Formation publications reproductibles avec
RMarkdown

2 - Les principaux templates pour créer des publications

Maël THEULIERE - Marouane ZELLOU

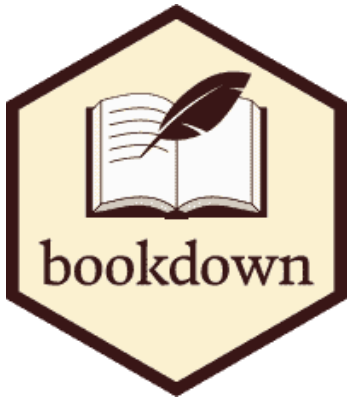
Introduction

Derrière Rmarkdown aujourd'hui existe tout un ecosystème permettant de mixer des traitements réalisés en R et du texte dans des formats très différents, que ce soit en matière de technologie (html, pdf, ...) ou de type de rendu (diaporama, rapport, livre, note, cv, ...).

Tous ces formats disposent de gabarits prédéfinis permettant de structurer rapidement vos projets.

Déroulé

Bookdown



Pagedown

Officer

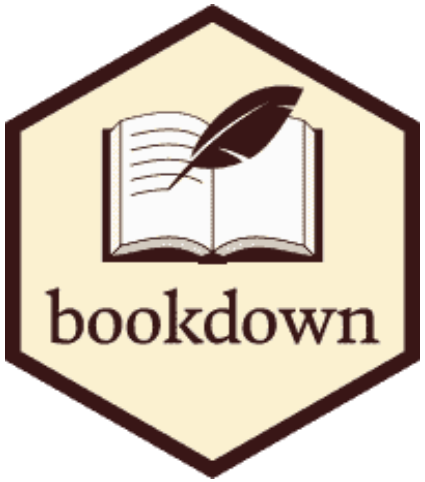
Bookdown

Bookdown

Le format Bookdown permet de gérer des projets Rmarkdown plus large qu'un simple document de quelques pages.

Par ailleurs, le format bookdown apporte plusieurs améliorations :

- Les livres et les rapports peuvent être créés à partir de plusieurs fichiers R Markdown.
- Des fonctionnalités de mise en forme supplémentaires sont ajoutées, telles que les références croisées et la numérotation des figures, des équations et des tableaux.
- Les documents peuvent être facilement exportés dans une gamme de formats adaptés à la publication (PDF, epub, HTML).



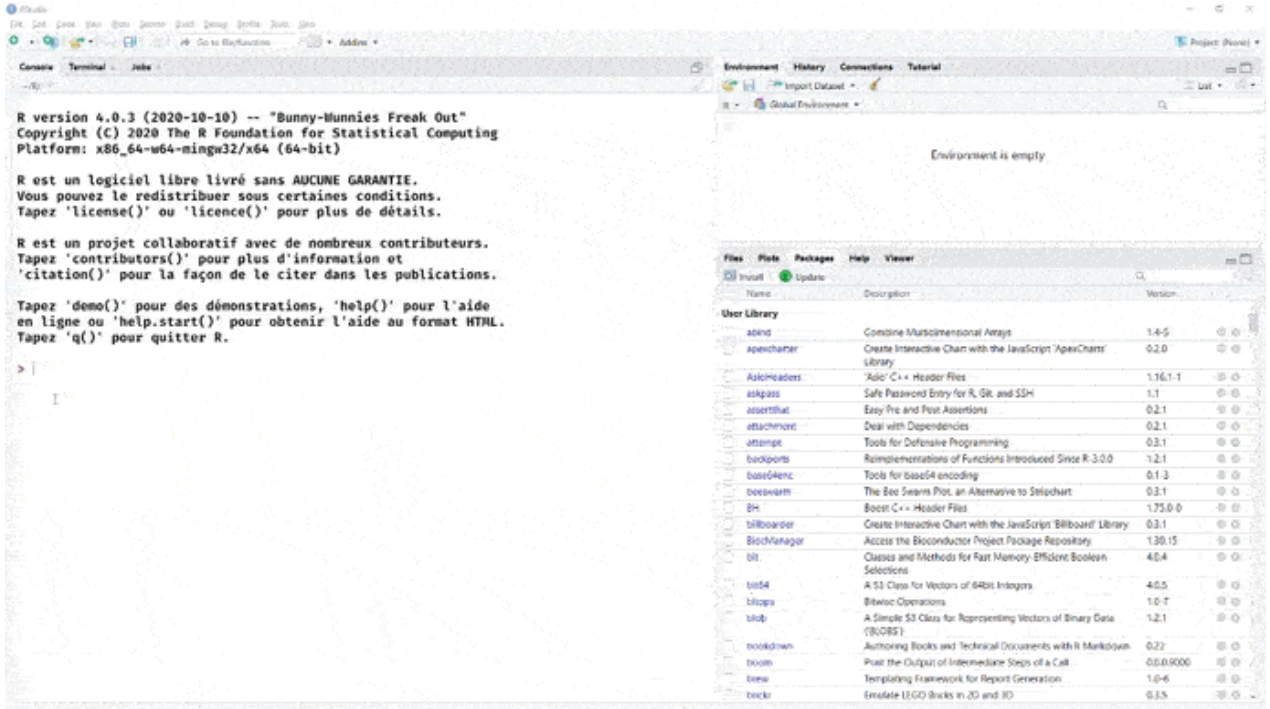
Comment créer son premier bookdown

Installer bookdown depuis le CRAN

```
install.packages('bookdown')
```

- Depuis Rstudio, cela vous apporte un nouveau type de projet, accessible depuis `File -> New Project -> New Directory -> Book Project using bookdown`.

- Une fois celui ci créé, vous pouvez cliquer sur `Build book` dans l'onglet `Build` de Rstudio. Cela vous compilera le document en html, qui sera accessible dans le répertoire du projet et visible par défaut dans le viewer.



La structure du projet

Quand vous créz un projet bookdown, vous avez dans votre projet automatiquement les fichiers suivants :

- `index.Rmd`, le seul fichier Rmarkdown qui contient comme usuellement un yaml en entête. C'est le premier chapitre de votre livre.
- `01-intro.Rmd` à `06-references.Rmd` sont les fichiers rmarkdown correspondant aux chapitres de vos livres. La structure classique d'un rmarkdown est d'un document Rmd par chapitre, qui seront ensuite pour la version html votre premier niveau de navigation. Chaque fichier commence par le titre du chapitre.
- `_bookdown.yml` Un fichier de configuration de votre document bookdown.
- `_output.yml` Un fichier de configuration des formats de sortie de votre document (pdf, html...).
- `book.bib` un fichier de bibliographie au format BibTeX.
- `preamble.tex` et `style.css` des fichiers de configuration de l'apparence de votre document pour sa version pdf (réalisé en LaTeX) et sa version html (réalisé en css).

```
directory/
├──  index.Rmd
├── 01-intro.Rmd
├── 02-literature.Rmd
├── 03-method.Rmd
├── 04-application.Rmd
├── 05-summary.Rmd
├── 06-references.Rmd
├── _bookdown.yml
├── _output.yml
├──  book.bib
├──  preamble.tex
├──  README.md
└──  style.css
```

La structure du projet - index.Rmd

La balise yaml du fichier `index.Rmd` contient des options spécifiques au format bookdown sur la gestion de la bibliographie.

```
title: "A Minimal Book Example"
author: "Yihui Xie"
date: "2021-10-08"
site: bookdown::bookdown_site
documentclass: book
bibliography: [book.bib, packages.bib]
biblio-style: apalike
link-citations: yes
description: "This is a minimal example of using the bookdown package to write a book. The output f
```

La structure du projet - _bookdown.yml

Le fichier `_bookdown.yml` permet de spécifier des options de configuration supplémentaires pour construire le livre.

Par exemple :

- changer le nom du fichier (`book_filename`)
- franciser le préfixe devant le numéro du chapitre (`chapter_name`) ou le préfixe des tableau et des graphiques (`fig` et `tab`)
- changer l'ordre de fusion des fichiers (`rmd_files`)

```
book_filename: "mon_premier_bookdown"
delete_merged_file: true
language:
  ui:
    chapter_name: "Chaptitre "
  label:
    fig: "Graphique "
    tab: "Tableau "
rmd_files: ["index.Rmd", "01-intro.Rmd", "05-su
```

La structure du projet - _output.yml

Le fichier `_output.yml` est utilisé pour spécifier les types de format de sortie (pdf, html...) et les options relatives à ces formats.

Pour le format html, c'est là par exemple que vous pouvez spécifier entre autre :

- la feuille de style css à utiliser
- les textes inscrit en haut et en bas du menu de navigation
- les options de partage sur les réseaux sociaux que vous voulez (si vous en voulez, vous pouvez aussi tous les désactiver avec l'option `sharing` ci contre)

```
bookdown::gitbook:
  css: style.css
  config:
    toc:
      before: |
        <li><a href="./">Mon premier bookdown</a
      after: |
        <li>Mon premier bookdown</li>
    download: ["pdf", "epub"]
    sharing: no
    info: no
```

Quelques trucs et à astuces

- Vous pouvez exclure un chapitre de la numérotation, en rajoutant `{-}` devant son nom dans le fichier Rmd.
- Vous pouvez structurer vos chapitres en partie. Pour commencer une partie rajouter `# (PART) Nom de votre partie {-}` devant un chapitre.

Pagedown

Pagedown

Pagedown est une implémentation pour Rmarkdown de paged.js, qui permet de réaliser des documents html paginés.

Paged.js

paged.js est une bibliothèque javascript visant à mettre en oeuvre les propriétés css dédiées au médias paginés du W3C.

Ces spécifications visent à pouvoir réaliser des documents prêt pour l'impression avec les technologies du web (html, css, js).

Ces spécifications sont toujours en draft pour le moment au sein du W3C, donc pas vraiment reconnues par les principaux navigateurs. D'où le besoin de cette bibliothèque javascript pour pouvoir les mettre en oeuvre.

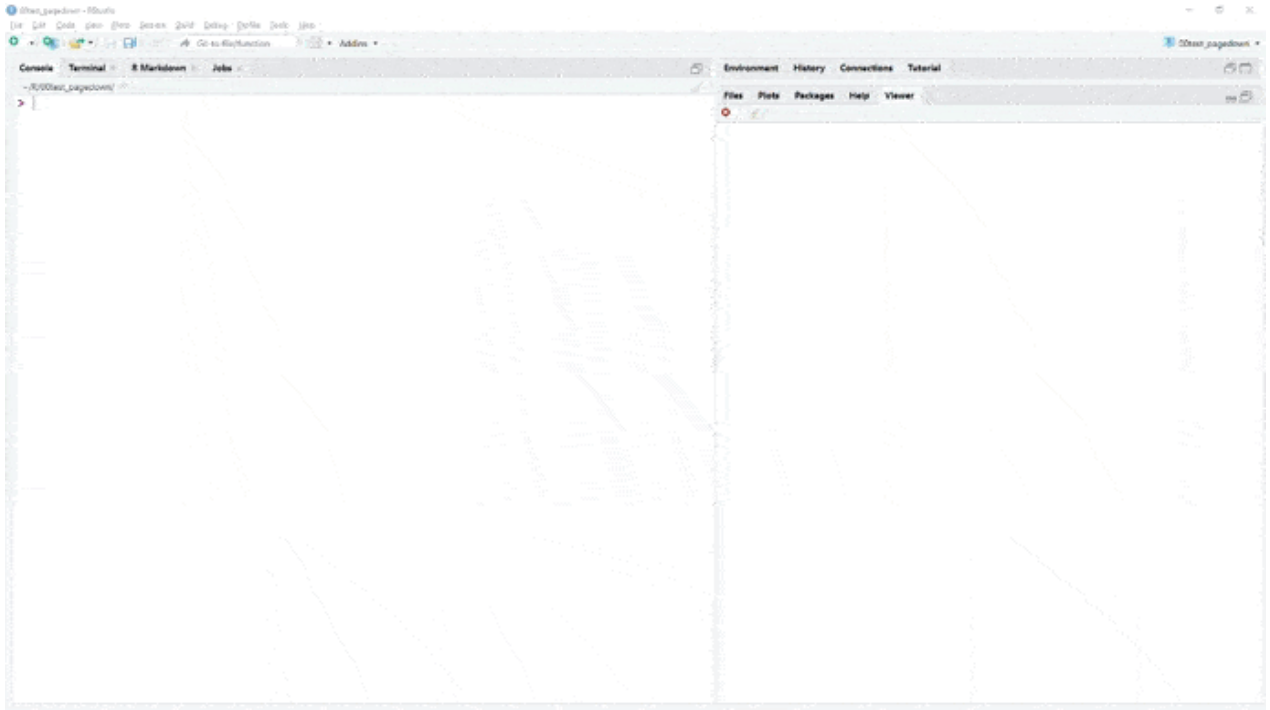
Comment créer son premier pagedown

Pour installer pagedown depuis le CRAN :

```
install.packages('rstudio/pagedown')
```

Depuis Rstudio, cela vous apporte un nouveau type de document rmarkdown, accessible depuis `File -> New File -> Rmarkdown... -> From template -> Paged HTML document`.

Une fois celui-ci créé, vous pouvez cliquer sur knit de l'interface de Rstudio. Cela vous compilera le document par défaut en html, qui sera accessible dans le répertoire du projet et visible par défaut dans le viewer.



La structure du yaml

La structure du yaml est relativement proche d'un document Rmarkdown classique.

A noter toutefois une option utile à retenir, la balise `knit: pagedown::chrome_print` qui vous permet de compiler directement votre document en pdf grâce à la fonction `pagedown::chrome_print()` livrée dans le package.

Cette fonction par ailleurs peut être utilisée pour imprimer en pdf tout document html en pdf ou en format image. Elle utilise la technologie d'impression de google chrome.

```
title: "A Multi-page HTML Document"
author: "Yihui Xie and Romain Lesur"
date: "2021-10-08"
output:
  pagedown::html_paged:
    toc: true
    \# change to true for a self-contained docu
    self_contained: false
\# uncomment this line to produce HTML and PDF
\#knit: pagedown::chrome_print
```

Configurer votre yaml

Voici quelques options du yaml intéressantes à connaître :

```
toc-title: sommaire
```

Pour modifier le nom du sommaire du document.

```
lot: true
lot-title: "Tableaux"
lof: true
lof-title: "Graphiques"
```

Pour ajouter et configurer une liste de tableau et de graphique dans le sommaire. Si vous ne voulez pas que les graphiques et tableaux apparaissent dans le sommaire, vous pouvez utiliser `lot-unlisted: true`.

```
chapter_name: "Chapitre\\ "
```

Pour modifier le préfixe du titre du chapitre. Vous pouvez ensuite insérer un chapitre en utilisant `# Ceci est un titre de chapitre {.chapter}`

Configurer votre yaml

Voici quelques options du yaml intéressantes à connaître.

```
links-to-footnotes: true
```

Cette option transforme automatiquement un lien url dans votre document en note de bas de page. Ainsi `[CRAN](https://cran.r-project.org/)` sera traduit comme `[CRAN]^(https://cran.r-project.org/)`.

```
---
output:
  pagedown::html_paged:
    front_cover: !expr syst
    back_cover: https://www
---
```

Pour rajouter une image pour la première de couverture et la 4ème de couverture. On peut utiliser le prefix !expr pour insérer du code R créant en sortie une image.

Quelques balises importantes à connaître

`.page-break-before` et `.page-break-after` vous permettent d'insérer un saut de page.

```
### Nouveau chapitre après un saut de page {.page-break-
```

`[Chapitre 1]` vous permet de rajouter un lien vers la section `Chapitre 1`.

Modifier le CSS

Apprendre à customiser un document pagedown va vous demander d'investir sur l'apprentissage du CSS en général et de pagedjs en particulier pour construire un template ad hoc.

C'est un investissement en soit, qui pourra vous être utile de la même façon que la gestion de la mise en page d'un document bureautique.

Vous pouvez aussi centraliser cet investissement pour votre équipe sur une ou deux personnes, ou travailler avec votre service web sur vos projets.

Quelques ressources pour apprendre le CSS :

- Le site de la fondation Mozilla
- Le site du W3C

Modifier le CSS

Voici déjà quelques recettes de base pour commencer à modifier votre document.

Première étape : importer les css par défaut de pagedown dans votre document.

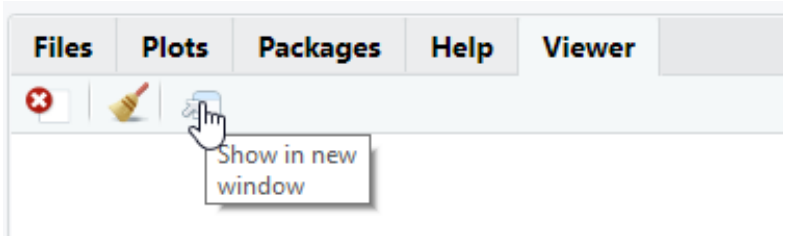
```
files <- c("default-fonts", "default-page", "default")
from <- pagedown:::pkg_resource(paste0("css/", files, ".css"))
to <- c("custom-fonts.css", "custom-page.css", "custom.css")
file.copy(from = from, to = to)
```

Ajouter une référence à ces fichiers dans le yaml :

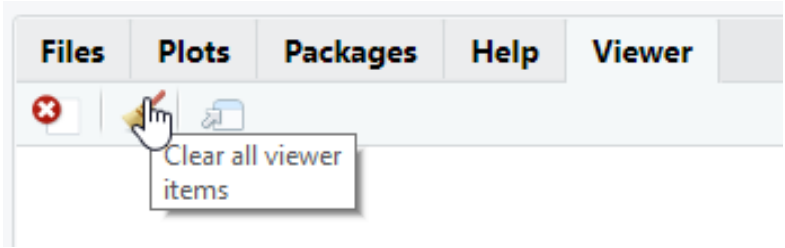
```
output:
  pagedown::html_paged:
    css:
    - custom-fonts.css
    - custom-page.css
    - custom.css
    toc: true
    self_contained: false
```

Modifier le CSS

- Lancer dans la console `xaringan::inf_mr()`, qui permet de compiler un document Rmarkdown en ayant un aperçu live du rendu.
    - Appuyer ensuite sur `Show in new window` pour consulter le document sur votre navigateur par défaut.



    - Puis sur `Clear all viewer item`, pour que vos modifications puisse s'afficher en live.



Modifier le CSS

Vous pouvez commencer à modifier vos fichiers css et voir le résultats immédiatement.

Les 3 fichiers correspondent aux éléments suivants :

- `custom-fonts.css` correspond aux polices de caractères
- `custom-page.css` correspond aux paramètres de la page (format de la page, orientation,...)
- `custom.css` intègre entre autre la façon dont les pages sont affichées à l'écran (couleur de fond, espacement entre les pages...)

Simplifier l'usage de {pagedown} avec {pagedreport}

`{pagedreport}` est un package permettant de simplifier l'appropriation de pagedown, et notamment évite de devoir rentrer tout de suite dans une modification du css.

Il n'est pas encore sur le cran, pour l'installer :

```
remotes::install_github("rfortherestofus/pagedreport)
```

Simplifier l'usage de {pagedown} avec {pagedreport}

`{pagedreport}` vous simplifie l'utilisation de pagedown en proposant trois templates avancés paramétrables via de nouvelles options à intégrer dans le yaml :

- `paged_grid::logo` vous permet d'ajouter le logo de votre institution à votre rapport
- `paged_grid::front_img` et `paged_grid::back_img` vous permettent de choisir deux images pour la première et la quatrième de couverture.
- `main-color` et `secondary-color` vous permette de choisir les couleurs principales et secondaires de votre document.
- `main-font` et `header-font` vous permettent de choisir une police de caractère pour le texte et vos titres. Si `google-font` est à TRUE, vous pouvez sélectionner toute police disponible sur google font.

```
---
title: "Vaccination contre la covid 19 en France
subtitle: "Au 1er juin 2021"
author: "Guillaume RRRRRozier"
date: "11 juin 2021"
output:
    pagedreport::paged_grid:
      logo: "https://upload.wikimedia.org/wikipe
knit: pagedown::chrome_print
toc-title: "Sommaire"
main-color: "#000091"
secondary-color: "#000091"
google-font: TRUE
main-font: "Open Sans"
header-font: "Open Sans"
---
```