Importing Data in R - Exploring Packages and Their Advantages

Yassaman Shafabakhsh (Matriculation Number:400806611)

International Management, M.A., Hochschule Fresenius - University of Applied Science

Author Note

Correspondence concerning this article should be addressed to Yassaman Shafabakhsh (Matriculation Number:400806611), Email: shafabakhsh.yassaman@stud.hs-fresenius.de

Abstract

Importing data is a critical first step in data analysis, ensuring accuracy and efficiency throughout the workflow. R offers several tidyverse-friendly packages—such as readr, readxl, haven, rio, and jsonlite—that simplify loading data from formats like CSV, Excel, SPSS, JSON, and databases. This paper provides an overview of the most widely used import packages in R, highlighting their functions, advantages, limitations, and example usage. Choosing the right package helps maintain data integrity and supports smooth, reproducible analysis in R.

Keywords: R programming, data import, tidyverse, readr, readxl, haven, jsonlite, rio, vroom, readODS, DBI, data preprocessing, R packages

Importing Data in R - Exploring Packages and Their Advantages

In data analysis, importing data is an essential initial step that allows analysts and researchers to load external data into R for examination, cleaning, visualization, and statistical analysis. R is highly versatile and provides multiple convenient methods for importing different data formats through dedicated packages, each optimized for specific file types and uses. Selecting the right method is crucial for maintaining data integrity and ensuring efficient analysis workflows.

Why Importing Data Matters

Importing data is one of the foundational steps in data analysis, setting the stage for accurate and effective research outcomes. The process of importing data goes beyond merely loading files but it also involves making sure your data is accurate, easy to work with, and compatible with your tools. Handling this first step correctly greatly affects the quality and reliability of your entire analysis.

1. Accurate data importing is crucial because any errors at this stage can propagate and amplify throughout the entire analysis process, leading to incorrect conclusions and misguided decisions. Proper import methods ensure:

- Data Integrity: Data types (numeric, factor, date) are recognized and assigned correctly, preserving original data meaning.
- Reduction of Errors: Correctly handling missing values, special characters, and encoding issues helps prevent common mistakes that can affect the validity of analytical results.
- Consistency: Ensuring data maintains its original structure and formatting across different analytical sessions.

2. Efficiency in importing data becomes especially critical when dealing with large datasets or frequent data updates. Efficient importing provides:

- Time Savings: Automation and streamlined packages such as rio or data.table reduce manual intervention, speeding up the analysis workflow significantly.

- Scalability: Properly optimized import methods allow analysts to easily manage and repeatedly import large datasets without performance degradation, enabling smoother data exploration and manipulation.

3. Data often originates from multiple sources and formats—CSV, Excel, SPSS, Stata, SAS, databases, and more. Compatibility in importing ensures:

- Flexibility: Analysts can seamlessly incorporate data from diverse software environments, allowing comprehensive, integrated analysis without manual conversion or data loss.

- Versatility: Use of specialized packages like haven or readxl ensures accurate handling of specific data formats, preserving metadata and minimizing the effort involved in translating data formats.

- Collaboration and Accessibility: Maintaining compatibility with common data formats supports easier collaboration among researchers who may use different software, fostering effective interdisciplinary teamwork.

In summary, correctly importing data in R is not merely a technical step—it is a critical phase ensuring accuracy, efficiency, and compatibility, thereby laying a strong foundation for successful and reliable data analysis.

Using R Packages for Importing Data

R provides specialized packages designed to simplify the process of importing data. These packages offer user-friendly tools that help you quickly and accurately load various data formats into your R environment. Each package is optimized for particular types of data files, such as CSV, Excel, SPSS, SAS, and Stata, enabling you to easily handle diverse datasets. By using the right package, you ensure smoother data processing, fewer errors, and a more efficient workflow throughout your analysis.

R has several powerful packages that simplify data import. These are popular data-importing packages from the tidyverse, along with clear examples of datasets you might typically import with each:

1. **readr**: Efficiently reads CSV, TSV, and other delimited text files.

   Function: read_csv(), read_tsv()

   Advantage:

- Fast and efficient and can handles large files
- Good handling of missing values

   Disadvantage:

- Only works with delimited text files (CSV, TSV)
- No support for Excel, SPSS.

   Example dataset: mtcars.csv

```
Rows: 32 Columns: 11
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (11): mpg, cyl, disp, hp, drat, wt, qsec, vs, am, gear, carb

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

   Furthermore, explore readr's other functions here

2. **readxl**: Used to read Excel (.xls and .xlsx files)

   Function: read_excel()

   Advantage:

- No dependency on external software like Excel
- Simple syntax

- Supports multiple sheets

Disadvantage:

- Can only read/import data from Excel files, but cannot write or export data back to Excel files.
- Does not support Excel formulas or advanced formatting.

Example dataset: “iris.xlsx”

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

3. **rio**: Used to all-in-one package for importing/exporting multiple formats.

Function: import()

Advantage:

- One function for many formats e.g. “import(”file.csv”)“.
- Supports .csv, .xlsx, .sav, .dta, .json, .tsv, .rds, and many more.
- Very beginner-friendly.

Disadvantage:

- Internally depends on other packages (e.g., readxl, haven, etc.)
- Less control for advanced users.

To know more about this package, refer to *Import and Export* (2025).

Example #1: mtcars dataset

	rownames	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
2	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
3	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
4	Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
5	Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
6	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2

Example #2: gapminder.csv

	country	year	pop	continent	lifeExp	gdpPercap
1	Afghanistan	1952	8425333	Asia	28.801	779.4453
2	Afghanistan	1957	9240934	Asia	30.332	820.8530
3	Afghanistan	1962	10267083	Asia	31.997	853.1007
4	Afghanistan	1967	11537966	Asia	34.020	836.1971
5	Afghanistan	1972	13079460	Asia	36.088	739.9811
6	Afghanistan	1977	14880372	Asia	38.438	786.1134

4. **readODS**: Used to import .ods files in R. This package is useful if you work with data from organizations, governments, or partners who use open-source tools and send .ods files.

   Function: read_ods()

   Advantage:

- No need for Excel: Works without requiring Microsoft Excel or Java.
- Simple syntax: Similar to read_excel() and read_csv() functions, making it easy to use.
- Cross-platform support: works on Windows, Mac, and Linux systems.

Disadvantage:

- Less commonly used in practice: .ods files are not as widely used as .xlsx or .csv formats.
- Limited features: It primarily supports reading .ods files. It does not support writing/saving .ods files.
- No advanced formatting support: cell formatting, formulas, and charts are not preserved or interpreted during import.

```
# A tibble: 6 x 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
         <dbl>       <dbl>        <dbl>       <dbl> <chr>
1          5.1         3.5          1.4         0.2 setosa
2          4.9         3            1.4         0.2 setosa
3          4.7         3.2          1.3         0.2 setosa
4          4.6         3.1          1.5         0.2 setosa
5          5           3.6          1.4         0.2 setosa
6          5.4         3.9          1.7         0.4 setosa
```

5. **DBI + RSQLite**: The DBI package lets R connect to different databases like SQLite, MySQL, MariaDB, PostgreSQL, SQL Server, and Oracle.

   Function: dbReadTable(), dbConnect()

   Advantage:

- Easy-to-use functions for many types of databases.
- Good for large amounts of data.
- Works with different database packages.

Disadvantage:

- Needs extra packages depending on the database.

- You must have access to the database.
- Some SQL knowledge helps for complex tasks.

6. **vroom**: Used for fast reading of large delimited files.

   Function: vroom()

   Advantages:

- Extremely fast for big files
- Tidyverse-friendly

   Disadvantages:

- Less stable than readr in rare edge cases

7. **haven**: Used to Import SPSS, Stata, and SAS data

   Function: read_sav(), read_dta(), read_sas()

   Advantage: - Keeps the extra information from the original file — like the full names of questions or what values mean (e.g., 1 = Male, 2 = Female) — so you don't lose important details when you import the data. - Tidyverse-friendly (returns tibbles) - Allows to easily work with data from those programs such as SPSS (.sav), Stata (.dta), and SAS (.sas7bdat) directly in R without needing to convert the files manually.

   Disadvantage:

   - Only for SPSS/Stata/SAS formats - Slightly slower on very large datasets compared to readr

   Example dataset: survey_data.sav

8. **jsonlite**: Used for reading JSON files (tidyverse-compatible).

   Function: fromJSON()

   Advantage:

- Handles nested data
- Converts to lists or data frames
- Good for API data
- real time data from websites or online services like weather, news, or stock prices.

Disadvantage:

- Output may need restructuring for tidy analysis
- Not part of core tidyverse (but compatible)

For deeper exploration of importing data using R packages, watch these videos StatQuest with Josh Starmer (2018) QuantPsych (2020).

Also refer to Team (2000) and Wickham et al. (2017) to get more information on importing data using R packages.

Furthermore, explore Import data with c() which gives a view on importing different kinds of datasets.

Conclusion

Importing data is a critical first step in any data analysis workflow, and R offers a wide range of packages tailored to different data formats and sources. Whether working with spreadsheets, statistical software files, databases, or web-based APIs, R provides efficient and tidyverse-friendly tools to streamline the process.

Packages like readr, readxl, and haven enable smooth integration of CSV, Excel, and SPSS data, while tools like rio and vroom offer flexibility and speed. For advanced needs, packages such as DBI, jsonlite, and rvest allow users to access structured databases, real-time APIs, and web data.

By choosing the right package for each data type and following tidy principles, analysts and researchers can ensure that their data is loaded accurately, cleanly, and in a format ready for immediate analysis and visualization in R.

References

Import and export. (2025). https://epirhandbook.com/en/new_pages/importing.html

QuantPsych. (2020). *Three ways to import data into RStudio*. https://www.youtube.com/watch?v=Owf71im-4BE.

StatQuest with Josh Starmer. (2018). *How to import data in r*. https://www.youtube.com/watch?v=e8B9YU_M5FM.

Team, R. C. (2000). *R data import/export*.

Wickham, H., Grolemund, G., et al. (2017). *R for data science* (Vol. 2). O'Reilly Sebastopol, CA.

Affidative

I hereby affirm that this submitted paper was authored unaided and solely by me. Additionally, no other sources than those in the reference list were used. Parts of this paper, including tables and figures, that have been taken either verbatim or analogously from other works have in each case been properly cited with regard to their origin and authorship. This paper either in parts or in its entirety, be it in the same or similar form, has not been submitted to any other examination board and has not been published.

I acknowledge that the university may use plagiarism detection software to check my thesis. I agree to cooperate with any investigation of suspected plagiarism and to provide any additional information or evidence requested by the university.

Checklist:

- ☒ The handout contains 3-5 pages of text.
- ☒ The submission contains the Quarto file of the handout.
- ☒ The submission contains the Quarto file of the presentation.
- ☒ The submission contains the HTML file of the handout.
- ☒ The submission contains the HTML file of the presentation.
- ☒ The submission contains the PDF file of the handout.
- ☒ The submission contains the PDF file of the presentation.

- ☒ The title page of the presentation and the handout contain personal details (name, email, matriculation number).
- ☒ The handout contains a abstract.
- ☒ The presentation and the handout contain a bibliography, created using BibTeX with APA citation style.
- ☒ Either the handout or the presentation contains R code that proof the expertise in coding.
- ☒ The handout includes an introduction to guide the reader and a conclusion summarizing the work and discussing potential further investigations and readings, respectively.
- ☒ All significant resources used in the report and R code development.
- ☒ The filled out Affidavit.
- ☒ A concise description of the successful use of Git and GitHub, as detailed here: https://github.com/hubchev/make_a_pull_request.
- ☒ The link to the presentation and the handout published on GitHub.

[Yassaman Shafabakhsh,] [28/05/2025,] [Cologne]