TRABAJO PRACTICO TyHM I*

Combes Diego[1] and Cicchitti Luciano[2] Adriano Torresi[3]

[1] Universidad Nacional de Cuyo - Catedra de Tecnicas y herramientas modernas I
[2] Facultad de Ingeniería - Ciudad Universitaria Mendoza, Capital

Abstract. En el contexto académico y profesional, la escritura de informes extensos como tesis y trabajos finales de ingeniería requiere de herramientas eficientes que permitan una redacción organizada, flexible y fácil de gestionar. Markdown ha emergido como una solución altamente efectiva para la redacción de documentos estructurados gracias a su simplicidad y potencia. Además, su aplicación en programas como Obsidian mejora la gestión y organización de la información, facilitando el trabajo del estudiante e investigador.
Este informe explora la importancia de Markdown en la escritura de informes extensos y analiza sus ventajas cuando se emplea en aplicaciones como Obsidian

Keywords: Importancia · Eficiencia· Documentacion.

1 La Importancia de la Escritura de Informes Largos

En la redacción de informes extensos, como tesis o trabajos finales de ingeniería, es fundamental utilizar herramientas que permitan estructurar la información de manera clara, eficiente y flexible. Markdown, Obsidian y GitHub forman un ecosistema de herramientas que facilitan la organización, edición y colaboración en documentos largos, optimizando el flujo de trabajo del usuario.

A continuación, exploraremos cada una de estas tecnologías y sus beneficios en la escritura de informes técnicos y científicos.

1.1 Markdown: Un Lenguaje de Escritura Ligero y Versátil

¿Qué es Markdown? Markdown es un lenguaje de marcado ligero que permite formatear texto de manera sencilla utilizando una sintaxis mínima. Creado por John Gruber en 2004, su principal objetivo es hacer que la escritura y edición de documentos sean intuitivas, sin necesidad de utilizar procesadores de texto complejos.

¿Para qué se usa Markdown? Markdown se utiliza principalmente en la redacción de documentos extensos, documentación técnica, artículos científicos, páginas web y notas personales. Gracias a su compatibilidad con múltiples plataformas y su fácil conversión a formatos como PDF, HTML y LaTeX, se ha convertido en una herramienta clave para la escritura académica y técnica.

* Instituto de Ingeniería Industrial UNCuyo

Ventajas de Markdown en la Escritura de Informes Largos

- **Simplicidad y rapidez:** No requiere menús complejos; el formato se aplica con caracteres simples.
- **Compatibilidad con múltiples plataformas:** Puede utilizarse en editores de texto plano, aplicaciones especializadas y entornos web.
- **Conversión a diferentes formatos:** Facilita la exportación a PDF, Word, LaTeX y HTML sin pérdida de formato.
- **Separación entre contenido y diseño:** Permite centrarse en la escritura sin preocuparse por el formato visual, lo que agiliza la producción de documentos largos.
- **Facilidad de integración con otros sistemas:** Markdown es compatible con GitHub, Obsidian y gestores de contenido, lo que lo hace ideal para proyectos colaborativos.

1.2 Obsidian: Organización Eficiente de Notas y Documentos

¿Qué es Obsidian? Obsidian es un software de gestión de notas basado en archivos Markdown. Funciona como un sistema de organización de información en formato local, sin depender de servidores externos, y está diseñado para la creación de bases de conocimiento interconectadas.

¿Para qué se usa Obsidian? Obsidian se usa para la toma de notas, la organización de ideas y la redacción de informes extensos. Su sistema de enlaces internos y gráficos visuales lo convierte en una herramienta ideal para estructurar documentos complejos, como tesis o informes técnicos, facilitando la navegación y recuperación de información.

Ventajas de Usar Obsidian en la Escritura de Informes

- **Organización modular:** Permite dividir un informe en secciones interconectadas, facilitando la estructuración de documentos extensos.
- **Búsqueda rápida y eficiente:** Su potente motor de búsqueda interna permite localizar información en segundos.
- **Gráficos de conocimiento:** Visualiza la relación entre diferentes secciones del informe, lo que ayuda a estructurar mejor los contenidos.
- **Edición sin distracciones:** Su modo de texto plano evita distracciones y mejora la concentración en la escritura.
- **Historial y control de cambios:** Permite realizar un seguimiento de las modificaciones realizadas en el documento, lo que es útil en trabajos de investigación y desarrollo.

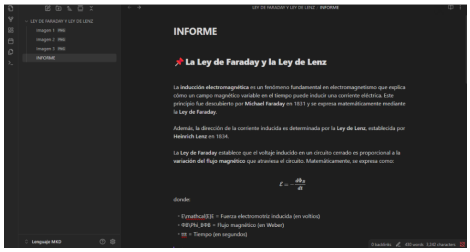


Fig. 1. Captura de pantalla de trabajo en Obsidian

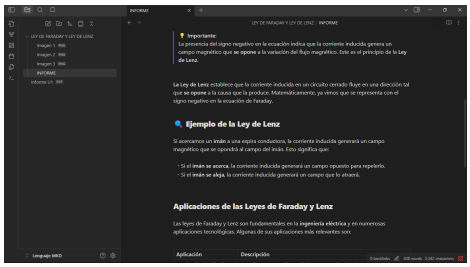


Fig. 2. Captura de Pantalla

1.3 GitHub: Control de Versiones y Colaboración en Proyectos de Escritura

¿Qué es GitHub? GitHub es una plataforma basada en Git que permite el control de versiones y la colaboración en proyectos de software y documentos. Aunque se asocia principalmente con la programación, también es una excelente herramienta para la gestión de documentos en proyectos de investigación y escritura técnica.

¿Para qué se usa GitHub en la Escritura de Informes? GitHub se utiliza para mantener un historial de versiones de documentos, colaborar en la redacción de informes largos y facilitar la integración con otras herramientas como LaTeX y Markdown. Es especialmente útil para equipos de trabajo que necesitan compartir, revisar y mejorar documentos en conjunto.

Ventajas de Usar GitHub en la Escritura de Informes

- **Control de versiones:** Guarda un historial completo de cambios en el documento, permitiendo revertir a versiones anteriores si es necesario.
- **Colaboración en equipo:** Varios autores pueden trabajar en un mismo documento sin riesgo de perder información.
- **Integración con Markdown y LaTeX:** Permite escribir documentos técnicos y científicos con herramientas especializadas.

- **Almacenamiento en la nube:** Mantiene una copia de seguridad de los archivos y permite acceder a ellos desde cualquier lugar.
- **Automatización de procesos:** Se pueden crear flujos de trabajo automatizados para generar versiones finales en diferentes formatos.

2 Conclusión

El uso de Markdown, Obsidian y GitHub en la escritura de informes largos proporciona una solución eficiente y flexible para la organización, edición y colaboración en documentos académicos y técnicos. Markdown simplifica la escritura y el formateo, Obsidian facilita la organización de información en estructuras interconectadas, y GitHub permite el control de versiones y la colaboración en equipo.

Para los estudiantes de ingeniería y profesionales que buscan optimizar su flujo de trabajo en la redacción de tesis, proyectos finales e informes técnicos, la combinación de estas herramientas representa una alternativa superior a los métodos tradicionales de escritura y edición de documentos.

R Markdown : : CHEAT SHEET

What is R Markdown?

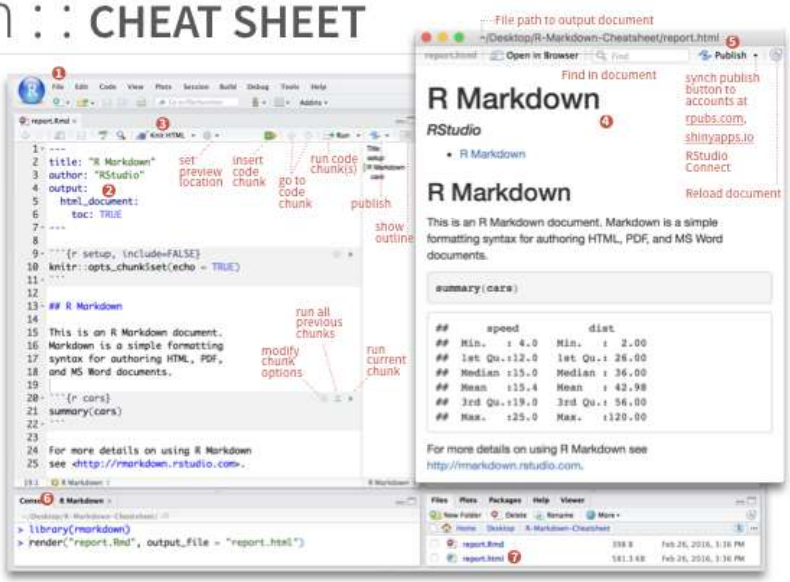
.Rmd files · An R Markdown (.Rmd) file is a record of your research. It contains the code that a scientist needs to reproduce your work along with the narration that a reader needs to understand your work.

Reproducible Research · At the click of a button, or the type of a command, you can rerun the code in an R Markdown file to reproduce your work and export the results as a finished report.

Dynamic Documents · You can choose to export the finished report in a variety of formats, including html, pdf, MS Word, or RTF documents; html or pdf based slides, Notebooks, and more.

Workflow

1. **Open a new .Rmd file** at File ▶ New File ▶ R Markdown. Use the wizard that opens to pre-populate the file with a template
2. **Write document** by editing template
3. **Knit document to create report**; use knit button or **render()** to knit
4. **Preview Output** in IDE window
5. **Publish** (optional) to web server
6. **Examine build log** in R Markdown console
7. **Use output file** that is saved along side .Rmd



.rmd Structure

YAML Header
Optional section of render (e.g. pandoc) options written as key:value pairs (YAML).

At start of file

Between lines of `---`

Text
Narration formatted with markdown, mixed with:

Code Chunks
Chunks of embedded code. Each chunk:

Begins with ```` ```{r} ````

ends with ```` ``` ````

R Markdown will run the code and append the results to the doc.

It will use the location of the .Rmd file as the **working directory**

Parameters

Parameterize your documents to reuse with different inputs (e.g., data, values, etc.)

1. **Add parameters** · Create and set parameters in the header as sub-values of params

```
---
params:
  n: 100
  d: !r Sys.Date()
---
```

2. **Call parameters** · Call parameter values in code as params$<name>

```
Today's date
is `r params$d`
```

3. **Set parameters** · Set values wth Knit with parameters or the params argument of render():

render("doc.Rmd", params = list(n = 1, d = as.Date("2015-01-01"))

render

Use rmarkdown::**render()** to render/knit at cmd line. Important args:

input - file to render **output_format**	**output_options** - List of render options (as in YAML)	**output_file** **output_dir**	**params** - list of params to use	**envir** - environment to evaluate code chunks in	**encoding** - of input file

Interactive Documents

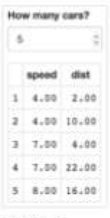
Turn your report into an interactive Shiny document in 4 steps

1. Add runtime: shiny to the YAML header.
2. Call Shiny input functions to embed input objects.
3. Call Shiny render functions to embed reactive output.
4. Render with rmarkdown::run or click Run Document in RStudio IDE

```
---
output: html_document
runtime: shiny
---

```{r, echo = FALSE}
numericInput("n",
  "How many cars?", 5)

renderTable({
  head(cars, input$n)
})
```
```



Embed a complete app into your document with shiny::**shinyAppDir()**

NOTE: *Your report will rendered as a Shiny app, which means you must choose an html output format, like* ***html_document****, and serve it with an active R Session.*

Embed code with knitr syntax

INLINE CODE

Insert with **`` `r <code>` ``**. Results appear as text without code.

Built with `` `r getRversion()` `` ➡ Built with 3.2.3

CODE CHUNKS

One or more lines surrounded with **```{r}** and **```**. Place chunk options within curly braces, after **r**. Insert with

```
```{r echo=TRUE}
getRversion()
```
```

GLOBAL OPTIONS

Set with knitr::**opts_chunk$set()**, e.g.

```
```{r include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```

IMPORTANT CHUNK OPTIONS

cache - cache results for future knits (default = FALSE)

cache.path - directory to save cached results in (default = "cache/")

child - file(s) to knit and then include (default = NULL)

collapse - collapse all output into single block (default = FALSE)

comment - prefix for each line of results (default = '##')

dependson - chunk dependencies for caching (default = NULL)

echo - Display code in output document (default = TRUE)

engine - code language used in chunk (default = 'R')

error - Display error messages in doc (TRUE) or stop render when errors occur (FALSE) (default = FALSE)

eval - Run code in chunk (default = TRUE)

fig.align - 'left', 'right', or 'center' (default = 'default')

fig.cap - figure caption as character string (default = NULL)

fig.height, fig.width - Dimensions of plots in inches

highlight - highlight source code (default = TRUE)

include - Include chunk in doc after running (default = TRUE)

message - display code messages in document (default = TRUE)

results (default = 'markup')
'asis' - passthrough results
'hide' - do not display results
'hold' - put all results below all code

tidy - tidy code for display (default = FALSE)

warning - display code warnings in document (default = TRUE)

Options not listed above: R.options, aniopts, autodep, background, cache.comments, cache.lazy, cache.rebuild, cache.vars, dev, dev.args, dpi, engine.opts, engine.path, fig.asp, fig.env, fig.ext, fig.keep, fig.lp, fig.path, fig.pos, fig.process, fig.retina, fig.scap, fig.show, fig.showtext, fig.subcap, interval, out.extra, out.height, out.width, prompt, purl, ref.label, render, size, split, tidy.opts