

# Rédiger avec R

Eric Marcon

13 février 2024

Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

# Ecrire avec RMarkdown

Markdown est un format d'écriture très simple et lisible.

[knitr](#) exécute le code R à l'intérieur des documents, produit les résultats et les figures puis appelle RMarkdown ou Quarto.

[RMarkdown](#) importe Markdown dans R et [Bookdown](#) étend RMarkdown pour produire des documents au format Markdown strict.

[Quarto](#) est un programme indépendant de R qui joue le même rôle que Bookdown.

[Pandoc](#) convertit les documents Markdown en HTML, LaTeX (à compiler en PDF), Word...

# Document RMarkdown simple

Rédiger avec  
R

Eric Marcon





Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

Dans RStudio : *File / New File / R Markdown...*

New R Markdown

-  Document
-  Presentation
-  Shiny
-  From Template

**Title:**

**Author:**

**Default Output Format:**

☒ HTML

Recommended format for authoring (you can switch to PDF or Word output anytime).

☐ PDF

PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

☐ Word

Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

OK

Cancel

Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

Etudier le modèle :

- En-tête YAML et premier bout de code ;
- Formatage du texte ;
- Bouts de code.

Anti-sèche.

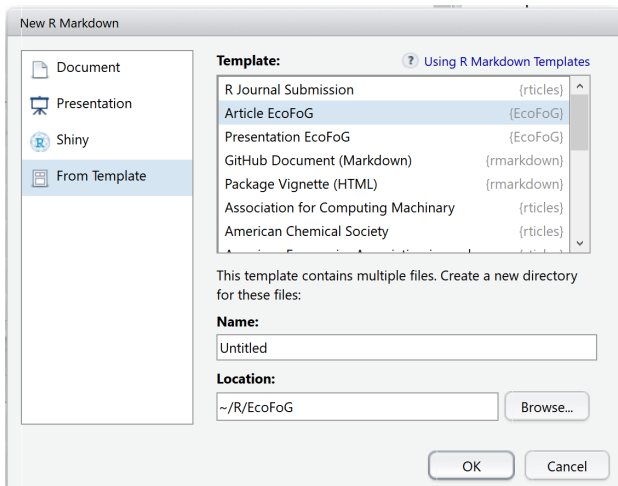
- Tester les 3 formats : HTML, PDF, Word.

PDF nécessite LaTeX, à installer par [tinytex](#).

```
library("tinytex")  
install_tinytex(bundle = "TinyTeX")
```

Le même document peut être utilisé sous différents format sans réécriture.

## Des packages fournissent des modèles.



Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

Article simple : HTML pour un partage rapide, PDF simple.

Article stylé : PDF pour l'autoarchivage et HTML pour la lecture. Word possible.

Présentation : Beamer et HTML (utiliser *ioslides*).

Mémoire (Mémoire de master, Thèse, HDR, livre) : PDF et HTML.

[Galerie](#)

Documentation dans les modèles.



Un projet R contient tout :

- Modèle de document ; Fichiers nécessaire à la mise en forme (styles de texte, de bibliographie, ...) ;
- Données ; Code R pour produire les résultats, y compris les figures ;
- Figures additionnelles.

Ce n'est pas un package :

- Un package a une organisation formelle inutile pour un article.

Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

Utiliser l'assistant *New Project / New Directory / Document Project using memoir*.

Tricoter pour vérifier le fonctionnement.

Possibilité de tricoter en HTML pour gagner du temps.

Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

Placer les données dans le projet, dans un format lisible par R (typiquement, CSV).

Lire les données dans le préambule de l'article.

Placer les calculs dans des bouts de code dans la section Matériels et Méthodes.

Utiliser les options des bouts de code :

- `echo` : affichage du code dans l'article (`FALSE` pour la publication) ;
- `cache` : pour ne pas répéter les calculs à chaque compilation.

Les figures sont produites directement par le code :

- insérer les bouts de code contenant les commandes `plot` dans la section *Résultats*.

Utiliser sa bibliographie générale, produite par Zotero et Better Bibtex :

- Pas de perte de temps pendant la rédaction ;
- Exportation en temps réel de la bibliographie du projet.

Ou utiliser directement une bibliographie spécifique, dans un fichier bib autonome.

Voir la [Documentation](#).

Modèles sauf *memoir* :

- Tricoter aux formats PDF et HTML.
- Exécuter `memoiR::build_githubpages()`.

Les fichiers produits (PDF, HTML, libs) sont déplacés dans */docs*.

Le fichier */README.md* est dupliqué dans */docs*.

Modèle *memoir* : *Build Book* tricote tout dans */docs*.

Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

Passer le projet sous Git et le pousser sur GitHub.

Ajouter des collaborateurs.

Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

## Activer les pages web du dépôt GitHub :

- Settings, GitHub Pages :
  - Source = Master Branch / docs Folder
  - choisir un thème.



Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

Dans README.md, ajouter les liens vers les fichiers produits :

- HTML pour la lecture en ligne ;
- PDF pour le téléchargement.

Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

# Intégration continue

Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

Sous-traiter à GitHub Actions (service en ligne) la construction des documents

Permet une mise à jour en continu, à chaque push.

Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

Entrer les secrets du projet : jeton Github et adresse de messagerie.

Créer le script : `memoiR::build_ghworkflow()`

Pages GitHub sur la branche `gh-pages`

# Figures pour la publication

Produire des figures à utiliser hors des documents Markdown.

Continuité entre l'analyse de données et la production de figures.

Eviter les copier-coller : créer directement des fichiers pour contrôler les tailles relatives.

Si les données changent, les figures sont refaites par le script.

# Exemple

Rédiger avec  
R

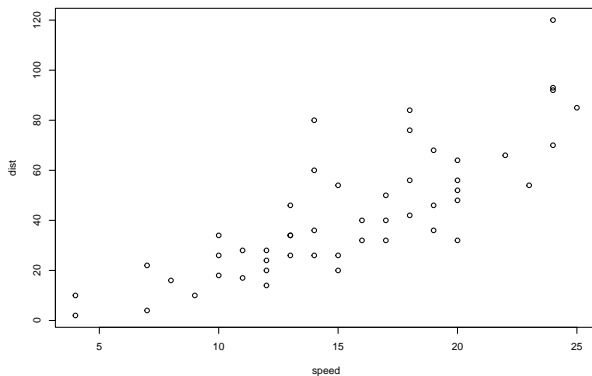
Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

```
plot(cars)
```



Demander à R d'écrire dans un fichier plutôt qu'à l'écran

```
postscript("Fig1.eps", width = 6, height = 4, horizontal=FALSE)  
plot(cars)  
dev.off()
```

```
## pdf  
## 2
```

postscript crée un fichier EPS, pdf un fichier PDF et svg un fichier SVG.

bmp, jpeg, png, tiff créent des fichiers raster.



Fichiers vectoriel pour les figures : PostScript ou PDF pour LaTeX, MetaFile pour Word:

```
library("devEMF")  
emf("Fig1.emf", width = 6, height = 4)  
plot(cars)  
dev.off()
```

```
## pdf  
## 2
```

Fichiers raster pour les rasters ou demandes particulières.

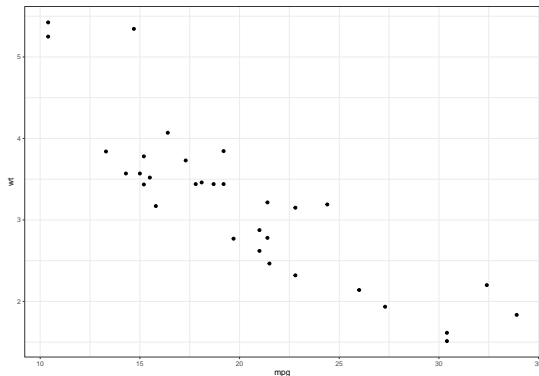
**ragg** améliore le rendu des figures en PNG, JPEG et TIFF.

Fonctions : `agg_png()`, `agg_jpeg()` et `agg_tiff()`.

Utilisable par RStudio  $\geq 1.4$  et par **knitr**.

La fonction `ggsave()` enregistre un graphique dans un fichier :

```
library("tidyverse")
mtcars |>
  ggplot(aes(mpg, wt)) +
  geom_point()
```



*# Sauvegarde du dernier graphique affiché*

Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication

Taille des caractères.

Couleurs ou non.

Ne passer en raster qu'en tout dernier recours. 300 dpi minimum.

Utiliser la documentation des fonctions pour les finitions (polices, taille, transparence...).

## Instructions

Les seuls formats acceptés sont PostScript et TIFF.

Les tailles sont précisées.

R n'encapsule pas les polices dans les fichiers EPS :

- utiliser `embedFonts()` pour le faire (nécessite *Ghostscript*)
- ou utiliser *Inkscape* pour transformer les polices en courbes avant publication.

Rédiger avec  
R

Eric Marcon

Ecrire avec  
RMarkdown

Intégration  
continue

Figures pour  
la publication