R Training Session 2: Matrices, Data Frames, and Data Manipulation

Data Avicenna

Department of Economics
Universitas Gadjah Mada

May 14, 2025

Outline

Review of Previous Session

In our previous session, we covered:

- Basic operations and functions in R
- Vectors - the fundamental data structure
- Factors for categorical data
- Logical operations and comparisons
- Conditional statements (if, else, if-else)
- Creating and using functions

Today, we'll expand our knowledge to:

- Working with directories and importing data
- Matrices - two-dimensional data structures
- Data frames - tabular data with mixed types
- Introduction to data manipulation with tidyverse packages

Matrices: Introduction

- Matrices are collections of vectors.
- All elements must be of the same type (numeric, character, etc.)
- The dimensions are $r \times k$, where r is the number of rows and k is the number of columns.

Creating Matrices

There are multiple ways to create matrices in R:

```
# Method 1: From a vector with matrix() function
vec <- 1:12
mat1 <- matrix(vec, nrow = 4, ncol = 3)

# Method 2: By binding vectors together
x <- c(1, 2, 3, 4)
y <- c(5, 6, 7, 8)
z <- c(9, 10, 11, 12)

# Column binding
mat2 <- cbind(x, y, z)

# Row binding
mat3 <- rbind(x, y, z)
```

Creating Matrices

```
# Method 3: Changing dimensions of a vector
vec2 <- 1:9
dim(vec2) <- c(3, 3)  # Converts vector to a 3x3 matrix
```

Matrix Options

Control how data fills matrices with options:

```
# Data is filled by column by default
matrix(1:6, nrow = 2, ncol = 3)
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6

# Fill by row instead
matrix(1:6, nrow = 2, ncol = 3, byrow = TRUE)
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

Matrix Options

```
# Create a matrix with specific values
matrix(0, nrow = 3, ncol = 4)  # Matrix of zeros
##      [,1] [,2] [,3] [,4]
## [1,]    0    0    0    0
## [2,]    0    0    0    0
## [3,]    0    0    0    0
```

Matrix Indexing

Accessing elements, rows, or columns in a matrix:

```r
mat <- matrix(1:9, nrow = 3, ncol = 3)
mat
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9

# Entire second row
mat[2, ]    # [1] 2 5 8

# Entire third column
mat[, 3]    # [1] 7 8 9
```

Matrix Indexing

Accessing elements, rows, or columns in a matrix:

```
# Individual element at row 2, column 3
mat[2, 3]  # [1] 8
```

Matrix Operations

Matrices support various mathematical operations:

```
A <- matrix(1:4, nrow = 2)
B <- matrix(5:8, nrow = 2)

# Element-wise operations
A + B          # Addition
##      [,1] [,2]
## [1,]    6   10
## [2,]    8   12

A * B          # Element-wise multiplication
##      [,1] [,2]
## [1,]    5   21
## [2,]   14   32
```

Matrix Operations

Matrices support various mathematical operations:

```
# Matrix multiplication
A %*% B       # True matrix multiplication
##      [,1] [,2]
## [1,]   19   22
## [2,]   43   50
```

More Matrix Operations

```
# Transpose
t(A)
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4

# Matrix diagonal
diag(A)       # [1] 1 4

# Create diagonal matrix
diag(1, 3)    # 3x3 identity matrix
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1

# Matrix inversion (requires square matrix)
C <- matrix(c(3, 1, 2, 4), nrow = 2)
solve(C)      # Inverse of matrix C
```

Matrix Functions

Useful functions for working with matrices:

```r
mat <- matrix(1:9, nrow = 3)

# Dimensions
dim(mat)      # [1] 3 3
nrow(mat)     # [1] 3
ncol(mat)     # [1] 3

# Row and column names
rownames(mat) <- c("row1", "row2", "row3")
colnames(mat) <- c("col1", "col2", "col3")

# Apply functions to rows or columns
rowSums(mat)  # [1] 12 15 18
colMeans(mat) # [1] 2 5 8

# Apply any function to rows or columns
apply(mat, 1, sum)    # Apply sum to rows (margin = 1)
apply(mat, 2, min)    # Apply min to columns (margin = 2)
```

Data Frames: Introduction

- Data frames are the most common data structure for analysis in R
- Unlike matrices, data frames can contain different types of data in each column
- Each column is a vector of the same length (but different types allowed)

```r
# Creating a basic data frame
economic_data <- data.frame(
  country = c("Indonesia", "Malaysia", "Thailand",
  "Vietnam"),
  gdp_billion = c(1119, 364, 505, 261),
  population_million = c(270, 32, 70, 97),
  inflation_rate = c(3.2, 2.8, 1.7, 4.1)
)
```

Data Frame Structure

Let's examine the structure of the data frame:

```
# View the data frame
economic_data

# Check the structure
str(economic_data)
```

Accessing Data Frame Elements

```
# Access a column using $ notation
economic_data$country
# [1] "Indonesia" "Malaysia" "Thailand" "Vietnam"

# Access a column using bracket notation
economic_data[, "gdp_billion"]
# [1] 1119  364  505  261

# Access a row
economic_data[2, ]
# country gdp_billion population_million inflation_rate
# 2 Malaysia   364              32                  2.8
```

Accessing Data Frame Elements

```r
# Access a specific element
economic_data[3, "inflation_rate"]
# [1] 1.7

# Multiple rows or columns
economic_data[1:2, c("country", "gdp_billion")]
```

Manipulating Data Frames

```
# Add a new column
economic_data$gdp_per_capita <-
economic_data$gdp_billion * 1e9 /
(economic_data$population_million * 1e6)

# Modify values
economic_data$inflation_rate[1] <- 3.5
# Update Indonesia's inflation rate
```

Manipulating Data Frames

```r
# Add a new row
new_country <- data.frame(
  country = "Philippines",
  gdp_billion = 331,
  population_million = 110,
  inflation_rate = 4.5,
  gdp_per_capita = 331 * 1e9 / (110 * 1e6)
)

economic_data <- rbind(economic_data, new_country)
```

Data Frame Operations

Common operations with data frames:

```
# Sorting a data frame
sorted_by_gdp <-
economic_data[order(economic_data$gdp_billion,
                                        decreasing = TRUE), ]

# Filtering rows based on conditions
high_inflation <-
economic_data[economic_data$inflation_rate > 3, ]

# Select subset of columns
gdp_data <- economic_data[, c("country",
"gdp_billion", "gdp_per_capita")]
```

Data Frame Operations

```
# Summary statistics
summary(economic_data)

# Number of rows and columns
dim(economic_data)  # [1] 5 5
nrow(economic_data) # [1] 5
ncol(economic_data) # [1] 5
```

Built-in Data Sets in R

R includes many datasets for learning:

```
# List all available datasets
data()

# Load a dataset into your environment
data(mtcars)  # Motor Trend Car Road Tests
head(mtcars)
```

Importing Data: Smoking Data Overview

We will be using a dataset on smoking for the rest of the module:

- Cross-sectional dataset with observations on 10,000 indoor workers
- Collected as part of the National Health Interview Survey (1991 & 1993)
- Contains information on:
  - Whether individuals were subject to a workplace smoking ban
  - Smoking status
  - Demographic characteristics
- Used in the paper "Do Workplace Smoking Bans Reduce Smoking?" (Evans, Farrelly, & Montgomery, 1999)

Smoking Data Variables

Variable	Definition
smoker	=1 if current smoker, =0 otherwise
smkban	=1 if there is a work area smoking ban, =0 otherwise
age	age in years
hsdrop	=1 if high school dropout, =0 otherwise
hsgrad	=1 if high school graduate, =0 otherwise
colsome	=1 if some college, =0 otherwise
colgrad	=1 if college graduate, =0 otherwise
black	=1 if black, =0 otherwise
hispanic	=1 if Hispanic, =0 otherwise
female	=1 if female, =0 otherwise

Note: The educational binary indicators refer to the highest level attained and are mutually exclusive.

Working Directories in R

Understanding working directories is essential for importing data:

```r
# Check current working directory
getwd()
# [1] "C:/Users/username/Documents"

# Set working directory
setwd("/path/to/your/folder")

# List files in current directory
list.files()
# [1] "Smoking.csv" "Smoking.xlsx" "Smoking.dta"
```

Importing Smoking Data - CSV Format

```r
# Importing Data ####
# Using base R
smoking_df <- read.csv("Smoking.csv")

# Common parameters
smoking_df <- read.csv("Smoking.csv",
                       header = TRUE, # First row contains headers
                       sep = ";", # Delimiter (comma)
                       )

# Using readr package from tidyverse
# install.packages("tidyverse")
# install.packages("readr")
library(tidyverse) # Or alternatively, library(readr)
smoking_df <- read_delim("Smoking.csv")
```

Importing Smoking Data - Excel Format

```
# Using readxl package
# install.packages("readxl")
library(readxl)

# Basic import
smoking_df_xl <- read_excel("Smoking.xlsx")

# We can also specify the sheet if there are multiple sheets
smoking_df_xl <- read_excel("Smoking.xlsx", sheet = 1)

# Compare dimensions of our imported data
dim(smoking_df)      # From CSV
dim(smoking_df)   # From Excel
# Should be the same: [1] 10000    10
```

Importing Smoking Data - Stata Format

```r
# Using haven package
# install.packages("haven")
library(haven)

# Import Stata file
smoking_df_dta <- read_dta("Smoking.dta")
```

Introduction to tidyverse

- The tidyverse is a collection of R packages designed for data science
- Core packages include:
  - **dplyr**: data manipulation (filter, sort, summarize)
  - **tidyr**: data reshaping and cleaning
  - **ggplot2**: data visualization
  - **readr**: data import
  - **tibble**: modern data frames
- Benefits:
  - Consistent syntax and grammar
  - More readable code with the pipe operator (%>%)
  - Optimized for working with data frames
  - Designed for data analysis workflows

Installing and Loading tidyverse

To get started with tidyverse:

```
# Install the complete tidyverse (one-time setup)
# install.packages("tidyverse")

# Or install individual packages as needed
# install.packages("dplyr")
# install.packages("tidyr")

# Load the entire tidyverse
library(tidyverse)

# Or load individual packages
library(dplyr)
library(tidyr)

# Let's use our smoking dataset for all examples
# Convert to a tibble for better printing
smoking_df <- as_tibble(smoking_df)
```

dplyr: Key Functions - filter()

The first key dplyr function is `filter()`, which subsets rows:

```
# 1. filter() - subset rows based on values

# Find only smokers
smokers <- filter(smoking, smoker == 1)
dim(smokers)  # Should have fewer rows than the original dataset

# Find smokers who are subject to workplace smoking bans
banned_smokers <- filter(smoking, smoker == 1 & smkban == 1)

# Find female smokers with at least some college education
female_educated_smokers <- filter(smoking,
                                  smoker == 1,
                                  female == 1,
                                  colsome == 1 | colgrad == 1)
```

dplyr: Key Functions - select()

The second key dplyr function is `select()`, which selects columns:

```
# 2. select() - select columns

# Select only demographic variables
demographics <- select(smoking_df, age, female, black, hispanic)
head(demographics)

# Select variables related to education
education <- select(smoking, hsdrop, hsgrad, colsome, colgrad)
head(education)

# You can also exclude columns with -
no_race <- select(smoking_df, -black, -hispanic)
head(no_race)

# Select columns that match a pattern
education_vars <- select(smoking_df, starts_with("hs"), starts_with("col"))
head(education_vars)
```

dplyr: Key Functions - arrange()

The third key dplyr function is `arrange()`, which reorders rows:

```
# 3. arrange() - reorder rows

# Sort by age (youngest first)
youngest_first <- arrange(smoking_df, age)
head(youngest_first)

# Sort by age (oldest first)
oldest_first <- arrange(smoking_df, desc(age)) # desc = descendi

head(oldest_first)

# Sort by multiple columns: education level then age
educated_by_age <- arrange(smoking_df,
                           desc(colgrad),
                           desc(colsome),
                           desc(hsgrad),
                           age)
```

dplyr: Key Functions - mutate()

The fourth key dplyr function is `mutate()`, which creates new variables:

```
# 4. mutate() - create new variables

# Create an age category variable
smoking_with_age_cat <- mutate(smoking_df,
                               age_category = case_when(
                                 age < 30 ~ "Young",
                                 age < 50 ~ "Middle",
                                 TRUE ~ "Senior" # default case
                               ))
# Convert string variables of the catagories as factors
as.factor(smoking_with_age_cat$age_category)
class(smoking_with_age_cat$age_category)
# Check the generated age categories
levels(smoking_with_age_cat$age_category)
```

dplyr: Key Functions - summarize()

The fifth key dplyr function is `summarize()`, which creates summaries:

```
# 5. summarize() - reduce multiple values to a single summary

# Overall smoking rate
smoking_rate <- summarize(smoking,
                          pct_smokers = mean(smoker) * 100,
                          count = n())

# Often used with group_by() for group summaries
by_gender <- group_by(smoking, female)
gender_summary <- summarize(by_gender,
                            pct_smokers = mean(smoker) * 100,
                            avg_age = mean(age),
                            pct_banned = mean(smkban) * 100,
                            count = n())
gender_summary
```

The Pipe Operator (%>%)

The pipe operator makes code more readable by chaining operations:

```
# Without pipes
banned_smokers_age <- filter(smoking, smkban == 1, smoker == 1)
banned_smokers_age <- select(banned_smokers_age, age)
banned_smokers_age <- summarize(banned_smokers_age,
                                avg_age = mean(age))

# With pipes
banned_smokers_age <- smoking %>%
  filter(smkban == 1, smoker == 1) %>%
  select(age) %>%
  summarize(avg_age = mean(age))

# The pipe takes the output from one function and passes it
# as the first argument to the next function
```

Complex Data Analysis with Pipes

Example: Analyzing smoking rates by demographic characteristics

```
# Smoking rates by gender and education level
smoking_by_demo <- smoking %>%
  # Group by gender and education level
  group_by(female, colgrad) %>%
  # Calculate summary statistics for each group
  summarize(
    avg_age = mean(age),
    smoking_rate = mean(smoker) * 100,
    ban_rate = mean(smkban) * 100,
    count = n()
  ) %>%
  # Sort by smoking rate
  arrange(desc(smoking_rate))

# View results
smoking_by_demo
```

Data Reshaping with tidyr

Let's reshape our smoking data to understand patterns:

```
# First, create a summary by gender and education
education_summary <- smoking_df %>%
  group_by(female, hsdrop, hsgrad, colsome, colgrad) %>%
  summarize(
    smoking_rate = mean(smoker) * 100,
    count = n()
  ) %>%
  ungroup()

# Convert from multiple education columns to a single column
edu_long <- education_summary %>%
  pivot_longer(
    cols = c(hsdrop, hsgrad, colsome, colgrad), # columns to collapse
    names_to = "education_level",     # new column with the level name
    values_to = "is_level"      # new column that holds the 0/1 values
  ) %>%
  filter(is_level == 1) %>%  # Keep only the actual education level
  select(-is_level)          # Remove the indicator column
```

Merging Data with dplyr

Let's demonstrate joining with a small additional dataset:

```
# Create a small dataset with education descriptions
education_labels <- tibble(
  education_level = c("hsdrop", "hsgrad", "colsome", "colgrad", "higher"),
  description = c("High School Dropout", "High School Graduate",
                  "Some College", "College Graduate",
                  "Post-Graduate Degree"),
  years_edu = c(10, 12, 14, 16, 18)
)

# Join our long education data with the labels
edu_labeled <- edu_long %>%
  left_join(education_labels, by = "education_level")

# Now we have both the smoking rates and the education descriptions
head(edu_labeled)
```

Assignment 2

You need to submit your answers to this assignment to obtain a training certificate. Submit your R script to my email: **`dataavicenna@mail.ugm.ac.id`**. Contact my email for any questions.

Submission deadline: **20 May 2025 at 23.59pm**

Assignment 2

1. Write the answers to the following questions in an R script:

a. Import the smoking dataset into RStudio and name the object "smoking_df". Convert the data into a tibble if you want.

b. Create a categorical variable for education level (name the variable as "edu_level). The categories are as follows: High school dropouts are coded as 1, high school graduates are coded as 2, those attended some college are coded as 3, and college graduates are coded as 4. (Hint: Use mutate() from dplyr)

c. Create a tibble consisting of the following vectors:
   - edu_level = c(1, 2, 3, 4)
   - edu_desc = c("High School Dropout", "High School Graduate", "Some College", "College Graduate")

d. Merge the tibble with "smoking_df" using the following code:

```
smoking_df %>% left_join(name_of_your_tibble, by = "edu_level")
```

e. Lastly, convert edu_desc as factors. Print the merged data.

Assignment 2

2. Using smoking_df, answer the following questions:

   a. Using the mutate() function, make a new variable called male_smoker, which takes the value of 1 if the observed individual is both a male and a smoker.
   b. Print the data and check whether male_smoker is already generated.
   c. Calculate the percentage of male smokers.