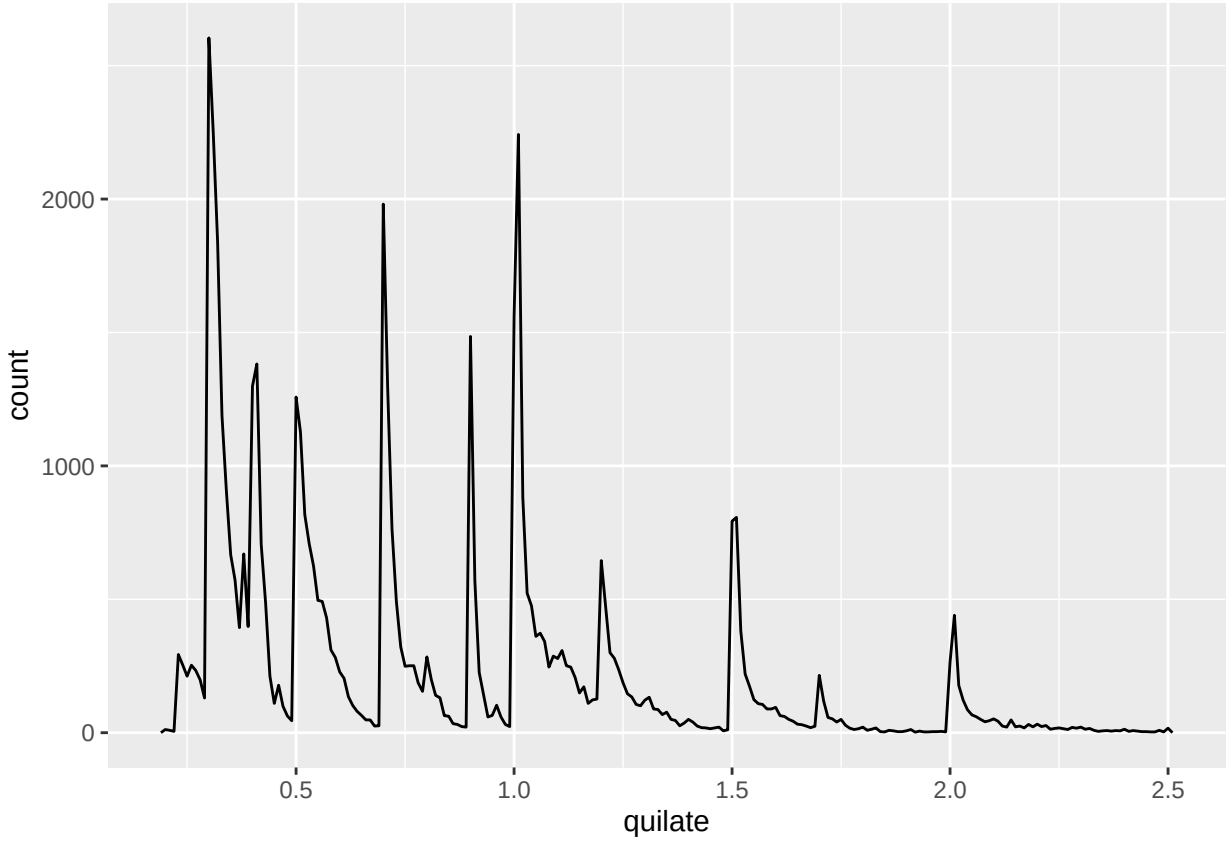
RMarkdown cap 27, parte 28, 29 y 30

Rmarkdown

https://github.com/cienciadedatos/r4ds/blob/traduccion/27-rmarkdown.Rmd

Tenemos datos respecto de 53940 diamantes. Únicamente 126 son mayores a 2,5 quilates.

La distribución de los diamantes pequeños se muestra a continuación:



Render: Para producir un reporte completo que contenga todo el texto, código y resultados, clic en "Knit" o presiona Ctrl + Shift + K. Puede hacerse también de manera programática con rmarkdown::render("1-example.Rmd").

Knit: Con las opciones de Knit se puede variar el tipo de salida (HTML, PDF, Word)

Cuando haces knit al documento, R Markdown envía el .Rmd a knitr que ejecuta todos los bloques de código y crea un nuevo documento markdown (.md) que incluye el código y su output. El archivo markdown generado por knitr es procesado entonces por pandoc que es el responsable de crear el archivo terminado.

Figure 1: Problemas carga Flujo Rmarkdown

Formato de texto

cursiva o *cursiva* **negrita negrita** `code` superíndice^2 y subíndice$_2$

Para provocar salto de linea basta con tipear 2 espacios

Encabezados

Encabezado de primer nivel

Encabezado de segundo nivel

Encabezado de tercer nivel

Listas

- Elemento 1 en lista no enumerada
- Elemento 2
  - Elemento 2a
  - Elemento 2b

1. Elemento 1 en lista enumerada
2. Elemento 2. La numeración se incrementa automáticamente en el output.

Enlaces e imágenes

si la imagen se encuentra al posicionarse encima, aparece la imagen como grisada

ojo!! que esta linea de abajo no funciona con salida pdf por eso esta puesta como chunk y el contenido comentario

```
#![](https://github.githubassets.com/images/modules/logos_page/GitHub-Mark.png){width='100px'}
```

Dentro de los corchetes, se puede escribir el texto alternativo. Este es opcional y solo entra en acción cuando no se puede cargar la imagen correctamente.

http://ejemplo.com

texto del enlace

faltaría agregar otra forma como se hace en HTML, con un bloque

Tablas

Primer encabezado	Segundo encabezado
Contenido de la celda	Contenido de la celda
Contenido de la celda	Contenido de la celda

Tabla ascii imagen

Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car	Dec	Hex	Car
32	20	espacio	48	30	0	64	40	@	80	50	P	96	60	`	112	70	p
33	21	!	49	31	1	65	41	A	81	51	Q	97	61	a	113	71	q
34	22	"	50	32	2	66	42	B	82	52	R	98	62	b	114	72	r
35	23	#	51	33	3	67	43	C	83	53	S	99	63	c	115	73	s
36	24	$	52	34	4	68	44	D	84	54	T	100	64	d	116	74	t
37	25	%	53	35	5	69	45	E	85	55	U	101	65	e	117	75	u
38	26	&	54	36	6	70	46	F	86	56	V	102	66	f	118	76	v
39	27	'	55	37	7	71	47	G	87	57	W	103	67	g	119	77	w
40	28	(	56	38	8	72	48	H	88	58	X	104	68	h	120	78	x
41	29	)	57	39	9	73	49	I	89	59	Y	105	69	i	121	79	y
42	2a	*	58	3a	:	74	4a	J	90	5a	Z	106	6a	j	122	7a	z
43	2b	+	59	3b	;	75	4b	K	91	5b	[	107	6b	k	123	7b	{
44	2c	,	60	3c	<	76	4c	L	92	5c	\	108	6c	l	124	7c	|
45	2d	-	61	3d	=	77	4d	M	93	5d	]	109	6d	m	125	7d	}
46	2e	.	62	3e	>	78	4e	N	94	5e	^	110	6e	n	126	7e	~
47	2f	/	63	3f	?	79	4f	O	95	5f	_	111	6f	o			

Tabla 2. Caracteres ASCII imprimibles

Bloques (Chunks)

Para insertar un chunk de código se puede hacer con alt+ctrl+i o bien con el símbolo insert cuadradito c en verde, y con la flecha se puede elegir lenguaje, o también manualmente tipeando `{r} y`

Dentro del chumk sepuede ejecutar como antes con ctrl + enter... pero también esta el atajo ctrl+shift+enter que ejecuta todo el chunk.

El encabezado de bloque, consiste en "'{r, seguido por un nombre opcional para el bloque, seguido luego por opciones separadas por comas y un }. Luego el código y al final ´´´ Hay un chunk especial llamado setup: cuando te encuentras en modo Notebook, el bloque llamado setup se ejecutará automáticamente una vez, antes de ejecutar cualquier otro código.

Opciones dentro de Bloques

Veremos las más importantes, el resto se puede encontrar en http://yihui.name/knitr/options/ Estos, controlan si tu bloque de código es ejecutado y qué resultados estarán insertos en el reporte final:

- eval = FALSE no se evalua el código y por ende no se ejecutará. Sirve por ejemplo para mostrar código como ejemplo o deshabilitar un bloque sin #.
- include = FALSE ejecuta el código, pero no muestra el código ni los resultados en el documento final. Se usa para configuración, por ejemplo se incluyen library()...
- echo = FALSE evita que el código se vea, pero muestra los resultados. Se usa mucho para usuarios finales q no quieren ver código.
- message = FALSE o warning = FALSE, evita que aparezcan mensajes o advertencias en el documento final.
- results = 'hide' oculta output impreso; fig.show = 'hide' oculta gráficos.
- error = TRUE **causa que el render continúe aunque el código devuelva error. En general no se incluye en el código final, pero puede ser muy util para depurar código. Tambien es útil si quieres incluir un error al enseñar. Por default error = FALSE provoca el el knit falle incluso si hay error en documento**.

Para lista completa de opciones en http://yihui.name/knitr/options/

Opción	Ejecuta	Muestra	Output	Gráficos	Mensajes	Advertencias
`eval = FALSE`	-		-	-	-	-
`include = FALSE`		-	-	-	-	-
`echo = FALSE`		-				
`results = "hide"`			-			
`fig.show = "hide"`				-		
`message = FALSE`					-	
`warning = FALSE`						-

collapse = TRUE muestra el block de fuente y de Output en un sólo bloque, si FALSE (es el default) lo muestra separado. **solo aplica para Markdown docs**.

Tablas y Graficos:

Por defecto, R Markdown imprime data frames y matrices tal como se ven en la consola:

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
```

pero si se quiere con otro formato, se puede utilizar la función kable de knitr:

```
knitr::kable(
  mtcars[1:5, ],
  caption = "Un kable de knitr."
)
```

Table 3: Un kable de knitr.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2

?knitr::kable para ver los otros modos en los que puedes personalizar la tabla. Sino indagar en los paquetes: **xtable, stargazer, pander, tables y ascii**. Cada uno provee un set de herramientas para generar tablas con formato a partir código de R.
En Markdown el párrafo que inicia con > lo hace un **bloque de cita**. si además queremos un píe de cita alineado a derecha se puede generar con tuffle::quote_footer() en una expresión inline por ejemplo:

> Citas:
> Al Pan, pan... y al vino,vino No por mucho madrugar amanece más temprano Autor Anonimo

Para quote individuales usar acento al revés alrededor de la palabra o frase que se quiere destacar.

Redimensionar graficos en Rmarkdown

Hay cinco opciones principales que controlan el tamaño de la figura: `fig.width`, `fig.height`, `fig.asp`, `out.width` y `out.height`. El tamaño de la imagen es un desafío porque hay dos tamaños (el tamaño de la figura creada por R y el tamaño al que se inserta en el documento de salida) y varias formas de especificarlo (es decir, altura, ancho y relación de aspecto: elige dos de tres). Estas son tres de las cinco:

- que los gráficos tengan un ancho consistente, se puede configurar: `fig.width = 6` (6 ") y `fig.asp = 0.618` (la proporción áurea) en los valores predeterminados. Luego, en bloques individuales, solo ajusto `fig.asp`.
- Controlo el tamaño de salida con `out.width` y lo configuro a un porcentaje del ancho de línea).De manera predeterminada, out.width = "70%" y fig.align = "center"
- Para poner múltiples gráficos en una sola fila, establezco out.width en 50% para dos gráficos, 33% en 3 gráficos, o 25% en 4 gráficos, y setfig.align = "default".

Si observas que tienes que entrecerrar los ojos para leer el texto de tu gráfico, debes ajustar fig.width. Si fig.width es mayor que el tamaño de la figura en el documento final, el texto será demasiado pequeño; si fig.width es más pequeño, el texto será demasiado grande. A menudo necesitarás experimentar un poco para calcular la proporción correcta entre fig.width y el ancho asociado en tu documento.

Si deseas asegurarte que el tamaño de fuente es el mismo en todas tus figuras, al establecer out.width, también necesitarás ajustar fig.width para mantener la misma proporción en relación al out.width predeterminado. Por ejemplo, si tu valor predeterminado de fig.width es 6 y out.width es 0.7, cuando establezcas out.width = "50%" necesitarás establecer fig.width a 4.3 (6 * 0.5 / 0.7).

Otras opciones importantes

1. configurar `fig.show = "hold"` para que los gráficos se muestren después del código
2. Para agregar un título al gráfico, usa fig.cap. En R Markdown esto cambiará la figura de "inline" a "floating".
3. Si estás produciendo resultados en formato PDF, el tipo de gráficos predeterminado es PDF, es bueno porque son graficos vectoriales de alta calidad. Pero puede ser muy grande la figura, entonces se puede configura `dev = "png"` para forzar el uso de PNG. Son de calidad ligeramente inferior, pero serán mucho más compactos.
4. darles nombres a los bloques de código que producen figuras. Etiquetar el bloque se utiliza para generar el nombre de archivo del gráfico en el disco, por lo que darle un nombre a los bloques hace que sea mucho más fácil seleccionar gráficas y reutilizarlas.

Caching:

Si tienes cómputos que toman mucho tiempo, quiza no quieras empezar desde cero a procesar los bloques de codigo (aunque es muy bueno para la reprodcibilidad del documento). En ese caso, la solución es `cache = TRUE`. Guarda el output del bloque en un archivo con un nombre especial en el disco. En ejecuciones siguientes, knitr revisará si el código ha cambiado, y si no ha hecho, reutilizará los resultados del caché. Pero por default no controla dependencia:por ejemplo si hay un chunk anterior que lea los datos a utilizar de disco. Para eso se utiliza la opción `dependson`. Knitr actualizará los resultados para el bloque cacheado cada vez que detecta que una de sus dependencias ha cambiado. **A tener en cuenta: cache solo hace seguimiento de los cambios dentro del archivo .Rmd**. Para seguir también los cambios hechos en ese archivo, puedes usar la opción `cache.extra`.

Acordarse de limpiar todos los cachés con `knitr::clean_cache()`.

El siguiente chunk se definió con cache.extra = file.info("un_archivo_muy_grande.csv"), y eval FALSE solo para poder usar a fines didácticos, para que no se evalue y trate de leer.

```
datos_crudos <- readr::read_csv("un_archivo_muy_grande.csv")
```

Opciones globales

Puedes definirlas Por ejemplo para tutoriales o libros usamos el sig. parrafo de global options, que utiliza nuestro formato preferido de ># para comentarios y se asegura que el código y el output se mantengan entrelazados: 'knitr::opts_chunk$set()' para setear opciones y no dejar las default. ** Se puede incluir en el chunk de setup**

knitr::opts_chunk$set(
comment = "#>",
collapse = TRUE
)

Esto ocultará por defecto el código, así que solo mostrará los bloques que deliberadamente has elegido mostrar (con echo = TRUE). Puedes considerar fijar message = FALSE y warning = FALSE, pero eso puede hacer más díficil la tarea de depurar problemas, porque no verías ningún mensajes en el documento final.

knitr::opts_chunk$set(echo = FALSE)

Con este chunk vas a conseguir que no se muestre el fuente ejecutado, y si deliberadamente en alguno se quiere mostrar en ese chunk hay que agregar echo = TRUE

##Código R en linea de documento

Como ya vimos se puede insertar con 53940. Cuando el resultado es un número se puede formatear usando la función format(). En este caso, definimos la función coma para que no tenga un número demasiado grande de decimales y el separador de miles con coma.

coma <- function(x) format(x, digits = 2, big.mark = ",")
coma(3452345)
#> [1] "3,452,345"
coma(.12358124331)
#> [1] "0.12"

HACER LOS EJERCICIOS DEL LIBRO DE ESTE TEMA QUE SON INTERESANTES

SOLUCIONANDO PROBLEMAS

Es más complicado en RMarkdown, porque al ejecutar no estas en un ambiente interactivo R.

Tips:

1. **Tratar de recrear el problema en una sesión interactiva.**

2. **Reinicia R, ejecuta todos los bloques de código. Si tienes suerte, eso recreará el problema y podrás descubrir lo que está ocurriendo interactivamente**

3. **Si lo anterior no ayuda, debe haber algo diferente entre nuestro ambiente interactivo y el ambiente de R Markdown -> hay que explorar sistemáticamente las opciones: La diferencia más común es el directorio de trabajo: el directorio de trabajo de R Markdown es el directorio en el que se encuentra: Revisar que el directorio de trabajo es el que esperas incluyendo getwd() en un bloque.**

4. **A continuación, piensa en todas las cosas que podrían causar el error. Necesitarás revisar sistemáticamente que tu sesión de R y tu sesión de R Markdown sean la misma. La manera más fácil de hacer esto es fijar error = TRUE en el bloque que causa problemas, y luego usa print() y str() para revisar que la configuración es la esperada.**

YAML: YET ANOTHER MARKUP LANGUAGE

Este lenguaje de marcado está diseñado para representar datos jerárquicos de modo facil para leer y escribir. R Markdown lo utiliza para controlar muchos detalles del output. Aquí sólo dos: parámetros del documento y bibliografías.

Parámetros:

Los documentos R Markdown pueden incluir uno o mas parámetros cuyos valores pueden ser fijados cuando se renderiza el reporte. Para declarar uno o más parámetros, utiliza el campo `param`.

Este ejemplo utiliza el parámetro `my_class` para determinar que clase de auto mostrar:

**IMPORTANTE: COMO QUERÍA QUE SALIERA EL .RMD Y NO DIERA ERROR TUVE QUE GENERAR UNA SUBDIRECTORIO EN RMARKDOWN LLAMADO rmarkdown_para_mostrar, copiar

ahí el ejemplo que quería mostrar dandole el nombre de fuel_economy.rmd, y luego en este chunk hago un cat que sería como un print del read_file(rmarkdown_para_mostrar/fuel-economy.Rmd) ... al tener el chunk un echo = false no se muestra lo que se ejecuta que es el cat(de lo que se lee: el contenido del archivo fuel-economy.Rmd)

````
---
output: html_document
params:
  mi_clase: "suv"
---

```{r setup, include = FALSE}
library(datos)
library(ggplot2)
library(dplyr)
clase <- millas %>% filter(clase == params$mi_clase)
```

# Economía de combustible en vehículos de tipo `r params$mi_clase`

```{r, message = FALSE, warning= FALSE}
ggplot(clase, aes(cilindrada, autopista)) +
  geom_point() +
  geom_smooth(se = FALSE)
```
````

- Puedes escribir vectores atómicos directamente en el encabezado YAML.
- Puedes también ejecutar expresiones arbitrarias de R agregando !r antes del valor del parámetro. Esta es una buena manera de especificar parámetros de fecha/hora.

Los parámetros están disponibles dentro de los bloques de código como una lista de solo lectura llamada params.

```
params:
 start: !r lubridate::ymd("2015-01-01")
 snapshot: !r lubridate::ymd_hms("2015-01-01 12:30:00")
```

Observar que el trozo de arriba no es un chunk de r, sino que es un quote para escribir params (y que no lo tome como parte del .Rmd?? no se lo probe sin el yaml y también funciono

Para ejecutar se puede hacer: * knitr común, y se generará el render con el parámetro definido como default * knit con parámetros: En RStudio, puedes hacer clic en la opción "Knit with Parameters" en el menú desplegable Knit para fijar parámetros, renderizar y previsualizar en un solo paso. Puedes personalizar el diálogo fijando otras opciones en el encabezado. Para ver más opciones de parametrización ir aqui * si necesitas producir varios reportes parametrizados, puedes ejecutar rmarkdown::render() con una lista de params:

```
rmarkdown::render("rmarkdown_para_mostrar/fuel-economy.Rmd", params = list(mi_clase = "suv"))
```

Esto es particularmente poderoso en conjunto con `purrr:pwalk()`. El siguiente ejemplo crea un reporte para cada valor de `clase` que se encuentra en `millas`. Primero creamos un data frame que tiene una fila para cada clase, dando el nombre de archivo (`filename`) del reporte y los `params`:

```
reportes <- tibble(
  clase = unique(millas$clase),
  filename = stringr::str_c("economia-combustible-", clase, ".html"),
  params = purrr::map(clase, ~ list(mi_clase = .))
)
reportes
```

```
## # A tibble: 7 x 3
##   clase       filename                              params
##   <chr>       <chr>                                 <list>
## 1 compacto    economia-combustible-compacto.html    <named list [1]>
## 2 mediano     economia-combustible-mediano.html     <named list [1]>
## 3 suv         economia-combustible-suv.html         <named list [1]>
## 4 2asientos   economia-combustible-2asientos.html   <named list [1]>
## 5 minivan     economia-combustible-minivan.html     <named list [1]>
## 6 pickup      economia-combustible-pickup.html      <named list [1]>
## 7 subcompacto economia-combustible-subcompacto.html <named list [1]>
```

Reportes es un tibble con 3 elementos: * un vector con las clases (únicas) dentro del archivo millas. En este caso un vector de 7 elementos * un vector con el nombre que llevará cada html que se va a generar. En este caso un vector de 7 elementos * una lista con sublistas: cada sublistas son los parámetros, en este caso hace una lista de 7 sublistas donde cada sublista tiene 1 elemento Entonces unimos los nombres de las columnas con los nombres de los argumentos de `render()`, y utilizamos la función `pwalk()` (*parallel walk*) del paquete **purrr** para invocar `render()` una vez en cada fila:

```
reportes %>%
  select(output_file = filename, params) %>%
  purrr::pwalk(rmarkdown::render, input = "rmarkdown_para_mostrar/fuel-economy.Rmd")
```

Bibliografía y citas

Pandoc puede generar automáticamente citas y bibliografía en varios estilos. Hay que especificar un archivo de bibliografía usando el campo `bibliography` en el YAML. El campo debe incluir una ruta del directorio que contiene tu archivo .Rmd al archivo que contiene el archivo de bibliografía:

```
bibliography: rmarkdown.bib
```

Puedes usar muchos formatos comunes de biliografía incluyendo BibLaTeX, BibTeX, endnote, medline.

Para crear una cita dentro de tu archivo .Rmd, usa una clave compuesta de '@' + el identificador de la cita del archivo de la bibliografía. Después, ubica esta cita entre corchetes. Aquí hay algunos ejemplos:

```
Multiple citas se separan con un ';': Bla bla[@smith04; @doe99].

Puedes incluir comentarios arbritarios dentro de los corchetes:
Bla bla [ver @doe99, pp. 33-35; también @smith04, ch. 1].

Remover los corchetes para crear una cita dentro del texto: @smith04
dice bla, o @smith04 [p. 33] dice bla.

Agrega un signo '-' antes de la cita para eliminar el nombre del autor:

Smith dice bla [-@smith04].
```

Cuando R Markdown *renderice* tu archivo, construirá y agregará una bibliografía al final del documento. La bibliografía contendrá cada una de las referencias citadas de tu archivo de bibliografía, pero no contendrá un encabezado de sección. Como resultado, es una práctica común finalizar el archivo con un encabezado de sección para la bibliografía, tales como `# Referencias` or `# Bibliografía`.

Puedes cambiar el estilo de tus citas y bibliografía referenciando un archivo CSL (sigla de "citation style language", es decir, lenguaje de estilo de citas) en el campo `csl`:

```
bibliography: rmarkdown.bib
csl: apa.csl
```

Tal y como en el campo de bibliografía, tu archivo csl debería contener una ruta al archivo. Aquí asumimos que el archivo csl está en el mismo directorio que el archivo .Rmd. Un buen lugar para encontrar archivos CSL para estilos de bibliografía comunes es http://github.com/citation-style-language/styles.

#Formatos de rmarkdow Se pueden indicar:
- Permanente, modificando el YAML -> output: html_document - transitorio, indicandolo dentro de rmarkdown::render("diamonds.Rmd", output_format="word_document") - Seleccionando salida con boton Knit

usar ?rmarkdown::html_document para saber que opciones se pueden utilizar cuando se hace render con formato html

si se quiere utilizar parámetros distintos a los de default utilizar un campo output extendido. Por ejemplo:

```
output:
  html_document:
    toc: true
    toc_float: true
```

También se pueden generar múltiples salidas, indicándolo en el YAML. Si no quieres modificar ninguna de las opciones por defecto debes indicar la palabra *default*:

```
output:
  html_document:
    toc: true
    toc_float: true
  pdf_document: default
```

Los formatos que se pueden generar son: -html
-pdf
-doc
-odt
-rtf
-md_document
-github_document

Documentos sin visualizar codigo

Recordar que se puede hacer indicando en el chunk setup,

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

Otra opción para los `html_document`es hacer que los fragmentos de código estén escondidos por defecto, pero visibles con un clic:

```
output:
  html_document:
    code_folding: hide
```

Notebooks

Es similar a un html_document, pero esta màs orientado para colaborar, por ej. en ciencia de datos. Estos propósitos diferentes llevan a que la salida HTML sea usada de diferentes maneras. Ambas contendrán todo el output renderizado, pero el notebook también contendrá el código fuente completo. Esto significa que puedes usar el archivo `.nb.html` generado por el notebook de dos maneras: 1. Puedes verlo en un navegador web y ver el output generado. A diferencia del `html_document`, esta renderización siempre incluye una copia incrustada del código fuente que lo generó.

2. Puedes editarlo en RStudio. Cuando abras un archivo `.nb.html`, RStudio automáticamente recreará el archivo `.Rmd` que lo creó. En el futuro, también podrás incluir archivos de soporte (por ej., archivos de datos `.csv`), que serán extraídos automáticamente cuando sea necesario.

Enviar archivos `.nb.html` por correo electrónico es una manera simple de compartir los análisis con tus colegas. Otra opciòn ùtil es

```
output:
  html_notebook: default
  github_document: default
```

`html_notebook` te da una vista previa local y un archivo que puedes compartir por correo electrónico. `github_document` crea un archivo md mínimo que puedes ingresar en Git. Puedes revisar fácilmente cómo los resultados de tus análisis (no solo el código) cambian con el tiempo y GitHub lo renderizará muy bien en línea.

Presentaciones:

Las presentaciones funcionan dividiendo tu contenido en diapositivas, con una nueva diapositiva que comienza en cada encabezado de primer nivel (#) o de segundo nivel (##). También puedes insertar una regla horizontal (***) para crear una nueva diapositiva sin encabezado.

R Markdown viene con tres formatos de presentación integrados: 1. ioslides_presentation - Presentación HTML con ioslides.

1. slidy_presentation - Presentación HTML con W3C Slidy.
2. beamer_presentation - Presentación PDF con LaTeX Beamer.

Dashboards:

Los dashboards o tableros de control son una forma útil de comunicar grandes cantidades de información de forma visual y rápida. Flexdashboard hace que sea particularmente fácil crear dashboards usando R Markdown y proporciona una convención de cómo los encabezados afectan el diseño:

- Cada encabezado de Nivel 1 (#) comienza una nueva página en el dashboard.
- Cada encabezado de Nivel 2 (##) comienza una nueva columna.

- Cada encabezado de Nivel 3 (###) comienza una nueva fila.

Por ejemplo, puedes producir este dashboard:

Usando este código:

Para obtener más información (en inglés) acerca de Flexdashboard visita http://rmarkdown.rstudio.com/flexdashboard/.