Introduction to R - Young Researchers Fellowship Program

Lecture 6 - RMarkdown and Quarto

Daniel Sánchez Pazmiño

Laboratorio de Investigación para el Desarrollo del Ecuador

October 2024

Document markup tools

Document markup tools

What are Markup Tools?

Markup tools allow you to write documents that combine text, code, and outputs in a reproducible way. These tools are essential in data analysis, allowing users to blend analysis with narrative explanations.

Popular Markup Tools

1. **RMarkdown** (`.Rmd`):
   - **Language**: R, with support for Python, Julia and more.
   - **Output**: HTML, PDF, Word, and slides.
   - **Uses**: Dynamic reports, reproducible research, dashboards, and presentations.
2. **Quarto** (`.qmd`):
   - The next generation RMarkdown. Developed by Posit in 2021-2022.
   - **Language**: Supports R, Python, Julia, and Observable.
   - **Output**: HTML, PDF, Word, slides, blogs, and books.
   - **Uses**: Scientific communication, data science documents, multi-language reports.

Popular Markup Tools

3. **Jupyter Notebooks** (`.ipynb`):
   - **Language**: Primarily Python, but also supports R, Julia, and other languages.
   - **Output**: Interactive notebooks, HTML, PDF.
   - **Uses**: Data exploration, machine learning models, and educational material.
4. **LaTeX** (`.tex`):
   - **Language**: LaTeX markup language for typesetting.
   - **Output**: High-quality PDF documents.
   - **Uses**: Academic papers, technical reports, and books with advanced mathematical notation.

Comparison

Tool	Main Code Support	Output Formats	Best For
RMarkdown	R, Python, Julia	HTML, PDF, Word	Reports, presentations
Quarto	R, Python, Julia, Observable	HTML, PDF, Reveal Slides	Multi-language reports, blogs
Jupyter Notebook	Python, R, Julia	HTML, PDF	Interactive data exploration
*LATEX**	LATEX, R	PDF	Complex typesetting, academia

Why Use Markup Tools?

- **Reproducibility**: Code and analysis are embedded in the document.
- **Automation**: Automatically updates outputs when code is re-run.
- **Communication**: Makes it easy to share analysis with both technical and non-technical audiences.
- **Flexibility**: Generate various output formats from a single source file.

Which Tool Should You Use?

- **RMarkdown/Quarto**: For dynamic, reproducible reports and presentations in R, Python, or other languages.
- **Jupyter**: Best for interactive analysis in Python or multi-step data exploration.
- **LaTeX**: Ideal for high-quality PDF reports with advanced math and styling.

What is RMarkdown?

Definition

RMarkdown is a file format for creating dynamic documents that combine text, code, and output in a single document.

- **File extension:** `.Rmd`
- **Components**:
    - Text (written in Markdown)
    - Executable code (R, Python, Julia, etc.)
    - Non executable code (virtually all languages)
    - Rendered outputs (plots, tables, etc.)

Why Use RMarkdown?

Key Benefits

- **Reproducibility**: Maintain all code, analysis, and outputs in one document.
- **Automation**: Code is run automatically every time the document is rendered.
- **Versatility**: Output to different formats like HTML, PDF, Word, and even slides.
- **Communication**: Ideal for blending narrative text and analysis results to communicate effectively with stakeholders.

Document Structure

Basic Structure of an RMarkdown File

1. YAML Header (Metadata)

The YAML header defines metadata like the title, author, and output format.

```
---
title: "My Analysis"
author: "Daniel Sanchez"
output: html_document
---
```

The YAML Header

RMarkdown can generate different types of output documents: - These are controllable options from the YAML header.

- **HTML**: Interactive reports, easily shared.
- **PDF**: High-quality documents for formal reports.
- **Word**: Common for business reporting and collaboration.
- **Slides**: Generate presentation slides (like these).

Changing the Output Format

To change the output format, modify the YAML header:

```
---
title: "My Report"
author: "Daniel Sanchez"
output: pdf_document
---
```

Basic Structure of an RMarkdown File

2. Markdown for Text

Use markdown syntax for formatting text:
Bold: **Bold** Italics: *Italics* Headers: #, ##, ###

Markdown Syntax

You can use standard markdown syntax for text formatting in RMarkdown documents:

- **Headers**: Use # for headers.
- **Lists**: Use - for bulleted lists, or `1.` for numbered lists.
- **Emphasis**: Use * or _ for italics, and ** or __ for bold.

Example:

```
# Header
This is a **bold** word and *italic* text.

- Item 1
- Item 2
```

Basic Structure of an RMarkdown File

3. Code Chunks

Insert executable code using chunks. The results of the code can be displayed in the output document.

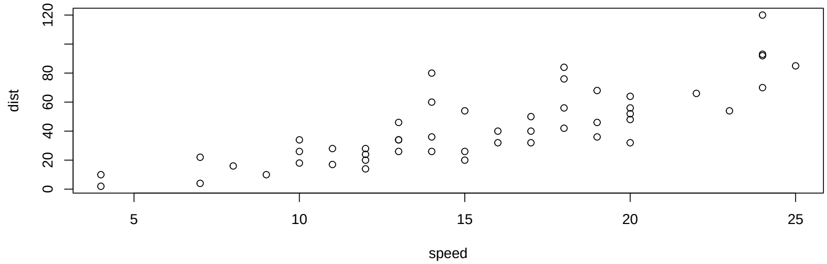
```
summary(cars)
```

```
     speed           dist       
 Min.   : 4.0   Min.   :  2.00  
 1st Qu.:12.0   1st Qu.: 26.00  
 Median :15.0   Median : 36.00  
 Mean   :15.4   Mean   : 42.98  
 3rd Qu.:19.0   3rd Qu.: 56.00  
 Max.   :25.0   Max.   :120.00  
```

Code Chunks

- Code chunks contain the code to be executed in the document. Basic syntax for an R chunk:

```
# Your code here
plot(cars)
```



Chunk Options

You can control how code chunks behave using options. Some common options include:

- **eval**: Controls whether the code is run (TRUE/FALSE).
- **echo**: Controls whether the code is shown in the output (TRUE/FALSE).
- **message**: Shows or hides messages from the code.

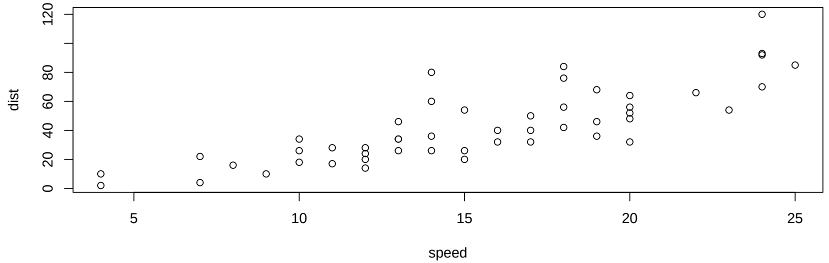
Example:

```
summary(cars)
```

Embedding Plots

Use code chunks to add plots directly into your RMarkdown document:

```
plot(cars)
```



The resulting plot will automatically appear in your output.

Plot Chunk Options

- `fig.width` and `fig.height`: Control the size of the plot in inches.
  - Example: `fig.width=7, fig.height=5`
- `fig.align`: Controls the alignment of the plot in the output.
  - Possible values: `'left'`, `'right'`, `'center'`
- `echo`: Controls whether the code is displayed in the final output.
  - `TRUE`: Show the code.
  - `FALSE`: Hide the code.

More Advanced Plot Chunk Options

- `fig.cap`: Adds a caption to the plot.
  - Example: `fig.cap="This is a plot of the cars dataset"`
- `out.width` and `out.height`: Control the size of the plot in the output document (e.g., when knitting to HTML or PDF). These values can be percentages or specific units (e.g., px).
  - Example: `out.width="80%", out.height="300px"`
- `fig.show`: Controls whether the plot is displayed or just the code is run.
  - `'hold'`: Only show the plot once all code in the chunk has run.
  - `'asis'`: Display plots as they are generated (default).
- `dpi`: Controls the resolution of the plot, useful for high-quality images in PDFs.
  - Example: `dpi=300`
- `dev`: Specifies the type of graphical device to use for rendering plots.
  - Example: `dev='png'`, `dev='pdf'`

Example: Customizing a Plot

Customizing Styles

For PDF output, RMarkdown lets you customize the document's appearance using LATEX commands or style templates.

For example, you can add custom LaTeX code in the YAML header for better control over formatting.

```
---
output:
  pdf_document:
    latex_engine: xelatex
    keep_tex: true
header-includes:
  - \usepackage{booktabs}
---x
```

How to Knit an RMarkdown Document

Knitting in RStudio

Knitting is the process of rendering your `.Rmd` file into a final output document such as HTML, PDF, or Word. It combines your text, code, and the results of that code into a unified document.

Steps to Knit

1. **Open RStudio** and load your `.Rmd` file.
2. **Click the “Knit” button** in the toolbar at the top of the editor.
3. **Select the output format** (HTML, PDF, Word, etc.), or use the default specified in the YAML header.
4. **View the output**: After knitting, the document will open in the selected format.

Knitting Shortcut

- Use the **keyboard shortcut**:
  - On **Windows/Linux**: `Ctrl + Shift + K`
  - On **Mac**: `Cmd + Shift + K`

Output Format

The output format is specified in the YAML header:

```
---
title: "My Report"
author: "Daniel Sanchez"
output: html_document
---
```

Using LATEX in RMarkdown

What is LATEX?

LATEX is a typesetting system widely used for writing mathematical formulas and equations. RMarkdown supports embedding LaTeX code to display high-quality mathematical expressions in your documents.

Inline Math

For inline math expressions, use single dollar signs $ to wrap the LaTeX code.

Example:

```
The equation of a line is given by $y = mx + b$.
```

Output:

The equation of a line is given by $y = mx + b$.

Common RMarkdown Issues

1. **Knit Errors**:
   - If code fails or packages are missing, knitting will stop. Check the **knit log** for errors.
   - Ensure your code runs without errors in RStudio before knitting.
2. **Working Directory**:
   - By default, RMarkdown runs from the **document's directory**.
   - Be careful with file paths—use relative paths for consistency.
   - Set a different working directory using:

```
knitr::opts_knit$set(root.dir = "your/path/here")
```

3. **Large Data or Plots**:
   - If the data or plots are too large, knitting to PDF or Word can fail.
   - Use `cache=TRUE` to avoid re-running heavy computations or limit the size of embedded plots using `fig.width` and `fig.height` options.
4. **Inline Code**:
   - When using inline code (e.g., `x`), ensure the object is defined beforehand in a code chunk.
   - Example:

Isolated Environments in RMarkdown

- **RMarkdown documents** are rendered in a clean, isolated environment.
- Each time you knit an RMarkdown file, RStudio creates a **new R session** where all code chunks are executed.
- This ensures **reproducibility**, meaning that the document can be knitted from scratch without relying on objects or variables from previous sessions.

Why Objects from the Global Environment Are Not Available

- Objects created in the **global environment** (outside of the RMarkdown file) are not automatically available in the knitting process.
- This prevents accidental dependencies on variables or objects that might exist in your workspace but are not defined within the document itself.

Quarto vs RMarkdown

What is Quarto?

- **Quarto** is a next-generation publishing system that supports multiple languages, including R, Python, Julia, and more.
- It allows users to create **HTML, PDF, Word documents, blogs, websites**, and even **slides**.
- Quarto extends the functionality of RMarkdown and is designed for **multi-language** projects, with improved syntax and output options.

Key Differences

Feature	**Quarto**	**RMarkdown Slides**
Language Support	R, Python, Julia, Observable	R (primary), some Python & SQL
Output Formats	HTML, PDF, Word, Slides, Websites	HTML, PDF, PowerPoint, Revealjs
Multi-language	Yes (seamless)	Limited (mostly R)
Customization	More flexible, supports custom themes, advanced options	Basic theming (more limited)
Rendering Engine	Quarto CLI (command-line)	Knitted in RStudio via Knit button

Why Choose Quarto?

1. **Multi-language Support**:
   - Quarto seamlessly integrates R, Python, Julia, and Observable, allowing you to use multiple languages within one document.
2. **More Output Flexibility**:
   - Quarto offers more control over output formats (websites, scientific papers, etc.) and additional output options like `PDF` and `docx` are enhanced.
3. **Advanced Customization**:
   - Quarto has more customization options for slides, including better control over styles, custom themes, and even LaTeX-like math typesetting across formats.