#YAML

title: “Clase 1 — RMarkdown” author: “Constanza Trujillo y José Ruiz-Tagle” date: “23 noviembre 2025” output: html_document: #Si es un documento HTML que tenga css: styles.css #forma de estilizar el informe toc: true #tabla de contenido toc_depth: 3 #sangría de títulos number_sections: true #número de las secciones code_folding: hide #que se muestren o se oculten los códigos df_print: paged # theme: simple #tema word_document:#Características cuando es un word toc: true toc_depth: ‘3’ pdf_document: #Características cuando es pdf toc: true number_sections: true fig_caption: true —

> **Objetivo:** entender **qué es R Markdown**, cómo funciona la combinación **texto + código**, qué es un **chunk**, cómo controlar su ejecución con **opciones** (`echo`, `eval`, `include`, `results`, `message`, `warning`, `fig.*`), y cómo **compilar (Knit)** a **HTML** y **PDF**.

¿Qué es R Markdown?

R Markdown es un formato que permite escribir documentos que mezclan **prosa** (Markdown) y **código ejecutable** (R, y otros).
Al “tejer” el documento (Knit), **knitr** ejecuta el código, **rmarkdown** arma el informe y lo exporta a **HTML**, **PDF** o **Word**.

- **Markdown** = sintaxis ligera para dar formato (títulos, listas, negritas, tablas).
- **Chunks** = bloques de código R delimitados por ```` ```{r} ```` y ```` ``` ````.
- **YAML** = cabecera al inicio entre `---` con **metadatos** (título, autor, fecha) y **formatos de salida**.

Anatomía de un `.Rmd`

Un archivo `.Rmd` típico tiene tres partes:

1. **YAML** (entre líneas `---`) con título/autor/fecha y `output:`.
2. **Prosa en Markdown** (texto con formato).
3. **Chunks de código** (ejecutables) y **código inline** (en línea) con `` `r` ``.

Chunks: crear, nombrar y ejecutar

En **RStudio**:

- Insertar chunk: **Ctrl/Cmd + Alt + I** o botón *Insert Chunk*.
- Estructura básica:

```
# tu código aquí
```

- Ejecutar un chunk: ícono "Run" (triángulo) en el borde del chunk o **Ctrl/Cmd + Shift + Enter** para ejecutar el documento de arriba hacia abajo.

Opciones de chunk más usadas (knitr)

Estas van en la cabecera `{r ...}` o como **globales** en el chunk `setup`.

- `echo = TRUE/FALSE` → ¿se muestra el **código** en el informe?
- `eval = TRUE/FALSE` → ¿se **ejecuta** el código?
- `include = TRUE/FALSE` → ¿incluye **código + output** en el informe (aunque se ejecute)?
- `message = TRUE/FALSE` y `warning = TRUE/FALSE` → mostrar/ocultar mensajes y advertencias.
- `results = 'hide'|'asis'` → controlar salida textual.
- `fig.width`, `fig.height`, `fig.align`, `fig.cap`, `out.width` → control de gráficos.

  Diferencias clave:
  - `echo=FALSE` oculta el **código**, pero muestra el **resultado**.
  - `eval=FALSE` muestra el **código** sin ejecutarlo (no hay resultado).
  - `include=FALSE` ejecuta pero **no** incluye nada en el informe (útil para cargas y setups).

Setup global de chunks

Se recomienda un chunk inicial **(setup)** con `include=FALSE` para fijar opciones globales:

DEMO: texto, inline code y listas

Markdown básico

(Deja **una línea en blanco** antes de listas como esta.)

- **Negrita** con `**así**` y *cursiva* con `*así*`.
- Lista:
  - item 1
  - item 2

Código inline: hoy es 23-11-2025 y `mtcars` tiene 32 filas. *El texto inline (o código inline) en R Markdown es código R corto que se evalúa dentro de una línea de texto y su resultado se inserta en el párrafo al compilar. Se escribe entre backticks con r al inicio.*

Ejemplos de chunks y opciones

mtcars es un dataset incorporado en R (viene "de fábrica"). Resume datos de 32 automóviles probados por la revista Motor Trend (1974), con 11 variables como consumo y desempeño.

Chunk básico

```r
1 + 1   # suma simple
```

```
## [1] 2
```

```r
mean(c(1, 2, 3, 10))  # media de una serie de números
```

```
## [1] 4
```

`echo=FALSE` (oculta código, muestra resultados)

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.40   15.43   19.20   20.09   22.80   33.90
```

`eval=FALSE` (muestra código, no ejecuta)

```r
plot(mtcars$wt, mtcars$mpg)  # plot(x,y): dispersión; X=mtcars$wt (peso, 1000 lbs), Y=mtcars$mpg (rendi
```

Gráfico con `fig.cap` y `out.width`

```r
plot(mtcars$wt, mtcars$mpg,                          # dispersión X=wt (peso 1000 lbs), Y=mpg (millas/galó
     pch = 19,                                       # tipo de punto: círculo sólido
     xlab = "Peso (1000 lbs)",                       # etiqueta eje X
     ylab = "Millas por galón")                      # etiqueta eje Y

abline(lm(mpg ~ wt, data = mtcars),                 # añade la recta de regresión mpg ~ wt
       lwd = 2)                                     # grosor de la línea (más visible)
```

`message` y `warning`

```r
library(dplyr)         # Carga el paquete dplyr para manipulación de datos (pipes, group_by, summarise,
mtcars %>%                 # Toma el data.frame mtcars y pásalo por la pipe (%>%) Ctrl + Shift + M
  group_by(cyl) %>%        # Agrupa las filas por número de cilindros (cyl: 4, 6, 8)
  summarise(               # Calcula resúmenes por cada grupo de 'cyl'
    mpg_prom = mean(mpg), # promedio de millas/galón dentro de cada grupo
    .groups = "drop")      # quita el atributo de agrupación en el resultado final
```

Tablas con `knitr::kable`

- head(iris): toma las primeras 6 filas del dataset iris (incluye Sepal/ Petal medidas y Species)
- knitr::kable(): crea una tabla "bonita" para el informe (funciona en HTML, PDF y Word)
- caption = "..." : pone un pie de tabla (se numera si el formato lo permite y usas fig/tab captions)

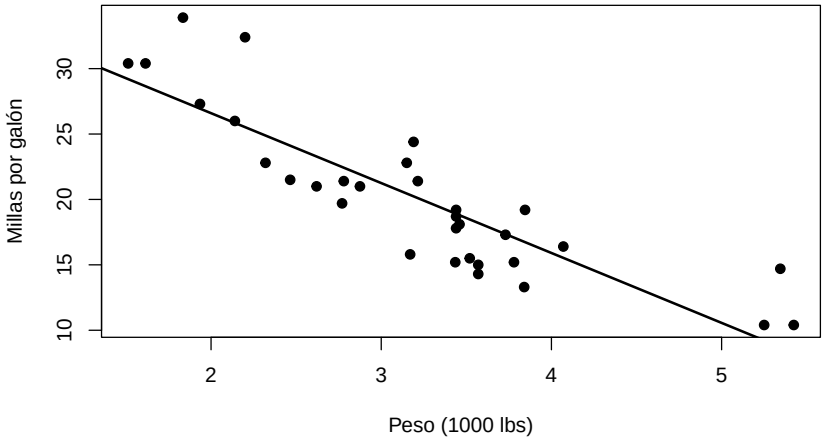

Figure 1: Relación peso vs rendimiento (mtcars)

- con kableExtra se pueden crear múltiples tipos de tablas, con colores, con imágenes, con gráficos, etc.

```
knitr::kable(
  head(iris), # datos a mostrar (primeras 6 filas)
  caption = "Primeras filas de iris") # título/pie de la tabla
```

Table 1: Primeras filas de iris

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

```
# --- Variantes útiles  -------------------------

# Alinear columnas: "l"=left, "c"=center, "r"=right (uno por columna)
 knitr::kable(head(iris), align = c("l","r","r","r","c"),
              caption = "Iris alineado")
```

Table 2: Iris alineado

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

```
# Renombrar encabezados de columnas:
 knitr::kable(head(iris),
              col.names = c("Largo Sépalo","Ancho Sépalo","Largo Pétalo","Ancho Pétalo","Especie"),
              caption = "Iris con encabezados en español")
```

Table 3: Iris con encabezados en español

Largo Sépalo	Ancho Sépalo	Largo Pétalo	Ancho Pétalo	Especie
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

```
# En HTML puedes combinar con kableExtra para estilo avanzado:
if (knitr::is_html_output()) {
   knitr::kable(head(iris), caption = "Iris con estilo") |>
     kableExtra::kable_styling(full_width = FALSE) #no queremos que su ancho sea de toda la pantalla
 }
```

```
library(dplyr)
library(kableExtra)
```

```
text_tbl <-data.frame(
 Items= c("Item1", "Item2", "Item3"),
 Features =c(
 "Loremipsumdolorsit amet,consectetur adipiscingelit.Proinvehicula temporex.Morbimalesuadasagittis",
 "In euurna atmagnaluctusrhoncus quis innisl.Fusceinvelitvarius, posuere risuset,cursusaugue.",
 "Vivamus venenatisegestas eros uttempus. Vivamus idest nisi.Aliquam molestie erat etsollicitudin"
 )
 )
 kbl(text_tbl,booktabs =T)%>%
 kable_styling(full_width= F)%>%
 column_spec(1,bold = T,color= "red") %>%
 column_spec(2,width= "30em")
```

```
that_cell<-c(rep(F,7),T)
 mtcars[1:8,1:8] %>%
 kbl(booktabs =T,linesep = "") %>%
 kable_paper(full_width= F)%>%
 column_spec(2,color=spec_color(mtcars$mpg[1:8]),
 link = "https://haozhu233.github.io/kableExtra") %>%
 column_spec(6,color= "white",
 background= spec_color(mtcars$drat[1:8],end =0.7),
```

Items	Features
Item1	Loremipsumdolorsit amet,consectetur adipiscingelit.Proinvehicula temporex.Morbimalesuadasagittis
Item2	In euurna atmagnaluctusrhoncus quis innisl.Fusceinvelitvarius, posuere risuset,cursusaugue.
Item3	Vivamus venenatisegestas eros uttempus. Vivamus idest nisi.Aliquam molestie erat etsollicitudin

	mpg	cyl	disp	hp	drat	wt	qsec	vs
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	~~1~~

```
popover = paste("am:",mtcars$am[1:8])) %>%
column_spec(9,strikeout=that_cell,bold = that_cell,
color= c(rep("black",7),"red"))
```

results='hide' y include=FALSE

```
# Se ejecuta pero no muestra resultados
x <- rnorm(1e5)       # Genera 100.000 (1e5) números aleatorios ~ N(0, 1) y los guarda en 'x'
                      #  rnorm(n) produce valores con media 0 y desviación estándar 1 por defecto.

media_x <- mean(x)    # Calcula la media (promedio) de esos 100.000 valores y la guarda en 'media_x'
                      #  Por la ley de los grandes números, este valor debería estar cerca de 0.
```

```
## [1] 0.0009767488
```

¿Cómo compilar (Knit)?

Desde RStudio

1. Guarda el `.Rmd`.
2. Usa la flecha del botón **Knit** para elegir **HTML** o **PDF**, y clic en **Knit**.

Con código (automatización)

```r
#install.packages("rmarkdown")
#install.packages("LaTeX")
# HTML
rmarkdown::render("clase_rmarkdown.Rmd", output_format = "html_document")
# PDF
rmarkdown::render("clase_rmarkdown.Rmd", output_format = "pdf_document")
```

Exportar a PDF (TinyTeX)

Para generar PDF necesitas LaTeX. La opción más simple es **tinytex**:

```r
install.packages("tinytex")
tinytex::install_tinytex()  # una sola vez
```

> Si ya tienes MiKTeX/TeX Live, no instales tinytex. Si faltan paquetes LaTeX: `tinytex::tlmgr_install("<paquete>")` o `tinytex::reinstall_tinytex()`.

HTML vs PDF vs Word

- **HTML**: liviano, interactivo (plegado de código, tablas paginadas), fácil de compartir.
- **PDF**: estable para impresión/entrega formal, requiere LaTeX.
- **Word**: editable por terceros.

Puedes definir múltiples salidas en `output:` y elegir desde la flecha de **Knit**.

Ejercicios

1. Crea un `.Rmd` con YAML mínimo (título/autor/fecha) y `output: html_document`.
2. Inserta **3 chunks**:
   - a) cálculo simple (p.ej., media de un vector),
   - b) un gráfico con `fig.cap`,
   - c) una tabla con `knitr::kable`.
3. Juega con `echo`, `eval`, `include`, `message`, `warning`, `results`.
4. Agrega **código inline** con length(unique(iris$Species)).
5. Activa `code_folding: show` y **compila a HTML**.
6. Instala **tinytex** y **compila a PDF**.

Problemas comunes

- **No compila a PDF** → instala `tinytex` (o verifica LaTeX), reinicia RStudio y vuelve a knit.
- **Paquetes faltantes** → `install.packages("<paquete>")` o `tinytex::tlmgr_install("<paquete LaTeX>")`.
- **Rutas con tildes/espacios** → evita caracteres raros en carpetas.
- **Demasiados mensajes/advertencias** → usa `message=FALSE, warning=FALSE` o limpia el ambiente y re-knit.
- **Resultados/gráficos desordenados** → ajusta `fig.width/height`, `out.width`, `fig.align`