Writing a reproducible paper with R Markdown and Pagedown[2]

Paul C. Bauer
Mannheim Centre for European Social
Research
mail@paulcbauer.eu

Camille Landesvatter
Mannheim Centre for European Social
Research
camille.landesvatter@mzes.uni-
mannheim.de

First version: 20 June, 2021
This version: 23 June, 2021

Download: https://osf.io/k8jhx

Abstract

The present paper provides a template to write a reproducible scientific paper with R Markdown and Pagedown.[1] Below we outline some of the "tricks"/code (e.g., referencing tables, sections etc.) we had to figure out to produce this document. The underlying files which produce this document can be downloaded here. Importantly, we also provide different CSS and HTML files that can be used to achieve a pdf output with the look of a "working paper". We are convinced that in the future there will be many improvements and developments with regards to RStudio, R markdown and Pagedown. We intend to update this file when we discover more convenient code. You can follow any updates on the [github repository](https://github.com/paulcbauer/Writing_a_reproducable_paper_in _pagedown/.

Keywords: open science, transparency, replication, R, markdown, pagedown.

---

[2]Corresponding adress: mail@paulcbauer.eu, camille.landesvatter@mzes.uni-mannheim.de

[1]Based on an earlier R Markdown template that uses Latex and can be downloaded under https://osf.io/q395s (see Bauer 2018).

1. Why reproducible research (in R)?

Some arguments…

- **Access**: Research is normally funded by taxpayers (researchers are also taxpayers). Hence, it should be freely accessible to everyone without any barriers, e.g., without requiring commercial software. Importantly, researchers from developing countries are even more dependent on free access to knowledge (Kirsop and Chan 2005).

- **Reproducability**: Even if you have written a study and analyzed the data yourself you will forget what you did after a few months. A fully reproducible setup will help you to trace back your own steps. Obviously, the same is true for other researchers who may want to understand your work and built on it. It may sound like a joke but why not aim for a document that can be used to reproduce your findings in 500 years.

- **Errors**: Manual steps in data analysis (e.g., manually copy/pasting values into a table etc.) may introduce errors. R Markdown allows you to **automatize** such steps and/or avoid them.

- **Revisions**: Revising a paper takes much less time if you have all the code you need in one place, i.e., one `.rmd` file. For instance, if you decide to exclude a subset of your data you simply need to insert one line of your code at the beginning and everything is rebuilt/re-estimated automatically.

2. Why Pagedown?

Formatting text as PDF is probably one of the most widespread standards in the scientific community, especially when it comes to submitting papers and similar documents. The traditional way to well-formatted and good-looking PDFs is often through LaTeX or Word. However, if you have spent hours and hours debugging latex code (or getting it to run) you may be on the lookout for something new.

The fairly new `pagedown` R package takes a completely new approach. While the main purpose of pagedown is to create high-quality PDFs, the idea is to take advantage of modern web technologies (HTML/JSS/Javascript) with which one can design web pages and eventually print those to PDF.

While web pages are usually single-page scrollable documents, pagedown uses the JavaScript library `Paged.js` which allows documents to be paginated with elements like headers, footers and everything a readable scientific paper will need. Additionally, pagedown documents are based on R Markdown. In our view, Pagedown and the underlying technology may replace Latex in the long run. In the near future it should also be possible to produce a PDF with static graph and an equivalent html with interactive graps (see dicsussion here).

3. Prerequesites

We assume that you are using R on a day-to-day basis and you may have even started to work in R Markdown. If you don't know what R Markdown is there are many great resources that you should use (e.g. watch this short video). An older template [see Bauer (2018); `https://osf.io/q395s`] on which this newer template is based, may provide a quick entry point to writing a reproducible with R Markdown and Latex.

Based on R Markdown, Pagedown allows you to create custom and well-formatted (paged) HTML Documents. For a comprehensive overview watch this video which is a record of a talk introducing `pagedown` given by Yihui Xie (who in addition to Romain Lesur developed the `pagedown` package). If you are not in a video watching mood find the slides here.

Then…

- …install R and Rstudio (most recent versions) (R Core Team 2017; RStudio Team 2015).
- …install the "pagedown"-package from github using the code below (Xie and Lesur 2021; Xie et al. 2021).

```
R> remotes::install_github('rstudio/pagedown')
```

- …also install the packages below using the code below (Xie 2014, 2015, 2016, 2017, 2018; Zhu 2017).

```
R> install.packages(c("rmarkdown", "knitr", "kableExtra",
R+                    "stargazer", "modelsummary", "knitr"))
```

- …download the 4 input files we created — `paper.rmd`, `references.bib`, `data.csv` and `american-sociological-association.csl` — from this folder. Ignore the other files.
- …also download the 4 styling files we created: `wp_paged.html`, `wp.css`, `wp-fonts.css` and `wp-pages.css`.
- …store all 8 files from above together in one folder (and use this folder as your working directory later on)
- …learn R and read about the other underlying components namely Markdown, R Markdown and Pagedown.
- …pagedown comes with several Rmd-templates (presentations, poster, thesis, etc.) and via this review we provide another template for a working paper style. If however you want to modify single aspects or create your own template, you will need to at least gain some basic skills in CSS and HTML.

4. Basics: Input files, output files and the YAML header

All the files you need to produce the present PDF file are:

1. the input files:

- `paper.rmd` (the underlying R Markdown file).
- `references.bib` (the bibliography).
    - I use paperpile to manage my references and export the `.bib` file into the folder that contains my `.rmd` file.
- `data.csv` (some raw data).
- `american-sociological-association.csl` (defines the style of your bibliography).[3]

2. the "styling" files:

Basically, these are files you will need to specify in the YAML of your rmd-file, so that R and ultimately pagedown recognizes the certain style you want to achieve for your document. With using our templates, you will create a document that has the "look" of a working paper (we based our files on the jss_paged pagedown format).

- `wp_paged.html` (based on `jss_paged.html`)
- `wp.css`
- `wp-fonts.css`
- `wp-pages.css`

Take `paper.rmd` (the underlying R Markdown file of this pdf) and have a look at the YAML (line #18 - #22) to see how to specifiy these files. Basically, what happens here is that within the jss_paged function we additionally specify that we want to use custom CSS and custom HTML.

Download these files and save them into a folder. Close R/Rstudio and directly open `paper_pagedown.rmd` with RStudio. Doing so assures that the working directory is set to the folder that contains `paper.rmd` and the other files.[4]

Once you run/compile the `paper.rmd` file in Rstudio it creates a output file called `paper_pagedown.html`.

---

[3]You can download various citation style files from this webpage: `https://github.com/citation-style-language/styles`.

[4]You can always check your working directory in R with `getwd()`.

By using pagedown's `chrome_print` function in the YAML (line #25) your html based web page will be printed to `paper_pagedown.pdf` (the one you are reading right now).

Both outputs will be saved in your working directory.

5. Referencing within your document

To see how referencing works simply see the different examples for figures, tables and sections below. For instance in Section 8 you can find different ways of referencing tables. The code of the underlying `paper.rmd` will show you how I referenced Section 8 right here namely with 'Section \@ref(sec:tables)'.

6. Software versioning

Software changes and gets updated, especially with an active developer community like that of R. Luckily you can always access old versions of R and old version of R packages in the archive. In the archive you need to choose a particular package, e.g dplyr and search for the right version, e.g., `dplyr_0.2.tar.gz`. Then insert the path in the following function: `install.packages("https://....../dplyr_0.2.tar.gz", repos=NULL, type="source")`. Ideally, however, results will be simply reproducible in the most current R and package versions.

I would recommend to use the command below and simply add it to the appendix as I did here in Appendix 14.1. This will make sure you always inform the reader about the package versions your relied on in your paper. For more advanced tools see packrat.

```
R> cat(paste("#", capture.output(sessionInfo()), "\n", collapse =""))
R>   # or use message() instead of cat()
```

7. Data

7.1. Import

Generally, code is evaluated by inserting regular R Markdown blocks.

```
R> x <- 1:10
R> x
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

Below we import an exemplary dataset (you can find `data.csv` in the folder with the other files).

```
R> data <- read.csv("data.csv")
R> head(data)
```

```
  Fertility Agriculture Examination Education Catholic Infant.Mortality
1      80.2        17.0          15        12     9.96              22.2
2      83.1        45.1           6         9    84.84              22.2
3      92.5        39.7           5         5    93.40              20.2
4      85.8        36.5          12         7    33.77              20.3
5      76.9        43.5          17        15     5.16              20.6
6      76.1        35.3           9         7    90.57              26.6
```

7.2. Putting your entire data into the .rmd file

Applying the function `dput()` to an object gives you the code needed to reproduce that object. So you could paste that code into your `.rmd` file if you don't want to have extra data files. This makes sense were data files are small.

```
R> dput(data[1:5,]) # here we only take a subset
```

```
structure(list(Fertility = c(80.2, 83.1, 92.5, 85.8, 76.9), Agriculture = c(17,
45.1, 39.7, 36.5, 43.5), Examination = c(15L, 6L, 5L, 12L, 17L
), Education = c(12L, 9L, 5L, 7L, 15L), Catholic = c(9.96, 84.84,
93.4, 33.77, 5.16), Infant.Mortality = c(22.2, 22.2, 20.2, 20.3,
20.6)), row.names = c(NA, 5L), class = "data.frame")
```

You can then insert the dput output in your `.rmd` as below.

```
R> data <- structure(list(Fertility = c(80.2, 83.1, 92.5, 85.8, 76.9), Agriculture = c(17,
R+ 45.1, 39.7, 36.5, 43.5), Examination = c(15L, 6L, 5L, 12L, 17L
R+ ), Education = c(12L, 9L, 5L, 7L, 15L), Catholic = c(9.96, 84.84,
R+ 93.4, 33.77, 5.16), Infant.Mortality = c(22.2, 22.2, 20.2, 20.3,
R+ 20.6)), class = "data.frame", row.names = c(NA, -5L))
```

8. Tables

Producing good tables and referencing these tables within a R Markdown document has been a hassle but got much better. Examples that you may use are shown below.

8.1. Tables with kable() and kable_styling()

A great function is `kable()` (knitr package) in combination with `kableExtra`. Table 8.1 provides an example. To reference the table produced by the chunk you need to add ´tab:´ to the chunk name, i.e., ´tab:table-2´ and would reference it by adding "Table \@ref(tab:table-2)" in your text.

```
R> library(knitr)
R> library(kableExtra)
R>
R> kable(swiss[1:10,], row.names = TRUE,
R+        caption = 'Table with kable() and kablestyling()',
R+        format = "html", booktabs = T) %>%
R+          kable_styling(full_width = T,
R+                      latex_options = c("striped",
R+                                        "scale_down",
R+                                        "HOLD_position"),
R+                  font_size = 10)
```

	Fertility	Agriculture	Examination	Education	Catholic	Infant.Mortality
Courtelary	80.2	17.0	15	12	9.96	22.2
Delemont	83.1	45.1	6	9	84.84	22.2
Franches-Mnt	92.5	39.7	5	5	93.40	20.2
Moutier	85.8	36.5	12	7	33.77	20.3
Neuveville	76.9	43.5	17	15	5.16	20.6
Porrentruy	76.1	35.3	9	7	90.57	26.6
Broye	83.8	70.2	16	7	92.85	23.6
Glane	92.4	67.8	14	8	97.16	24.9
Gruyere	82.4	53.3	12	7	97.67	21.0
Sarine	82.9	45.2	16	13	91.38	24.4

Table 8.1: Table with kable() and kablestyling()

8.2. Tables with modelsummary

The `modelsummary` package provides a variety of tables and plots to summarize statistical models and data in R. Modellsummary plots and tables are highly customizable and they can be saved to almost all formats, e.g., HTML, PDF and Markdown. This makes ist especially easy to embed them in dynamic documents. Please look at the package's extensive documentation where they also show examples for almost any plot or table you might be looking for. In this template we demonstrate an example for modelsummary's `datasummary` function. `Datasummary` creates frequency tables, crosstab tables, correlation tables, balance tables and many **more**.

8.2.1. Summarize numeric variables

Table 8.2 shows a summary table for numeric variables.

```
R> library(modelsummary)
R> datasummary_skim(swiss,
R+               type="numeric",
R+               histogram=T,
R+               title = "Summary: Numeric variables")
```
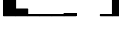
	Unique (#)	Missing (%)	Mean	SD	Min	Median	Max	
Fertility	46	0	70.1	12.5	35.0	70.4	92.5	
Agriculture	47	0	50.7	22.7	1.2	54.1	89.7	
Examination	22	0	16.5	8.0	3.0	16.0	37.0	
Education	19	0	11.0	9.6	1.0	8.0	53.0	
Catholic	46	0	41.1	41.7	2.1	15.1	100.0	
Infant.Mortality	37	0	19.9	2.9	10.8	20.0	26.6	

Table 8.2: Summary: Numeric variables

8.2.2. Summarize categorical variables

Table 8.3 shows a summary table for categorical variables.

```
R> # Create categorical variables
R> swiss$Education_cat <- cut(swiss$Education,
R+               breaks=c(-Inf, 6, 12, Inf),
R+               labels=c("low","middle","high"))
R> swiss$Infant.Mortality_cat <- cut(swiss$Infant.Mortality,
R+               breaks=c(-Inf, 18.15, 21.70, Inf),
R+               labels=c("low","middle","high"))
R>
R> library(flextable)
R> tab_cat <- datasummary_skim(swiss,
R+                    type="categorical",
R+                    title = "Summary: Categorical variables",
R+                    output = 'flextable')
R>
R> # additionally we want to change the font, fontsize and spacing
R> library("gdtools")
R>
R> library(dplyr)
R> tab_cat <- tab_cat %>%
R+   font(fontname="Times New Roman", part="header") %>%
R+   font(fontname="Times New Roman", j=1:4) %>%
R+   fontsize(size=12, part="header") %>%
R+   fontsize(size=10, j=1:4) %>%
R+   line_spacing(space = 0.3, part = "all")
R>
R> tab_cat
```

		N	%
Education_cat	low	14	29.8
	middle	22	46.8
	high	11	23.4
Infant.Mortality_cat	low	12	25.5
	middle	23	48.9
	high	12	25.5

Table 8.3: Summary: Categorical variables

8.3. Regression table

Table 8.4 shows the output for a regression table. Make sure you name all your models and explicitly refer to model names (M1, M2 etc.) in the text.

```
R> library(modelsummary)
R> M1 <- lm(Fertility ~ Education + Agriculture, data = swiss)
R> M2 <- lm(Fertility ~ Education + Catholic, data = swiss)
R> M3 <- lm(Fertility ~ Education + Infant.Mortality + Agriculture, data = swiss)
R> models <- list("M1" = M1, "M2" =  M2, "M3" = M3)
R>
R> tab_reg <-modelsummary(models, title = 'A regression table.', output = 'flextable')
R>
R> # additionally we want to change the font, font size and spacing
R> library("gdtools")
R>
R> tab_reg <- tab_reg %>%
R+   font(fontname="Times New Roman", part="header") %>%
R+   font(fontname="Times New Roman", j=1:4) %>%
R+   fontsize(size=10, part="header") %>%
R+   fontsize(size=10, j=1:4) %>%
R+   line_spacing(space = 0.3, part = "all")
R>
R> tab_reg
```

	M1	M2	M3
(Intercept)	84.080	74.234	51.101
	(5.782)	(2.352)	(10.995)
Education	-0.963	-0.788	-0.857
	(0.189)	(0.129)	(0.173)
Agriculture	-0.066		-0.026
	(0.080)		(0.073)
Catholic		0.111	
		(0.030)	
Infant.Mortality			1.493
			(0.439)
Num.Obs.	47	47	47
R2	0.449	0.575	0.566
R2 Adj.	0.424	0.555	0.536
AIC	349.7	337.6	340.5
BIC	357.1	345.0	349.7
Log.Lik.	-170.846	-164.782	-165.243
F	17.945	29.705	18.699

Table 8.4: A regression table.

9. Inline code & results

Reproduction reaches new heights when you work with inline code. For instance, you can automatize the display of certain coefficients within the text. An example is to include estimates, e.g., the coefficient of `dist` of the model we ran above. `` `r round(coef(M1)[2], 2)` `` will insert the coefficient as follows: -0.96. Or `` `r 3 + 7` `` will insert a 10 in the text. Inline code/results that depend on earlier objects in your document will automatically be updated once you change those objects. For instance, imagine a reviewer asks you to omit certain observations from your sample. You can simply do so in the beginning of your code and push play subsequently.. at time you might have to set `cache = FALSE` at the beginning so that all the code chunks are rerun. Researchers often avoid referring to results in-text etc. because you easily forget to change them when revising a

manuscript. At the same it can make an article much more informative and easier to read, e.g., if you discuss a coefficient in the text you can directly show it in the section in which you discuss it. Inline code allows you to do just that. R Markdown allows you to that do so in a reproducible and automatized manner.

10. Graphs

10.1. R base graphs

Inserting figures can be slightly more complicated. Ideally, we would produce and insert them directly in the `.rmd` file. It's relatively simple to insert R base graphs as you can see in Figure 10.1.
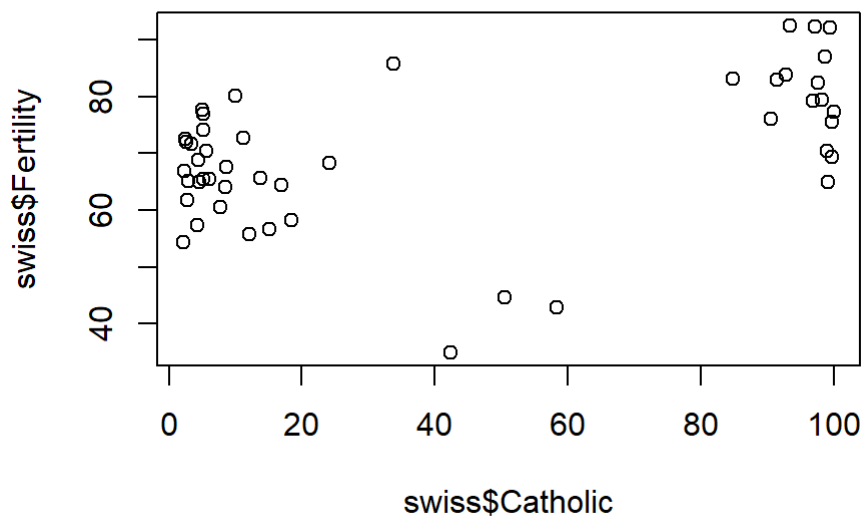
```
R> plot(swiss$Catholic, swiss$Fertility)
```


Figure 10.1: Scatterplot of Speed and Distance

But it turns out that it doesn't always work so well.

10.2. ggplot2 graphs

Same is true for ggplot2 as you can see in Figure 10.2.

```
R> library(ggplot2)
R> ggplot(swiss, aes(x=Catholic, y=Fertility, shape=Education_cat)) + geom_point() +
R+    labs(x="Agriculture", y = "Fertility",
R+        shape="Education") + theme_classic()
```
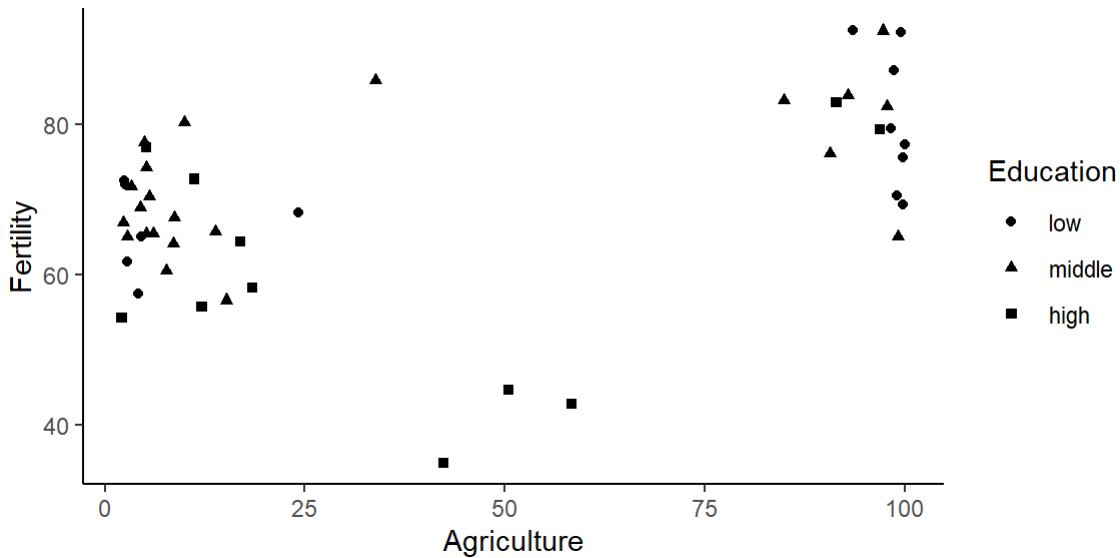
Figure 10.2: Miles per gallon according to the weight

11. Compiling the document

To view your paper, pagedown requires a web server (since it is based on paged.js)[5]. By compiling a document, R Studio will display your HTML page through a local web server, i.e., paged.js will work in RStudio Viewer.

There are several options, depending on your intention:

- click on the `Knit` button in R Studio which by default will provide a HTML document in the RStudio viewer pane (the HTML will be stored in your working directory)

- use pagedown's `chrome_print` function in the YAML (uncomment line #24 of this `Rmd` file) if additionally you want your HTML based web page to be printed to be PDF (the PDF will be stored in your working directory)

- to "live"-preview your pages do not click on the `Knit` button but use the **xaringan** (Xie 2021) RStudio add-in *Infinite Moon Reader*. You can simply call the function `xaringan::inf_mr()` (within your console). This will launch a local web server via the **servr** package (Xie 2021a) and display your pages in the RStudio viewer. Each time you save your document (*Ctrl+S*) xaringan updates your pages in the viewer.

- If you use the option `self_contained: false` (see line #21 of this `Rmd` file) (change to true for a self-contained document, but it'll be a litte slower for Pandoc to render), don't click on the `Knit` button in RStudio. Use instead the **xaringan** (Xie 2021) RStudio add-in *Infinite Moon Reader*.

12. Good practices for reproducibility

Every researcher has his own optimized setup. Currently we would recommend the following:

- Keep all files of your project (that matter for producing the PDF) in one folder without subfolders. You can zip and directly upload that folder to the Harvard dataverse.[6]

---

[5]open-source library to paginate content in the browser

- Make sure that filenames have a logic to them.
  - Main file with text/code: "paper.rmd", "report.rmd"
  - Data files: "data_xxxxxx.*"
  - Image files: "fig_xxxxxx.*"
  - Tables files: "table_xxxx.*"
  - etc.
  - Ideally, your filenames will correspond to the names in the paper. For instance, Figure 1 in the paper may have a corresponding file called `fig_1_xxxxx.pdf`.
- Use the document outline in R studio (Ctrl + Shift + O) when you work with R Markdown.
- Name rchunks according to what they do or produce:
  - "`fig-...`" for chunks producing figures
  - "`table-...`" for chunks producing tables
  - "`model-...`" for chunks producing model estimates
  - "`import-...`" for chunks importing data
  - "`recoding-...`" for chunks in which data is recoded
- Use "really" informative variable names:
  - Q: What do you think does the variable *trstep* measure? It actually measures trust in the European parliament.
    - How could we call this variable instead? Yes, `trust.european.parliament` which is longer but will probably be understood by another researcher in 50 years.
  - If your setup is truly reproducible you will probably re-use the variable names that you generate as variable names in the tables you produce. Hence, there is an incentive to use good names.
- Use unique identifiers in the final R Markdown document paper.rmd that you upload:
  - Think of someone who wants to produce Figure 1/Model 1/Table 1 in your paper but doesn't find it in your code…
    - Name the chunks "fig-1", "fig-2" as the are named in the published paper.
    - Name the chunks that produce tables "table-1", "table-2" etc. as they are named in the published paper.
    - Name your statistical models in your R code "M1", "M2" as they are named in the published paper.

13. Additional tricks for publishing

- Make your script anonymous
  - Simply put a `<!-- ... -->` around any identifying information, e.g., author names, title footnote etc.
- Counting words
  - Use adobe acrobat (commerical software) to convert your file to a word file. Then open in word and delete all the parts that shouldn't go into the word count. The word count is displayed in the lower right.
  - Use an one of the online services to count your words (search for "pdf word count")
- Appendix: You can change the numbering format for the appendix in the rmd file
  - What is still not possible in this document is to automatically have separate reference sections for paper and appendix.
- Journals may require you to use their tex style: Sometimes you can simply use their template in your rmarkdown file. See here for a PLOS one example.

14. Citation styles

---

[6]Another good folder setup would be to store all files needed as input files for the R Markdown manuscript in a subfolder called "input" and all output files that are produced apart from paper.html and paper.pdf in a subfolder called "output".

If your study needs to follow a particular citation style, you can set the corresponding style in the header of your `.rmd` document. To do so you have to download the corresponding `.csl` file.

In the present document we use the style of the American Sociological Association and set it in the preamble with `csl: american-sociological-association.csl`. However, you also need to download the respective `.csl` file from the following github page: `https://github.com/citation-style-language/styles` and copy it into your working directory for it to work.

The github directory contains a wide variety of citation style files depending on what discipline you work in.

References

Bauer, Paul. 2018. "Writing a Reproducible Paper in R Markdown." *Open Science Framework Preprint* 1–14.

Kirsop, Barbara and Leslie Chan. 2005. "Transforming Access to Research Literature for Developing Countries." *Serials Review* 31(4):246–55.

R Core Team. 2017. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

RStudio Team. 2015. *RStudio: Integrated Development Environment for R*. Boston, MA: RStudio, Inc.

Xie, Yihui. 2014. "Knitr: A Comprehensive Tool for Reproducible Research in R." in *Implementing reproducible computational research*, edited by V. Stodden, F. Leisch, and R. D. Peng. Chapman; Hall/CRC.

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC.

Xie, Yihui. 2016. *Bookdown: Authoring Books and Technical Documents with R Markdown*. Boca Raton, Florida: Chapman; Hall/CRC.

Xie, Yihui. 2017. *Bookdown: Authoring Books and Technical Documents with R Markdown*.

Xie, Yihui. 2018. *Knitr: A General-Purpose Package for Dynamic Report Generation in R*.

Xie, Yihui. 2021. *Xaringan: Presentation Ninja*.

Xie, Yihui and Romain Lesur. 2021. "Pagedown: Create Paged HTML Documents for Printing from R Markdown."

Xie, Yihui, Romain Lesur, Brent Thorne, and Xianying Tan. 2021. *Pagedown: Paginate the Html Output of R Markdown with Css for Print*.

Zhu, Hao. 2017. *KableExtra: Construct Complex Table with 'Kable' and Pipe Syntax*.

Online appendix

14.1. Attach R session info in appendix

Since R and R packages are constantly evolving you might want to add the R session info that contains information on the R version as well as the packages that are loaded.

```
R version 4.1.0 (2021-05-18)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 17763)

Matrix products: default

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ggplot2_3.3.4     dplyr_1.0.6      gdtools_0.2.3     flextable_0.6.6
[5] modelsummary_0.8.0 kableExtra_1.3.4  knitr_1.33

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.6        svglite_2.0.0    tidyr_1.1.3       ps_1.6.0
 [5] assertthat_0.2.1  digest_0.6.27    utf8_1.2.1        R6_2.5.0
 [9] backports_1.2.1   evaluate_0.14    httr_1.4.2        highr_0.9
[13] pillar_1.6.1      rlang_0.4.11     uuid_0.1-4        rstudioapi_0.13
[17] data.table_1.14.0 jquerylib_0.1.4  checkmate_2.0.0   rmarkdown_2.9
[21] labeling_0.4.2    webshot_0.5.2    servr_0.22        stringr_1.4.0
[25] munsell_0.5.0     broom_0.7.7      compiler_4.1.0    httpuv_1.6.1
[29] xfun_0.24         pkgconfig_2.0.3  systemfonts_1.0.2 base64enc_0.1-3
[33] htmltools_0.5.1.1 websocket_1.4.0  tidyselect_1.1.1  tibble_3.1.2
[37] bookdown_0.22     fansi_0.5.0      viridisLite_0.4.0 crayon_1.4.1
[41] withr_2.4.2       later_1.2.0      tables_0.9.6      grid_4.1.0
[45] jsonlite_1.7.2    gtable_0.3.0     lifecycle_1.0.0   DBI_1.1.1
[49] magrittr_2.0.1    scales_1.1.1     zip_2.2.0         stringi_1.6.1
[53] farver_2.1.0      promises_1.2.0.1 xml2_1.3.2        bslib_0.2.5.1
[57] ellipsis_0.3.2    generics_0.1.0   vctrs_0.3.8       tools_4.1.0
[61] glue_1.4.2        officer_0.3.18   purrr_0.3.4       processx_3.5.2
[65] yaml_2.2.1        colorspace_2.0-1 rvest_1.0.0       pagedown_0.14
[69] sass_0.4.0
```

14.2. All the code in the paper

To simply attach all the code you used in the PDF file in the appendix see the R chunk in the underlying `.rmd` file:

```
R> knitr::opts_chunk$set(cache = FALSE)
R> # Use chache = TRUE if you want to speed up compilation
R> # A function to allow for showing some of the inline code
R> rinline <- function(code){
R+   html <- '<code  class="r">``` `r CODE` ```</code>'
R+   sub("CODE", code, html)
R+ }
R> remotes::install_github('rstudio/pagedown')
R> install.packages(c("rmarkdown", "knitr", "kableExtra",
R+                 "stargazer", "modelsummary", "knitr"))
R> cat(paste("#", capture.output(sessionInfo()), "\n", collapse =""))
R>   # or use message() instead of cat()
R> x <- 1:10
R> x
R> data <- read.csv("data.csv")
R> head(data)
R> dput(data[1:5,]) # here we only take a subset
R> data <- structure(list(Fertility = c(80.2, 83.1, 92.5, 85.8, 76.9), Agriculture = c(17,
R+ 45.1, 39.7, 36.5, 43.5), Examination = c(15L, 6L, 5L, 12L, 17L
R+ ), Education = c(12L, 9L, 5L, 7L, 15L), Catholic = c(9.96, 84.84,
R+ 93.4, 33.77, 5.16), Infant.Mortality = c(22.2, 22.2, 20.2, 20.3,
R+ 20.6)), class = "data.frame", row.names = c(NA, -5L))
R> library(knitr)
R> library(kableExtra)
R>
```

```
R> kable(swiss[1:10,], row.names = TRUE,
R+      caption = 'Table with kable() and kablestyling()',
R+      format = "html", booktabs = T) %>%
R+        kable_styling(full_width = T,
R+                      latex_options = c("striped",
R+                                        "scale_down",
R+                                        "HOLD_position"),
R+                      font_size = 10)
R>
R>
R> library(modelsummary)
R> datasummary_skim(swiss,
R+                  type="numeric",
R+                  histogram=T,
R+                  title = "Summary: Numeric variables")
R> # Create categorical variables
R> swiss$Education_cat <- cut(swiss$Education,
R+                  breaks=c(-Inf, 6, 12, Inf),
R+                  labels=c("low","middle","high"))
R> swiss$Infant.Mortality_cat <- cut(swiss$Infant.Mortality,
R+                  breaks=c(-Inf, 18.15, 21.70, Inf),
R+                  labels=c("low","middle","high"))
R>
R> library(flextable)
R> tab_cat <- datasummary_skim(swiss,
R+                             type="categorical",
R+                             title = "Summary: Categorical variables",
R+                             output = 'flextable')
R>
R> # additionally we want to change the font, fontsize and spacing
R> library("gdtools")
R>
R> library(dplyr)
R> tab_cat <- tab_cat %>%
R+   font(fontname="Times New Roman", part="header") %>%
R+   font(fontname="Times New Roman", j=1:4) %>%
R+   fontsize(size=12, part="header") %>%
R+   fontsize(size=10, j=1:4) %>%
R+   line_spacing(space = 0.3, part = "all")
R>
R> tab_cat
R> library(modelsummary)
R> M1 <- lm(Fertility ~ Education + Agriculture, data = swiss)
R> M2 <- lm(Fertility ~ Education + Catholic, data = swiss)
R> M3 <- lm(Fertility ~ Education + Infant.Mortality + Agriculture, data = swiss)
R> models <- list("M1" = M1, "M2" =  M2, "M3" = M3)
R>
R> tab_reg <-modelsummary(models, title = 'A regression table.', output = 'flextable')
R>
R> # additionally we want to change the font, font size and spacing
R> library("gdtools")
R>
R> tab_reg <- tab_reg %>%
R+   font(fontname="Times New Roman", part="header") %>%
R+   font(fontname="Times New Roman", j=1:4) %>%
R+   fontsize(size=10, part="header") %>%
R+   fontsize(size=10, j=1:4) %>%
R+   line_spacing(space = 0.3, part = "all")
R>
R> tab_reg
R>
R> plot(swiss$Catholic, swiss$Fertility)
R> library(ggplot2)
R> ggplot(swiss, aes(x=Catholic, y=Fertility, shape=Education_cat)) + geom_point() +
R+   labs(x="Agriculture", y = "Fertility",
R+        shape="Education") + theme_classic()
R> print(sessionInfo(), local = FALSE)
```