# R Markdown Template Example

Last Update: 12-Aug-25

## Summary

This paper covers an example R Markdown template file. While it is itself part of a larger general workflow structure that I created, it can provide insight into basic concepts useful for a template program. The paper assumes a working knowledge of R syntax and R Markdown.

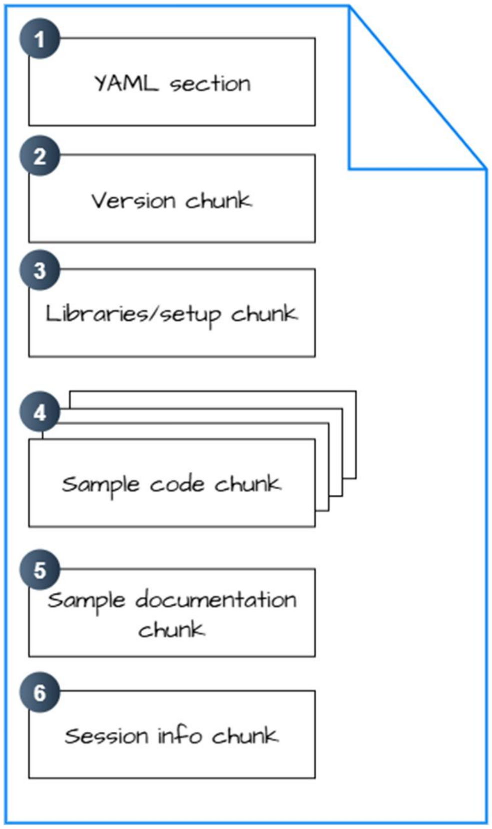
## Background

Using a template can speed up development, reduce errors, produce a consistent look, feel and execution. The R Markdown template was produced to give anyone (programmer or not) a fast way to write code that produces output and allows repeatability.

## General Structure

We will now show the general flow of the parts.

1. YAML section customizes the output behavior and content
2. Version chunk (in R) documents changes to the program.
3. Libraries/setup chunk (in R) brings in the libraries and calls the configuration file (see the R Configuration Model).
4. Sample code chunk(s) (in R or other languages) can be repeated as needed for the programming needs.
5. Sample documentation chunk (in R) provides a standardized approach to documentation the work products of the program.
6. Session Info chunk provides metrics and a description of the session that created the output.



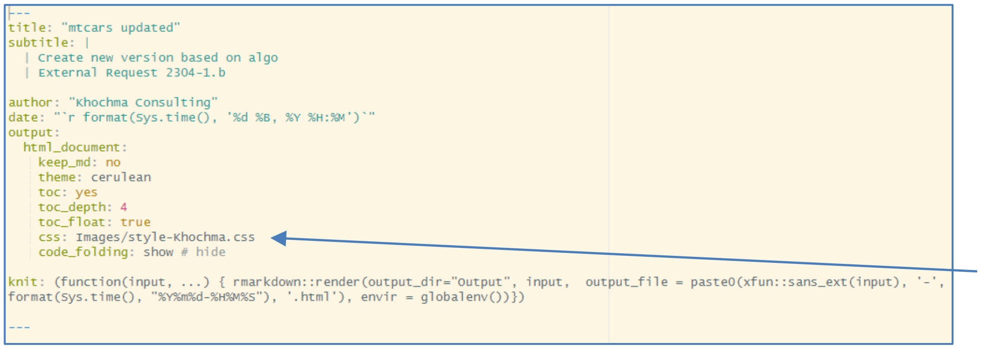
## Detailed Structure

The sample R Markdown template is bare-bones example of the template concept. For some of my work, I create function specific templates (e.g., MatchIT template, Geospatial joining, etc.) Those have specific parameters and options, whereas the example below is meant to show basic functionality.

**A sample .RMD file**

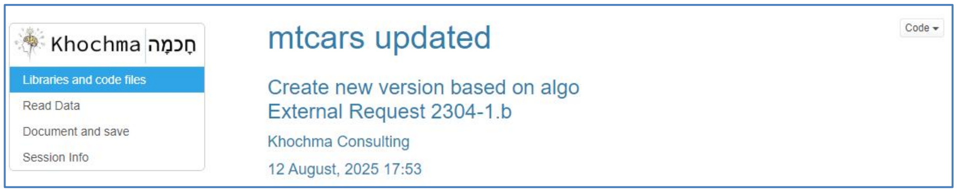
```
---
title: "Step00-Blank"
subtitle: |
  | Description/Purpose
  | Study

author: "Author"
date: "`r format(Sys.time(), '%d %B, %Y %H:%M')`"
output:
  html_document:
    keep_md: no
    theme: cerulean
    toc: yes
    toc_depth: 4
    toc_float: true
    code_folding: show # hide

knit: (function(input, ...) { rmarkdown::render(output_dir="Output", input,
output_file = paste0(xfun::sans_ext(input), '-',  format(Sys.time(),
"%Y%m%d-%H%M%S"), '.html'), envir = globalenv())})

---
```

*Example from a program*



```
---
title: "mtcars updated"
subtitle: |
  | Create new version based on algo
  | External Request 2304-1.b

author: "Khochma Consulting"
date: "`r format(Sys.time(), '%d %B, %Y %H:%M')`"
output:
  html_document:
    keep_md: no
    theme: cerulean
    toc: yes
    toc_depth: 4
    toc_float: true
    css: Images/style-Khochma.css
    code_folding: show # hide

knit: (function(input, ...) { rmarkdown::render(output_dir="Output", input,  output_file = paste0(xfun::sans_ext(input), '-',
format(Sys.time(), "%Y%m%d-%H%M%S"), '.html'), envir = globalenv())})

---
```

*Example output (notice the "Code" button on the right which can globally show/hide code)*



mtcars updated

Create new version based on algo
External Request 2304-1.b

Khochma Consulting

12 August, 2025 17:53

**Explanation**

YAML (https://yaml.org/) controls the output.

For purposes of this paper, the items in yellow are edited and modified to show their use.

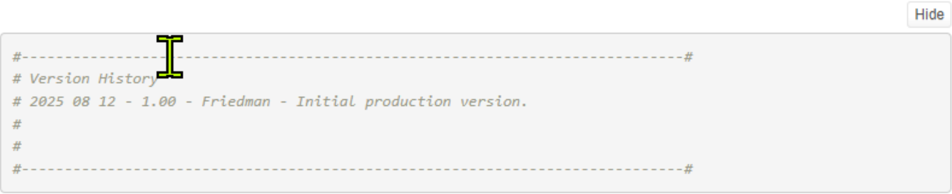
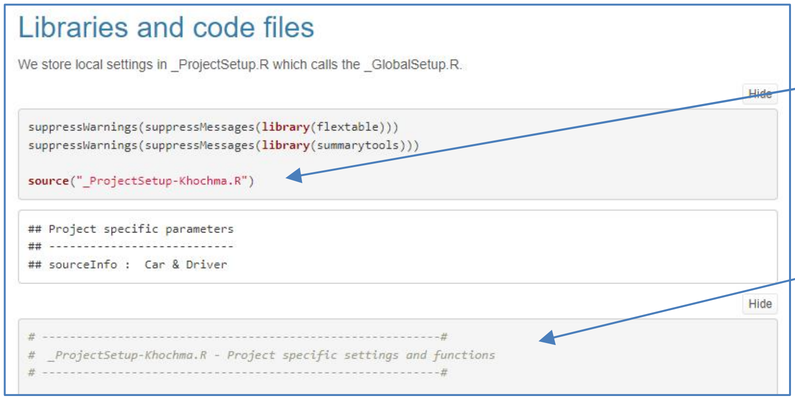
The theme value can be used to change the theme of the output.
https://www.datadreaming.org/posts/2018-04-11-r-markdown-theme-gallery/2018-04-11-r-markdown-theme-gallery.html

Code folding is an extremely useful tweak. You can show or hide all the code with a toggle at the top and above each individual code section in the output.

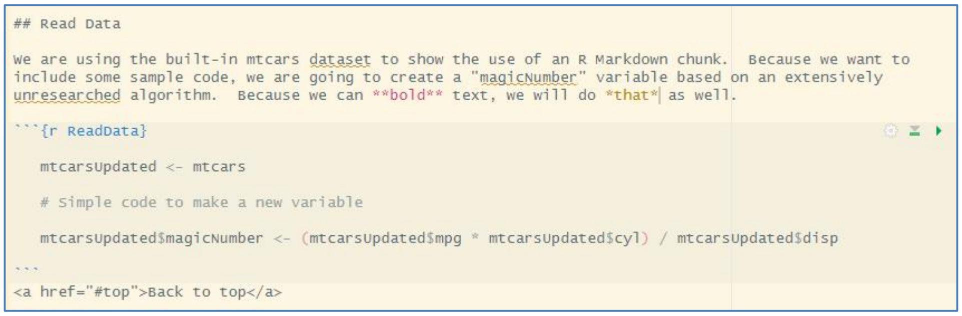
The knit: call does the work of identifying where and how to name the output. In this case, the output goes into a subfolder called "Output" and is named based on the input file, and date/timestamp.

In the example on the left, a custom .css file was written to put a logo on top of the table of contents.

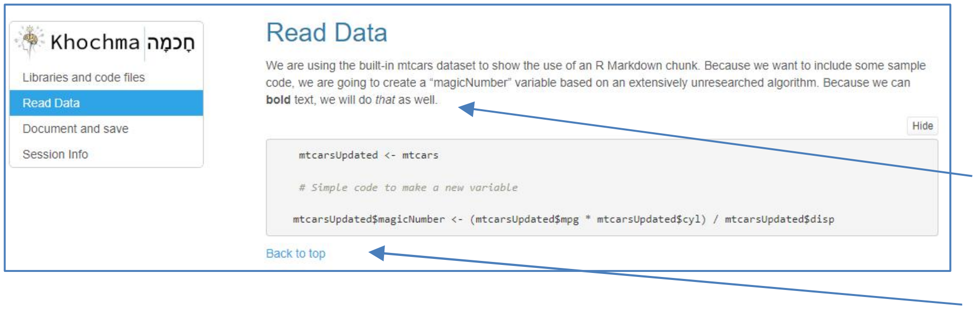
Each row in the table of contents relates to a specific section chunk in the program.

| A sample .RMD file | Explanation |
|---|---|
| ```{r VersionInfo}<br>#-----------------------------------------------------------------------#<br># Version History<br># 202x mm dd - 1.00 - Author - Initial production version.<br>#<br>#<br>#-----------------------------------------------------------------------#<br>```<br><br>*Example from a program  (notice the “Hide” button on the right)*<br><br>Hide<br>#------------------------------------------------------------------------------#<br># Version History<br># 2025 08 12 - 1.00 - Friedman - Initial production version.<br>#<br>#<br>#------------------------------------------------------------------------------# | Best practice suggests you document changes to code. Even in less formal programming, it is useful to put a high-level note and version programs when you make meaningful changes to a program.<br>Having the boilerplate chunk in a template can (hopefully) help people document changes. |
| ```{r libraries}<br><br>library(packageX)<br><br>source("_ProjectSetup.R")<br><br>```<br><br>```{r, code = readLines("_ProjectSetup-Khochma.R"), echo=TRUE, eval=FALSE}<br>```<br><br>*Example from a program*<br>## Libraries and code files<br><br>We store local settings in _ProjectSetup.R which calls the _GlobalSetup.R.<br><br>```{r libraries}<br><br>library(package)<br><br>source("_ProjectSetup.R")<br><br>```<br><br>*Example output*<br><br>Libraries and code files<br>We store local settings in _ProjectSetup.R which calls the _GlobalSetup.R.<br>Hide<br>suppressWarnings(suppressMessages(library(flextable)))<br>suppressWarnings(suppressMessages(library(summarytools)))<br><br>source("_ProjectSetup-Khochma.R")<br><br>## Project specific parameters<br>## ---------------------------<br>## sourceInfo :  Car & Driver<br>Hide<br># ----------------------------------------------------------#<br>#  _ProjectSetup-Khochma.R - Project specific settings and functions<br># ----------------------------------------------------------# | These two chunks are examples of bringing in packages.<br><br>The second chunk is a workaround for ensuring the “_ProjectSetup.R” code appears as a code chunk in the output. Doing it this way allows you to show/hide the code chunk, instead of having it appear as output.<br><br>In the output on the left, you can see the source statement and below it is the produced output.<br><br>Below that is the start of showing the code in “_ProjectSetup-Khochma.R” for documentation completeness. |

**A sample .RMD file**

~~~
## Section

Description of the section.

```{r Section}

   # Your Code Here

~~~
<a href="#top">Back to top</a>
```

*Example from a program*



```
## Read Data

We are using the built-in mtcars dataset to show the use of an R Markdown chunk.  Because we want to
include some sample code, we are going to create a "magicNumber" variable based on an extensively
unresearched algorithm.  Because we can **bold** text, we will do *that* as well.

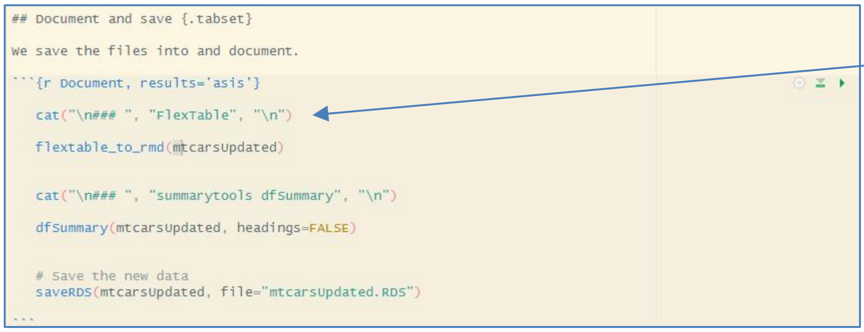
```{r ReadData}

   mtcarsUpdated <- mtcars

   # Simple code to make a new variable

   mtcarsUpdated$magicNumber <- (mtcarsUpdated$mpg * mtcarsUpdated$cyl) / mtcarsUpdated$disp
```
<a href="#top">Back to top</a>
```

*Example output*



Khochma חָכְמָה
Libraries and code files
Read Data
Document and save
Session Info
Hide

Read Data

We are using the built-in mtcars dataset to show the use of an R Markdown chunk. Because we want to include some sample code, we are going to create a "magicNumber" variable based on an extensively unresearched algorithm. Because we can **bold** text, we will do *that* as well.

```
   mtcarsUpdated <- mtcars

   # Simple code to make a new variable

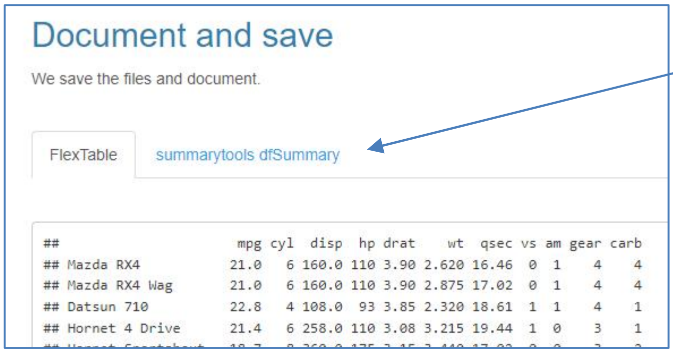
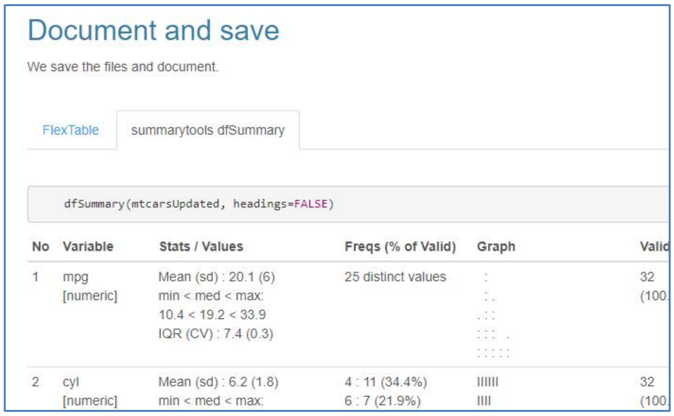
  mtcarsUpdated$magicNumber <- (mtcarsUpdated$mpg * mtcarsUpdated$cyl) / mtcarsUpdated$disp
```

Back to top

**Explanation**

This sample section and chunk let you name a section and describe it in as much detail as you want using R Markdown.

It is important to ALWAYS use unique names in the {r Section} statement.   You cannot knit a document if there are sections with the same name.

This html code produces a simple “back to top” in the output.

As we move through the document, the TOC reflects what section we are in.

Examples of bold and italics are shown in the R Markdown text as well.

You can see the “back to top” link in action.
```

## A sample .RMD file

~~~
## Document and save {.tabset}

We save the files into and document.

```{r Document, results='asis'}



~~~

<a href="#top">Back to top</a>
```

*Example from a program*



```
## Document and save {.tabset}

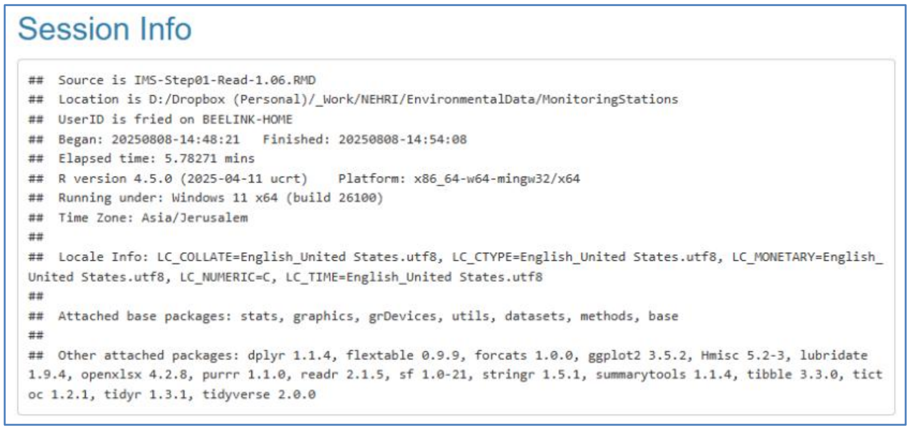
We save the files into and document.

```{r Document, results='asis'}

  cat("\n### ", "FlexTable", "\n")

  flextable_to_rmd(mtcarsUpdated)


  cat("\n### ", "summarytools dfSummary", "\n")

  dfSummary(mtcarsUpdated, headings=FALSE)


  # Save the new data
  saveRDS(mtcarsUpdated, file="mtcarsUpdated.RDS")

```
```

*Example output*



Document and save

We save the files and document.

FlexTable
summarytools dfSummary

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
```



Document and save

We save the files and document.

FlexTable
summarytools dfSummary

```
dfSummary(mtcarsUpdated, headings=FALSE)
```

| No | Variable | Stats / Values | Freqs (% of Valid) | Graph | Valid |
|---|---|---|---|---|---|
| 1 | mpg<br>[numeric] | Mean (sd) : 20.1 (6)<br>min < med < max:<br>10.4 < 19.2 < 33.9<br>IQR (CV) : 7.4 (0.3) | 25 distinct values | :<br>: .<br>. : :<br>: : : .<br>: : : : : | 32<br>(100 |
| 2 | cyl<br>[numeric] | Mean (sd) : 6.2 (1.8)<br>min < med < max: | 4 : 11 (34.4%)<br>6 : 7 (21.9%) | IIIIII<br>IIII | 32<br>(100 |

## Explanation

This section and code chunk have some interesting capabilities. The {.tabset} statement will use tabs in the output (instead of vertically placing the output) which is useful if you have a few dozen datasets you are looping through to produce documentation for.

The `cat("\n### " …` statement at the 3rd level (3 hash tags means 3rd level) places the output in its own tab.

We see two tabs:

- FlexTable
- Summarytools dfSummary

Clicking on each brings up the associated output.
```

A sample .RMD file

~~~
## Session Info

```{r SessionInfo, echo=FALSE}
     fnSessionInfo()
```
~~~

*Example output*



Session Info

```
##  Source is IMS-Step01-Read-1.06.RMD
##  Location is D:/Dropbox (Personal)/_Work/NEHRI/EnvironmentalData/MonitoringStations
##  UserID is fried on BEELINK-HOME
##  Began: 20250808-14:48:21   Finished: 20250808-14:54:08
##  Elapsed time: 5.78271 mins
##  R version 4.5.0 (2025-04-11 ucrt)    Platform: x86_64-w64-mingw32/x64
##  Running under: Windows 11 x64 (build 26100)
##  Time Zone: Asia/Jerusalem
##
##  Locale Info: LC_COLLATE=English_United States.utf8, LC_CTYPE=English_United States.utf8, LC_MONETARY=English_
United States.utf8, LC_NUMERIC=C, LC_TIME=English_United States.utf8
##
##  Attached base packages: stats, graphics, grDevices, utils, datasets, methods, base
##
##  Other attached packages: dplyr 1.1.4, flextable 0.9.9, forcats 1.0.0, ggplot2 3.5.2, Hmisc 5.2-3, lubridate
1.9.4, openxlsx 4.2.8, purrr 1.1.0, readr 2.1.5, sf 1.0-21, stringr 1.5.1, summarytools 1.1.4, tibble 3.3.0, tict
oc 1.2.1, tidyr 1.3.1, tidyverse 2.0.0
```

Explanation

This section and chunk are boilerplate and call a custom written function based on the package SessionInfo.  It produces documentation and metrics.

In the example to the left, you can see the following:

- the input program name
- where it was located
- who ran it on what computer
- when it ran
- how long it took to run
- what version of R was used
- what operating system
- locale information
- R package information, including version number