Lab: Lab Journal 1

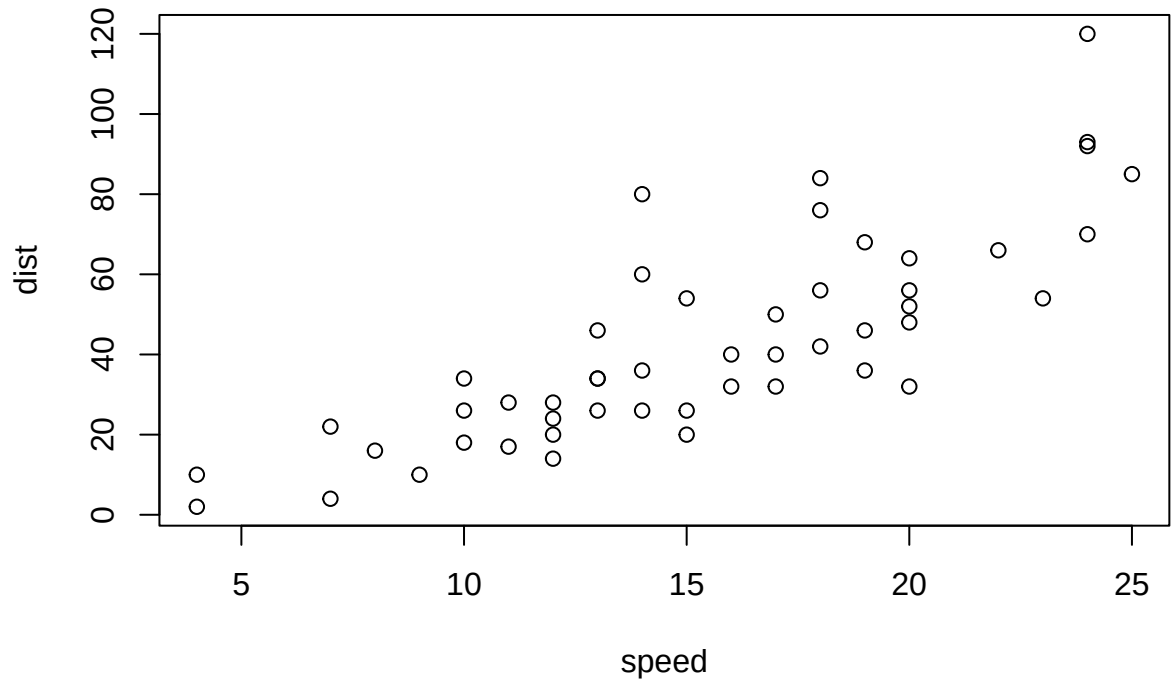
Reflection on this week exercises and reading

1. Vectorized Operations: In R you can perform operations on vectors and matrices in a very clean and efficient manner. In many languages, you might have to write loops to perform element-wise operations, but R handles this elegantly with its vectorized operations. For example, adding two arrays can be done directly with a simple +.

```
a <- c(1, 2, 3)
b <- c(4, 5, 6)
c <- a + b  # c will be c(5, 7, 9)
```

2. Data Manipulation with tidyverse: The tidyverse package in R includes libraries like dplyr and ggplot2 which make data wrangling and visualization much more straightforward than in many other languages.

```
plot(cars)
```



3 Notebook Support through R Markdown: RStudio supports R Markdown, a document format that allows you to create notebooks. This allows for a mix of code and rich text, enabling very effective data storytelling.

Part 1: GitHub

https://github.com/Keiichiro1101/IndividualProject

Part 2: Starting with R

1

Exercises: Mean of Positive Values

```
library(magrittr)
# Generate samples and calculate mean of positive values
mean_value <- rnorm(1000) %>%
  { .[. < 0] <- NA; . } %>%
  mean(na.rm = TRUE)

print(paste("The mean of the positive samples is:", mean_value))
```

```
## [1] "The mean of the positive samples is: 0.850527238413835"
```

Exercises: Root Mean Square Error

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Create a sample data frame
df <- data.frame(t = c(1, 2, 3, 4, 5),
                 y = c(1.1, 2.1, 2.9, 3.8, 5.2))

# Compute RMSE using a pipeline
rmse <- df %>%
  mutate(sq_diff = (t - y)^2) %>%
  summarise(mean_sq_diff = mean(sq_diff)) %>%
  mutate(rmse = sqrt(mean_sq_diff)) %>%
  pull(rmse)

print(paste("Root Mean Square Error is:", rmse))
```

```
## [1] "Root Mean Square Error is: 0.148323969741913"
```

2

```r
# Custom function to illustrate the use of seq() in R
generate_sequence <- function(from = 1, to = 10, by = 1, length.out = NULL, along.with = NULL) {

  if (!is.null(along.with)) {
    return(seq(along.with = along.with))
  }

  if (!is.null(length.out)) {
    return(seq(from = from, to = to, length.out = length.out))
  }

  return(seq(from = from, to = to, by = by))
}

# Examples
print("Sequence from 1 to 10 by 1:")
```

```
## [1] "Sequence from 1 to 10 by 1:"
```

```r
print(generate_sequence())
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
print("Sequence from 5 to 50 by 5:")
```

```
## [1] "Sequence from 5 to 50 by 5:"
```

```r
print(generate_sequence(from = 5, to = 50, by = 5))
```

```
##  [1]  5 10 15 20 25 30 35 40 45 50
```

```r
print("Sequence with length 5, from 0 to 1:")
```

```
## [1] "Sequence with length 5, from 0 to 1:"
```

```r
print(generate_sequence(from = 0, to = 1, length.out = 5))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

3

```r
# Create a numeric vector
num_vector <- c(12, 5, 8, 33, 7)

# Get the ranks of the elements
ranks <- rank(num_vector)

print(paste("Original vector:", toString(num_vector)))
```

```
## [1] "Original vector: 12, 5, 8, 33, 7"
```

```
print(paste("Ranks:", toString(ranks)))
```

```
## [1] "Ranks: 4, 1, 3, 5, 2"
```

```
# Get the indices that would sort the ranks
sorted_rank_indices <- order(ranks)

sorted_by_rank_vector <- num_vector[sorted_rank_indices]

print(paste("Sorted indices of ranks:", toString(sorted_rank_indices)))
```

```
## [1] "Sorted indices of ranks: 2, 5, 3, 1, 4"
```

```
print(paste("Vector sorted by ranks:", toString(sorted_by_rank_vector)))
```

```
## [1] "Vector sorted by ranks: 5, 7, 8, 12, 33"
```

4

R applies a technique called "recycling." This means that the shorter vector is recycled, or repeated, until it matches the length of the longer vector. However, R expects that the length of the shorter vector is a divisor of the length of the longer vector. If this is not the case, R will still proceed but will give a warning.

Part 3: Reflection and self-check questions on readings

1

Good data visualization helps to turn raw data into understandable, actionable insights. In careers like data science or market research, it's a core skill that can drive decisions. Imagine you are a Public Health Analyst. During an outbreak of a contagious disease, it's not sufficient to merely have access to data like infection rates, hospitalization rates, or demographic details of affected individuals. How this data is presented can influence public policy, hospital resource allocation, and general public perception. For instance, a well-designed heatmap could instantly highlight outbreak hotspots, enabling quicker interventions.

2

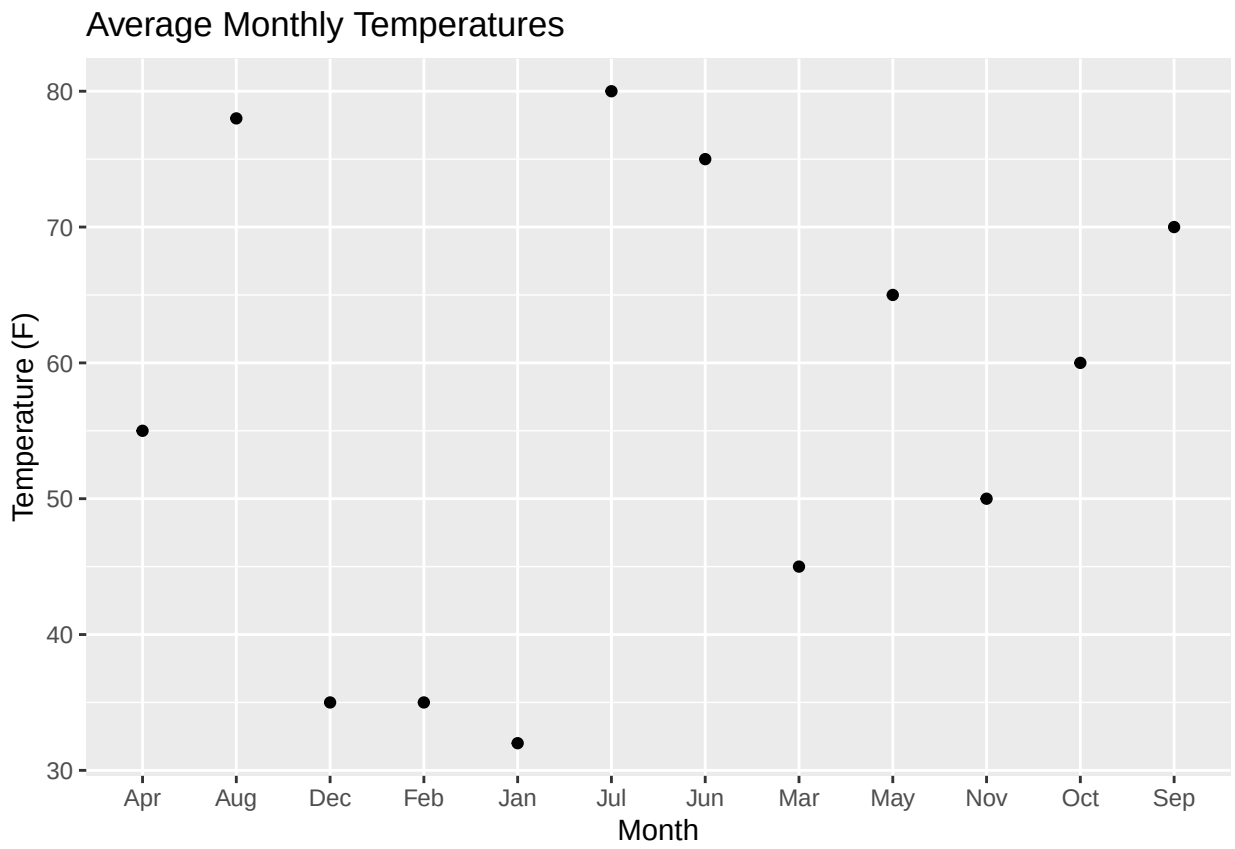
Visualization 1: Line Chart Showing Temperature Over Time

```
# Load the ggplot2 package
library(ggplot2)

# Create data frame for average monthly temperatures
months <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
temperatures <- c(32, 35, 45, 55, 65, 75, 80, 78, 70, 60, 50, 35)
data <- data.frame(months, temperatures)
```

```r
# Generate the line chart
ggplot(data, aes(x = months, y = temperatures)) +
  geom_line() +
  geom_point() +
  ggtitle("Average Monthly Temperatures") +
  xlab("Month") +
  ylab("Temperature (F)")
```

```
## `geom_line()`: Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```



What: The data represents average temperatures recorded over a year in a city.

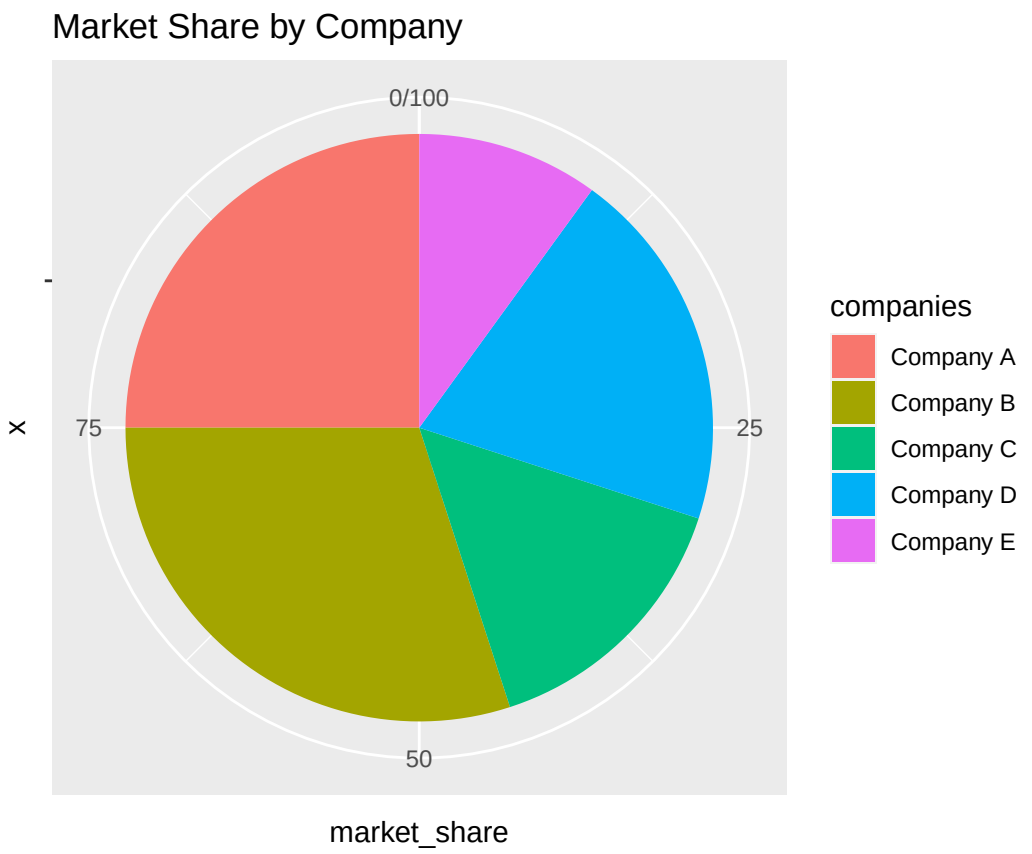
Why: The goal is to understand temperature trends to prepare for future weather conditions.

How: A line chart is used, with months on the X-axis and temperature on the Y-axis.

Visualization 2: Pie Chart Representing Market Share

```r
# Create data frame for market share
companies <- c("Company A", "Company B", "Company C", "Company D", "Company E")
market_share <- c(25, 30, 15, 20, 10)
data_market <- data.frame(companies, market_share)
```

```r
# Generate the pie chart
ggplot(data_market, aes(x = "", y = market_share, fill = companies)) +
  geom_bar(width = 1, stat = "identity") +
  coord_polar("y") +
  ggtitle("Market Share by Company")
```



What: The data represents the market share of five different companies in the same industry.

Why: The goal is to understand the current competitive landscape in this market.

How: A pie chart is used, with each slice representing a company, and the size of each slice proportional to its market share.

3

Effective Communication Regular stand-up meetings to discuss what each team member is working on, what challenges they're facing, and what support they need from the team.

Stakeholder Involvement Schedule bi-weekly meetings with stakeholders to update them on project progress and to involve them in critical decisions. This could include live demos or reports.

Reference

Name(s) of all authors: Keiichiro Watanabe, Destany Best

Lab: Lab Journal 1

Due Monday, 4 September 2023

Written/online sources used: (enter "none" if no sources other than the textbook and course website used) : none

Help obtained (Acknowledgments): none

"I/we confirm that the above list of sources is complete AND that I/we have not talked to anyone else about the solution to this problem."