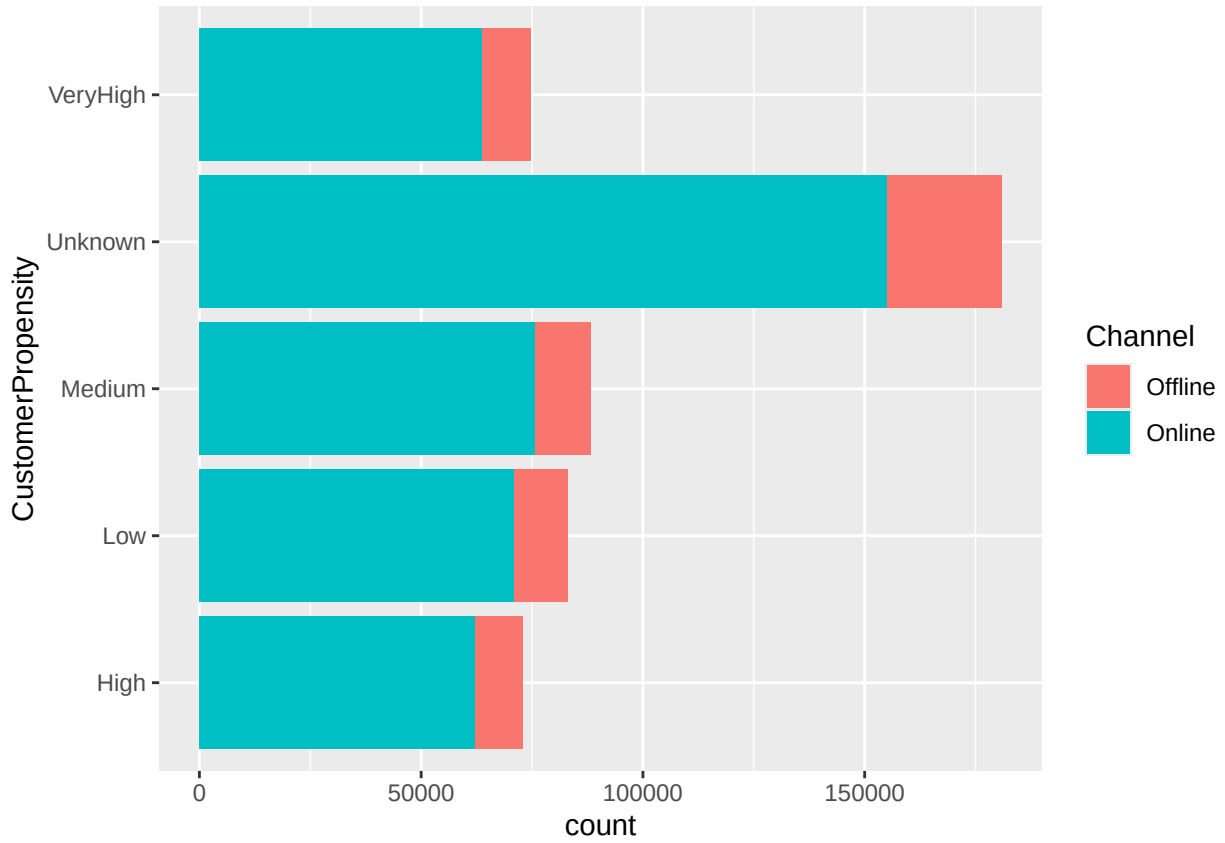


Different coordinate systems can be employed for specific visualization needs. For instance, if a box plot is preferred, the coordinate system can be flipped to enhance readability.

```
# Define the bargraph object
bargraph <- ggplot(data = purchaseprediction) +
  geom_bar(mapping = aes(x = CustomerPropensity, fill = Channel))

# Use the `bargraph` object with coord_polar()
bargraph + coord_polar()
```


```
# Or use the `bargraph` object with coord_flip()
bargraph + coord_flip()
```



Chapter 2

This chapter explore the basics of running R code and explored the most helpful features of R studio.

- R can be used as a calculator for basic arithmetics.

```
1 / 200 * 30
```

```
## [1] 0.15
```

```
(59 + 73 + 2) / 3
```

```
## [1] 44.66667
```

```
sin(pi / 2)
```

```
## [1] 1
```

- We can also create new objects using <-. This is also demonstrated in Chapter 1 summary where created object purchase predictions and assigned it to the data set.Object names in R must start with a letter and can contain letters, numbers, _, and .
- R is case-sensitive, and typos matter.
- The continuation character + indicates that R is waiting for more input.

Chapter 3

In Chapter 3 of R for Data Science, the focus is on using the dplyr package, a core member of the tidyverse, for data manipulation.

The chapter introduces five key dplyr functions that address most data manipulation challenges:

1. filter(): Selects observations based on their values.
2. arrange(): Reorders the rows.
3. select(): Picks variables by their names.
4. mutate(): Creates new variables from existing ones.
5. summarize(): Collapses many values into a single summary.

These functions can be combined with group_by() to operate on groups within the dataset.

Using the Purchase Prediction dataset, we shall explore the 5 dplyr functions. The chapter also introduces the pipe %>% to chain multiple operations together, making the code easier to read.

1.filter(): Select Observations Based on Their Values

Suppose you want to filter for customers who are over 40 years old.

```
# Select customers over 40 years old
older_customers <- purchaseprediction %>%
  filter(CustomerAge > 40)

#Display first 10 rows of older_customer
head(older_customers$CustomerAge, 10)
```

```
##  [1] 55 75 71 72 56 72 67 55 72 58
```

2.arrange(): Reorder Rows

To see the data ordered by Membership Points in descending order, use arrange():

```
# Order by Membership Points, highest first
ordered_data <- purchaseprediction %>%
  arrange(desc(MembershipPoints))

# Display the first five rows of MembershipPoints from ordered_data
head(ordered_data$MembershipPoints, 15)
```

```
##  [1] 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
```

3. select(): Pick Variables by Their Names

To focus on only a few columns, like CustomerID, CustomerAge, and MembershipPoints:

```
# Select specific columns
selected_columns <- purchaseprediction %>%
  select(CUSTOMER_ID, CustomerAge, MembershipPoints)

head(selected_columns)
```

```
##   CUSTOMER_ID CustomerAge MembershipPoints
## 1           1          55                6
## 2           2          75                2
## 3           3          34                4
## 4           4          26                2
## 5           5          38                6
## 6           6          71                6
```

4. mutate(): Create New Variables

You could create a new column for MembershipLevel, where you categorize customers based on Membership Points.

```
# Create a new variable based on Membership Points
mutated_data <- purchaseprediction %>%
  mutate(MembershipLevel = ifelse(MembershipPoints > 5, "High", "Low"))

head(mutated_data$MembershipLevel)
```

```
## [1] "High" "Low"  "Low"  "Low"  "High" "High"
```

5. summarize(): Collapse Many Values into a Single Summary

To find the average CustomerAge and MembershipPoints, you can use summarize():

```
# Calculate average age and membership points
summary_data <- purchaseprediction %>%
  summarize(
    avg_age = mean(CustomerAge, na.rm = TRUE),
    avg_points = mean(MembershipPoints, na.rm = TRUE)
  )
summary_data
```

```
##    avg_age avg_points
## 1 44.95875    4.22006
```

The summarize() function collapses a data frame to a single row. When used with group_by(), it provides grouped summaries.

```
# Group by ProductChoice and summarize with average MembershipPoints, ordered by MembershipPoints
ordered_data <- purchaseprediction %>%
  group_by(ProductChoice) %>%
  summarize(avg_membership_points = mean(MembershipPoints, na.rm = TRUE)) %>%
  arrange(desc(avg_membership_points))

# Display the first five rows of the result
head(ordered_data, 5)
```

```
## # A tibble: 4 x 2
##   ProductChoice avg_membership_points
```

```
##           <int>                <dbl>
## 1             3                 4.52
## 2             1                 4.32
## 3             2                 4.10
## 4             4                 3.65
```

Chapter 4

Chapter 4 explores running the code on the R-Script rather than the console.

Key Take Aways - The script editor can be used to store code that works - Always start scripts with the necessary library() calls to make it clear what packages are needed. - Avoid including install.packages() or setwd() in shared scripts to prevent changing settings on someone else's computer. - Hovering over the cross when an error is identified reveals the problem, helping to quickly identify and fix errors.

Chapter 5

This chapter focuses on using visualisation and exploratory data analysis. It combine the knowledge of dplyr and ggplot.

Exploratory Data Analysis is driven by questions that guide the investigation.

1. What type of variation occurs within my variables?
2. What type of covariation occurs between my variables?

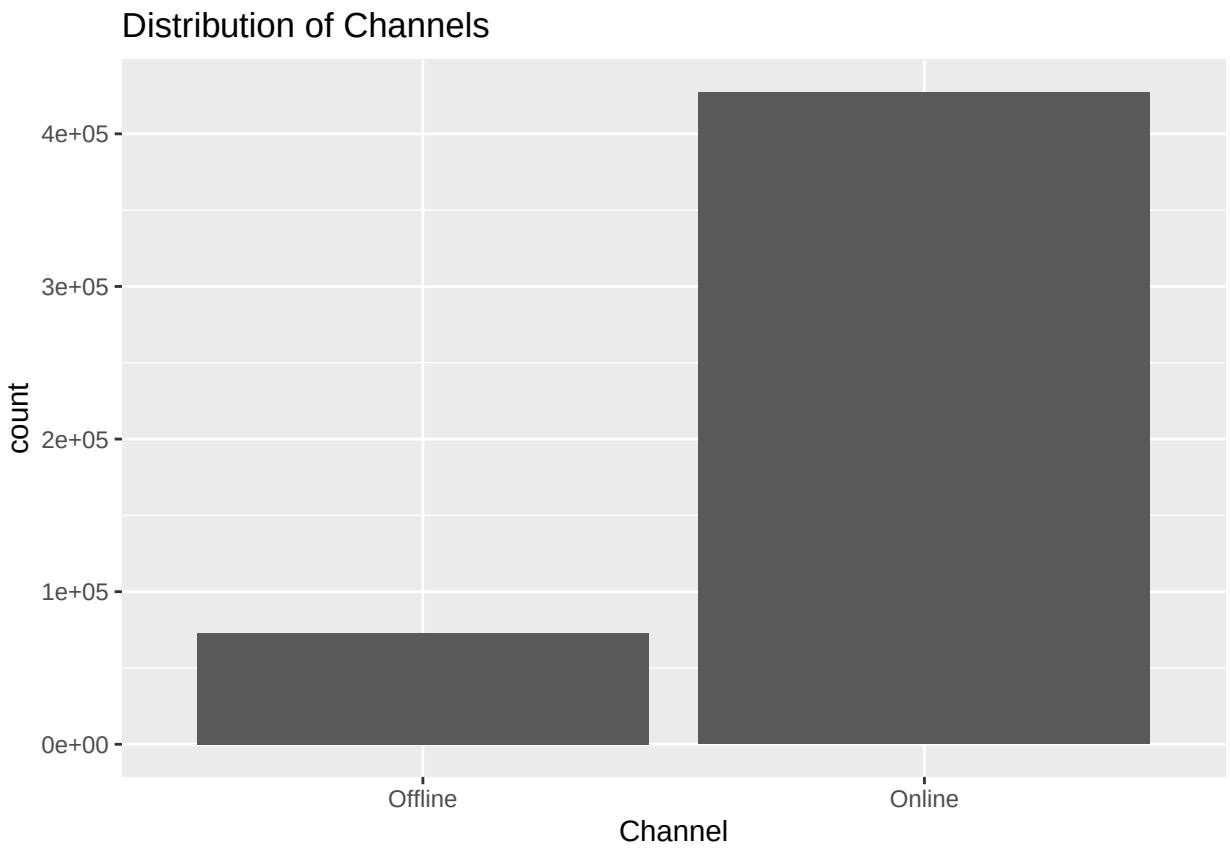
Variation refers to the tendency of a variable's values to change from measurement to measurement.

Visualizing Distributions

This depends on whether the data is categotical or continous.

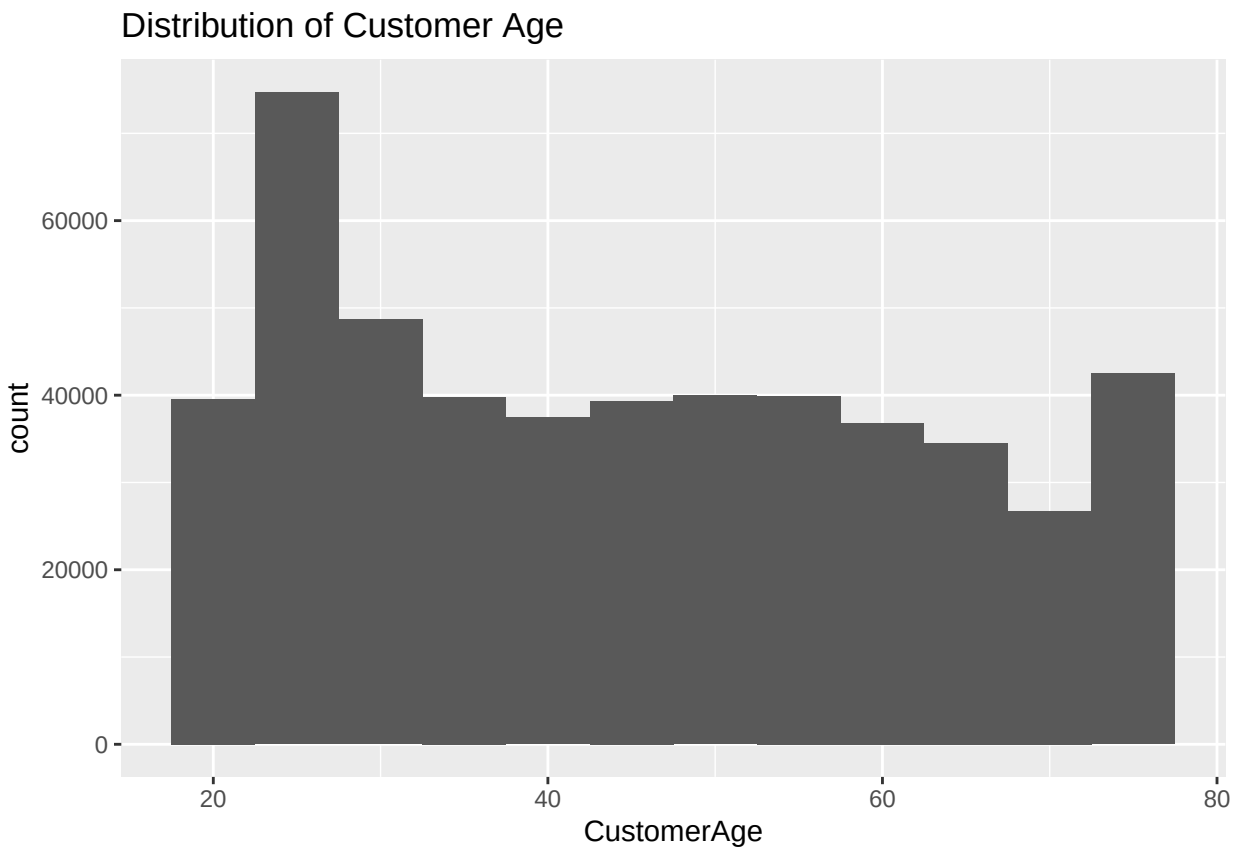
Categorical

```
ggplot(data = purchaseprediction) +
  geom_bar(mapping = aes(x = Channel)) +
  labs(title = "Distribution of Channels")
```



Continous

```
ggplot(data = purchaseprediction) +
  geom_histogram(mapping = aes(x = CustomerAge), binwidth = 5) +
  labs(title = "Distribution of Customer Age")
```



Identifying Outliers

Outliers don't fit the pattern. We can handle outliers by replacing values outside a certain range by NA.

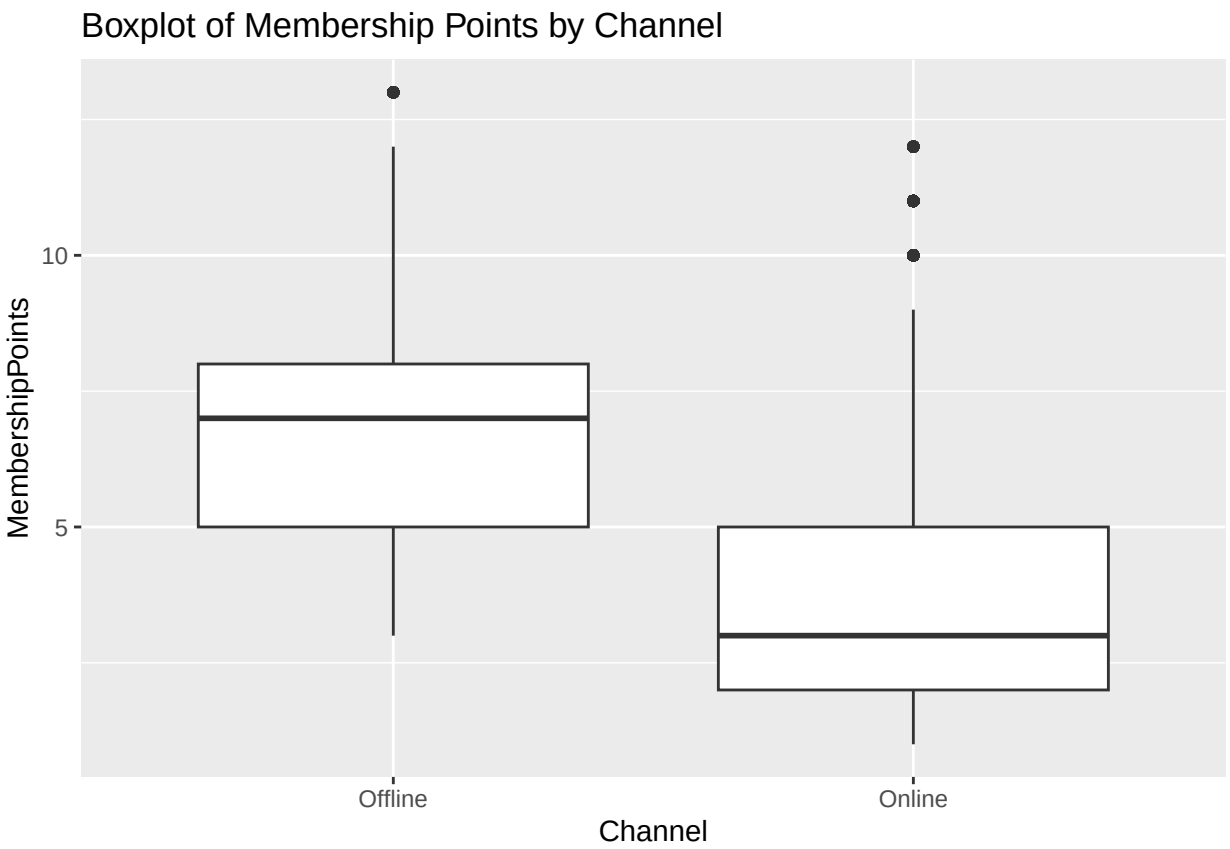
By replacing extreme values with NA, the code helps in handling outliers that could skew the analysis.

Visualizing Covariation

Covariation means that changes in one variable are associated with changes in another variable.

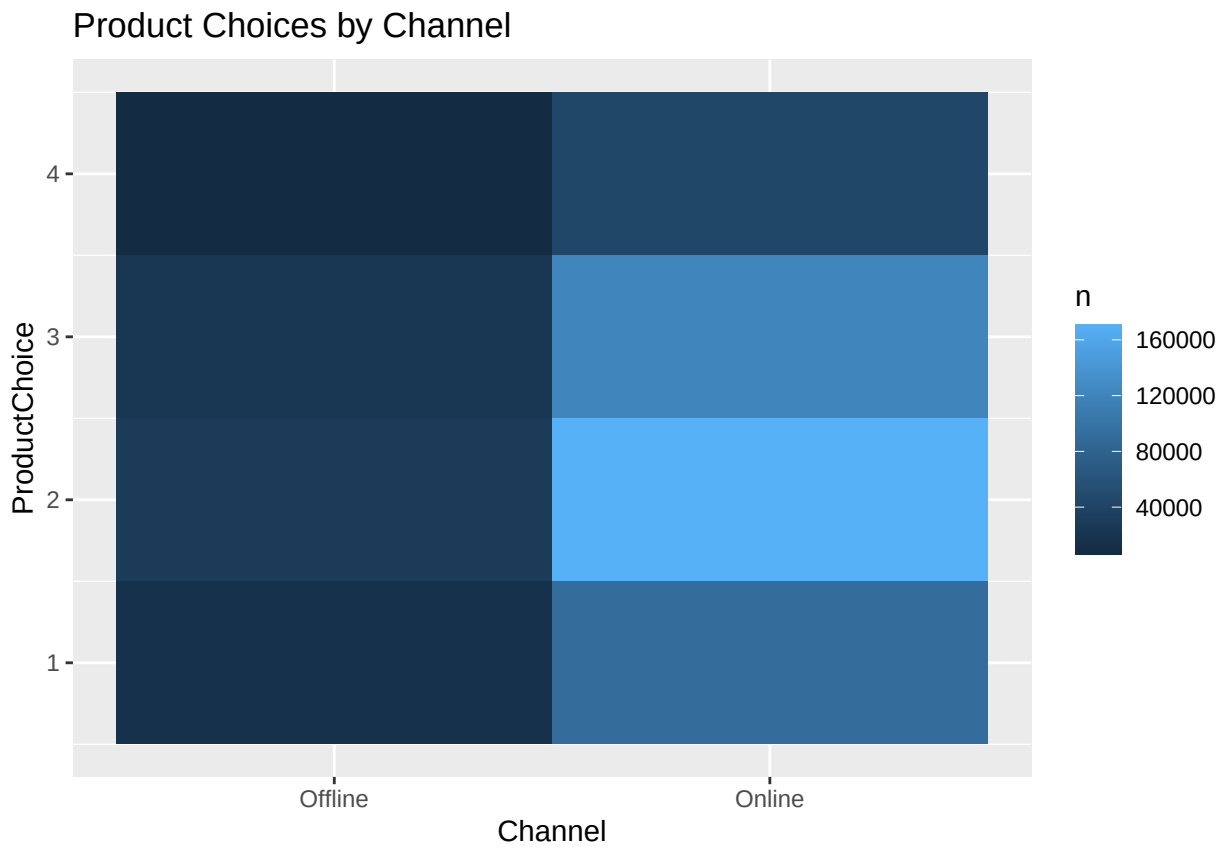
This can be done using a box plot or a density plot when dealing with a categorical variable and continous variable. ### Box plot

```
ggplot(data = purchaseprediction, mapping = aes(x = Channel, y = MembershipPoints)) +
  geom_boxplot() +
  labs(title = "Boxplot of Membership Points by Channel")
```



When visualizing with two categorical variables, a count plot or a tile plot can be used.

```
purchaseprediction %>%
  count(Channel, ProductChoice) %>%
  ggplot(mapping = aes(x = Channel, y = ProductChoice, fill = n)) +
  geom_tile() +
  labs(title = "Product Choices by Channel")
```



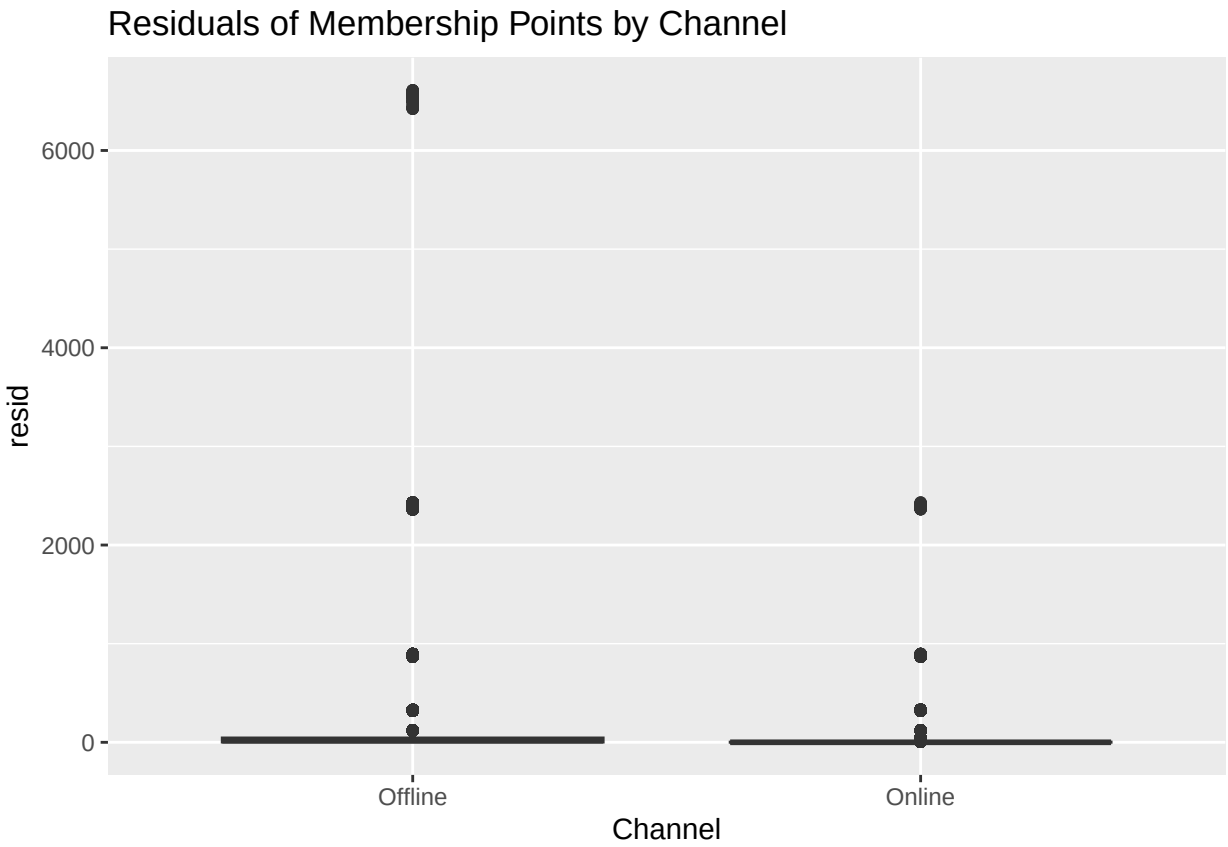
Analyzing Patterns and Building Models

Here we are creating a simple linear model for prediction. we shall use package modelr

```
# Fit a linear model
model <- lm(MembershipPoints ~ CustomerAge, data = purchaseprediction)

# Add residuals to the data frame
prediction <- purchaseprediction %>%
  add_residuals(model) %>%
  mutate(resid = exp(resid))

# Plot residuals by Channel to check patterns
ggplot(data = prediction) +
  geom_boxplot(mapping = aes(x = Channel, y = resid)) +
  labs(title = "Residuals of Membership Points by Channel")
```



Chapter 6

This chapter addresses practical aspects of managing R projects, including saving work, organizing analyses, and handling data files. It emphasizes the importance of considering R scripts as the "real" record of analysis and using RStudio projects to manage work effectively.

Key Points - Path Separators: Mac/Linux use slashes (/), while Windows uses backslashes (\). R can work with either, but using forward slashes is recommended.

- RStudio projects help keep all files associated with a project together, including input data, R scripts, analytical results, and figures.
- Organize projects into directories and set the working directory to the associated directory.
- Save figures and results using R code, not manually, to ensure reproducibility.

Chapter 7

This chapter explores tibbles. Tibbles are part of the tidyverse and offer several advantages over traditional data frames, particularly in terms of printing and subsetting.

To create a tibble from the existing purchaseprediction data frame, the as_tibble() function was employed.

```r
purchase_tibble <- as_tibble(purchaseprediction)
print(purchase_tibble, n=6, width = Inf)
```

```
## # A tibble: 500,000 x 12
##   CUSTOMER_ID ProductChoice MembershipPoints ModeOfPayment ResidentCity
##         <int>         <int>            <int> <chr>         <chr>
## 1           1             2                6 MoneyWallet   Madurai
## 2           2             3                2 CreditCard    Kolkata
## 3           3             2                4 MoneyWallet   Vijayawada
## 4           4             3                2 MoneyWallet   Meerut
## 5           5             2                6 MoneyWallet   Madurai
## 6           6             3                6 DebitCard     Kolkata
##   PurchaseTenure Channel IncomeClass CustomerPropensity CustomerAge
##            <int> <chr>         <int> <chr>                     <int>
## 1              4 Online            4 Medium                       55
## 2              4 Online            7 VeryHigh                     75
## 3             10 Online            5 Unknown                      34
## 4              6 Online            4 Low                          26
## 5              3 Online            7 VeryHigh                     38
## 6              3 Online            4 High                         71
##   MartialStatus LastPurchaseDuration
##           <int>                <int>
## 1             0                    4
## 2             0                   15
## 3             0                   15
## 4             0                    6
## 5             1                    6
## 6             0                   10
## # i 499,994 more rows
```

Subsetting tibbles can be accomplished using the $ operator or the double bracket [[]] notation.

```r
membership_points <- purchase_tibble$MembershipPoints
# or
membership_points <- purchase_tibble[["MembershipPoints"]]
print(head(membership_points))
```

```
## [1] 6 2 4 2 6 6
```

In scenarios where older R functions do not support tibbles, converting a tibble back to a data frame is straightforward using the as.data.frame() function.

```r
purchase_df <- as.data.frame(purchase_tibble)
head(purchase_df)
```

```
##   CUSTOMER_ID ProductChoice MembershipPoints ModeOfPayment ResidentCity
## 1           1             2                6   MoneyWallet      Madurai
## 2           2             3                2    CreditCard      Kolkata
## 3           3             2                4   MoneyWallet   Vijayawada
## 4           4             3                2   MoneyWallet       Meerut
## 5           5             2                6   MoneyWallet      Madurai
```

```
## 6           6            3               6     DebitCard      Kolkata
##   PurchaseTenure Channel IncomeClass CustomerPropensity CustomerAge
## 1              4  Online           4             Medium          55
## 2              4  Online           7           VeryHigh          75
## 3             10  Online           5            Unknown          34
## 4              6  Online           4                Low          26
## 5              3  Online           7           VeryHigh          38
## 6              3  Online           4               High          71
##   MartialStatus LastPurchaseDuration
## 1             0                   4
## 2             0                  15
## 3             0                  15
## 4             0                   6
## 5             1                   6
## 6             0                  10
```