MGT4018/MGT4090 Lab 1

Bernd Wurth

Table of contents

1 Introduction

This lab is the same as the SPSS Lab 1 for MGT4018 and MGT4090. We use base R functions as the default. While there are many R packages available, understanding base R operations provides a strong foundation for data analysis.

Alternatives using R packages

Alternatives for achieving the same outcomes using different R packages are provided in green boxes. If you want to explore these alternatives, we need to install and load the necessary R packages. Run this code once to install packages:

```
install.packages(c(
  "tidyverse",  # for data manipulation and visualization
  "gt",         # for creating beautiful tables
  "gtsummary",  # for summary statistics
  "janitor"     # for tabulation functions
))
```

Now load the packages:

```
library(tidyverse)
library(gt)
library(gtsummary)
library(janitor)
```

2 Exercise A: Creating and Exploring a Data Set in R

This lab will guide you through creating a dataset, assigning labels, and conducting basic analyses using R. You will learn how to create variables, enter data, and generate summary tables similar to those you would in SPSS.

2.1 Step A1: Setup

Before starting any work in R, it is important to organize your files. This ensures that your scripts, datasets, and outputs are easy to manage and reproducible. Projects are strongly recommended for better organization and reproducibility, but setting a working directory is an alternative if needed (see also the respective section in `0. Getting Started`).

The best way to organize your work is to create an RStudio project. This keeps all related files in a single folder. Follow these steps:

1. Open RStudio.
2. Go to **File > New Project**.
3. Select **New Directory** and then **New Project**.
4. Choose a location on your computer and give the project a name, e.g., `ResearchMethodsLab`. Avoid spaces in folder and file names.
5. Click **Create Project**. RStudio will open a new session within the project folder.
6. Create a new **R script**: Go to **File > New File > R Script**.
7. Save the script in your project folder with an appropriate name, e.g., `Lab_Exercise_A.R`.

If you are not using a project, you will need to set a working directory manually. The working directory is the folder where R looks for files and saves outputs.

1. In your Finder (Mac) or Explorer (Windows), create a new folder for the R Labs (e.g., `ResearchMethodsLab`). Avoid spaces in folder and file names.
2. Open RStudio.
3. Create a new **R script**: Go to **File > New File > R Script**.
4. Save the script in the folder that you previously created with an appropriate name, e.g., `Lab_Exercise_A.R`.
5. Set the working directory in your script using the `setwd()` function. Copy the following code into your script, change the path to the older that you created, and execute the script.

```
setwd("path/to/your/folder")
```

Now you are ready to begin your work in R!

2.2 Step A2: Creating Variables

In base R, we will create our vectors and combine them into a data frame. Use the following code to create your variables:

```
# Create vectors for our data
id <- 1:30

# Create age bands with labels
age_bands <- factor(
  c(1,3,2,4,2,5,5,2,3,1,4,1,3,2,4,2,5,5,2,3,1,4,2,4,2,5,5,2,3,1),
  levels = 1:5,
  labels = c("<21", "21-30", "31-40", "41-50", ">50")
)

# Create gender categories with labels
gender <- factor(
  c(0,1,1,0,0,0,1,1,1,0,9,0,2,1,0,0,1,1,0,0,1,0,0,0,1,1,1,0,9,9),
  levels = c(0,1,2,9),
  labels = c("Male", "Female", "Other", "Prefer not to say")
)
```

> **Note**
>
> In R, we use `factor()` to create categorical variables. This is similar to value labels in SPSS. The `levels` argument specifies the underlying codes, while `labels` provides the human-readable labels.

2.3 Step A3: Create the Data Frame

We now combine the variables that we created in the previous step into a `data.frame`.

```
# Combine into a data frame
survey_data <- data.frame(
  ID = id,
  AgeBand = age_bands,
  Gender = gender
)
```

You can easily explore and check the basic structure of your data and get a summary:

```
# View the first few rows using head()
head(survey_data)

# Examine the structure
str(survey_data)

# Get basic summary statistics
summary(survey_data)
```

Alternatively, you can download the dataset.

2.4 Step A4: Frequency Tables

Create frequency tables using base R functions:

```
# Age Band frequencies
age_freq <- table(survey_data$AgeBand)
age_prop <- prop.table(age_freq) * 100

# Combine frequencies and percentages
age_summary <- cbind(
  Frequency = age_freq,
  Percentage = round(age_prop, 1)
)

# Display the results
print("Age Band Distribution:")
print(age_summary)
```

```
# Gender frequencies
gender_freq <- table(survey_data$Gender)
gender_prop <- prop.table(gender_freq) * 100

# Combine frequencies and percentages
gender_summary <- cbind(
  Frequency = gender_freq,
  Percentage = round(gender_prop, 1)
)

# Display the results
print("Gender Distribution:")
print(gender_summary)
```

2.5 Step A5: Cross-Tabulation

Create a cross-tabulation using base R:

```
# Create cross-tabulation
cross_tab <- table(survey_data$AgeBand, survey_data$Gender)

# Add row and column totals
cross_tab_with_totals <- addmargins(cross_tab)

# Display the results
print("Age Band by Gender Cross-Tabulation:")
print(cross_tab_with_totals)

# Calculate percentages (optional)
prop_table <- round(prop.table(cross_tab) * 100, 1)
print("\nPercentage Distribution:")
print(prop_table)
```

2.6 Step A6: Loading and Viewing Data

2.7 Step A7: Exploring Value Labels

2.8 Step A8: Basic Exploration with Summary Statistics

2.9 Step A9: Correcting Errors

2.10 Step A10: Descriptive Statistics

2.11 Step A11: Setting Variable Types

2.12 Step A12: Cross-tabulations

2.13 Optional Expansion: Plotting the Data

3 Exercise B: Reports and Tables

3.1 Step 1: