Résumé

Marc Thevenin

2024-03-20

Résumé

> Formation Bordeaux 21-03-2024
>
> - Chapitres non traités (mars 2024): les parties relatives à Git et aux contenus de type website, book, blog… un peu difficile à résumer.
> - Pas vraiment relu le contenu de la page

Chapitre 2: De Rmarkdown à Quarto

- Pour des documents simple, un simple changement de l'extension peu suffire: `.rmd` en `.qmd`
- Le chunk `{r setup...}` peut être supprimé.
- Dans le yaml du document (metadonnées) pensez à changer l'option pour le format: `format:` et non `output:`
- Compilation des fichiers dans le terminal avec Quarto:
  - `quarto render`: par défaut html, tous les fichiers du projets (si book, wensite)
  - `quarto render nom_fichier.qmd` pour un fichier particulier.
  - On peut changer le format de l'output, par exemple pour un format pdf:
    - `quarto render <file> --to pdf`

Chapitre 3: Edition d'un document

Balises markdown

- Mettre un commentaire dans un fichier .qmd

```
<!-- Commentaire -->
```

Formatage texte:

- `*`: texte en italique → `*texte*` *texte*
- `**`: texte en gras → `**texte**` **texte**
- `**`: texte en gras et italique → `***texte***` ***texte***
- `~~`: texte barré → `~~texte~~` ~~texte~~

Saut de ligne:

- A la fin de la phrase: `\`
- Deux espaces à la fin de la phrase
- Ajout d'une ligne vide pour augmenter l'espace entre deux lignes

Titre/header:

- `#`: de 1 à 6 => `#`, `##`,..., `######`

Listes:

- Une ligne vide obligatoire avec la phrase précédent la liste
- Non ordonnée: un symbole `*`, `+`, `-` en début de ligne pour une liste non ordonnée.
- Ordonnée: une série numérique `1.`, `2.`
- Sous liste: au niveau du premier caractère de la phrase de liste de niveau supérieur.

Insertion d'un lien:

- Chemin explicite sans titre: `<lien>`
- Titre: `[Titre](lien)`

Insertion d'une image:

- Sans option: `![](lien insertion image)`
- Avec option: `![](lien insertion image){options}`

Insertion d'un tableau markdown:

- Système de pipe.
- Utiliser mode visual de Quarto ou un éditeur en ligne

Insertion d'une équation/formule:

- Dans une ligne de texte: `$ équation/formule $`
- Dans une ligne pleine: `$$ équation/formule $$`

Propre à Quarto

- Section libre (
  en html)
  - Ouverte et fermée au minimum par 3: `:` → `:::`
  - Classe css associée

```
::: {.classe css ou insertion d'un style css}

Texte/image....

:::
```

- Formatage texte: `[texte]{ classe css ou insertion d'un style css}`
- Shortcode d'insertion: `{{< élément_inséré >}}`

Chapitre 4: Introduction au yaml

- Dans un fichier `.yml`
- – Dans un fichier `.qmd`: début du document .qmd ouvert et fermé par `---`
- Règle d'intendation (retrait) des options :
  - Au moins un retrait (second caractère), généralement (3ème caractère).
- Une même option ne peut être renseignée deux fois
- Une option inconnue n'engendre pas de message d'erreur si elle respecte les règles de syntaxe.

```yaml
---
option1: true/false

option2:
  sous-option2:

option3: "Texte"
---
```

Chapitres 5: blocs de codes

Déclaration d'un bloc et options

- Ouvert et fermé par ```
- Ajout d'un moteur entre des accolades: r, python,...: ```{r}
- Ajout d'un point au moteur si on ne souhaite pas qu'il soit exécuté: ```{.r}

````
```{r}
programme R
```
````

Ajouter des options dans le bloc:

- Hérité premières version de rmarkdown: ```{r, options}
- Privilégier dans le bloc de code la syntaxe: `#| option: true/false/...`

````
```{r}
#| option1: true/false
#| option2: "texte"

programme R
```
````

Options dans le yaml:

- Options partagées par tous les blocs.
- On peut revenir en arrière dans un ou plusieurs bloc individuellement avec la méthode précédente.

```yaml
---
option-bloc: true
---
```

````
```{r}
#| option-bloc: false

programme R
```
````

Quelques options d'exécution:

- Exécution du programme: `eval: true/false`
  - Par défaut `true`
- Report du programme: `echo: true/false/fenced`
  - Par défaut `true` **à l' exception du format revealjs**
- Affichage message et warning: `message/warning: true/false`
  - Par défaut true
- Enregistrer et réutilisé les résultats d'un programme (graphique,tableau formaté):
  - `cache: true/false`
- Report d'un résultat dans le corps du document (inline code)
  - `{moteur} fragment_programme`

Dans le yaml: sous option de `execute:`

```yaml
---
execute:
  eval: true/false
  echo: true/false
  message: true/false
  warning: true/false
```

Dans un bloc de codes:

```
```{r}
#|  eval: true/false
#|  echo: true/false
#|  message: true/false
#|  warning: true/false
```
```

- Report d'un résultat dans le corps du document (inline code)
  - `{moteur} fragment_programme`

Quelques options pour les graphiques et tableaux

Graphiques

- Dans le yaml et/ou dans le bloc de code
- préfixées par `fig-`: `fig-nom_option`
  - Exemple dans le bloc: `#| fig-cap: "Title"`
  - Exemple dans le yaml (seulement): `#| fig-cap-location: top` [par défaut `bottom`]

Tableaux

- Dans le yaml et/ou dans le bloc de code
- Préfixés par `tbl-`: `tbl-nom_option`
  - Exemple dans le bloc: `#| tbl-cap: "Title"`
  - Exemple dans le yaml (seulement): `#| tbl-cap-location: bottom` [par défaut `top`]

Options propres à Quarto

- Afficher/cacher à la volée le programme: `code-fold: false/true/show`. Par défaut `false`
  - Attention: option principale du yaml et non sous option du format html
- Afficher le code source de la page .qmd:
  - Option `code-tools: true/false`

  - Sous option de `code-tools` avec `source: true/false`

- Si associé à `code-fold` un menu est disponible.
- Syntaxes du yaml possibles avec `code-fold` et `code-tools`:

```yaml
---
format:
  html:
    code-tools: true/false

code-fold: true/false/show
---
```

```yaml
---
format:
  html:
    code-tools:
      source: true/false

code-fold: true/false/show
---
```

Si pas d'autres formats compilés:

```yaml
---
code-tools: true/false
code-fold: true/false/show
---
```

```yaml
---
code-tools:
  source: true/false
code-fold: true/false/show
---
```

- Accéder à la documentation web des fonctions: dans le yaml → `code-link: true/false`
- Titre d'un bloc: directement après avoir indiqué le moteur → ```` ```{r filename="Titre du bloc"} ````

- Annotations/commentaires de ligne du programme:
  - A la fin d'une ligne ajouter: `# <1>`, `# <2>`…
  - Sous le bloc du programme renseigner les commentaire:
    * `1. commentaire_1`
    * `2. commentaire_2` …

Chapitre 9: Introduction aux formats

- Formats compilé déclaré dans le yaml avec l'option `format`.
- Par défaut, le format document/notebook html est compilé. Sa déclaration est donc optionnelle.

Un seul format

Si déclarée:

```
---
format: html
---
```

```
---
format:
  html: default
---
```

```
---
format:
  html:
    option_html1: `true/false/...`
    option_html2: `true/false/...`
---
```

Si non délarée (seulement format document/notebook html):

```
---
option_html1: `true/false/...`
option_html2: `true/false/...`
---
```

- Si autre que html, les format doivent être déclarés. Exemple pdf:

```
---
format:
  pdf:
    option_pdf1: `true/false/...`
    option_pdf2: `true/false/...`
---
```

Si plusieurs format (html + pdf):

```
---
format:
  html:
    option_html1: `true/false/...`
    option_html2: `true/false/...`
  pdf:
    option_pdf1: `true/false/...`
    option_pdf2: `true/false/...`
---
```

- Si options identiques à tous les formats, elles peuvent être renseignées comme option principales dans le yaml. Exemple pour un titre et une table des matières:

```
---
titre: "XXXX"

format:
  html:
    toc-depth: 3
  pdf: default
    toc-depth: 5

toc: true/false
---
```

Chapitre 10: format html

- Deux types: le document/notebook ou la présentation (revealjs)
- Partagent des options communes

Exemples d'options communes document/notebook et présentation revealjs

- **Lightbox**: agrandir et ouvrir les images/graphiques dans un fenêtre → dans le yaml `lightbox: true`.
  - On peut désactiver l'option pour un.e ou plusieurs image/graphique:
    - Image (insertion markdown): option `{.nolightbox}`.
    - Graphique généré dans un programme: option `#| lightbox: false`.
- **Onglet**:
  - Dans une section libre (`:::`), classe css `{.tabset}`
  - Contenu de chaque onglet dans une section (## titre). Utiliser un nombre de # cohérent avec le nombre de sections dans la page. Si dans la profondeur de la table des matières est de 3, le nombre de `#` sera de préférence `####`:

```
::: {.tabset}

#### Titre onglet 1
Contenu onglet 1

#### Titre onglet 2
Contenu onglet 2
```

Document/notebook

- Changer les langues des titres (caractéristiques auteur.e, table des matières…) → dans le yaml `lang: fr` pour le français
- Résumé et description du document: dans le yaml options `abstract: |` et `description |`
  - Contenu sur la ligne suivante avec au moins un espace après le début de la ligne

```yaml
---
abstract: |
 Description de mon document
---
```

```yaml
---
abstract: |
 Description de mon document
---
```

- Modification du grid d'une page: dans le yaml option `grid` composé de 3 éléments:
  - A gauche la sidebar: sous option `sidebar: valeur`
  - Au centre le body: `body: valeur`
  - A droite la margin: `margin: valeur`
  - Attention les valeurs renseignées (en px) ne sont pas des largeurs absolues, mais des largeurs maximales qui s'adapteront au contenu.
- Largeurs par défaut:

```yaml
---
format:
  html:
    grid:
      sidebar-width: 250px
      body-width:    800px
      margin-width:  250px
---
```

Présentation revealjs

Les blocs de codes:

- Rappel: les blocs de code ne sont pas affichés par défaut. Dans le yaml ou dans des blocs spécifiques indiquer explicitement qu'on souhaite les afficher (`echo: true code-fold:true`).
- Ligne(s) en surbrillance: option `#| code-lines-numbers: "choix ligne(s)"`
  - Par exemple:
    * `"2"` ligne 2.
    * `"2,5"` lignes 2 et 5.

    * `"2-5"` lignes 2 à 5.
    * `"|2|4"` toutes les lignes, puis ligne 2, puis ligne 4

- Animation des blocs: dans le titre des slides option `{auto-animate: "true"}`

    - On répète dans plusieurs slides les lignes qui seront affichés
    - Penser à ce que dans chaque chaque slide le début du programme se trouve sur une ligne identique pour avoir un bon rendu.

Affichage:

- liens sous forme d'iframe (conseillé): activer dans le yaml l'option `preview-links: true`
- Affichage incrémental des éléments d'une liste:

    - Pour toute la présentation: dans le yaml `incremental: true`
    - Pour une slide en particulier, dans une section div (:::): `::: {.incremental}`
    - Si activé pour toutes les slides, on peut désactiver l'option pour une slide: `::: {.nonincremental}`

- Ajouter un effet: `{.fragment option}`. Exemple `{.fragment .fade-up}`
- Réduction de la taille d'affichage du texte: dans le titre d'une slide option `{.smaller}`
- Dépasser la limite verticale des slides: dans le yaml option `scrollable: true`

Facilité l'affichage dans une slide:

- Réduction de la taille d'affichage du texte: dans le titre d'une slide option `{.smaller}`
- Dépasser la limite verticale des slides: dans le yaml option `scrollable: true`
- Edition en plusieurs colonnes avec des section div (:::) css `::: {.columns}` et `::: {.columns width: "x%"}`. Exemple avec une colonne vide pour générer l'espace entre les textes des deux colonnes:

```
::: {.columns}
{.columns width: "47.5%"}
texte colonne 1
:::
::: {.columns}
{.columns width: "5%"}
```

```
:::
::: {.columns}
{.columns width: "47.5%"}
texte colonne 2
:::
```

- Pour l'affichage d'un graphique généré dans un programme qui est reporté: option `#| output-location: option`
  - options: `column`, `fragment`, `column-fragment`, `slide`

Chapitre 11: documents pdf

- Même si le document est seulement compilé en pdf on doit indiquer explicitement le format pdf: `format: pdf`
- 3 types de document: articles, rapport, livres. Dans le yaml on peut indiquer le type (par défaut article) avec l'option `documentclass: scrartcl/scrreprt/scrbook`.
- En plus de la table des matières, on peut ajouter une table des graphiques/images et/ou une table des tableaux.
  - Dans le yaml, options `lof: true/false` et `lot: true/false`. Par défaut false.
  - Les éléments sont insérés lorsqu'ils ont explicitement un titre:
    - Insertion markdown: `![TITRE](img)` pour une image, `: TITRE` sous un tableau.
    - Généré par un programme: `#|fig-cap: "TITRE"` pour un graphique, `#|tbl-cap: "TITRE"`
- Saut de pages: shortcode `{{< newpage >}}`
- Modifier les marges du document: dans le yaml option `geometry:` avec sous forme de liste `top`, `bottom`, `left`, `right`.

```
---
geometry:
  - top=Amm
  - left=Bmm
  - right=Cmm
  - bottom=Dmm
---
```

- Changement de la police: dans le yaml option **`fontfamily: nom_font`**

Contenu conditionnel

- On peut réservé un contenu à un ou plusieurs formats spécifiques. Dans certaines situations, cela s'avère nécessaire : mise en pages de plusieurs images sous forme de vignettes avec l'option lightbox donnera un rendu insatisfaisant en pdf ou docx.
- Des section
  permettent de réservé ou d'exclure un contenu à un ou plusieurs format:
  - `::: {.content-visible when-format="type_format"}`
  - `::: {.content-visible unless-format="type_format"}`
  - `::: {.content-hidden when-format="type_format"}`
  - `::: {.content-hidden unless-format="type_format"}`

Chapitre 13: theming html

Les thèmes bootstrap

- 26 thèmes bootstrap embarqués dans Quarto: dans le yaml option `theme: nom_theme`

```yaml
---
format:
  html:
    theme: nom_theme
---
```

- Possibilité de switcher entre un thème light et dark.
- Dans Quarto deux thèmes sont directement associés avec ces deux modes: **flatly** et **darkly**.

```yaml
---
format:
  html:
    theme:
      light: flatly
      dark:  darkly
---
```

- A partir d'un thème light (ou dark) ou peut créér un thème dark (ou light) à l'aide d'un fichier .scss. Plus généralement n'importe quel thème peut être modifié à l'aide de ce fichier (voir plus loin)

```yaml
---
format:
  html:
    theme:
      light: zephyr
      dark:  [zephyr, zephy_dark.scss]
---
```

- Quarto dispose de quelques options qui permettent de changer directement des éléments du thème sans passer par des fichiers .css ou .scss. Par exemple pour la couleur des liens avec l'option `linkcolor` (sous option de `theme`):

```yaml
---
format:
  html:
    theme: zephyr
    linkcolor: "#d63384"
---
```

Fichers CSS et SCSS

- Fichiers additionnels qui doivent être déclarés dans le yaml.

Fichier .css seulement:

```yaml
---
format:
  html:
    css: styles.css
---
```

Fichier .scss seulement avec le thème par défaut:

```yaml
---
format:
  html:
    theme: nom_fichier.scss
---
```

Fichier .scss seulement avec un thème:

```
---
format:
  html:
    theme: [superhero, styles.scss]
---
```

ou

```
---
format:
  html:
    theme:
      - superhero
      - styles.scss
---
```

On peut utiliser simultanément un fichier .css et un fichier .scss:

```
---
format:
  html:
    theme:
      - superhero
      - styles.scss
    css: styles.css
---
```

CSS

- Suite de classes de styles qui seront appliquées à tout le document ou à des éléments du document:
- Automatiquement: body (couleur de fond du document par exemple), style des liens ….
- Après déclaration d'une balise dans le document par exemple avec `:::{.nom_classe}`

Exemple mettre un contour à une image:

Dans le fichier .css:

```
.contour  {
border: 2px solid red;
padding: 5px;
margin: 5px;
}
```

Dans le fichier .qmd

```
::: {.contour}
![](lien_image)
:::
```

SCSS

- Variables SASS: Affectation en amont d'attributs de styles comme des couleurs qui seront affectées plus facilement à des styles css
- Quarto met à disposition une série de variable Sass qui seront directement appliquées au document. Très pratique
- Un fichier scss doit comprendre au moins une section d'affectation des variables Sass:
  - Cette section est introduite par : `/*-- scss:defaults --*/`. **Avec Quarto on doit absolument mettre cet élément au début du fichier**.
  - Les variables Sass sont déclaré avec un `$`: `$nom_var: attribut_css;`:

```
/*-- scss:defaults --*/

// variables sass couleur

$couleur_bleu: #3498db;
$couleur_rouge: #e74c3c
```

- Les variables Sass définies vont pouvoir être appliquée à la documentation via:
  - Directement à l'aide de variables Sass propres à Quarto, par exemple `$link-color:`
  - Indirectement via des classes css renseignées dans le fichier .scss.

Variables Sass Quarto:

```
/*-- scss:defaults --*/

// variables sass couleur

$couleur_bleu: #3498db;
$couleur_rouge: #e74c3c

// Variables Sass Quarto

$link-color: $couleur_bleu;
$toc-color:  $couleur_rouge;
```

Classes CSS dans un fichier .scss:

- On doit déclarer avec Quarto une section css dans le fichier .scss avec `/*-- scss: rules --*/`.
- On utilise la syntaxe css directement dans le ficher scss en affectant directement les variables Sass déclarées précédemment:

```
/*-- scss:rules --*/

.contour  {
border: 2px solid $couleur_rouge;
padding: 5px;
margin: 5px;
```

Un fichier .scss peut alors prendre l'aspect suivant:

```
/*-- scss:defaults --*/

// variables sass couleur

$couleur_bleu: #3498db;
$couleur_rouge: #e74c3c

// Variables Sass Quarto

$link-color: $couleur_bleu;
$toc-color:  $couleur_rouge;
```

```
/*-- scss:rules --*/

// classe css contour avec la la variable Sass $couleur_rouge

.contour  {
border: 2px solid $couleur_rouge;
padding: 5px;
margin: 5px;
```

Chapitre 14: notes, référence croisée et bibliographie

Notes

Deux méthodes

- Dans une phrase on déclare un numéro de note à l'endroit souhaité avec `^[numero_note]` et on renseigne plus loin le contenu de la note: `[^numero_note]: texte de la note`.
- Dans une phrase on renseigne directement le contenu de la note avec `^[contenu de la note]`.

```
On va insérer une note ici^[1]

[^1]: texte de la note
```

```
On va insérer une note ici^[texte de la note]
```

Références croisées

Seulement le principe avec quelques exemples.

- Principe de liens internes qui va s'appliquer à plusieurs types d'éléments d'un document: section, tableau, graphiques, callout, formules…….
- On déclare l'élément qui sera renvoyé plus tard à l'aide d'un label: `élement {#type-nom_label}` ou si l'élément provient d'un bloc de code avec l'option `#| type-label: nom_label` . * Le type est un prefixe qui dépend de l'élement: section: `sec`, image: `fig`, tableau `tbl`, équation `eq`….

Exemples:

- Une section du document:

```
## TitreI {#sec-titre1}
```

- Un graphique généré dans un programme

````
```{r}
#| fig-cap: "titre du graphique"
#| fig-label: graph1

<programme>
```
````

- A un autre moment du document pour afficher l'élément appelé: `@type-nom_label`.

Exemples:

```
Se reporter à la section 1 [@sec-titre1]
```

```
Se reporter au graphique 1 [@fig-titre1]
```

Dans cette page on a mis un label à la sous section *markdown:

```
### Balises markdown {#sec-md}
```

A ce moment du document, on l'appelle avec {Section } :

```
A ce moment du document, on l'appelle avec {@sec-md}
```

Références bibliographiques

- Dans le yaml associer un fichier .bib avec l'option `bibliography: nom_fichier.bib`.
- Dans le document, insérer le label de la référence bibliographique avec `@label_ref_biblio`.

Dans le fichier .bib on la référence suivante:

```
@article{bidibule_2099, title={blabla}, year = {2099}}
```

On insère la référence dans une phrase avec: `@bidibule_2099`

```
Comme l'a si bien écrit @bidibule_2099 .......
```

Comme l'a si bien écrit Bidibule (2099) …….

- On peut insérer toutes les références citées dans une bibliographie générale à l'aide d'une section div (`:::`) vide de contenu : `::: {#refs}`.

```
#### Références bibliographiques

::: {#refs}
:::
```

Chapitre 15: Elements de mise en page

Edition multicolonne

- A réserver à du texte de préférence
- Une colonne: balise div (`:::`) `{.column width= "x%"}`
- Plusieurs colonne: on entoure les colonnes avec la balise `{.columns}`
- On peut ajouter des colonnes vides pour gérer l'espace du texte entre deux colonnes

```
::: {.column}

::: {.columns width="47.5%"}
Texte colonne 1
:::

::: {.columns width="5%"}
:::

::: {.columns width="47.5%"}
Texte colonne 1
:::

:::
```

Mettre des éléments dans la marge du document

- Marge du document: colonne à droite du grid
- Balise css générique: `::: {.column-margin}` pour image et tableau markdown, liste, formule

```
::: {.column-margin}
Element à mettre dans la marge du document
:::
```

- Options dans les blocs de programme (graphique et tableau): option `#| column: margin`
- Pour les notes on doit utiliser une classe css qui sera placée juste après la note: `{.aside}`
  - La note est renseignée entre des crochets: `[texte de la note]{.aside}`
  - Attention aucun numéro de note sera ajouté.

```
On peut insérer une note dans la margin du
document[Attention il n'y aura pas de numéro de note]{.aside}
```

Mise en page des images sous forme de vignettes

- A réserver de préférence au html
- Tirer partie de l'option lightbox
- Deux types d'insertion: fixe ou libre
- Insertion fixe: `layout-ncol`
  - Dans une balise css: `::: {layout-ncol=nbre_colonne}`
  - Dans un bloc de codes: `#| layout-ncol=nbre_colonne`
- Insertion libre: `layout: "[liste sous forme de taille relative]"` => 2 images [50,50] ou [1,1] pour une taille égale
  - Balise css: `::: {layout: "[t1,t2,t3,...]"}`
    - Si les hauteurs des différentes images sont différentes on peut aligner en bas toutes les images avec l'option `layout-valign="bottom"`
  - Option bloc de codes: `#| layout: [t1,t2,t3,...]`

- On peut ajuster l'espace entre deux images avec une valeur négative, par exemple: `"[50,-1,50]"`

Références bibliographiques

Bidibule. 2099. "J'ai Des Choses à Dire."