Session 1: Basic R Operations

Wali Reheman

2024-09-03

Contents

1 Introduction

- **Overview:** Introduces fundamental R data structures, operations, and basic data manipulation.
- **Objectives:**
  - Understand vectors, lists, and data frames.
  - Learn basic arithmetic, relational, and logical operations.
  - Import data from external files.
  - Perform data manipulation using `tidyverse`.
  - Export datasets in different formats.

2 Basic R Operations

- **Arithmetic Operations**
  - Perform addition, subtraction, multiplication, division, and exponentiation.

  - Example:

```r
# Arithmetic operations
a <- 10
b <- 3

addition <- a + b
subtraction <- a - b
multiplication <- a * b
division <- a / b
exponentiation <- a^b

addition
```

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

```r
subtraction
```

```
## [1] 7
```

```r
multiplication
```

```
## [1] 30
```

```r
division
```

```
## [1] 3.333333
```

```r
exponentiation
```

```
## [1] 1000
```

- **Relational Operations**
  - Compare values using operators like `==`, `!=`, `<`, and `>`.
  - Example:

```r
# Relational operations
equal_check <- a == b
not_equal_check <- a != b
greater_than <- a > b
less_than <- a < b

equal_check
```

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

```r
not_equal_check
```

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

```r
greater_than
```

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

```r
less_than
```

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

- **Logical Operations**
  - Combine or invert boolean expressions with `&`, `|`, and `!`.
  - Example:

```r
# Logical operations
logical_and <- (a > 5) & (b < 5)
logical_or <- (a > 15) | (b < 5)
logical_not <- !(a > b)

logical_and
```

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

```r
logical_or
```

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

```r
logical_not
```

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

- **Vectorized Operations**
  - Apply operations element-wise on vectors.
  - Example:

```r
# Vectorized operations
vector1 <- c(1, 2, 3)
vector2 <- c(4, 5, 6)
vector_sum <- vector1 + vector2

vector_sum
```

```
## [1] 5 7 9
```

3 1. Vectors and Objects

- **Definition:** A vector is a basic R data structure containing elements of the same type.
- **Example:**

```
# Create vectors
Names <- c("Jack", "Amy")    # Character vector for names
Apple <- c(5, 7)             # Numeric vector for apple counts
Orange <- c(4, 8)            # Numeric vector for orange counts
```

4 2. Lists

- **Definition:** A list can store elements of different types, such as vectors, data frames, or other lists.
- **Examples:**

```
# List containing vectors
List_of_Variables <- list(Names, Apple, Orange)

# List containing datasets (data frames)
List_of_Datasets <- list(cars, mtcars)

# List with mixed types: integer, vector, and data frame
List_of_anything <- list(1, Orange, mtcars)

# Extracting a dataset from a list
mtcars <- List_of_Datasets[[2]]  # Assigns the second dataset to 'mtcars'
```

5 3. Datasets

- **Definition:** In R, datasets are typically stored as data frames or tibbles.
- **Formats:**
  - **Wide Format:** Each column represents a variable.
  - **Long Format:** Data is organized with one column for variable types.
- **Examples:**

```
# Wide format data frame
Names <- c("Jack", "Amy")
Apple <- c(5, 7)
Orange <- c(4, 8)
Wide <- data.frame(Names, Apple, Orange)
head(Wide)  # Display first rows
```

```
##   Names Apple Orange
## 1  Jack     5      4
## 2   Amy     7      8
```

```
# Long format data frame
Names2 <- c("Jack", "Jack", "Amy", "Amy")
Fruit2 <- c("Apple", "Orange", "Apple", "Orange")
Amount <- c(5, 4, 7, 8)
Long <- data.frame(Names2, Fruit2, Amount)
head(Long)  # Display first rows
```

```
##   Names2 Fruit2 Amount
## 1   Jack  Apple      5
## 2   Jack Orange      4
## 3    Amy  Apple      7
## 4    Amy Orange      8
```

6 4. Reading External Data

- **Purpose:** Import data from external files into R.
- **Instructions:**
  - Use the `haven` package for SPSS and Stata files.
  - Use `load()` to import RData files.
- **Examples:**

```r
# Install and load 'haven' package (run once)
#install.packages("haven")
library(haven)

# Read a Stata file
gss22 <- read_dta("GSS2022.dta")
head(gss22)
```

```
## # A tibble: 6 x 1,156
##   year          id wrkstat hrs1        hrs2        evwork      wrkslf  occ10
##   <dbl+lbl> <dbl> <dbl+l> <dbl+lbl>   <dbl+lbl>   <dbl+lbl>   <dbl+l> <dbl+lbl>
## 1 2022          1 1 [wor~    40       NA(i) [iap] NA(i) [iap] 2 [som~  430 [man~
## 2 2022          2 5 [ret~ NA(i) [iap] NA(i) [iap]     1 [yes] 2 [som~   50 [mar~
## 3 2022          3 1 [wor~    52       NA(i) [iap] NA(i) [iap] 2 [som~ 4610 [per~
## 4 2022          4 3 [wit~ NA(i) [iap]    25       NA(i) [iap] 2 [som~ 4120 [foo~
## 5 2022          5 8 [oth~ NA(i) [iap] NA(i) [iap]     1 [yes] 2 [som~ 7330 [ind~
## 6 2022          6 1 [wor~    50       NA(i) [iap] NA(i) [iap] 2 [som~ 4610 [per~
## # i 1,148 more variables: prestg10 <dbl+lbl>, indus10 <dbl+lbl>,
## #   marital <dbl+lbl>, martype <dbl+lbl>, divorce <dbl+lbl>, widowed <dbl+lbl>,
## #   spwrksta <dbl+lbl>, sphrs1 <dbl+lbl>, sphrs2 <dbl+lbl>, spevwork <dbl+lbl>,
## #   cowrksta <dbl+lbl>, coevwork <dbl+lbl>, cohrs1 <dbl+lbl>, cohrs2 <dbl+lbl>,
## #   spwrkslf <dbl+lbl>, sppres80 <dbl+lbl>, spocc10 <dbl+lbl>,
## #   sppres10 <dbl+lbl>, spind10 <dbl+lbl>, coocc10 <dbl+lbl>,
## #   coind10 <dbl+lbl>, pawrkslf <dbl+lbl>, paocc10 <dbl+lbl>, ...
```

```r
# Read an SPSS file
gss18 <- read_sav("GSS2018.sav")
head(gss18)
```

```
## # A tibble: 6 x 1,065
##   ABANY     ABDEFECT  ABFELEGL ABHELP1 ABHELP2 ABHELP3 ABHELP4 ABHLTH   ABINSPAY
##   <dbl+lbl> <dbl+lbl> <dbl+lb> <dbl+l> <dbl+l> <dbl+l> <dbl+l> <dbl+lb> <dbl+lb>
## 1  2 [NO]    1 [YES]  NA       1 [Yes] 1 [Yes] 1 [Yes] 1 [Yes]  1 [YES] 1 [Peop~
## 2  1 [YES]   1 [YES]   3 [It ~ 2 [No]  2 [No]  2 [No]  2 [No]   1 [YES] 2 [Peop~
## 3 NA        NA        NA       1 [Yes] 2 [No]  1 [Yes] 1 [Yes] NA       2 [Peop~
## 4 NA        NA         1 [Sho~ 1 [Yes] 1 [Yes] 1 [Yes] 1 [Yes] NA       1 [Peop~
## 5  2 [NO]    1 [YES]  NA       2 [No]  2 [No]  2 [No]  1 [Yes]  1 [YES] 2 [Peop~
## 6  1 [YES]   1 [YES]   1 [Sho~ 1 [Yes] 1 [Yes] 1 [Yes] 1 [Yes]  1 [YES] 1 [Peop~
## # i 1,056 more variables: ABMEDGOV1 <dbl+lbl>, ABMEDGOV2 <dbl+lbl>,
## #   ABMELEGL <dbl+lbl>, ABMORAL <dbl+lbl>, ABNOMORE <dbl+lbl>,
## #   ABPOOR <dbl+lbl>, ABPOORW <dbl+lbl>, ABRAPE <dbl+lbl>, ABSINGLE <dbl+lbl>,
## #   ABSTATE1 <dbl+lbl>, ABSTATE2 <dbl+lbl>, ACQNTSEX <dbl+lbl>,
## #   ACTSSOC <dbl+lbl>, ADMINCONSENT <dbl+lbl>, ADULTS <dbl+lbl>,
## #   ADVFRONT <dbl+lbl>, AFFRMACT <dbl+lbl>, AFRAIDOF <dbl+lbl>,
## #   AFTERLIF <dbl+lbl>, AGE <dbl+lbl>, AGED <dbl+lbl>, AGEKDBRN <dbl+lbl>, ...
```

```r
# Load an RData file containing one or more datasets
load("electionpe.rda")

# Extract a dataset from nested lists (example 1)
R1 <- electionpe[["presidential"]][["y2021"]][["dataR1"]]

# Extract a dataset using index notation (example 2)
R1 <- electionpe[[1]][[1]][[2]]
```

7 5. Data Manipulation

- **Objective:** Modify and prepare datasets using the `tidyverse` package.
- **Instructions:**
  - Install and load `tidyverse` (if not already installed).
  - Use functions like `mutate`, `select`, and `filter` for data transformation.
- **Example:**

```r
# Install and load 'tidyverse' (run once)
#install.packages("tidyverse")
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 1.9.4     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 error
```

```r
# Create a new variable and select specific columns
gss22_new <- gss22 %>%
  mutate(mom_degr = madeg) %>%  # Create new variable 'mom_degr'
  select(mom_degr, year, id)    # Keep specific columns

# Filter dataset for rows where 'mom_degr' equals 4
gss22_new <- gss22_new %>%
  filter(mom_degr == 4)
```

8 6. Outputting Data

- **Objective:** Save datasets to files for later use or sharing.
- **Instructions:**
  - Use the `openxlsx` package for Excel files.
  - Use `write.csv` for CSV files.
  - Use `write_dta` from `haven` for Stata files.
  - Use `save()` to create an RData file containing multiple datasets.
- **Example:**

```r
# Install and load 'openxlsx' package (run once)
#install.packages("openxlsx", dependencies = TRUE)
library(openxlsx)

write.xlsx(gss22_new, file = "gss22_new.xlsx")
# Export dataset to a CSV file
write.csv(gss22_new, file = "gss22_new.csv")

# Export dataset to a Stata file using 'haven'
write_dta(gss22_new, path = "gss22_new.dta")

# Save multiple datasets into an RData file
save(gss22, gss22_new, file = "gss22_both.rda")

# Load to verify the saved datasets
load("gss22_both.rda")
# Note: Datasets load as separate objects, not as a list.
```

9 Summary

- **Key Takeaways:**
  - Vectors, lists, and data frames form the core R data structures.
  - Basic arithmetic, relational, and logical operations are essential for data manipulation.
  - External data can be imported using appropriate packages.
  - The `tidyverse` simplifies data transformation tasks.
  - Datasets can be exported in various formats for further analysis.