Markdown Basics

Level 1 heading

Paragraph text. Note that comments in Rmd use html style <!- –> format instead of R # format. That's because the # symbols are used to define headings. Fortunately you can use the RStudio comment/uncomment shortcut (Command + Shift + C) and it will comment text the right way for whatever type of code you are writing.

Level 2 heading

Paragraph text. Use single asterisks for *italics*, double asterisks for **bold**. Notice that if you hit enter and create multiple lines, markdown interprets this as a continuous paragraph.

If you want a new paragraph, put a blank line in between. Github README files use these same conventions but are saved as .md (plain markdown instead of R Markdown). These are good to know for documenting your project repo, not just for using in Rmd.

Level 3 heading

Lists are easy to create.

- Top level of an un-ordered list
  - 2nd level of list using tab
  - 2nd level again
- Back to top level

1. Numbered lists work the same way
2. But just keep using "1".
3. Rmd will figure it out for you
4. See!

Hyperlinks are easy to embed in text, like so: click here. Or if you want the URL to display you can do so like this https://padlab.ucr.edu.

Let's get some R in this markdown

Backticks (the funny looking apostrophes above tab) let us embed R code in the document. You can do this in a text sentence like so: 25. Why do we need to put 'r' in there? Let's try it without: `5*5`. You can use backticks to just format something as code (which is nice for sharing coding examples). Putting 'r' in the code tells Rmd to run it as R code. You need to specify R because Rmd can actually run other types of code, including Python and shell scripts.

Code chunks

For more extended code, we need to insert a code chunk. You can use the insert menu in the upper right corner, or you can put three backticks on a line followed by {r} to specify that it is an R language chunk. Every line will be treated as R code until you put a line with another three backticks.

```r
5*5
```

```
## [1] 25
```

```r
x <- 5*5
```

Now we are back to markdown. Notice that by default, the R chunk prints the code, all warnings/messages, and the result. This might be desirable sometimes (such as when demonstrating code and its result), but at other times you might prefer to tailor the output of a chunk. You can do this with chunk options.

To simply display code without running it, use `eval = FALSE`:

```r
5*5
z <- 'Hello'
```

To run code quietly without displaying anything, use `include = FALSE`:

`include = FALSE` is good for creating a 'setup chunk', which normally would go at the top of the script. A setup chunk is good to have at the start of your file to load libraries and data and do other data cleaning that you might not want to scroll through.

To suppress the code but show the results, use `echo = FALSE`:

```
## [1] "Hello"
```

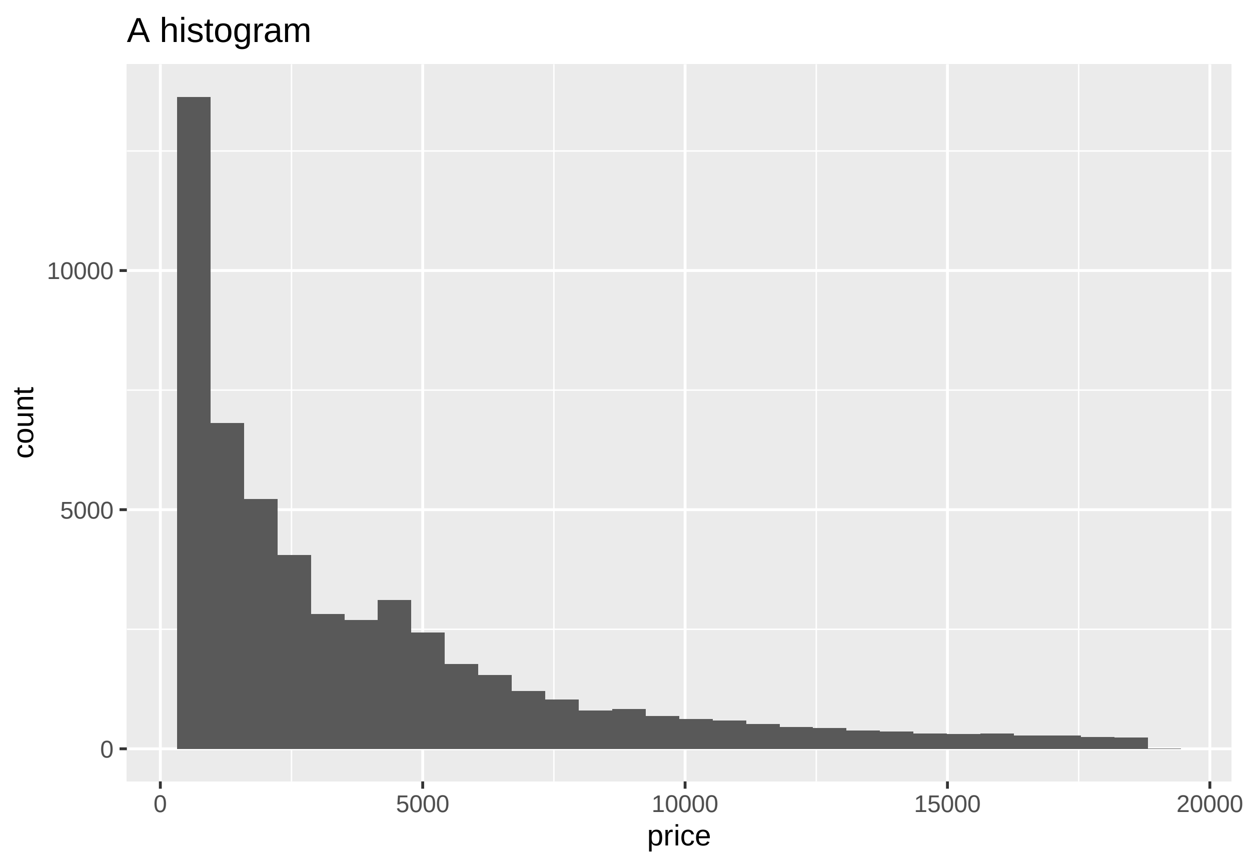
Other helpful options include `message = FALSE` and `warning = FALSE` to keep R warnings and messages from printing in your document.

Knitting will run all code chunks, but when working on a Rmd document you may want to test code chunks without knitting them (or see the results/warnings/messages if you chose to suppress them in the output). Using the little green play button in the upper-right corner will run a chunk and create a little output area beneath. This is similar to highlighting lines and running them from within a script (which you can also do in Rmd). The "triangle with a green line under it" button means "run everything up to here", which is helpful to make sure all of your processing steps have been run up to the point that you're working on.

Figures

A code chunk that produces a figure will insert that figure into the output document.

```r
ds <- diamonds
ggplot(ds, aes(x = price)) + geom_histogram() + ggtitle("A histogram")
```



Have graphics from other sources? Easy! It's just like inserting a hyperlink (I added the `{width=50%}` tag at the end to keep it from filling the entire width of the page, but that's optional:

Tables

Tibbles can become tables really easily. You can print them as is:

```r
ds %>% head()
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

A much nicer option is to use a formatting function such as `kable`, which is a function from the `knitr` package (what turns R Markdown files into documents) that's designed to control the formatting of tables.

```r
library(knitr)
kable(ds %>% head(), caption = "The first 5 diamond cuts and prices")
```

Figure 1: R Markdown hex logo

Table 1: The first 5 diamond cuts and prices

carat	cut	color	clarity	depth	table	price	x	y	z
0.23	Ideal	E	SI2	61.5	55	326	3.95	3.98	2.43
0.21	Premium	E	SI1	59.8	61	326	3.89	3.84	2.31
0.23	Good	E	VS1	56.9	65	327	4.05	4.07	2.31
0.29	Premium	I	VS2	62.4	58	334	4.20	4.23	2.63
0.31	Good	J	SI2	63.3	58	335	4.34	4.35	2.75
0.24	Very Good	J	VVS2	62.8	57	336	3.94	3.96	2.48

How to create a new Rmd file

File > New File > R Markdown. Pick a format, give it a title and author. But don't stress about your choices, because you can always change the YAML header later. RStudio will create a nice template depending on which format you choose to get you started.

There are many different output options to change the format of your document and its appearance. Check https://bookdown.org/yihui/rmarkdown/documents.html for an overview.

```
Sys.time()
```

```
## [1] "2025-03-09 18:24:00 PDT"
```

```
sessionInfo()
```

```
## R version 4.4.2 (2024-10-31)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK ve
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Los_Angeles
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] knitr_1.49      lubridate_1.9.4 forcats_1.0.0   stringr_1.5.1
##  [5] dplyr_1.1.4     purrr_1.0.2     readr_2.1.5     tidyr_1.3.1
##  [9] tibble_3.2.1    ggplot2_3.5.1   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.6      compiler_4.4.2    tidyselect_1.2.1  scales_1.3.0
##  [5] yaml_2.3.10       fastmap_1.2.0     R6_2.5.1          labeling_0.4.3
##  [9] generics_0.1.3    munsell_0.5.1     pillar_1.10.0     tzdb_0.4.0
## [13] rlang_1.1.4       utf8_1.2.4        stringi_1.8.4     xfun_0.49
## [17] timechange_0.3.0  cli_3.6.3         withr_3.0.2       magrittr_2.0.3
## [21] digest_0.6.37     grid_4.4.2        rstudioapi_0.17.1 hms_1.1.3
## [25] lifecycle_1.0.4   vctrs_0.6.5       evaluate_1.0.1    glue_1.8.0
## [29] farver_2.1.2      colorspace_2.1-1  rmarkdown_2.29    tools_4.4.2
## [33] pkgconfig_2.0.3   htmltools_0.5.8.1
```