# Topic 3: Importing data into R

2023-08-08

In this topic, you will learn about :

- Importing from flat files with "utils"base package
- Importing Excel data file
- Importing data from other statistical software
- Exporting data

## Importing from flat files

### Importing from Flat Files with "utils" Base Package in R

The "utils" package is a part of the base R installation, and it provides useful functions for various utility operations, including importing data from flat files. Flat files are simple text files where data is stored in a tabular format, such as CSV (Comma-Separated Values) files.

Here are some commonly used functions from the "utils" package for importing data from flat files:

**1. read.table( ) and read.csv( ):**

These functions are used to read data from text files (including CSV files) and create data frames. The **read.table( )** function is more generic and can handle different file formats, whereas **read.csv( )** is specifically designed to read CSV files.

```
# Example: Reading data from a CSV file using read.csv()
my_data <- read.csv("data.csv")
```

**2. read.delim( ) and read.delim2( ):**

These functions are similar to **read.table( )**, but they are specifically designed to handle tab-delimited data.

```
# Example: Reading tab-delimited data from a text file using read.delim()
my_data <- read.delim("data.txt")
```

**3. read.csv2( ):**

This function is used to read CSV files with European-style decimal separators (using semicolons as separators and commas as decimal points).

```
# Example: Reading data from a CSV file with European-style decimal separators
my_data <- read.csv2("data.csv")
```

**4. read.fwf( ):**

This function is used to read data from a file with fixed-width format (each column has a fixed width).

```
# Example: Reading fixed-width data from a text file using read.fwf()
my_data <- read.fwf("data.txt", widths = c(10, 15, 8))
```

**5. read.table( ) with sep argument:**

The **read.table( )** function can be used to read data from files with custom **separators (e.g., semicolon, tab, space)** by specifying the sep argument.

```
# Example: Reading data from a file with a custom separator using read.table()
my_data <- read.table("data.txt", sep = ";")
```

Remember that the actual file paths may need to be specified based on your working directory and the location of the data files. Also, make sure that the flat files are correctly formatted to match the chosen function for importing.

Using the **"utils"** package in R provides a quick and efficient way to import data from various flat file formats, allowing you to perform data analysis and manipulation using the powerful features of R.

## Importing Excel data file

R provides several packages that allow you to read data from Excel files. Here, we'll focus on two popular packages: **readxl** and **openxlsx**.

**1. readxl** Package:

The readxl package is a simple and lightweight option for reading Excel data files. It reads both .xls and .xlsx formats.

Installation:

You need to install the readxl package first if you haven't done it already.

```
install.packages("readxl")
```

**Example: Reading Excel Data with readxl**

```
# Load the readxl package
library(readxl)

# Read data from an Excel file (xlsx format)
my_data <- read_excel("data.xlsx", sheet = "Sheet1")

# Print the first few rows of the data
print(head(my_data))
```

**2. openxlsx** Package:

The openxlsx package is more versatile and allows you to read, write, and manipulate Excel files. It supports **.xlsx** format.

Installation:

If you haven't installed openxlsx yet, you can do it using the following command:

```
install.packages("openxlsx")
```

**Example: Reading Excel Data with openxlsx**

```
# Load the openxlsx package
library(openxlsx)

# Read data from an Excel file (xlsx format)
my_wb <- loadWorkbook("data.xlsx")
my_data <- readWorkbook(my_wb, sheet = "Sheet1")

# Print the first few rows of the data
print(head(my_data))
```

Make sure to provide the correct path to your Excel file in the examples above. You may also need to specify the sheet name or index from which you want to read data.

Both readxl and openxlsx packages are excellent choices for importing Excel data into R. Choose the one that best fits your needs and data format. Once you've loaded the data into R, you can perform various data analysis and manipulation tasks using the power of R's data manipulation capabilities.

## Importing data from other statistical software

R provides various packages that enable you to import data from other statistical software formats. Some of the commonly used packages are **foreign, haven, and readstat.**

**1. foreign Package:** The foreign package is useful for reading data from other statistical software formats such as **SPSS, Stata, and SAS.**

Installation:

The foreign package is typically included with the base R installation, so you don't need to install it separately.

**Example: Reading Data from SPSS File**

```
# Load the foreign package (usually not required as it comes with base R)
library(foreign)

# Read data from an SPSS file (.sav format)
my_data <- read.spss("data.sav")

# Print the first few rows of the data
print(head(my_data))
```

**2. haven Package:**

The haven package is designed specifically for reading data from **SPSS and Stata** file formats. It provides more consistent handling of data types and value labels.

Installation:

If you haven't installed haven yet, you can do it using the following command:

```
install.packages("haven")
```

**Example: Reading Data from Stata File**

```
# Load the haven package
library(haven)

# Read data from a Stata file (.dta format)
my_data <- read_dta("data.dta")

# Print the first few rows of the data
print(head(my_data))
```

**3. readstat Package:**

The readstat package is a versatile package that supports reading data from **SPSS, Stata, and SAS** file formats.

Installation:

If you haven't installed readstat yet, you can do it using the following command:

```r
install.packages("readstat")
```

**Example: Reading Data from SAS File**

```r
# Load the readstat package
library(readstat)

# Read data from a SAS file (.sas7bdat format)
my_data <- read_sas("data.sas7bdat")

# Print the first few rows of the data
print(head(my_data))
```

Choose the appropriate package based on the format of the data you want to import. The examples above demonstrate how to read data from SPSS, Stata, and SAS files, but each package supports other file formats as well. Once you've successfully imported the data, you can perform data analysis and visualization in R as needed.

## Exporting data

R provides several functions and packages for exporting data to various file formats. Exporting data is essential when you want to save the results of your data analysis, share data with others, or use the data in other applications. Here are some common methods for exporting data in R:

**1. write.table( ) and write.csv( ):**

These functions are used to write data frames to text files in tabular format, such as CSV files.

**Example: Exporting Data to CSV**

```r
# Create a sample data frame
my_data <- data.frame(
  Name = c("John", "Jane", "Mike"),
  Age = c(25, 30, 22),
  Score = c(80, 85, 70)
)

# Export data to CSV file
write.csv(my_data, file = "data.csv", row.names = FALSE)
```

**2. write.xlsx( ):**

The **openxlsx** package provides the **write.xlsx( )** function to export data frames to Excel files **(.xlsx format)**.

Installation:

If you haven't installed **openxlsx** yet, you can do it using the following command:

```r
install.packages("openxlsx")
```

**Example: Exporting Data to Excel**

```r
# Load the openxlsx package
library(openxlsx)

# Export data to Excel file
write.xlsx(my_data, file = "data.xlsx")
```

**3. write_dta( ):**

The haven package provides the **write_dta( )** function to export data frames to **Stata files (.dta format)**.

**Example: Exporting Data to Stata**

```r
# Load the haven package
library(haven)

# Export data to Stata file
write_dta(my_data, file = "data.dta")
```

**4. write_sav( ):**

The haven package also provides the **write_sav( )** function to export data frames to **SPSS files (.sav format)**.

**Example: Exporting Data to SPSS**

```r
# Load the haven package
library(haven)

# Export data to SPSS file
write_sav(my_data, file = "data.sav")
```

**Custom Export Formats:**

You can also create custom export functions to save data in formats not natively supported by R. For example, you can write custom functions to save data in **JSON, XML**, or other file formats using packages like **jsonlite or XML**.

Remember to specify the correct file path when exporting data. You can use either an absolute path or a relative path based on your working directory.

Exporting data in R allows you to save the results of your analysis for further use or share the data with colleagues and collaborators. Choose the appropriate method based on the desired file format and the specific requirements of your project.