Sagol School of Neuroscience
Tel Aviv University

Creating FAIR R code

Introduction: What is FAIR?

FAIR is an acronym describing how data, code, and analysis should ideally be managed and presented:

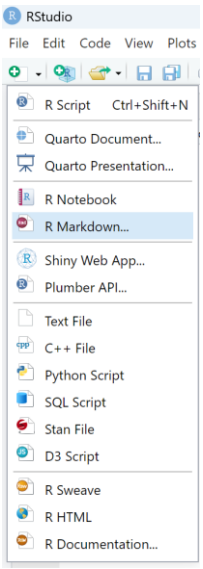
- **Findable**: Easy to locate, well-labeled, and clearly documented.
- **Accessible**: Available and easily retrievable by others.
- **Interoperable**: Usable across various platforms and software.
- **Reusable**: Clear enough for others (and your future self) to understand and apply to other situations.

Being FAIR with r code: R Markdown

The following guide assumes you have RStudio (a widely used IDE for R code).

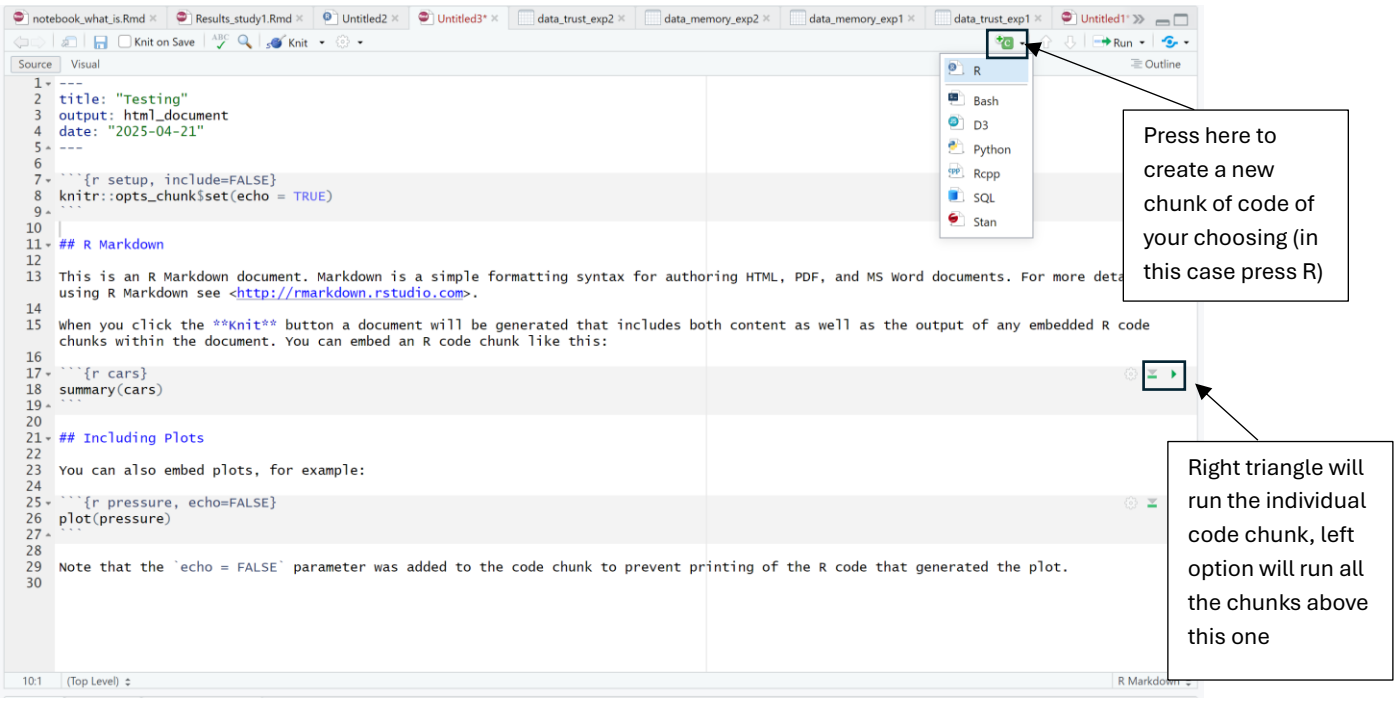
We will use R Markdown, which facilitates the organization of both code and statistical output in a manner that is transparent, reproducible, and accessible. This format is particularly useful for structuring code, as it allows for the execution of individual code chunks without rerunning the entire script. Additionally, R Markdown enables the conversion of analyses into well-formatted HTML or PDF documents, which is especially valuable for clearly presenting the steps taken during data analysis.

Start by opening your rStudio and clicking the + sign at the top left corner, then click on R Markdown:



1. Set Up Your R Markdown Document

Enter a clear and descriptive title for your R Markdown document (from now on, referred to as the *notebook*), along with the names of all authors associated with the work. Keep the default output format as HTML. Once these fields are completed, click "OK" to generate the notebook, which should appear as follows:



The top section of the notebook contains the metadata variables you entered in the previous step. Below this, the document is structured into multiple *chunks*. Gray chunks are used for executable code, while white chunks are designated for text and section headings. These white sections are used to clarify the purpose and functionality of each corresponding code chunk.

White (text) chunks: These sections are used to explain the purpose of various code chunks by providing titles and brief descriptions. Use the formatting symbols below to control how the content will appear in the final output.

Syntax	Output
#	Title (Level 1, Main section)
##	Subtitle (Level 2, sub-section)
###	Sub-subtitle (Level 3, smaller topics)

Gray (code) chunks: These sections begin and end with triple backticks (```)—these delimiters should not be altered. Immediately following the opening backticks, curly braces {} are used to specify chunk options, which control how the code and its output are rendered in the final HTML or PDF document. Below are several key options you may find useful.

Syntax	Description
echo	Show the code in the final document (TRUE or FALSE)
include	Include both code and output (FALSE hides both but runs the code)
warning	Show warnings generated by the code
fig.width	Width of plots (in inches)
fig.height	Height of plots (in inches)
fig.align	Plot alignment: 'left', 'center', 'right'
error	Whether to display errors in the document (TRUE or FALSE)
message	Show messages from the code (e.g., from library() calls)
tidy	Organize your code before displaying it (TRUE or FALSE)

2. Start writing code

Few important notes before you start writing code

- **Comment your code**: Describe the purpose of each chunk clearly as well as of complicated lines of code.
- **Use descriptive names**: Name variables and datasets meaningfully.
- **Consistent formatting**: Keep your style uniform.
- It is important to always **set a constant seed** (set.seed(123)) **for reproducibility**.

We begin with an example in which we read an Excel file and perform basic data manipulation. First, we load all the required libraries needed to execute the code. Each subsequent chunk is used for a specific task, and its purpose is explained using comments placed within or above the corresponding code.

````
---
title: "Testing"
output: html_document
date: "2025-04-21"
---

```{r warning=FALSE, message=FALSE, error=FALSE}
# Load necessary libraries
library(dplyr)
library(readxl)
library(lme4)
library(emmeans)
library(sjPlot)
library(ggplot2)
library(lme4)

```

## Data manipulation and cleaning
Here, we will briefly upload the data and make it suitable for analysis


```{r warning=FALSE}
#Loading the files
setwd("G:/My Drive/sync_new_comp/codes_for_lab")  #make sure to change the working directory to your own
data_experiment_1 <- read_excel("Experiment_1_data.xlsx")

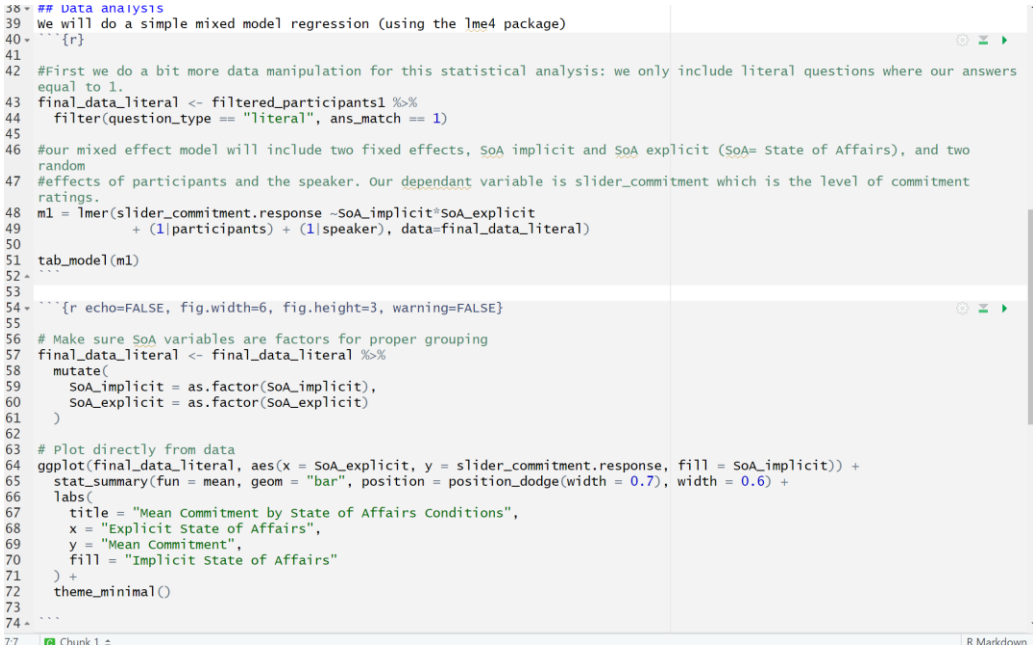
```

```{r warning=FALSE}
#filter based on incorrect responses (greater than 2 is excluded)
filtered_participants1 <- data_experiment_1 %>%
  group_by(participants) %>%
  filter(sum(is_corr_q1 == 0) <= 2) %>%
  ungroup()
```
````

3. Stats and graphs

- Clearly label tables and graphs.
- Use summaries (mean, median, standard deviation) appropriately.
- Recommended: Include concise interpretations immediately following results.

Let's do a short analysis, include some stats, and then in a separate chunk do a visualization:



```
## Data analysis
We will do a simple mixed model regression (using the lme4 package)
```{r}

#First we do a bit more data manipulation for this statistical analysis: we only include literal questions where our answers equal to 1.
final_data_literal <- filtered_participants1 %>%
  filter(question_type == "literal", ans_match == 1)

#our mixed effect model will include two fixed effects, SoA implicit and SoA explicit (SoA= State of Affairs), and two random
#effects of participants and the speaker. Our dependant variable is slider_commitment which is the level of commitment ratings.
m1 = lmer(slider_commitment.response ~SoA_implicit*SoA_explicit
          + (1|participants) + (1|speaker), data=final_data_literal)

tab_model(m1)
```

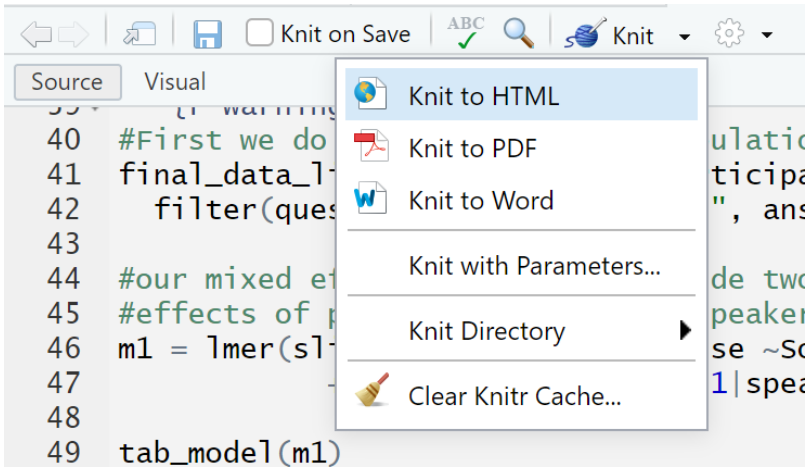
```{r echo=FALSE, fig.width=6, fig.height=3, warning=FALSE}

# Make sure SoA variables are factors for proper grouping
final_data_literal <- final_data_literal %>%
  mutate(
    SoA_implicit = as.factor(SoA_implicit),
    SoA_explicit = as.factor(SoA_explicit)
  )

# Plot directly from data
ggplot(final_data_literal, aes(x = SoA_explicit, y = slider_commitment.response, fill = SoA_implicit)) +
  stat_summary(fun = mean, geom = "bar", position = position_dodge(width = 0.7), width = 0.6) +
  labs(
    title = "Mean Commitment by State of Affairs Conditions",
    x = "Explicit State of Affairs",
    y = "Mean Commitment",
    fill = "Implicit State of Affairs"
  ) +
  theme_minimal()
```
Chunk 1
R Markdown

4. Create output file

To create your beautiful output file simply click on the Knit option and choose your preferred output format. Always add a section on recording the R version and all package versions with sessionInfo().


```