pagedown: Create Paged HTML Documents for Printing from R Markdown

A Less Traveled Road to PDF and Printing

Yihui Xie and Romain Lesur

2019-03-01

Contents

List of Tables

List of Figures

1 Introduction

When talking about PDF and printing, we often think of tools like LaTeX and Microsoft Word. When talking about HTML and CSS, we may have never imagined their possible off-screen use such as printing to PDF.

Can we print a book with HTML and CSS? W3C published the first working draft (https://www.w3.org/1999/06/WD-css3-page-19990623) on "Paged Media Properties for CSS(3)", which was last updated in 2013. Although the working draft has been there for nearly two decades, it is still not common to see authors write or print books with HTML and CSS. The main reason is that the W3C specs are still in the draft mode, so most web browsers have not really implemented them.

HTML and CSS still cannot beat other dominating tools like Word or LaTeX when it comes to typesetting content under the constraint of "pages". You may be disappointed by a lot of typesetting details on a paged HTML page. However, HTML and CSS can be extremely powerful and flexible in other aspects, especially when combined with the power of JavaScript. By the way, HTML works almost anywhere because it only requires a web browser.

Although most web browsers have not implemented the W3C specs for Paged Media, a JavaScript polyfill library named "paged.js" (https://www.pagedmedia.org/paged-js/) is currently being developed to fill the gap. This library

is still experimental and has many rough edges, but looks promising enough to us, so we created an R package **pagedown** (Xie and Lesur 2019) based on this JavaScript library to paginate the HTML output of R Markdown documents. You can install the package from Github (https://github.com/rstudio/pagedown):

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

To learn more about paged.js and CSS for Paged Media, you may check out the cheatsheet of paged.js (https://www.pagedmedia.org/about-paged-media/pagedjs-cheatsheet/).

The **pagedown** package contains output formats for paged HTML documents, letters, resumes, posters, business cards, and so on. Usually there is an R Markdown template for each output format, which you can access from RStudio's menu `File -> New File -> R Markdown -> From Template`.

2 Paged HTML documents

To create a paged HTML document, you can use the output format `pagedown::html_paged`, e.g.,

```
output:
  pagedown::html_paged:
    toc: true
    number_sections: false
```

2.1 Preview paged HTML documents

This format is based on paged.js. Some other formats in this package are extensions of `html_paged`, such as `html_letter` and `html_resume`. Please note that you need a web server to view the output pages of these formats, because paged.js requires a web server. The web server could be either a local server or a remote one. When you compile an R Markdown document to HTML in RStudio, RStudio will display the HTML page through a local web server, so paged.js will work in RStudio Viewer. However, when you view such pages in a real web browser, you will need a separate web server. The easiest way to preview these HTML pages in a web browser may be through the RStudio addin "Infinite Moon Reader", which requires the **xaringan** package (**???**). Or equivalently, you can call the function `xaringan::inf_mr()`. This will launch a local web server via the **servr** package (Xie 2019).

Please note that the layout of the pages is very sensitive to the zoom level in your browser. Elements on a page are often not zoomed strictly linearly, e.g., as you zoom out, certainly elements may start to collapse into each other. The 100% zoom level usually gives the best result (press `Ctrl + 0` or `Command + 0`). You are strongly recommended to use this level when printing the page to PDF.

2.2 The CSS overriding mechanism

We have provided a few default CSS stylesheets in this output format:

```
output:
  pagedown::html_paged:
    css: ["default-fonts", "default-
page", "default"]
```

To find the actual CSS files behind these names, use `pagedown:::list_css()`. For example, `default-fonts` means the file `resources/css/default-fonts.css` in the installation directory of **pagedown**. The stylesheet `default-fonts` defines the typefaces of the document, `default-page` defines some page properties (such as the page size, running headers and footers, and rules for page breaks), and `default` defines the style of various elements (such as the table of contents).

If you do not like any of these default stylesheets, you can use a subset of them, or override certain CSS rules.

For example, if you do not like the default typeface, you may create a CSS file `my-fonts.css` (assuming it is under the same directory of your Rmd file):

```
body {
  font-family: "Palatino Linotype", "Book
Antiqua", Palatino, serif;
}
```

Then include this CSS file via the `css` option:

```
output:
  pagedown::html_paged:
    css: ["my-fonts.css", "default-page",
"default"]
```

Note that this overriding mechanism also works for other output formats in **pagedown**.

2.3 Print to PDF

You can print HTML pages to PDF using Google Chrome or Chromium using the menu "Print" or by pressing `Ctrl + P` (or `Command + P` on macOS). Remember to allow background graphics to be printed. You may also consider using the function `pagedown::chrome_print()`, but currently it is experimental and you may get an incomplete PDF, so we recommend that you open the page in a real browser and print to PDF there instead.

3 Applications

3.1 Resume

Currently **pagedown** has one resume format: `pagedown::html_resume`. See https://pagedown.rbind.io/html-resume/ for an example.

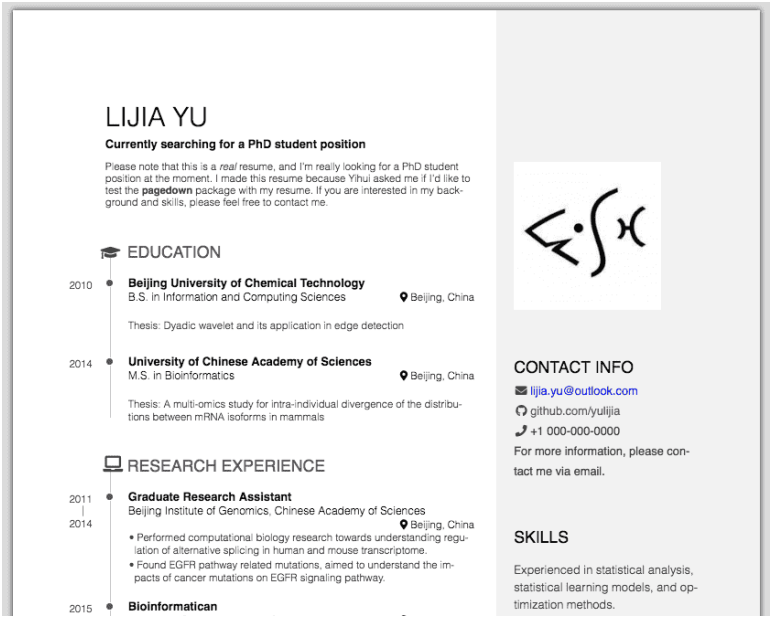

Figure 3.1: The HTML resume in pagedown.

The R Markdown source document should contain two parts titled “Aside” (for the sidebar) and “Main” (for the main body), respectively. Each part can contain any

numbers of sections (e.g., education background and working experience). Each section can contain any numbers of subsections, and you can write more content in a subsection.[1] Below is a quick example:

```
---
title: "Someone's resume"
author: Your Name
output: pagedown::html_resume
---


Aside
==================================

Picture, etc.

Contact info
----------------------------------

Email, phone number, ...

Skills
----------------------------------

- One
- Two
- Three

Disclaimer
----------------------------------

A footer in the sidebar.
```

[1] In case you are not very familiar with the Markdown syntax for section headings, a series of = under a heading is equivalent to # before a heading, and a series of - is equivalent to ##. See Pandoc's manual for details: https://pandoc.org/MANUAL.html#headers.

```
Main
====================================

Your Name {#title}
------------------------------------

Arbitrary content.

Education {data-icon=graduation-cap}
------------------------------------

### University One

Title

Location

Year

Arbitrary content

### University Two

...
```

The “Aside” part will be displayed in the right sidebar. This part can contain arbitrary content (not necessarily structured). For example, you can include a picture in the beginning. The “Disclaimer” section will be placed at the bottom of the first page. All icons for this resume template are from Font Awesome (https://fontawesome.com). For example, `<i class="fa fa-envelope"></i>` will generate an envelope icon. You can look up all icon names on the Font Awesome website if you want to use other icons.

For the “Main” part, all sections must follow a specific structure except the first section. The first section

usually shows your name and information that you want to highlight. For the rest of sections, they should contain a title and a number of subsections. You can specify an icon for a section title via the attribute `data-icon`, e.g., `{data-icon=graduation-cap}` means an icon of a graduation cap (which can be used as a symbol for education). For each subsection, it should contain a title, followed by at least three paragraphs:

- The first paragraph is a brief description of the subsection.
- The second paragraph is the location.
- The third paragraph is the time period. If this subsection has both a starting and ending time, separate them by a dash, e.g., `2014 - 2015` or `2014/07/27 - 2015/07/23`.

The description, location, and time period can each be `N/A` if the relevant information is not available.

You can write artibrary content after the third paragraph (e.g., more paragraphs, bullet lists, and so on). If you want to write content in two columns, you may use a "concise" block, e.g.,

```
::: concise
- Taught R language to beginners.
- Wrote Shiny app demos.
- Converted statistical tutorials from
SPSS to R language.
:::
```

If you want to write a side note, use an “aside” block, e.g.,

```
Section
---------------

### Subsection

Title

Location

Year

More info

::: aside
Some notes in the sidebar.
:::
```

Figure 3.2: An example of side notes.

There is a caveat about page breaks. By default, we allow page breaks within a subsection. Sometimes this may lead to odd output like the example in Figure 3.3. The first bullet should not be split into two columns, and the rest of bullets should have a larger left margin.

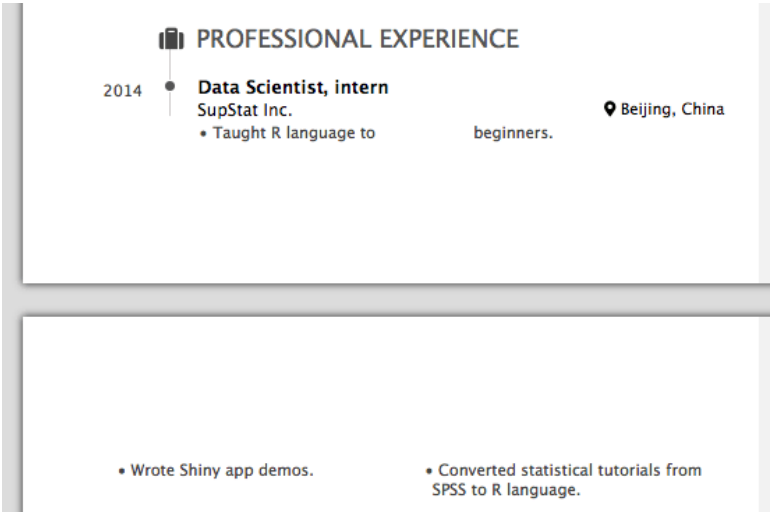

Figure 3.3: Allow page breaks in a subsection.

If you want to avoid layout problems like this, you may disallow page breaks via CSS:

```
.blocks {
  break-inside: avoid;
}
```

However, this may lead to a new issue demonstrated in Figure 3.4: there may be a large bottom margin on the previous page. At this point, you may start to miss your old friends, Word and LaTeX.

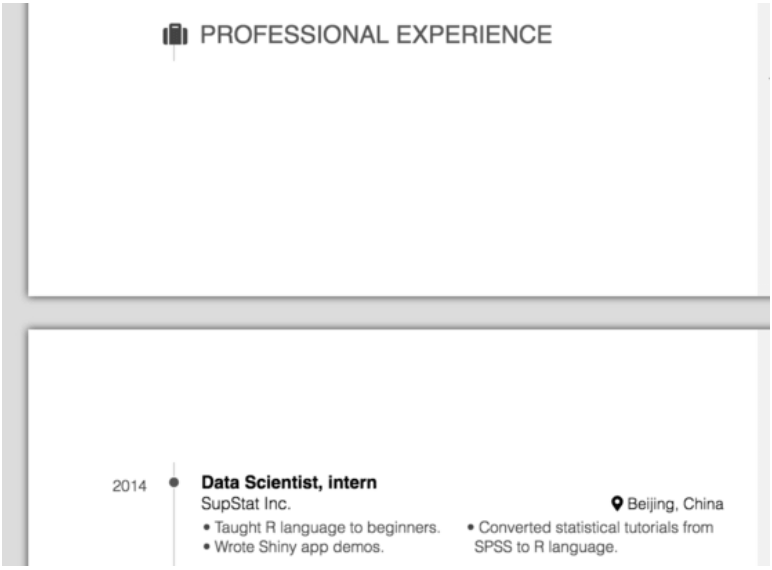

Figure 3.4: Do not allow page breaks in a subsection.

3.2 Poster

You can create a poster with the output format `pagedown::poster_relaxed` or `pagedown::poster_jacobs`. See https://pagedown.rbind.io/poster-relaxed/ and https://pagedown.rbind.io/poster-jacobs/ for examples.

We do not have time to document the poster formats yet, but here is a caveat: the layout of poster sections is hardcoded in CSS styleseets, which means you cannot add/delete sections unless you know CSS (in particular, CSS Grid Layout). If you are interested in learning CSS Grid Layout, you may take a look at the CSS of `poster_jacobs` (https://github.com/rstudio/pagedown/blob/master/inst/resources/css/poster-jacobs.css).

3.2.1 The ReLaXed style

3.2.2 The Jacobs University style

3.3 Business card

To create a simple business card, you can use the `pagedown::business_card` format. See https://pagedown.rbind.io/business-card/ for an example.

3.3.1 Single card

A single business card can be created with the following R Markdown file (this file contains only a YAML header).

```
---
name: Jane Doe
title: Miss Nobody
phone: "+1 123-456-7890"
email: "jane.doe@example.com"
url: www.example.com
address: |
  2020 South Street
  Sunshine, CA 90000
logo: "logo.png"
output: pagedown::business_card
---
```

You can repeat the card on multiple pages using the `repeat` variable. The following example produces as many pages as cards (12).

```
---
name: Jane Doe
title: Miss Nobody
phone: "+1 123-456-7890"
email: "jane.doe@example.com"
url: www.example.com
address: |
  2020 South Street
  Sunshine, CA 90000
logo: "logo.png"
repeat: 12
output: pagedown::business_card
---
```

In order to print the cards, you may prefer a layout with several cards on the same page: you can adjust the paper size with the `paperwidth` and `paperheight` variables and define a grid layout with the `cols` and `rows` variables (you may test some combinations of these parameters to find the most appropriate one).

```
---
name: Jane Doe
title: Miss Nobody
phone: "+1 123-456-7890"
email: "jane.doe@example.com"
url: www.example.com
address: |
  2020 South Street
  Sunshine, CA 90000
logo: "logo.png"
repeat: 12
paperwidth: 8.5in
paperheight: 11in
cols: 4
rows: 3
output: pagedown::business_card
---
```

You also can use markdown to define a card. Be aware to use the `slot` attributes as follows.

```
---
logo: "logo.png"
paperwidth: 8.5in
paperheight: 11in
cols: 4
rows: 3
output: pagedown::business_card
---

::::: {.wrapper data-repeat="12"}
[Jane Doe]{slot="name"}
[Miss Nobody]{slot="title"}
[+1 123-456-7890]{slot="phone"}
[jane.doe@example.com]{slot="email"}
[www.example.com]{slot="url"}

::: {.address slot="address"}
2020 South Street
Sunshine, CA 90000
:::
:::::
```

3.3.2 Different cards with shared informations

You can produce business cards for members of an organization sharing some informations (address, website...).
Common informations are declared as top level variables in the YAML header. Custom cards are defined using the `person` variable: each `key: value` pair of a `person` block overrides the corresponding top level pair.

```
---
phone: "+1 123-456-7890"
url: www.example.com
address: |
  2020 South Street
  Sunshine, CA 90000
logo: "logo.png"
person:
  - name: Jane Doe
    title: Miss Nobody
    email: "jane.doe@example.com"
    repeat: 6
  - name: John Doe
    title: Mister Nobody
    phone: "+1 777-777-7777" # overrides
the default phone
    email: "john.doe@example.com"
    repeat: 6
paperwidth: 8.5in
paperheight: 11in
cols: 4
rows: 3
output: pagedown::business_card
---
```

If you prefer, you can use markdown to create a card as follows.

```
---
name: Jane Doe
title: Miss Nobody
phone: "+1 123-456-7890"
email: "jane.doe@example.com"
url: www.example.com
address: |
  2020 South Street
  Sunshine, CA 90000
logo: "logo.png"
repeat: 6
```

```
paperwidth: 8.5in
paperheight: 11in
cols: 4
rows: 3
output: pagedown::business_card
---

::: {.wrapper data-repeat="6"}
[John Doe]{slot="name"}
[Mister Nobody]{slot="title"}
[+1 777-777-7777]{slot="phone"}
[john.doe@example.com]{slot="email"}
[my.domain.com]{slot="url"}
:::
```

3.3.3 Styling business cards

3.3.3.1 Fonts

You can change the text font with the `mainfont` and/or `googlefonts` top level YAML variables:

- `mainfont` will use the local font installed on your computer, e.g.

```yaml
---
name: Jane Doe
title: Miss Nobody
phone: "+1 123-456-7890"
email: "jane.doe@example.com"
url: www.example.com
address: |
  2020 South Street
  Sunshine, CA 90000
logo: "logo.png"
mainfont: Arial
output: pagedown::business_card
---
```

- `googlefonts` to use fonts from https://fonts.google.com, e.g.

```
---
name: Jane Doe
title: Miss Nobody
phone: "+1 123-456-7890"
email: "jane.doe@example.com"
url: www.example.com
address: |
  2020 South Street
  Sunshine, CA 90000
logo: "logo.png"
googlefonts: ["Roboto Condensed",
"Raleway"]
output: pagedown::business_card
---
```

3.3.3.2 Card sizing

You can modify the card size with the `cardwidth` and `cardheight` variables. You can get a landscape card with:

```
---
name: Jane Doe
title: Miss Nobody
phone: "+1 123-456-7890"
email: "jane.doe@example.com"
url: www.example.com
address: |
  2020 South Street
  Sunshine, CA 90000
logo: "logo.png"
cardwidth: 3.5in
cardheight: 2in
```

```
output: pagedown::business_card
---
```

If you render this card, you will see that the default style does not suit well with a landscape card. Read the next section to find an example of a landscape card with a better style.

3.3.3.3 CSS

Finally, you can modify the style of the card using CSS rules.
The markup of a card can be represented as follows[2]:

```
<div class="wrapper" data-repeat="1">
  <img class="logo" src="logo.png"
alt="Logo" />
  <div class="me">
    <div class="name"><span>Jane
Doe</span></div>
    <div class="title"><span>Miss
Nobody</span></div>
    <div class="coordinates">
      <p class="phone"><span>+1 123-456-
7890</span></p>
      <p class="contact-email">
<span>jane.doe@example.com</span></p>
      <p class="website">
<span>www.example.com</span></p>
      <div class="address">2020 South
Street Sunshine, CA 90000</div>
    </div>
  </div>
</div>
```

[2]This is technically incorrect since the card template use a shadow DOM.

You can use these built-in classes to style your card with CSS.
A landscape card could be styled like this:

````
---
name: Jane Doe
title: Miss Nobody
phone: "+1 123-456-7890"
email: "jane.doe@example.com"
url: www.example.com
address: |
  2020 South Street
  Sunshine, CA 90000
logo: "logo.png"
cardwidth: 3.5in
cardheight: 2in
output: pagedown::business_card
---

```{css}
.logo {
  display: block;
  height: 20%;
  margin-right: .3in;
  padding: .3in 0 0;
  float: right;
}
.name {
  margin-top: .1in;
}
```
````

3.4 Letter

You can write a letter with the `pagedown::html_letter` format. See

https://pagedown.rbind.io/html-letter/ for an example.

3.5 A Journal of Statistical Software article

You can write an article for the Journal of Statistical Sofware (https://jstatsoft.org/). See https://pagedown.rbind.io/jss-paged/ for an example.

4 Miscellaneous features

4.1 Lists of tables and figures

Lists of tables and/or figures can be inserted in the document using the `lot` and `lof` variables in the YAML metadata. You also can customize their titles with the `lot-title` and `lof-title` variables. For instance:

```
---
title: "A document with lists of tables
and figures"
output: pagedown::html_paged
toc-title: Contents

lot: true
# default: "List of Tables"
lot-title: "Tables"

lof: true
# default: "List of Figures"
lof-title: "Figures"
---
```

4.2 Front matter

By default, the front matter is composed of the cover, the table of contents and the lists of figures and tables if any. The only difference between the front matter pages and the main content pages is the style of the page numbers.

You can add extra sections to the front matter using the `front-matter` class. For instance, if you want to add a preface to the front matter, you need to write:

```
# Preface {.front-matter .unnumbered}
```

4.3 Links

In Markdown, the usual ways to insert links are *automatic* and *inline* links.

4.3.1 Automatic links

Automatic links are created using pointy brackets, e.g., `<https://bookdown.org>`. The full URL is then inserted in the final document https://bookdown.org. This is convenient for a short and meaningful URL.

4.3.2 Inline links

Inline links are useful when you do not want to show the full URL but an alternative text (because URLs are usually long and ugly). In Markdown, the link text is inserted in square brackets and the URL in parentheses, e.g. `[bookdown website](https://bookdown.org)`. On a website, the URL is hidden and replaced by the link text: bookdown website. The user can interactively access the URL by clicking on the link.

When printing a document, we lose interactivity. So we need to show the hidden URLs. By default, the `html_paged` format adds the URLs after the link text in parentheses, for instance `[bookdown website](https://bookdown.org)` is rendered as bookdown website (https://bookdown.org).

You also can use the `links-to-footnotes` top-level YAML parameter: it transforms all the URLs to footnotes. You will get the same result as `bookdown website^[https://bookdown.org]`. To activate the `links-to-footnotes` option, insert `links-to-footnotes: true` in the YAML header. For instance:

```
---
title: "A paged HTML document"
output: pagedown::html_paged
links-to-footnotes: true
---
```

4.4 Footnotes

The default behavior of pagedown is to render notes as endnotes because Paged.js does not natively support footnotes for now. However, we introduced an experimental support for footnotes. You can test it by including `paged-footnotes: true` in the YAML header of your document. If you get any trouble with this experimental feature, please open an issue in the **pagedown** repository on GitHub.

4.5 Custom running headers

Sometimes a section title may be too long to fit the header or footer area. In this case, you can specify a shorter title for the running headers/footers via the attribute `data-short-title` after a title, e.g.,

```
## The actual long long long title {data-
short-title="An alternative title"}
```

4.6 MathJax

The following test comes from http://www.cs.toronto.edu/~yujiali/test/mathjax.html.

Some RBM stuff:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i,j} w_{ij} v_i h_j - \sum_i b_i v_i - \sum_j c_j h_j$$

Multiline equations:

$$p(v_i = 1|\mathbf{h}) = \sigma\left(\sum_j w_{ij} h_j + b_i\right)$$
$$p(h_j = 1|\mathbf{v}) = \sigma\left(\sum_i w_{ij} v_i + c_j\right)$$

Here is an example of an inline expression: $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$.

4.7 Page references

Internal links will be followed by page numbers by default. For example, you can refer to another section using either `[Section Title]` or `[Link Text](#section-id)` (where `section-id` is the ID of the section header).

Do you still remember Paged.js that we mentioned earlier?

4.8 Figures and tables

Table 4.1:

```
knitr::kable(head(iris[, -5]), caption =
'An example table.')
```

Table 4.1: An example table.

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

Bibliography

Xie, Yihui. 2019. *Servr: A Simple Http Server to Serve Static Files or Dynamic Documents*. https://github.com/yihui/servr.

Xie, Yihui, and Romain Lesur. 2019. *Pagedown: Paginate the Html Output of R Markdown with Css for Print*. https://github.com/rstudio/pagedown.