Lab 00 Week 03 Worksheet

R Functions Glossary

This glossary provides an overview of key R functions used in **Lab 03**, explaining their **purpose** and **general use** in data processing and manipulation.

---

1. Working with Directories and Files

`setwd("path")`

Purpose: Sets the working directory so that R knows where to look for files.
Example: `setwd("~/Documents/arec330/lab_03")`

`getwd()`

Purpose: Returns the current working directory.
Example: `getwd()`

`dir()`

Purpose: Lists all files in the working directory.
Example: `dir()`

---

2. Installing and Loading Packages

`install.packages("package_name")`

Purpose: Installs a package in R (only needs to be done once per computer).
Example: `install.packages("dplyr")`

`library(package_name)`

Purpose: Loads an installed package into the current session.
Example: `library(dplyr)`

---

3. Reading and Viewing Data

`read_csv("file.csv")`

Purpose: Reads a CSV file into R as a **tibble** (modern dataframe).
Example: `df <- read_csv("supermarket_sales.csv")`

`View(df)`

Purpose: Opens a spreadsheet-like viewer for a dataframe in RStudio.
Example: `View(df)`

`glimpse(df)`

Purpose: Provides a compact summary of a dataframe's structure.
Example: `glimpse(df)`

`str(df)`

Purpose: Shows the structure of an object, including variable types.
Example: `str(df)`

---

4. Cleaning and Transforming Data

`clean_names(df)`

Purpose: Cleans column names (e.g., converts spaces to underscores).
Example: `df <- clean_names(df)`

`mutate(df, new_column = operation)`

Purpose: Creates or modifies columns in a dataframe.
Example: `df <- mutate(df, total_cost = unit_price * quantity)`

`rename(df, new_name = old_name)`

Purpose: Renames columns in a dataframe.
Example: `df <- rename(df, market = city)`

---

5. Filtering and Selecting Data

`filter(df, condition)`

Purpose: Keeps only rows that meet a condition.
Example: `df_yangon <- filter(df, city == "Yangon")`

`select(df, column1, column2)`

Purpose: Selects specific columns from a dataframe.
Example: `df_subset <- select(df, city, total, product_line)`

---

6. Sorting and Summarizing Data

`arrange(df, column)`

Purpose: Sorts rows by a column in ascending order (default).
Example: `df_sorted <- arrange(df, total)`

`arrange(df, desc(column))`

Purpose: Sorts rows in descending order.
Example: `df_sorted <- arrange(df, desc(total))`

`summarize(df, new_column = function(existing_column))`

Purpose: Aggregates data by computing summary statistics.
Example: `df_summary <- summarize(df, avg_price = mean(unit_price))`

`group_by(df, column)`

Purpose: Groups data by a categorical variable before summarization.
Example: `df_grouped <- group_by(df, product_line)`

---

7. Chaining Commands (Piping)

`df %>% function()`

Purpose: Passes the result of one function directly to another.
Example:

```
summary_df <- df %>%
  group_by(product_line) %>%
  summarize(avg_price = mean(unit_price))
```

---

8. Appending and Combining Data

`bind_rows(df1, df2)`

Purpose: Appends two dataframes (stacks rows).
Example: `df_combined <- bind_rows(df_yangon, df_mandalay)`

---

9. Capturing and Running Scripts

`sink("output.txt")`

Purpose: Redirects R console output to a text file.
Example: `sink("lab_03_log.txt")`

`source("script.R")`

Purpose: Runs an external R script.
Example: `source("lab_03.R")`

Using This Glossary

- Reference this list while working through Lab 03.
- Experiment with each function in R to solidify understanding.
- Combine multiple functions using pipes (`%>%`) to streamline analysis.

By understanding these core functions, you'll be able to efficiently clean, manipulate, and analyze data in R!