Description of functions covered in the 2024 training:

1. Directories and reading files
   1. **setwd():** Sets the working directory for your R session to the specified file path. *Example: setwd("C:/Projects").*
   2. **getwd():** Returns the current working directory as a string. *Example: getwd().*
   3. **dir():** Lists the files and directories in the current working directory or a specified path. *Example: dir() or dir("C:/Projects").*
   4. **readRDS():** Reads R data saved in RDS format and restores it as an R object. *Example: data <- readRDS("datafile.rds").*
   5. **read\_csv():** Reads a CSV file into a data frame. It is part of the readr package in the tidyverse. *Example: data <- read\_csv("datafile.csv").*
   6. **read\_excel():** Reads Excel files into R. Requires the readxl package. *Example: data <- read\_excel("datafile.xlsx").*
   7. **General Tip:** There are many R functions for reading specific file types. If you encounter a different format (e.g., .json, .xml), search for the corresponding R package and function online.
2. Data manipulation
   1. **head():** Displays the first few rows of a data frame or vector. *Example: head(data).*
   2. **str():** Displays the structure of an object, including its data types and a preview of its content. *Example: str(data).*
   3. **View():** Opens a tab to view a data frame or table in a spreadsheet-like format. *Example: View(data).*
   4. **%>%:** The pipe operator. It allows chaining commands for cleaner and more readable code. *Example: data %>% filter(x > 10).*
   5. **filter():** Extracts rows that meet a certain condition. *Example: filtered\_data <- data %>% filter(column\_name == "value").*
   6. **mutate():** Adds new columns or modifies existing ones in a data frame. *Example: data <- data %>% mutate(new\_col = old\_col \* 2).*
   7. **group\_by():** Groups a data frame by one or more variables for summary operations. *Example: grouped\_data <- data %>% group\_by(group\_col).*
   8. **pivot\_longer():** Converts wide-format data to long-format. *Example: long\_data <- pivot\_longer(data, cols = c(col1, col2), names\_to = "variable", values\_to = "value").*
   9. **as.numeric():** Converts a variable to numeric type. *Example: data$numeric\_col <- as.numeric(data$char\_col).*
   10. **General Tip:** The tidyverse package provides a lot of tools for data manipulation. The DataCamp course on tidyverse is highly recommended for mastering these.
3. Analyze data
   1. **summarize():** Summarizes data by applying one or more aggregation functions. Often used with group\_by(). *Example: summary <- data %>% summarize(mean\_val = mean(column, na.rm = TRUE)).*
   2. **sum():** Computes the sum of a numeric vector. *Example: total <- sum(data$column, na.rm = TRUE).*
   3. **mean():** Computes the mean of a numeric vector. *Example: average <- mean(data$column, na.rm = TRUE).*
   4. **min():** Finds the minimum value in a numeric vector. *Example: min\_val <- min(data$column, na.rm = TRUE).*
   5. **max():** Finds the maximum value in a numeric vector. *Example: max\_val <- max(data$column, na.rm = TRUE).*
   6. **is.na():** Identifies missing values (NA) in a vector or data frame. *Example: missing\_values <- is.na(data$column).*
   7. **grepl():** Searches for patterns in a string and returns a logical vector (T/F). *Example: matches <- grepl("pattern", data$column).*
   8. **case\_when():** Implements multiple conditional statements to create new columns.
   9. **General Tip:** Functions in the dplyr package, such as summarize() and case\_when(), simplify data analysis tasks. Again, look at the datacamp tidyverse course to cover these!
4. Plotting
   1. **ggplot():** Initializes a ggplot object for creating graphics. Part of the ggplot2 package. *Example: ggplot(data, aes(x = x\_col, y = y\_col)).*
   2. **geom\_point():** Adds a scatter plot layer to a ggplot. *Example: ggplot(data, aes(x, y)) + geom\_point().*
   3. **geom\_line():** Adds a line plot layer to a ggplot. *Example: ggplot(data, aes(x, y)) + geom\_line().*
   4. **General Tip:** There are many geom\_ functions (e.g., geom\_bar(), geom\_histogram()) for creating different plot types. Explore online resources and the ggplot2 documentation (<https://r-graph-gallery.com/>).