R Markdown

Reproducible Reports & Documentation

Code + Narrative = Professional Output

Reproducible
Reports

Automated
Updates

Multiple
Formats

Dr. Saad Laouadi

AI Expert & Data Science Educator

Independent Consultant

Version 1.1 | January 17, 2026

Professional R Development Series

Contents

1 Why R Markdown?

R Markdown combines code, results, and narrative in one document. Update data, re-run, output updates automatically.

1.1 The Problem

You analyze data in R. Copy results to Word. Make a plot. Screenshot. Paste.
Data updates? Start over. Every. Single. Time.
Result: Hours wasted. Errors introduced. Version chaos.

1.2 The Solution

Without R Markdown

- Manual copy-paste
- Screenshots of plots
- Version confusion
- Error-prone updates

With R Markdown

- Code generates output
- Plots auto-embedded
- One source of truth
- Click to update

1.3 What R Markdown Does

- **Combines** code and narrative
- **Executes** R code inline
- **Generates** reports in HTML, PDF, Word
- **Updates** automatically when data changes

2 Getting Started

2.1 Installation

```
install.packages("rmarkdown")
install.packages("knitr")
```

Install TinyTeX for PDF generation:
`install.packages("tinytex")`
`tinytex::install_tinytex()`

2.2 Create Your First Document

In RStudio:
File New File R Markdown
Choose:

- Title and author
- Output format (HTML, PDF, Word)

RStudio creates template. Click **Knit** to render.

2.3 Basic Structure

````
---
title: "My Report"
author: "Your Name"
date: "2025-01-15"
output: html_document
---

# Introduction

This is text.

```{r}
# This is R code
summary(mtcars)
```

More text here.
````

1. **YAML header** (between `---`)
2. **Markdown text** (narrative)
3. **Code chunks** (R code in `` ```{r} ``)

3 Markdown Basics

3.1 Text Formatting

Syntax	Output
`*italic*`	*italic*
`**bold**`	**bold**
`` `code` ``	`code`
`# Header 1`	Large header
`## Header 2`	Medium header
`- item`	Bullet point
`1. item`	Numbered list

3.2 Headers

```
# Main Title
## Section
### Subsection
#### Sub-subsection
```

3.3 Lists

```
# Unordered list
- First item
- Second item
- Nested item


# Ordered list
1. First
2. Second
3. Third
```

3.4 Links and Images

```
# Links
[Link text](https://example.com)


# Images
![Caption](path/to/image.png)


# Images with size control
![](image.png){width=50%}


# Center images
![](plot.png){width=80% fig-align="center"}
```

Images are relative to the .Rmd file location.
Best practice: Create `images/` folder in project root.

4 Code Chunks

4.1 Basic Code Chunk

````
"‘{r}
x <- c(1, 2, 3, 4, 5)
mean(x)
"‘
````

Executes code and shows both code and output.

4.2 Chunk Options

Option	Effect
`echo=FALSE`	Hide code, show output
`eval=FALSE`	Show code, don't run
`include=FALSE`	Run code, hide everything
`message=FALSE`	Hide messages
`warning=FALSE`	Hide warnings
`fig.width=7`	Set figure width (inches)
`fig.height=5`	Set figure height (inches)
`fig.cap="Title"`	Add figure caption
`cache=TRUE`	Cache results

4.3 Common Patterns

Show Results Only

````
"‘{r echo=FALSE}
summary(data)
"‘
````

Hides code. Shows output. Clean for reports.

Plot with Caption

````
"‘{r echo=FALSE, fig.width=8, fig.height=5,
fig.cap="Sales by Region"}
ggplot(data, aes(x, y)) + geom_point()
"‘
````

Shows plot with caption. Hides code. Professional output.

Setup Chunk

```
"‘{r setup, include=FALSE}
library(dplyr)
library(ggplot2)
knitr::opts_chunk$set(
echo = FALSE,
warning = FALSE,
message = FALSE,
fig.width = 8,
fig.height = 5
)
"‘
```

Runs silently. Loads packages. Sets global defaults for all chunks.

5 Output Formats

5.1 HTML

```
--
output: html_document
--
```

Best for: Interactive reports, web publishing, quick previews

5.2 HTML Customization

```
--
output:
html_document:
theme: flatly
toc: true
toc_float: true
code_folding: hide
code_download: true
--
```

```
default, cerulean, journal, flatly, darkly, readable, spacelab, united, cosmo, lumen,
paper, sandstone, simplex, yeti
```

5.3 PDF

```
--
output: pdf_document
--
```

Best for: Print-ready reports, formal documents, archival

Install TinyTeX first: `tinytex::install_tinytex()`

5.4 Word

```
--
output:  word_document
--
```

Best for: Sharing with non-technical collaborators, editing needed

5.5 Multiple Formats

```
--
output:
html_document:  default
pdf_document:  default
word_document:  default
--
```

Click dropdown arrow next to "Knit" to choose format.

6 Tables

6.1 knitr::kable()

````
"‘{r}
library(knitr)
kable(head(mtcars),
caption = "First 6 rows",
digits = 2)
"‘
````

Clean, simple tables. Works in all output formats.

6.2 Advanced Tables

```
library(kableExtra)


mtcars %>%
head() %>%
kable() %>%
kable_styling(
bootstrap_options = c("striped", "hover"),
full_width = FALSE
)
```

Professional formatting with kableExtra (HTML output).

6.3 Interactive Tables

````
library(DT)


"‘{r}
datatable(mtcars,
filter = 'top',
````

```
options = list(pageLength = 10))
"`
```

Creates searchable, sortable, filterable tables. HTML output only.

6.4 Summary Statistics

```
"`{r}
summary_stats <- mtcars %>%
summarise(
Mean = mean(mpg),
SD = sd(mpg),
Min = min(mpg),
Max = max(mpg)
)

kable(summary_stats, digits = 2,
caption = "Summary Statistics")
"`
```

7 Inline Code

7.1 Insert Values in Text

```
The mean mpg is `r mean(mtcars$mpg)`.

We analyzed `r nrow(mtcars)` cars.

The correlation is `r round(cor(mtcars$mpg, mtcars$wt), 2)`.
```

Values update automatically when data changes.

7.2 Formatting Inline Output

```
"`{r include=FALSE}
avg_mpg <- round(mean(mtcars$mpg), 1)
total_cars <- nrow(mtcars)
best_mpg <- max(mtcars$mpg)
"`

The average fuel economy is `r avg_mpg` mpg across
`r total_cars` vehicles, with the best achieving `r best_mpg` mpg.
```

Calculate first, reference later. Keeps text readable.

8 Parameters

8.1 What Are Parameters?

Parameters make reports dynamic. Change input values without editing code.

8.2 Define Parameters

```
--
title:  "Sales Report"
output:  html_document
params:
region:  "North"
year:  2024
threshold:  1000
--
```

8.3 Use Parameters

```
"‘{r}
# Access with params$
filtered_data <- sales %>%
filter(
region == params$region,
year == params$year,
amount > params$threshold
)
"‘


# Results for ‘r params$region‘ in ‘r params$year‘
```

8.4 Render with Different Parameters

```
# From R console:
rmarkdown::render(
"report.Rmd",
params = list(
region = "South",
year = 2025,
threshold = 2000
)
)
```

Perfect for automated reports!

9 Advanced Features

9.1 Child Documents

Break large reports into smaller, manageable pieces.

```
# Main report.Rmd
"‘{r child="intro.Rmd"}
"‘


"‘{r child="analysis.Rmd"}
"‘


"‘{r child="conclusions.Rmd"}
"‘
```

Modular reports. Reuse sections. Team collaboration.

9.2 Interactive Plots

```
library(plotly)

"‘{r}
p <- ggplot(mtcars, aes(wt, mpg)) +
geom_point() +
geom_smooth()

ggplotly(p)
"‘
```

Hover, zoom, pan. HTML output only.

9.3 Tabbed Sections

```
## Results {.tabset}

### By Region
Content for Region tab

### By Product
Content for Product tab

### By Time
Content for Time tab
```

HTML output gets interactive tabs.

9.4 Render from Command Line

```
# Render specific format
rmarkdown::render("report.Rmd",
output_format = "pdf_document")

# Render all formats
rmarkdown::render("report.Rmd",
output_format = "all")

# Specify output file
rmarkdown::render("report.Rmd",
output_file = "Q4_2024.html")
```

Automate report generation. Perfect for scheduled tasks.

10 Best Practices

Golden Rule

Write for humans first. Code comes second.

1. Setup Chunk First

````
"‘{r setup, include=FALSE}
# Load packages
library(tidyverse)
library(knitr)


# Set global chunk options
knitr::opts_chunk$set(
echo = FALSE,
warning = FALSE,
message = FALSE,
fig.width = 8,
fig.height = 5,
dpi = 300
)
"‘
````

Load packages once. Set global options. Reduces repetition.

2. Name Your Chunks

````
"‘{r load-data}
"‘
"‘{r clean-data}
"‘
"‘{r plot-results}
"‘
````

Makes debugging easier. Better error messages. Reusable chunks.

3. Hide Code by Default

```
knitr::opts_chunk$set(echo = FALSE)
```

Reports show results, not code. Override with `echo=TRUE` when needed.

4. Use Meaningful Headers

Good structure = easy navigation. Use TOC for long reports.

5. Cache Long Computations

````
"‘{r expensive-calc, cache=TRUE}
# Long-running code here
model <- train_large_model(data)
"‘
````

Runs once. Uses cached results next time. Speeds up development.

6. One Sentence Per Line

Makes version control easier. Git diffs work better.

7. Separate Data and Code

````
"‘{r}
# Read data, don't hardcode
data <- read_csv("data/sales.csv")
"‘
````

Keep data files separate. Makes updates easier.

11 Common Workflows

Analysis Report

```
---
title: "Data Analysis"
output:
html_document:
toc: true
code_folding: hide
---

"`{r setup, include=FALSE}
library(tidyverse)
knitr::opts_chunk$set(echo = FALSE)
"`

# Load Data
"`{r load-data}
data <- read_csv("data.csv")
"`

# Summary Statistics
"`{r summary-stats}
summary(data) %>% kable()
"`

# Visualization
"`{r viz, fig.cap="Relationship"}
ggplot(data, aes(x, y)) + geom_point()
"`
```

Parameterized Report

```
---
title: "Sales Report"
date: "`r Sys.Date()`"
output: html_document
params:
region: "North"
quarter: "Q1"
---

# `r params$quarter` Results - `r params$region`

"`{r}
sales %>%
filter(
region == params$region,
quarter == params$quarter
) %>%
summarise(Total = sum(amount)) %>%
kable()
"`
```

Change parameters. Re-render. Instant reports for all regions.

Technical Documentation

```
---
title: "API Documentation"
output:
html_document:
toc: true
toc_float: true
code_folding: show
theme: cosmo
---
```

Table of contents. Collapsible code. Professional theme.

12 Troubleshooting

Issue 1: Knit Fails

```
Error in eval(...)
```

Fix:

- Check for errors in code chunks
- Run chunks individually first (Ctrl+Shift+Enter)
- Make sure all packages loaded
- Check file paths (relative to .Rmd location)
- Restart R session and try again

Issue 2: PDF Won't Generate

Fix: Install TinyTeX

```
install.packages("tinytex")
tinytex::install_tinytex()


# Wait for installation, then:
tinytex::tlmgr_update()
```

Issue 3: Plot Doesn't Appear

Fix: Explicit print

```
"`{r}
p <- ggplot(data, aes(x, y)) + geom_point()
print(p)  # Add this!
"`
```

Required when creating plots inside loops or functions.

Issue 4: Slow Rendering

Fix: Use caching

```
"`{r cache=TRUE}
# Expensive computation
results <- complex_analysis(big_data)
"`
```

Results cached until code or data changes.

Issue 5: Object Not Found

```
Error:  object 'x' not found
```

Fix: Check chunk order. Objects created in earlier chunks only.

Issue 6: Package Not Loaded

Fix: Add to setup chunk

````
```{r setup, include=FALSE}
library(package_name)
```
````

13 Quick Reference

R Markdown Essentials

Document Structure

````
---
title:  "Report"
output:  html_document
---


# Header

Text here.


```{r}
code()
```
````

Key Chunk Options

- `echo=FALSE` - Hide code
- `include=FALSE` - Run silently
- `eval=FALSE` - Show code, don't run
- `fig.width=7` - Set figure size
- `fig.cap="Title"` - Add caption
- `cache=TRUE` - Cache results

Formats

- `html_document`
- `pdf_document`
- `word_document`
- `ioslides_presentation`

Tables

- `knitr::kable()`
- `kableExtra`
- `DT::datatable()`
- `gt` package

Inline Code

```
The mean is `r mean(x)`.
We have `r nrow(data)` observations.
```

Core Principle

Code + Narrative = Reproducible Reports

Write once. Update automatically. Share professionally.