MGT4018/MGT4090 Lab 2

Bernd Wurth

Table of contents

1 Introduction

This lab is the same as the SPSS Lab 2 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, each box will introduce the respective package, how to install and use it, and the outputs it can produce.

2 Exercise D: Correlation and Linear Regression

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 D1: Setup

You can continue working in the same project (Step A1, Option 1) or working directory (Step A1, Option 2) that you created in the previous lab. You should, however, do the following:

1. Create a new **R script**: Go to **File > New File > R Script**.
2. Save the script in your `scripts` folder with an appropriate name, e.g., `Lab2_Exercise_D_E_F_G.R`.

> Note
>
> You can either work through the following steps and copy/paste the respective code into the `Lab_Exercise_D_E_F_G.R` file that you will create in Step D1 or download the R script for Exercises D, E, F, and G and follow the instructions below and save the downloaded file in the `scripts` folder that you will create.

Now you are ready to begin your work in R and continue with Step D2!

2.2 Step D2: Loading the Dataset

Please download the dataset and save it in the `data` folder within your project folder or working directory.

```r
# Load the CSV file stored in the "data" folder
survey_data_full <- read.csv("data/lab2-survey.csv")
```

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

```r
# View the first few rows using head()
head(survey_data_full)

# Examine the structure
str(survey_data_full)

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

Simialr to the Exercises B and C in Lab 1, we want to assign labels to the demographic variables in the `survey_data_full` data frame using **factors** with labeled levels. This method ensures the data remains categorical but with human-readable labels for easier interpretation and analysis.

```r
# Convert demographic variables to factors with labeled levels

# Assign labels for "sex"
survey_data_full$sex <- factor(
  survey_data_full$sex,
  levels = c(1, 2),
  labels = c("Male", "Female")
)

# Assign labels for "marital"
survey_data_full$marital <- factor(
  survey_data_full$marital,
  levels = c(1, 2, 3, 4, 5, 6, 7, 8),
  labels = c(
    "Single", "Steady relationship", "Living with partner",
    "Married first time", "Remarried", "Separated",
    "Divorced", "Widowed"
  )
)

# Assign labels for "child"
survey_data_full$child <- factor(
  survey_data_full$child,
  levels = c(1, 2),
  labels = c("Yes", "No")
)

# Assign labels for "educ"
survey_data_full$educ <- factor(
  survey_data_full$educ,
  levels = c(1, 2, 3, 4, 5, 6),
  labels = c(
    "Primary", "Some secondary", "Completed high school",
    "Some additional training", "Completed undergraduate",
    "Postgraduate completed"
  )
)

# Assign labels for "source"
survey_data_full$source <- factor(
  survey_data_full$source,
  levels = c(1, 2, 3, 4, 5, 6, 7, 8, 9),
  labels = c(
    "Work", "Spouse or partner", "Relationships", "Children",
```

```
    "Family", "Health/illness", "Life in general",
    "Money/finances", "Lack of time, too much to do"
  )
)

# Assign labels for "smoke"
survey_data_full$smoke <- factor(
  survey_data_full$smoke,
  levels = c(1, 2),
  labels = c("Yes", "No")
)
```

The `summary()` function can be used to confirm that the labels have been applied correctly to the variables.

```
# Verify changes by printing a summary
summary(survey_data_full)
```

2.3 Step D3: Correlation

Correlation analysis helps us understand the relationship between two variables. Before we dive into the practical implementation, let's understand some key concepts.

2.3.1 What is Correlation?

Correlation measures the strength and direction of the relationship between two variables. The **Pearson correlation coefficient (PCC)**, often denoted as `r`, is a statistical measure that quantifies the strength and direction of a linear relationship between two continuous variables. It is one of the most widely used correlation metrics.

The correlation coefficient (r) ranges from -1 to +1:

- r = +1: Perfect positive correlation
- r = 0: No linear correlation
- r = -1: Perfect negative correlation

The magnitude of r (how close it is to ±1) indicates the strength of the relationship, while the sign indicates the direction.

> **Note**
>
> The correlation coefficient is standardized, meaning it's independent of the units of measurement of the original variables. This makes it useful for comparing relationships between different pairs of variables.

2.3.2 Assumptions for Pearson's Correlation

Before calculating correlations, we should check these assumptions:

1. Variables are measured at the interval or ratio level
2. Linear relationship between variables
3. No significant outliers
4. Approximate normal distribution of variables
5. Homoscedasticity (equal variances)

2.3.3 Visual Inspection: Scatterplots

Before calculating correlations, it's important to visualize the relationships:

```
# Install ggplot2 (NOTE: not needed if you completed the first lab)
install.packages("ggplot2")

# Load ggplot2
library(ggplot2)
```

```r
# Simple scatterplot
scatter_stress_control <- ggplot(data = survey_data_full, aes(x = tpcoiss, y = tpstress)) +
  geom_point(color = "blue", size = 2) +
  labs(
    title = "Scatterplot of Perceived Stress vs Coping Strategies",
    x = "Coping Strategies (tpcoiss)",
    y = "Perceived Stress (tpstress)"
  ) +
  theme_minimal()

# Show plot
scatter_stress_control
```

2.3.4 Computing Correlations

We can calculate correlation. In the first instance, we focus on the two variables `tpcoiss` and `tpstress`. Afterwards, we will explore how to calculate correlations for all variables at once.

Single Correlation

The `cor()` function calculates the correlation coefficient:

```r
# Calculate correlation between tpcoiss and tpstress
cor(survey_data_full$tpcoiss, survey_data_full$tpstress)

# Specify the method (default is Pearson)
cor(survey_data_full$tpcoiss, survey_data_full$tpstress, method = "pearson")
```

> **Note**
>
> **Best Practices in Base R**
>
> 1. Always visualize your data first
> 2. Check assumptions (e.g., normality):
>
> ```r
> # Test for normality
> shapiro.test(x)
> shapiro.test(y1)
> ```
>
> 3. Consider different correlation methods when appropriate:
>
> ```r
> # Pearson correlation (default)
> cor(x, y1, method = "pearson")
>
> # Spearman correlation (for non-normal data)
> cor(x, y1, method = "spearman")
>
> # Kendall correlation (for ordinal data)
> cor(x, y1, method = "kendall")
> ```

Correlation Matrix

We can calculate correlations for multiple variables variables at once.

First, let's create a smaller data frame with only continuous variables. We can do this in base R using the following code:

```r
survey_data_small <- survey_data_full[, c("tpcoiss", "tpstress", "toptim",
                                          "tposaff", "tnegaff", "tlifesat",
                                          "tslfest", "tmarlow")]
```

You can verify the structure of your new data frame using:

```r
# Check the structure of the new data frame
str(survey_data_small)
```

```
# Or see the first few rows
head(survey_data_small)
```

Tidyverse alternative

You can also use the `tidyverse` package to accomplish the same result: a new data frame containing only these eight variables:

```
library(tidyverse)

survey_data <- survey_data_full %>%
  select(tpcoiss, tpstress, toptim, tposaff,
         tnegaff, tlifesat, tslfest, tmarlow)
```

You can verify the structure of your new data frame using:

```
# Check the structure of the new data frame
str(survey_data_small)

# Or see the first few rows
head(survey_data_small)
```

```
# Calculate correlation matrix
correlation_matrix <- cor(survey_data_small)

# Round to 3 decimal places for clarity
round(correlation_matrix, 3)
```

2.3.5 Statistical Testing with `cor.test()`

The `cor.test()` function provides a full statistical test:

```
# Perform correlation test
correlation_test <- cor.test(x, y1)

# View complete results
print(correlation_test)
```

This output includes:

- The correlation coefficient
- The test statistic
- The p-value
- The confidence interval

2.3.6 Interpreting Correlation Results

When interpreting correlation results, consider:

1. **Strength**: Common guidelines for absolute values:
   - 0.00 to 0.19: "very weak"
   - 0.20 to 0.39: "weak"
   - 0.40 to 0.59: "moderate"
   - 0.60 to 0.79: "strong"
   - 0.80 to 1.00: "very strong"
2. **Direction**: Positive or negative relationship
3. **Statistical Significance**: Check the p-value
   - $p < 0.05$ typically indicates statistical significance
   - Consider effect size, not just significance
4. **Context**: What's meaningful in your field?

2.3.7 Common Pitfalls and Considerations

1. **Correlation Causation**: Correlation only indicates association, not causation
2. **Outliers**: Can strongly influence correlation coefficients
3. **Non-linear Relationships**: Pearson's correlation only measures linear relationships
4. **Missing Data**: Handle missing values appropriately

3 Summary

This lab introduced you to creating and loading datasets with R, manipulating data, and building basic tables and graphs (similar to the SPSS Lab 2). You can download the respective R scripts for Exercises A and B+C below, in case you created your own:

- R script for Exercises D, E, F, G