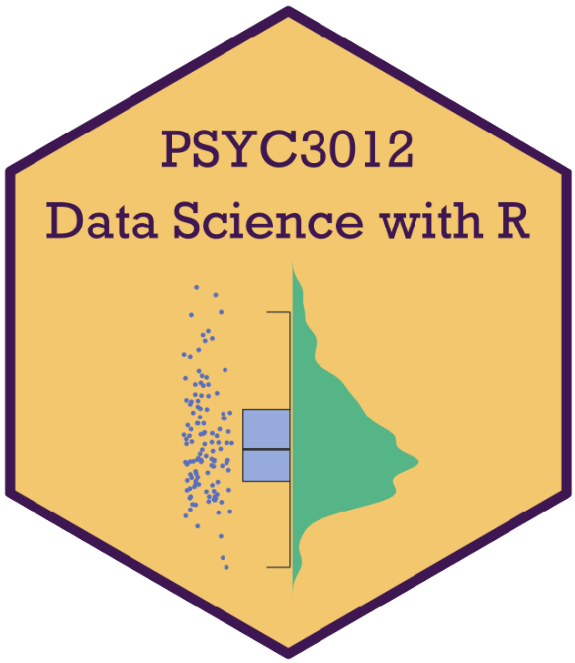
Data Science with R (PSYC3012)



true true true

Contents

Introduction

Data Science with R (PSYC3012) is a 15-credit optional module that aims to develop students' research skills relying on three main themes: Open Science, digital literacy, and science communication.

The module covers data wrangling, data visualization, categorical data analysis, general linear models, and robust statistics using R—a free software environment for statistical computing and graphics. Each session is grounded in Open Science best practices to favor reproducibility and transparency in science, following the guidelines of the 2019 Concordat to Support Research Integrity, the UK Reproducibility Network, and the 2016 Concordat on Open Research Data.

Upon successful completion of this module, students should be able to:

- Recognize, explain, and apply open science best practices following the Concordat on Open Research Data and the UK Reproducibility Network guidelines.
- Identify and describe the appropriate analytical procedures required to answer research questions related to categorical data analysis and general linear models.
- Apply R to organize and tidy the data, to model the data, and to generate appropriate data visualizations.
- Inspect and appraise the outputs generated with R to produce with Rmarkdown a reproducible research report that will communicate the results relying on the principles of transparency and reproducibility in science.

How to use this book

This book is intended to be the handbook of the optional module **Data Science with R** (PSYC3012); a module delivered for the Psychology undergraduate provisions at De Montfort University (Leicester, United Kingdom). This book includes all the content and reproducible examples that will be delivered in the 2-hour weekly sessions (Term 1, Academic Year 2023-24).

It is recommended to read in advance the corresponding chapters and sections that will be explained in class. Similarly, it is recommended to install **R**, **RStu-**

dio, and the required R packages to work on the examples of this book before attending the first lecture and workshop. Please, read the software installation guidelines before reading the book (Appendix A).

About the Author

Dr Carlos Crivelli is an Associate Professor/Reader in Affective Science and Social Interaction in the School of Applied Social Sciences at De Montfort University (Leicester, United Kingdom).

After completing a BSc (Hons) Psychology, Dr Crivelli received his MSc and PhD in Methodology of Behavioral and Health Sciences from Universidad Autónoma de Madrid (Madrid, Spain). His research interests focus on three main areas: social influence and emotion science, cross-cultural psychology, and data science. He has published on these topics in journals like *Trends in the Cognitive Sciences*, *Proceedings of the National Academy of Sciences, U.S.A.*, *Perspectives on Psychological Science*, and *Current Directions in Psychological Science*.

Dr. Crivelli is an Honorary Fellow of the University of Melbourne (Australia), a Fellow of the Higher Education Academy, and a member of the editorial board of *Nature–Scientific Data*. He teaches undergraduate and postgraduate courses in Psychometrics and Data Science in R, Personality and Intelligence, and Conceptual Issues and Critical Debates in Psychology, embedding the intercultural dimension, and open and reproducible research into the curriculum.

Chapter 1

What is Open Science?

LEARNING OUTCOMES

- Identify and explain Open Science's principles.
- Compare the different initiatives promoting Open Science nationally and internationally.
- Appraise the implications of using Open Science's principles of reproducibility, openness, and replicability in the research process.
- Apply Open Science to comply with the regulations of research institutions and funders, governments, and publishers.
- Identify the steps required and implement the best practices to promote data sharing, open code and software, and data curation.

According to the 2021 United Nations Educational, Scientific and Cultural Organization (UNESCO) Recommendation on Open Science, "… **Open Science** is defined as an inclusive construct that combines various movements and practices aiming to make multilingual scientific knowledge openly available, accessible and reusable for everyone, to increase scientific collaborations and sharing of information for the benefits of science and society, and to open the processes of scientific knowledge creation, evaluation and communication to societal actors beyond the traditional scientific community. It comprises all scientific disciplines and aspects of scholarly practices, including basic and applied sciences, natural and social sciences and the humanities, and it builds on the following key pillars: open scientific knowledge, open science infrastructures, science communication open engagement o societal actors and open dialogue with other knowledge systems." (2017, p. 7)

For The European Commission's task force on Research and Innovation (2021), "**Open Science** is a system change allowing for better science through open and collaborative ways of producing and sharing knowledge and data, as early

as possible in the research process, and for communicating and sharing results. This new approach affects research institutions and science practices by bringing about new ways of funding, evaluating, and rewarding researchers. Open Science increases the quality and impact of science by fostering reproducibility and interdisciplinarity. It makes science more efficient through better sharing of resources, more reliable through better verification and more responsive to society's needs" (p. 1)

In the United Kingdom, different regulations on open science (also named *Open Research* or *Open Scholarship*) has been developed. For example, the 2019 Concordat to Support Research Integrity (Universities UK) or the 2016 Concordat on Open Research Data (HEFCE, Research Councils UK, Wellcome Trust, Universities UK) have captured some of the initiatives on Open Science discussed and developed by other policy makers, institutions, publishers, funders, and research councils. It is important to note that the UK Reproducibility Network—supported by UK Research Institute, the British Psychological Society, Wellcome Trust, Cancer Research UK, UK Data Service, or Universities UK—has played a pivotal role in "seeking to understand the factors that contribute to poor research reproducibility and replicability, and to develop approaches to counter these and improve the quality of the research we produce" (UK Reproducibility Network, 2021, p. 1).

Open Science

Open Science is a broad term used to encompass the promotion of *transparency*, *reproducibility*, *research integrity*, and *societal impact (research and innovation)*.

Because of those principles, **Open Science** can be reconceived as a meta-scientific movement interested in enhancing the following skills:

Digital content creation

Science communication

Information and data literacy

To empower researchers at every career stage, different organizations and Higher Education institutions have launched innovative educational programes along with institutional statements and strategies to promote Open Science best practices. For example, the University of Sheffield has launched an Open Science strategy to promote a culture of research excellence. The University of Manchester has created the Office for Open Research led big five strategic priorities: (1) open research skills, (2) open research communities, (3) open research recognition, (4) open research workflows, and (5) open and FAIR research outputs. Similarly, the University of Surrey developed a 5-year open research strategic goals and action plan that emphasizes the awareness, training and advocacy of open practice across the university in research, teaching, and learning.

In December 2021, Loughborough University published an Open Research Po-

Table 1.1: Open Science Skills and Required Training

Skills	Training
Digital content creation	Copyright and intellectual property (CC-BY)
	Management and use of institutional repositories (DORA)
	OA publishers (e.g., bookdown)
	Open publication options (e.g., Gold, Green)
	Data repositories (OSF, Zotero, Figshare, Github)
	Data management plan
	Data presentation
Science communication	Bibliometrics, Altmetrics, and researcher impact
	DMU/Institutional webpage
	Personal brand (e.g., ORCID, Researchgate, Google scholar, Scopus)
	Public engagement (e.g., Leicester Business Festival)
	Research informs teaching (handbooks, monographs, tutorials)
	Innovation (e.g., the role of visual statistics during COVID-19)
Information and data literacy	Data analysis and visualization
	Data wrangling, modeling
	Text mining (qualitative research)
	Secondary sources (e.g., spatial data, census)
	Reproducibility and data reuse
	Publish in data journals
	Transparency
	Research integrity

sition Statement to develop, review, and promote an institutional policy framework for Open Research. Echoing the University of Manchester's strategic priorities, Loughborough University has emphasized the pivotal role of training and support for PhD students and senior academics to update their skills for open data and open methods (i.e., science communication, information and data literacy). Other universities stressing the importance of mastering digital literacy skills for open data and code/software (e.g., programming in R/Python, data sharing) are the Open Research Skills Framework at the University of York, King's College and their Open Research Group Initiative (KORGI), the University of Glasgow, the Reproducible Research Oxford group (RROx) at the University of Oxford, the Birmingham Environment for Academic Research's (BEAR) software carpentry training (in R, Python, Matlab, and Git) at the University of Birmingham, Open Research education for doctoral students at Imperial College London, or the Edinburgh Open Research Initiative of the University of Edinburgh.

1.1 The credibility of Science

The academic/research career runs in parallel with a cycle of credibility in which trust and reliability are sought (Miedema, 2022) (Figure 1.1).

To conduct any research project, we need staff (at least, one researcher) and equipment. Then, we need to design a research proposal and collect data. The data needs to be properly analyzed, requiring a set of research skills that enabled researchers to design the research proposal first, to collect data ensuring proper

levels of quality and rigor, and the necessary skills to describe, model, visualize, and interpret the findings. The findings are supported by arguments (some of them are theoretical, whereas others are methodological) and often presented as a manuscript or research report. However, to publish those manuscripts and research reports, researchers need to demonstrate appropriate and highly valued research skills (e.g., deep knowledge of the theoretical approaches and literature review, critical evaluation skills, understanding of the methods and data analytical techniques used), not only to create those products for internal consumption, but to be accepted by the scientific community after passing the judgment of senior and experts peers (the peer-review system).

Once the products of our research are published, different metrics are used to provide the context of the quality of the research process. Metrics such as the Journal Citation Report impact factors, H-index, Google Scholar metrics, or Altmetrics are used for recognition and are a proxy to secure or increase the chances to secure funding, a good research/academic job, promotions, or to accrue more equipment and staff (e.g., research associates, post-docs, PhD students).

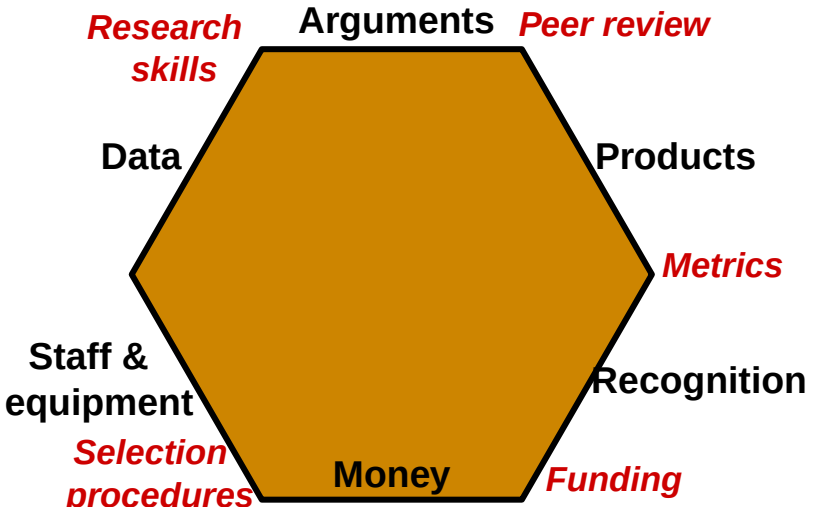


Figure 1.1: The credibility cycle

1.2 Questionable scientific practices

In the 2010s, disciplines such as psychology, economics, or the biomedical sciences were engulfed by a turmoil of distrust and lack of credibility. Although the Open Science movement is not new and it is at the core of the scientific method, a scientific reform movement emerged to inform on questionable scientific practices (John, Loewenstein, & Prelec, 2012). Synchronically, a heated

debate emerged to find solutions to the credibility crisis; a crisis that was triggered by the lack of openness, transparency, and replicability (Spellman et al., 2017).

Questionable Scientific Practices

Failing to report all dependent measures

Collecting more data after seeing whether results were significant

Failing to report all conditions

Stopping collecting data earlier than planned because one found the result that one had been looking for

Rounding down p values selectively reporting

Selectively reporting studies that "worked"

Excluding data after looking at the impact of doing so

Reporting an unexpected finding as having been predicted from the start

Falsely claiming that results are unaffected by demographic variables (e.g., gender) when one is unsure (or knows that they do)

Falsifying data

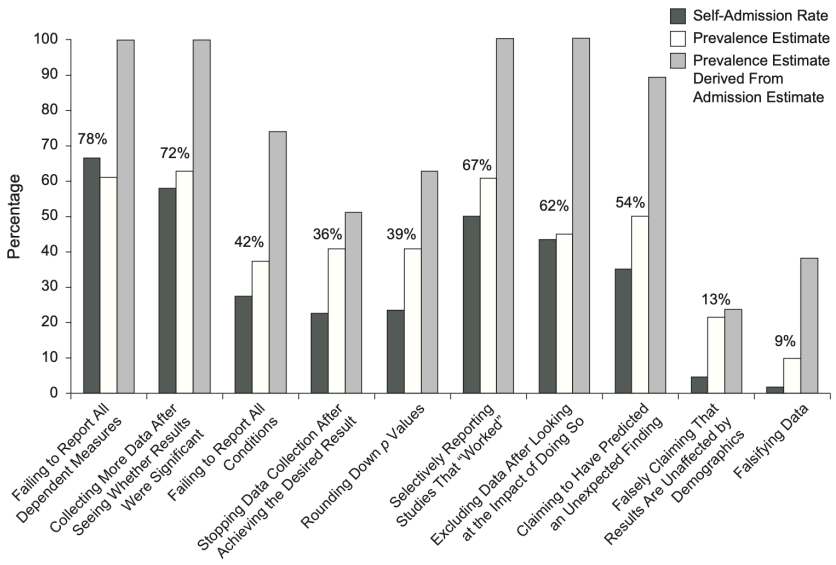


Figure 1.2: Prevalence of questionable research practices (John, Loewenstein, & Prelec, 2012)

1.3 The credibility crisis

One of the most cited papers of the last decade (7,243 citations in January 2023) (Open Science Collaboration, 2015) conducted a large-scale study in which 100 classic psychological experiments were put to a test using an international, multi-lab approach. The main goal of the study was to estimate the reproducibility of psychological science. Unfortunately, the results showed that most of the classic psychological studies that psychology students learn in college and that most researchers cite as settled science couldn't be replicated. This meta-scientific endeavor has been conducted with similar results in other fields such as the behavioral, cognitive, economic, health, and medical sciences.

The most telling results showed that:

- psychologists tend to use under-powered samples that overestimate the effect size and produce low reproducibility of results
- there is a lack of access to full methods
- there is a lack of access to analytic procedures and code
- the file-drawer problem is pervasive as editors and journals are keen to "find results" (i.e., rejection of the Null Hypothesis, rather that publish reports with null effects)
- there is a lack of access to publications
- there is discontent with reporting and use of standard statistics

1.4 Threats to the quality of the scientific process

Kovacs and colleagues (2021) conducted a research project to elucidate the origins and causes of the most frequent mistakes committed by researchers that damaged the quality of the scientific process. Using a mix of quantitative and qualitative approaches to data collection, they surveyed 488 researchers publishing in psychology journals between 2010 and 2018. They found four supra-ordinal categories or metagroups and their corresponding causes (Table 1.2).

Figure 1.4 shows the mistakes reported by Kovacs et al. (2021) organized by the stages of research data management. To mitigate these mistakes different solutions have been proposed. For example, to define the data properly and to avoid ambiguous naming or to improve poor documentation practices, the use of codebooks, data management plans, and transparent research workflows is advisable (e.g., see Arslan, 2019). Similarly, to avoid wrong data processing and analysis, programming errors or loss of materials/data, using statistical code languages such as R or Python with embedded comments and storing the code in public repositories like Open Science Framework, Figshare, or GitHub is recommended (Klein et al., 2018).

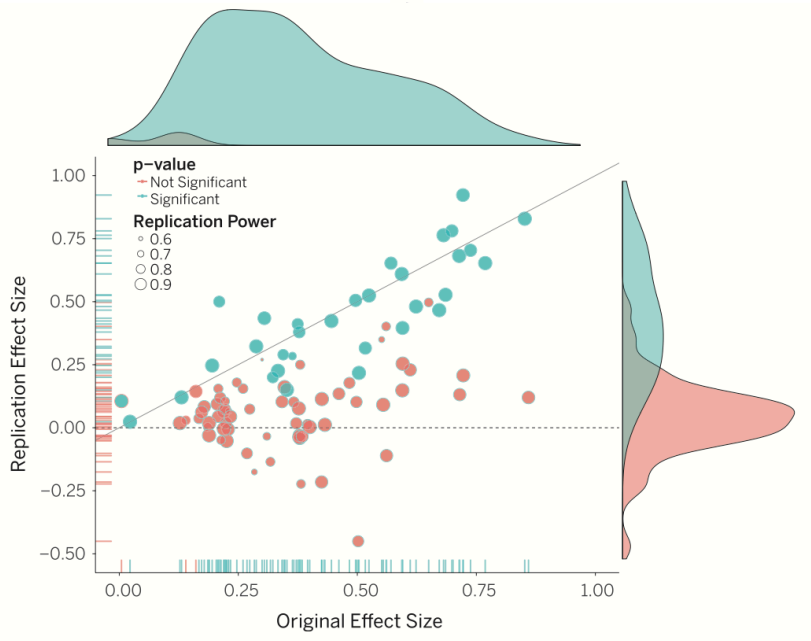


Figure 1.3: Reproducibility of psychological science (Open Science Collaboration, 2015)

Table 1.2: Metagroups for Mistake Causes

Metagroup	Cause group
Poor project preparation or management	Bad or lack of planning
	Bad or lack of standards
	Bad skill management
	Miscommunication
	Failure to automate an error prone task
	Time management issue
External difficulties	High task complexity
	Technical issues
Lack of knowledge	Lack of knowledge/experience
Personal difficulties	Carelessness
	Inattention
	Lack of control
	Overconfidence
	Physical or cognitive constraints

Note:
Table reproduced from Kovacs, Hoekstra, and Aczel (2021).

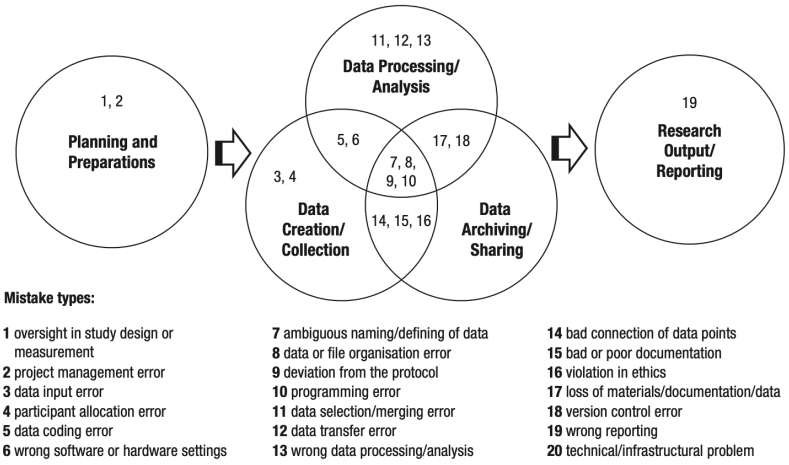


Fig. 6. Mistake types categorized by research data-management stage. The numbers indicate the mistake types (see https://osf.io/76d24/). Mistake 20 (technical or infrastructure problems) is not part of any stage because it is an external factor but can have an effect on the efficiency of the data-management pipeline.

Figure 1.4: Threats to the quality of the scientific process (Kovacs, Hoekstra, & Aczel, 2021)

1.5 Open Data

Data sharing is a core element in Open Science to enhance the credibility and trust on our research practices. Data sharing allows for the reproducibility of previous findings, while enabling others to inspect and spot mistakes in the data, the meta-data, or the research workflow. Additionally, it prevents—to a certain extent—the emergence of scientific fraud due to fabricated data/results/visualizations or to the use of questionable data analytical practices.

Data sharing also favors a swift progress in fields of research with difficulties in gaining access to relevant data (Gewin, 2016). It also enables researchers working at institutions based on countries with less resources and limited access to paywall journals and repositories.

However, sharing data has its drawbacks. Some researchers invest years to collect data that is very difficult to obtain (e.g., long-term primate behavior in Kibale National Park, Uganda) with the idea of completing a research project that will produce several outputs. If the data is shared in the first publication, sharing might be a problem for future publications (Hunt, 2019). To partially mitigate this problem, many journals allow researchers to publish the data sets in specialized data journals such as *Nature—Scientific Data*, *GigaScience*, *BMC Research Notes*, or *Data in brief*. For example, in *Nature—Scientific Data*, these data descriptors are curated and can be modified to incorporate new data collected later. Moreover, Nature-titled journals do not consider prior Data

Table 1.3: The FAIR Guiding Principles

Principles	Guidelines
Findable	F1. (meta)data are assigned a globally unique and persistent identifier F2. data are described with rich metadata (defined by R1 below) F3. metadata clearly and explicitly include the identifier of the data it describes F4. (meta)data are registered or indexed in a searchable resource
Accessible	A1. (meta)data are retrievable by their identifier using a standardized communications protocol A1.1 the protocol is open, free, and universally implementable A1.2 the protocol allows for an authentication and authorization procedure, where necessary A2. metadata are accessible, even when the data are no longer available
Interoperable	I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation I2. (meta)data use vocabularies that follow FAIR principles I3. (meta)data include qualified references to other (meta)data
Reusable	R1. meta(data) are richly described with a plurality of accurate and relevant attributes R1.1. (meta)data are released with a clear and accessible data usage license R1.2. (meta)data are associated with detailed provenance R1.3. (meta)data meet domain-relevant community standards

Descriptor publications to compromise the novelty of new manuscript submissions if those manuscripts go substantially beyond a descriptive analysis of the data, and report important new scientific findings appropriate for the journal in question (Nature—Scientific Data, 2023).

Similarly, sharing data is not as easy and straightforward as it seems (e.g., to upload my anonymized csv file into a public repository). Researchers—especially postgraduate students and early career researchers-—must learn a new set of skills in order to publish their research products:

- Data curation
- Data management plan
- Storing, saving, archiving, and data preservation
- Meta-data
- Data analysis and visualization
- Data wrangling
- Reproducibility and data reuse
- Compliance with FAIR data principles

1.6 FAIR Data

In 2016, the data journal *Nature—Scientific Data* published the *FAIR Guiding Principles for Scientific Data Management and Stewardship* to clarify and guide researchers with the FAIR principles: Findable, Accessible, Interoperable, Reusable. The FAIR principles pivot around the idea of a continuum of "machine-actionability" in which machines can act autonomously on data objects, (1) in relation to the contextual metadata surrounding a digital object (*what is it?*), and (2) when referring to the content (*how do I process it/integrate it?*) (Wilkinson, 2016).

1.7 Contributorship (CRediT)

The *Contributor Roles Taxonomy* (CRediT) has been designed to cover 14 key roles representing the spectrum of activities involved in the production of research outputs. Most of the big publishers (e.g., Nature, Elsevier, SAGE, Cell Press, Wiley) and psychological societies (American Psychological Association [APA]), Association for Psychological Science [APS]) have made the CRediT system compulsory.

The Contributor Roles Taxonomy (CRediT)

Conceptualization

Data curation

Formal analysis

Funding acquisition

Investigation

Methodology

Project administration

Resources

Software

Supervision

Validation

Visualization

Writing—original draft

Writing—review and editing

Some of the desired consequences of this contributor system are to avoid:

- **Ghost authorship**: Authors who contributed to the work but are not listed, generally to hide a conflict of interest from editors, reviewers, and readers.
- **Gift authorship**: Individuals given authorship credit who have not contributed in any substantive way to the research but are added to the author list by virtue of their stature in the organization.
- **Orphan authorship**: Authors who contributed materially to the work but are omitted from the author list unfairly by the drafting team.
- **Forged authorship**: Unwitting authors who had no part in the work but whose names are appended to the paper without their knowledge to increase the likelihood of publication.

1.8 References

Arslan, R. C. (2019). How to automatically document data with the codebook package to facilitate data reuse. *Advances in Methods and Practices in Psychological Science*, *2*(2), 169—187.

European Commission. (2021). Open Science [Fact sheet]. European Union.

Gewin, V. (2016). Data sharing: An open mind on open data. *Nature*, *529*, 117—119.

Hunt, L. T. (2019). The life-changing magic of sharing your data. *Nature Human Behavior*, *3*, 312—315.

John, L. K., Loewenstein, G., & Prelec, D. (2012). Measuring the prevalence of questionable research practices with incentives for truth telling. *Psychological Science*, *23*(5), 524—532.

Klein O., Hardwicke T. E., Aust F., Breuer J., Danielsson H., Mohr A. H., Ijzerman H., Nilsonne G., Vanpaemel W., Frank M. C. (2018). A practical guide for transparency in psychological science. *Collabra: Psychology*, *4*(1):20.

Kovacs, M., Hoekstra, R., & Aczel, B. (2021). The role of human fallibility in psychological research: A survey of mistakes in data management. *Advances in Methods and Practices in Psychological Science*, *4*(4), 1—13.

Miedema, F. (2022). Open Science: The very idea. Springer.

Nature–Scientific Data (2023). Frequently Asked Questions. Retrieved from Nature–Scientific Data website: https://www.nature.com/sdata/faq

Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. *Science*, *349*(6251), 1—8.

Spellman, B., Gilbert, E. A., & Corker, K. S. (2017, September 20). Open Science: What, Why, and How. https://psyarxiv.com/ak6jr/

UK Reproducibility Network. (2021, September). Terms of Reference. Version 3.3.

United Nations Educational, Scientific, and Cultural Organization. (2021, November). UNESCO Recommendation on Open Science (Programme Document SC-PCB-SPP/2021/OS/UROS). Retrieved from the United Nations Educational, Scientific, and Cultural Organization website: https://unesdoc.unesco.org/ark:/48223/pf0000379949

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J. W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., … Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, *3*:160018.

Chapter 2

Introduction to R

LEARNING OUTCOMES

- Identify the capabilities of R and RStudio's environment and appraise their functionality.
- Distinguish between R functions, objects, and diverse data wrangling approaches.
- Apply basic programming skills to import and organize the data using classic data wrangling approaches and the **tidyverse** grammar.
- Evaluate the R code and appraise the outputs to demonstrate a satisfactory level of basic programming skills in R.

R is a programming language and an open-source software environment for statistical computing and graphics available for Windows, MacOS, and Linux operating systems (R Core Team, 2023). R is an interactive environment for data science that enables us to import, manipulate, model, and visualize data to effectively communicate our results (Wickham & Grolemund, 2017). The use of R also promotes transparency and open research, with a clear focus on supporting reproducibility policies (Gandrud, 2015).

R, in its most basic form, is composed by (1) a *script* in which we write the lines of R code that we run, and (2) the *console* that displays the results and debugging (Figure 2.1). If we generate a plot, it will emerge as a pop-up window, showing one plot at a time. For more details on what is R, click here.

Why Use R?

It is a free and open-access programming environment for data science

Reproducible and transparent research in psychological science

Unlimited capabilities for data science (e.g., analysis, modelling, visualization)

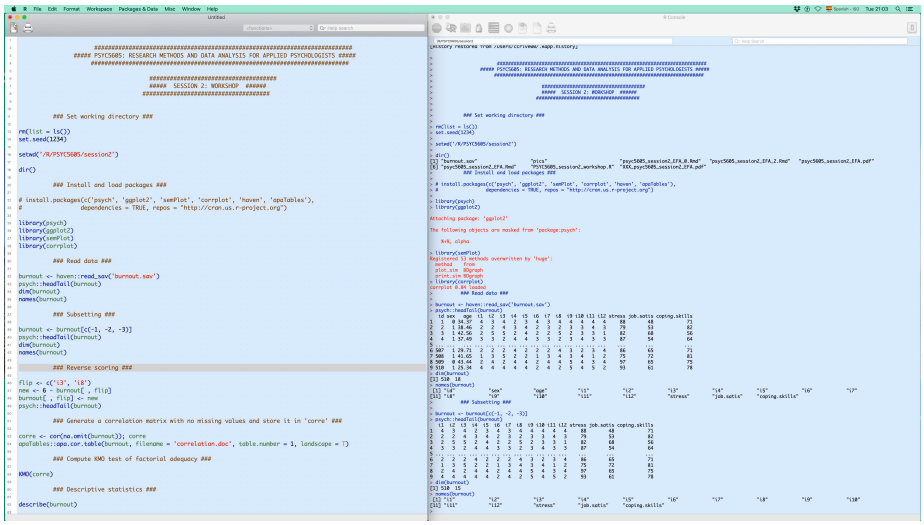


Figure 2.1: R displayed in two panels: Script (left) and Console (right).

It is dynamic. R is constantly evolving. It provides state-of-the-art solutions for any imaginable field (e.g., economics, psychology, genetics, medicine, ecology, sociology, computational physics, geography, psychometrics)

R has the biggest global community providing all sorts of free resources to support learners (e.g., reference manuals, books, on-line tutorials, R code and notebooks, cheat sheets, Internet forums)

R is more efficient than any other commercial software

UK universities teaching 21st century data science with R: University of Glasgow, Lancaster University, Nottingham Trent University, University of Edinburgh, University of Lincoln, University of Manchester, University of Plymouth, University of Sussex, Queen Mary University of London, University of Sheffield, University of Essex, Oxford University, King's College, University of Warwick...

Employability

2.1 What is RStudio?

RStudio is an Integrated Development Environment (IDE) for R. RStudio facilitates the task of coding by providing an enhanced programming experience. First, RStudio is free and easy to learn. Second, RStudio's elements are displayed in a four-panel display: the *code editor* (the script), the *console* (the results and debugging), the *global environment* (e.g., R objects, history of our session), and a *notebook* that includes different active tabs (e.g., plots, packages, help) (Figure 2.2). Last, there is an optimal integration of the different panels

and tabs to run R sessions and projects smoothly.

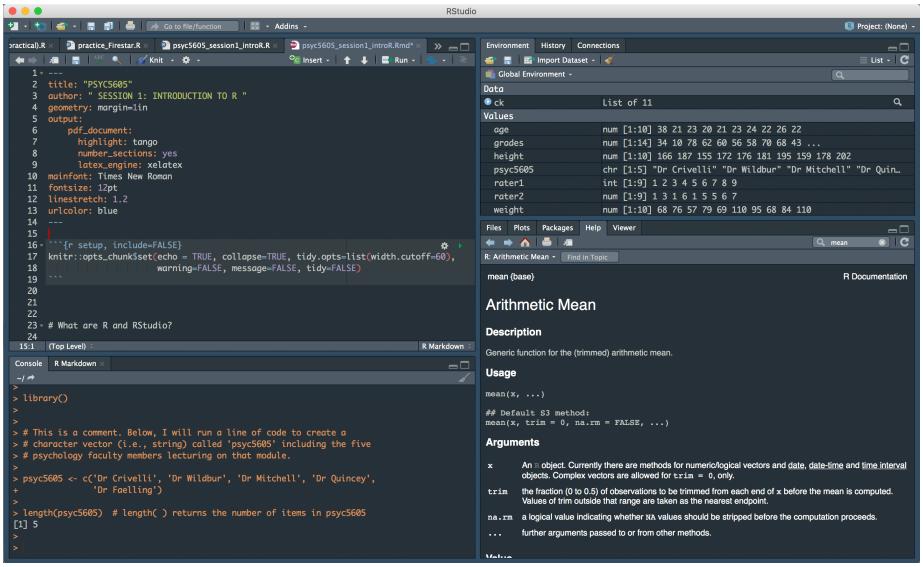


Figure 2.2: RStudio's four-panel display.

Why Use RStudio?

RStudio's main components are integrated into a four-pane layout: the *code editor* (the script), the *console* (the results and debugging), the *global environment* (the R objects, history of our session), and a *notebook* to hold tabs for the files/plots/packages/help/viewer

The code editor is feature-rich and integrated with the built-in console

The code editor and console are efficiently linked to the files/plots/packages/help/viewer panels

RStudio is available for Windows, MacOS, and Linux

RStudio is easy to learn

For more details on what is RStudio, click here

2.2 Let's get started!

2.2.1 Running a line of code

There are several ways to run a line of code:

- Highlight the line of code and click on the button **Run** (⇒ `Run`) displayed on the upper right side of the *code editor*.

- Highlight the line of code that you want to run and press **Ctrl + Enter** (Windows) or **Cmd + Enter** (MacOS).
- Place the cursor on the line of code and click on the button **Run** (⇒ `Run`).
- Press **Ctrl + Shift + S** (Windows) or **Cmd + Shift + S** (MacOS) to run the code of an entire script.
- Type a line of code directly into the *console* or copy a line of code from the *code editor* and paste it into the *console* after the prompt (`>`). Then, press **Enter** on the *console*.

2.2.2 Adding comments on R scripts

It is useful to add comments before chunks of R code to organize the session and to inform others that might be interested in reproducing our session and evaluating our outputs. The hash mark (`#`) is used to make comments by placing the symbol at the beginning of a line of code. R won't execute the line of code that is preceded by one or more hash mark symbols.

```
# This is a comment
# Raise 5 to the power of 2

5 ^ 2
## [1] 25

sqrt(25)  # sqrt(x) returns the square root of x
## [1] 5
```

2.3 Starting a session in RStudio

2.3.1 Setting the global options

It is possible to customize RStudio (e.g., code, appearance, organization of panels) to adapt it to our needs and preferences. To do so, click on the main **RStudio menu bar** → **Tools** → **Global Options**. Although there are many available options to customize RStudio sessions, we will mention below some of the most interesting features to inspect in the *global options* settings:

- **General**: It is convenient to start a new R session to avoid carrying over variable corruptions. Consequently, do not restore or save .RData files into the workspace at startup.
- **Code-Display options**: To enhance the usability experience, enable *soft-wrap R source files*, *highlight selected word*, *show line numbers*, and *blinking cursor*. In the option *show margin*, set the margin column to 60.
- **Appearance**: The customization of the *editor* and *console*'s appearance

(e.g., font size) is pivotal as we will spend a lot of time looking at the computer screen when programming. For example, the use of dark background colors (e.g., material, chrome) is very popular among programmers because it minimizes eye strain or fatigue. If you are a low vision user, the high contrast generated by a dark background and white text will be beneficial as well.

- **Pane Layout**: It is possible to reorganize the location of the panels. Likewise, it is possible to show or hide the different tabs available in the *global environment* and *notebook*. For instance, we could arrange the *source* on the upper left-hand side, inspecting the output of the code that we run underneath or on the right hand side (the *console* would be located on the bottom left corner or on the upper right corner respectively).

2.3.2 First lines of code

Every session will start with three lines of code that we will include at the beginning of our R script. First, we will use the function `rm()` (i.e., remove) to delete any R object stored in the *global environment*. We want to start each session anew, without carrying over R objects from previous sessions. Second, we will set the randomization seed with the function `set.seed()` to reproduce the R objects and outputs of any simulated data and computation. Last, we will turn off the scientific notation used in R.

```
rm(list = ls())
set.seed(1234)
options(scipen=999)
```

2.3.3 Setting the working directory

There is a point-and-click approach to set the working directory for each R session. To do so, click on the main **RStudio menu bar** → **Session** → **Set Working Directory** → **Choose Directory**. Then, select the desired folder to store all the files of the session (e.g., the R file with the code, data sets).

When exporting plots and files, these files will be saved and stored locally in the working directory already set at the beginning of the session. For example, we created a folder called *R* located in the main SSD MacOS/C: hard drive that includes the directories of all our R projects. At a lower level, we created another folder called *psyc3012* (Data Science with R) to store the current R project that, in turn, includes one folder per session (e.g., *session1*, *session2*, *session3*).

Setting the path for the working directory

The path should be short

Create a folder called *R* in your PC (C:) or MacOS (HD/SSD) hard drives. Then, create folders for different projects in the *R* folder

Use short names with no spaces (e.g., do not use a folder name such as *session 1*, but *session1* or *session_1*)

R is case sensitive (*psyc3012* differs from *PSYC3012*)

Use single ('*x*') or double quotes ("*x*") because the path is a character string

The function `getwd()` finds the path to the folder of our current working directory, whereas the function `setwd()` sets the working directory's path.

```
getwd()
setwd('~/R/psyc3012/session1')
```

2.4 R objects and functions

There are different kinds of entities that we identify as objects. Here, we will distinguish between R objects of different data structures (e.g., matrices, vectors, data frames, lists) and "recursive" objects such as those of mode *function* (Lander, 2017; Venables et al., 2009). We will refer to the latter as functions and to the former as R objects. In sum, we will use functions (e.g., `mean()`, `boxplot()`) to execute commands on R objects (e.g., atomic vectors, data frames, matrices).

2.4.1 Functions

Let's imagine that we would like to command a robot to do something for us. We simply need to learn the robot's language (i.e., rules of grammar, syntax, vocabulary, code), so it can decode our commands and execute our orders. Any programming language will work in a similar way. For instance, we will need to read data from files (e.g., text file, CSV file, SPSS file) or to simulate our own data. These data should be appropriately stored and saved into R's *global environment* to be able to manipulate and organize our data (e.g., merge, subset, select, rename), to model and visualize the data, and to publish the results. To complete those tasks, we must use appropriate functions on data structures of the same mode.

2.4.1.1 The `c()` function

The `c()` function is usually referred as the concatenate or combine function. It combines two or more elements of the same mode (e.g., numerical values, character strings, logical values) to create vectors.

```
c(16, 71, 172, 53, 77)
## [1]  16  71 172  53  77
```

```
c('Hylobates', 'Pongo', 'Gorilla', 'Pan', 'Homo')
## [1] "Hylobates" "Pongo"     "Gorilla"   "Pan"       "Homo"

c(FALSE, TRUE, TRUE, TRUE, TRUE)
## [1] FALSE  TRUE  TRUE  TRUE  TRUE
```

2.4.2 R objects

R objects such as character strings and numeric vectors, data frames, matrices or lists are saved and stored in R's *global environment* when we assign them to a name using the assignment operator (`name` ← `data`). The *console* shows the object only if it is recalled. To produce the assignment operator in Windows/Linux operating systems press **Alt** + **-**. Alternatively, press **Option** + **-** if using MacOS.

As a computing programming language, R executes our commands to produce certain outcomes. Sometimes those outcomes will appear on the *console* (e.g., estimating the arithmetic mean on a numerical vector). Other times, however, these commands won't produce effects on the *console*. For example, when we create a vector, it will only appear in the *global environment* tab unless it will be recalled in the same line of code preceded by a semicolon (;) or plaxed on a line of code below.

```
weight <- c(16, 71, 172, 53, 77); weight
## [1]  16  71 172  53  77

apes <- c('Hylobates', 'Pongo', 'Gorilla', 'Pan', 'Homo')
apes
## [1] "Hylobates" "Pongo"     "Gorilla"   "Pan"       "Homo"
```

If we don't assign a set of values to any given name to create an R object, these values won't be saved into the *global environment*. Therefore, we won't be able to recall the object for future computations.

```
c(16, 71, 172, 53, 77)
## [1]  16  71 172  53  77
```

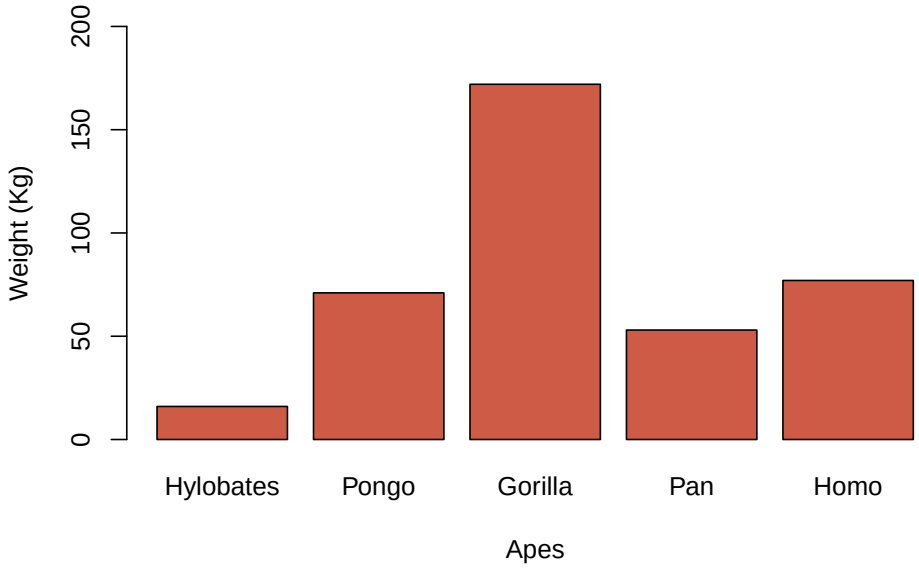
Instead of creating an R object first, we could be tempted to paste a vector into a function. This approach is not efficient—specially with large data structures—and it should be avoided. Given the outstanding possibilities that R offers, we should create R objects and use functions on them to model and visualize the data.

```
mean(c(16, 71, 172, 53, 77))
## [1] 77.8
```

```r
weight.apes <- c(16, 71, 172, 53, 77)

mean(weight.apes)
## [1] 77.8
mean(weight.apes, trim = .20)
## [1] 67
median(weight.apes)
## [1] 71
sd(weight.apes)
## [1] 57.78148
length(weight.apes)
## [1] 5

barplot(weight.apes, ylab = 'Weight (Kg)', xlab = 'Apes',
        col = 'coral3', ylim = c(0, 200),
        names.arg = c('Hylobates', 'Pongo', 'Gorilla', 'Pan', 'Homo'))
```



Naming objects

R is case sensitive (e.g., the R object *weight* differs from *Weight*)

Do not start with a number or special characters (e.g., +, -, &, *, ?)

Using lower (e.g., *weight.apes*, *x*, *m2spr34*) rather than an upper case letters (e.g., *Weight.Apes*, *X*, *M2spr34*) facilitates the programming flow

Do not use names of functions to name R objects (e.g., `data()`, `mean()`, `plot()`, `factor()`)

When using several words, do not leave spaces. Alternatively, use the full

stop, underscore, or a combination of UPPER and lower case letters (e.g., *g.spatial.males*, *g_fluid_females*, *gCrystallized*)

Use short, but informative names (e.g., *g.spatial.m*, *g_fluid_f*, *gCrystal*)

2.4.3 Applying functions to objects

Functions work on data stored in R objects and saved in the *global environment*. We will place the name of the R object within the parentheses of functions, executing the command of the function on R objects. Below, we will compute the arithmetic mean ($M = 52.1$) and estimate the length ($N = 10$) of one numeric vector named *age*.

```
age <- c(37, 19, 74, 62, 51, 56, 58, 76, 45, 43)
mean(age)
## [1] 52.1
length(age)
## [1] 10
```

Functions usually include arguments that allow us to modify their default settings. For example, the function `mean()` computes the arithmetic mean on the R object placed within parentheses. In the previous example, some outliers (e.g., 19, 76) might be distorting the estimation of the *age* distribution's central tendency. Luckily, the argument `trim` can be set from 0 (no trimming) to 0.5 (maximum trimming) to compute robust estimators of central tendency. By trimming 20% of the observations from each tail of the distribution (`trim = .20`), we will compute 20% trimmed means—a robust estimator of central tendency (Wilcox, 2016). The maximum trimming (`trim = .50`) computes the median as it only considers the central value of the distribution.

```
mean(age, trim = 0)
## [1] 52.1
mean(age, trim = .20)
## [1] 52.5
mean(age, trim = .50)
## [1] 53.5
median(age)
## [1] 53.5
```

2.4.4 Getting help

All R functions are included in packages that are created and maintained by their authors (e.g., statisticians, psychologists, biologists, sociologists). R packages are freely available on the Internet at the *Comprehensive R Archive Network* or CRAN. Every package has its own documentation that describes and explains how to use the functions of each package. We can also access to cheat sheets, online learning tools, forums and on-line communities, blogs, or even open-access

books created in R using **bookdown** (Xie, 2017; Xie, Allaire, & Grolemund, 2019).

RStudio includes the documentation of functions and packages in the *Help* tab. For example, if we search for the function `mean()`, we will find the documentation for this function included in R's package **base**. Another way to access to the documentation of packages and functions when help is required is to use the function `help()` or the question mark (`?`) followed by the name of the function.

```
help(mean)
?mean
```

2.5 Packages

A package is a collection of functions, data sets, and compiled R code for a given purpose (e.g., visualize data, perform specific data analytical techniques). First, we must install the packages on our computer. These packages will be downloaded and stored locally just once. Then, we will upload these packages into our R session when required. The function `.libPaths()` shows the directory in which our packages are installed.

```
.libPaths()
```

R comes with a standard set of packages (e.g., **base**, **stats**). However, most of the time we will use specific packages that include useful functions to organize, model, and visualize our data. The function `library()` is used to inspect the packages that we have previously installed on our computer.

```
library()
```

Currently (1st May 2023), there are 19,486 different R packages available on CRAN's (i.e., *the Comprehensive R Archive Network*) repository. These packages are grouped into 42 topics (e.g., psychometrics, graphics, genetics, social sciences, chemometrics and computational physics, econometrics, machine learning, finance, clinical trials, hydrology, medical image analysis, environmetrics, reproducible research).

Some packages' names are self-explanatory (e.g., **sem** for Structural Equation Models, **apaTables** for generating APA formatted Tables, **Kendall** for computing Kendall's nonparametric correlation tests), whereas other names include the abbreviation of specific tests, statistical techniques, or even research methodologies (e.g., **WRS2** for Wilcoxon's robust statistics, **lavaan** for latent variable analysis, **sna** for social network analysis).

To learn how to install R packages in your machine, please read carefully Appendix A.2. To inspect the list of R packages used on this book and that you will need to install to reproduce all the examples of the book, please read Appendix

A.3.

2.5.1 Loading R packages

Some basic packages (e.g., **base**, **stats**, **graphics**) are automatically uploaded when we start our session in R and RStudio. In contrast, most R packages will have to be uploaded at the beginning of our session in order to use them. The functions `library()` and `require()` load packages, using the name of the package as the first argument of the function. When the packages are successfully uploaded into our session, they are displayed with a ticked box (i.e., active) in the *packages* tab.

```
library(magrittr)
require(dplyr)
```

2.6 Importing and exporting data

The package **rio** enables us to import and export data sets from and into different formats (e.g., text, CSV, Excel, SPSS files). Traditionally, R users relied on functions such as `read.table()`, `read.csv()`, `read_excel()`, or `read_sav()` to import text files, csv comma-separated files, Excel files, or SPSS files. Similarly, the functions `write.table()`, `write.csv()`, `write_xlsx()`, and `write_sav()` were used to export data sets created and stored in the R session into an array of different formats. Unfortunately, to use most of these functions, R packages such as **haven**, **readxl**, or **writexl** were required.

Luckily, the package **rio** operates like a Swiss-army knife, allowing us to easily import and export data via generic functions whose arguments specify the type of file, the path, the name, etc.

2.6.1 Loading data

For example, we could read built-in data from the data sets loaded in our session using the function `data()`. From all the files available, we will read the data set `women` which includes the *height* and *weight* of 15 women.

```
data()
data(women)
women
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
```

```
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

Now, we will use the generic function `import()` from the package **rio** to read the `burnout` data set. The function `head()` shows the first six observations/participants of the R object included as the first argument of the function. In the following example, we are modifying the default settings that show the data of the first six observations/participants. We will request the first eight observations of the R object `burnout`.

```
my.burnout <- rio::import('./datasets/burnout.sav')
head(my.burnout, 8)
##   id sex age i1 i2 i3 i4 i5 i6 i7 i8 i9 i10 i11 i12 burnoutTot stress job.satis
## 1  1   0  34  4  3  4  2  3  4  3  4  4   4   4   4         34     88         48
## 2  2   0  38  2  2  4  3  4  2  3  2  3   3   4   3         28     79         53
## 3  3   1  43  2  5  5  2  4  2  2  5  2   3   3   1         22     82         68
## 4  4   1  37  3  3  2  4  4  3  3  2  3   4   3   3         33     89         54
## 5  5   0  43  2  2  4  3  3  1  3  4  3   2   4   2         23     76         51
## 6  6   1  35  3  3  4  3  3  3  3  3  3   3   4   2         29     83         47
## 7  7   1  31  2  4  1  4  4  4  4  2  4   2   5   3         37     86         59
## 8  8   1  25  3  4  1  4  3  2  4  2  4   5   3   4         38     84         50
##   coping.skills contract
## 1            71        2
## 2            82        2
## 3            56        3
## 4            64        1
## 5            73        3
## 6            74        3
## 7            75        1
## 8            80        2
```

CAUTION!

The data set that we want to import should be stored in the same folder of our working directory. If it is located in a different folder, the path should be specified in the code. For example, if the data set is in a folder named `data` inside of our working directory, the code should be as follows:

```
my.burnout <- rio::import('./data/burnout.sav')
```

IBM SPSS stores data frames into `.sav` files. Consequently, the file extension of the data set that we want to import needs to be explicitly included in the name of the file. To read CSV comma-delimited files, we will have to include the extension `.csv`. For Microsoft Excel files (i.e., spreadsheets) storing arrays of data sets using different sheets, we will use the file extension `.xlsx`.

Using two colons (::) to load specific functions

We can use functions from packages that we have not previously loaded by adding the name of the package followed by two colons before the specific function we intend to use [e.g., `rio::import()`, `psych::desbribeBy()`].

2.6.2 Exporting data

To export data from our R session, enter the R object (e.g., a data frame) as the first argument and the appropriate file extension as the second argument (e.g., `format = 'txt'`). Below, we will export the data set called `my.burnout` as an Excel file (i.e., `format = 'xlsx'`).

```
rio::export(my.burnout, format = 'xlsx')
```

2.7 References

Gandrud, C. (2015). *Reproducible Research with R and RStudio* (2nd ed.). Chapman and Hall/CRC.

Lander, J. P. (2017). *R for everyone* (2nd ed.). Addison-Wesley.

R Core Team (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing.

Venables, W. N., Smith, D. M., & the R Development Core Team. (2009). *An Introduction to R*. Network Theory Limited.

Wickham, H., & Grolemund, G. (2017). *R for data science: Import, tidy, transform, visualize, and model data*. O'Reilly.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., & Yutani, H. (2019). Welcome to the Tidyverse. *Journal of Open Source Software*, *4*(43), 1686. Retrieved at https://joss.theoj.org/papers/10.21105/joss.01686

Wickham, H., François, R., Henry, L., & Müller, K. (2022). *dplyr: A grammar of data manipulation*. Retrieved at https://dplyr.tidyverse.org

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Chapman and Hall/CRC.

Xie, Y., Allaire, J. J., & Grolemund, G. (2019). *R Markdown: The definite guide*. Chapman and Hall/CRC.

Chapter 3

Data Wrangling

You can add parts to organize one or more book chapters together. Parts can be inserted at the top of an .Rmd file, before the first-level chapter heading in that same file.

Add a numbered part: `# (PART) Act one {-}` (followed by `# A chapter`)

Add an unnumbered part: `# (PART\*) Act one {-}` (followed by `# A chapter`)

Add an appendix as a special kind of un-numbered part: `# (APPENDIX) Other stuff {-}` (followed by `# A chapter`). Chapters in an appendix are prepended with letters instead of numbers.

Important info

Open Science is a broad term used to encompass the promotion of *transparency*, *reproducibility*, *research integrity*, and *societal impact (research and innovation)*.

Because of those principles, **Open Science** can be reconceived as a meta-scientific movement interested in enhancing the following skills:

These blocks can be used inside of Tabs. The code block can also be used in Expandables, but cannot have other blocks inside it. The quote block and infobox can have headings, inline content and lists inside it.

Chapter 4

Descriptive Statistics and Data Visualization

4.1 Footnotes

Footnotes are put inside the square brackets after a caret `^[]`. Like this one [1].

4.2 Citations

Reference items in your bibliography file(s) using `@key`.

For example, we are using the **bookdown** package [Xie, 2023] (check out the last code chunk in index.Rmd to see how this citation key was added) in this sample book, which was built on top of R Markdown and **knitr** [Xie, 2015] (this citation was added manually in an external file book.bib). Note that the `.bib` files need to be listed in the index.Rmd with the YAML `bibliography` key.

The RStudio Visual Markdown Editor can also make it easier to insert citations: https://rstudio.github.io/visual-markdown-editing/#/citations

[1] This is a footnote.

Chapter 5

Blocks

5.1 Equations

Here is an equation.

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \tag{5.1}$$

You may refer to using `\@ref(eq:binom)`, like see Equation (5.1).

5.2 Theorems and proofs

Labeled theorems can be referenced in text using `\@ref(thm:tri)`, for example, check out this smart theorem 5.1.

Theorem 5.1. *For a right triangle, if c denotes the* length *of the hypotenuse and a and b denote the lengths of the* ***other*** *two sides, we have*

$$a^2 + b^2 = c^2$$

Read more here https://bookdown.org/yihui/bookdown/markdown-extensions-by-bookdown.html.

5.3 Callout blocks

The R Markdown Cookbook provides more help on how to use custom blocks to design your own callouts: https://bookdown.org/yihui/rmarkdown-cookbook/custom-blocks.html

Chapter 6

Sharing your book

6.1 Publishing

HTML books can be published online, see: https://bookdown.org/yihui/bookdown/publishing.html

6.2 404 pages

By default, users will be directed to a 404 page if they try to access a webpage that cannot be found. If you'd like to customize your 404 page instead of using the default, you may add either a `_404.Rmd` or `_404.md` file to your project root and use code and/or Markdown syntax.

6.3 Metadata for sharing

Bookdown HTML books will provide HTML metadata for social sharing on platforms like Twitter, Facebook, and LinkedIn, using information you provide in the `index.Rmd` YAML. To setup, set the `url` for your book and the path to your `cover-image` file. Your book's `title` and `description` are also used.

This `gitbook` uses the same social sharing data across all chapters in your book- all links shared will look the same.

Specify your book's source repository on GitHub using the `edit` key under the configuration options in the `_output.yml` file, which allows users to suggest an edit by linking to a chapter's source file.

Read more about the features of this output format here:

https://pkgs.rstudio.com/bookdown/reference/gitbook.html

Or use:

```
?bookdown::gitbook
```

Appendix A

Software

A.1 Installation of R and RStudio

kkfdslf kdlfkdslkf

A.2 Installation of R packages

Although there are different ways to install packages, we will describe two of them below: using RStudio's GUI and running a line of code.

- **RStudio's GUI**: First, we must click on the button **Install** in the *package* tab. Then, we need to enter the name of the package in the CRAN repository to install it locally on the library directory. It is important to enable the option **install dependencies** because many packages depend on the installation of others to be correctly loaded and work properly.

- **Running a line of code**: This is the recommended option. It can be done directly on the console. The function `install.packages()` uses the first argument of the function to install the designated package. The name of the package must be written in quotes and the argument *dependencies* should be set to *TRUE* (or *T*). We can combine several packages with the function `c()` and install them at the same time.

```
install.packages('binom', dependencies = TRUE)

install.packages(c('tidyverse', 'car', 'lavaan', 'rio', 'magrittr', 'psych',
                   'likert'), dependencies = TRUE)
```

Some packages include a lot of functions and/or require the installation of several dependencies. If this is the case, expect a long installation process. Always check that the prompt (>) symbol is displayed on the console as a confirmation that the installation process has ended.

The information provided on the console during the installation process informs you of potential problems that were encountered when installing the target packages and their dependencies. For example, you may end up experiencing a very common problem when reading the following warning message on the console:

```
Warning in install.packages : installation of package
'ZZZ' had non-zero exit status
```

The above warning message informs about the need to install or to update (i.e., reinstall) the **ZZZ** package because R couldn't install it. Restart your R session and install the latest version of the package. It is convenient to update from time to time the packages previously installed. R is a dynamic, collaborative, and open-source computing environment. Thus, R is constantly evolving while being updated by the developers of the packages. The function `update.packages()` will update all the packages already installed on our computer.

```
update.packages()
```

A.3 Packages used in this book

fewkfklew kewlfklew fewkfklew kewlfklew

Appendix B

Stylistic Conventions

kjgkjdfg kgldskglkg sdklgdslg kldsglj

Appendix C

Universal Design for Learning

fsdfdsfds fdsfsdf fdsfdsfs

Bibliography

Yihui Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. URL http://yihui.org/knitr/. ISBN 978-1498716963.

Yihui Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*, 2023. URL https://CRAN.R-project.org/package=bookdown. R package version 0.33.