Utiliser R Markdown pour créer des documents dynamiques

par Kevin Cazelles

21 août 2015

Résumé

Le package “rmarkdown” permet de créer des documents dynamiques qui intègrent des morceaux de code R et ce qu’ils génèrent (dont figures et tableaux). Ce document explique brièvement comment utiliser le package ‘rmarkdown’ et aussi la syntaxe “Pandoc Markdown”. Le code source de ce document est en lui-même un exemple d’application du package.

Table des matières

Table des figures

1 Introduction

En milieu universitaire, des travaux préliminaires aux manuscrits, nous produisons très souvent des documents intégrant analyses et figures. Il est très pratique de pouvoir les intégrer directement dans un document complet qui peut également, au besoin, inclure le code source associé. Avec R c'est possible depuis plusieurs années. Dans le package de base "utils", il existe une fonction, *Sweave()*, qui permet de remplacer du code R par son rendu. Le package "knitr", qui utilise cette fonctionnalité, permet de produire des fichiers Latex intégrant du code R et des figures. Le dernier package disponible sur le CRAN pour générer des documents dynamiques est le package "rmarkdown" dont je vais vous parler.

Dans "rmarkdown", il y a R et Markdown. Je pars du postulat que le lecteur est familier avec le langage R. Il me reste alors à parler de Markdown. Le "Mardown original" date de 2004, l'idée de départ est simple et élégante : produire un langage léger qui simplifie les balises HTML et qui serait facilement transformé en HyperText Markup Language (HTML). L'idée n'est pas de tant de remplacer le HTML mais plutôt d'augmenter l'efficacité d'écriture. Chemin faisant, Markdown séduit, en parallèle de la version originale (qui n'a jamais été standardisée), plusieurs extensions voient le jour. Il s'agit surtout de lever certaines limitations tout en préservant l'esprit d'origine, parmi elles:

— GitHub Flavored Markdown (GFM)
— Kramdown
— Markdown extra
— Multi Markdow
— Pandoc markdown

R Markdown version 2 [1] n'est pas une nouvelle variante Markdown. Il s'agit de l'utilisation de la variante Pandoc Markdown [2] à laquelle viennent s'ajouter les fonctionnalités des packages "knitr" et "yaml", le tout lié par l'utilisation de Pandoc. Une précision relative à Pandoc s'impose. Il s'agit, comme l'indique le site internet, d'un "convertisseur de document universel". Personnellement, Pandoc a changé mes pratiques : en une ligne de commande Pandoc convertit un document dans un format donné en de nombreux autres formats. Par exemple, on peut passer d'un fichier .tex (Latex) à un fichier .docx (Word)! L'intérêt du package "rmarkdown" est quadruple :

1. utiliser une syntaxe très épurée pour distinguer la mise en page, texte, le formatage du texte et le code R,
2. utiliser les fonctionnalités du package "knitr" pour créer morceaux de codes et figures dynamiquement,
3. utiliser le format de sérialisation YAML pour personnaliser la mise en page des documents produits grâce à la fonction *render* elle-même appelée en quelques clics avec Rstudio,
4. obtenir ainsi un document avec un contenu R dynamique en différents formats dont "html", "pdf" et "docx".

Concernant la documentation relative aux fonctionnalités du package "rmarkdown", je recommande le site officiel de R Markdown sur lequel vous trouverez deux documents précieux : un condensé d'utilisation sous forme de *Cheat Sheet* ainsi qu'un guide de référence. Le site dédié à knitr permet de bien comprendre l'intégration des morceaux de

1. La version 1 n'utilise pas Pandoc.
2. Pandoc est d'ailleurs capable de gérer différentes variantes de Markdown.

codes et des figures de R. Je tiens également à signaler la présentation de Mansun Kuo sur Rpubs. Pour apprendre la syntaxe Pandoc Markdown, vous pouvez vous reportez à la documentation disponible sur le site de Pandoc ou à la page internet de Jean-Daniel Bonjour pour une documentation en français.

Pour conclure cette introduction, je signale que la syntaxe Pandoc Markdown est, à mon sens, très complète, mais elle ne permet pas une personnalisation totale de la mise en page de votre document. Néanmoins, avant d'envisager l'un des recours proposés ci-dessous, il est très important de vous demander si cette personnalisation est nécessaire (surtout si le document est réservé à un comité restreint de lecteurs). Si toutefois vous souhaitez aller plus loin dans la mise en forme de votre document, plusieurs options vous sont offertes :

1. vous pouvez travailler sur le document dans le format qui vous intéresse après l'avoir générer,
2. vous pouvez modifier le *template* (fichier qui gère en grande partie la mise en forme de votre document) associé à votre document [3],
3. vous pouvez toujours composé en un langage donné, p. ex. HTML ou Latex, mais les commandes non valides pour un format donné, seront ignorées.

2 Comment utiliser un fichier .rmd

2.1 Quelques repères

Un fichier Rmarkdown (".rmd") est divisé en trois :

— Le YAML entre bloc de trois tirets : ---
— Les morceaux de code R entre bloc de trois accents graves : "‘
— Le reste du texte utilise la syntaxe Pandoc Markdown

Au début, il est délicat de se retrouver dans ces fichiers car on y croise différents langages. Trois langage sont pleinement utilisés pour créer le document : Markdown, R et YAML. De plus, dans la syntaxe Pandoc Markdown utilisée, on trouve des syntaxes communes à deux autres langages: - un ensemble de balises HTML, p. ex. celle utilisée pour insérer des commentaires : <-- le commentaire --> ou encore celle dédiée aux espaces insécables : , - la syntaxe Latex pour les équations.

Quand le fichier est complet, et que le package "rmarkdown" est installé [4], il suffit d'utiliser la fonction *render()* du package en lui indiquant le chemin de votre fichier ou de cliquer sur "Knit HTML" dans Rstudio. Je ne suis personnellement pas utilisateur de ce dernier, j'utilise donc simplement la fonction *render()*. Je vous invite dans tous les cas à regarder la documentation associée [5]. Pour convertir mon fichier ".rmd", j'entre la commande suivante dans la console R :

```
render("./ex_Rmardown.rmd", "all")
```

L'argument "all" permet d'obtenir l'ensemble des fichiers pour lequel il existe une spécification dans le fichier YAML. Les formats actuellement disponibles sont :

— pour les documents :
  — PDF
  — HTML
  — Word
  — Tufte handout (Pdf)
  — Package Vignette (Html)
— pour les présentations :
  — Beamer, un lien utile: https://bitbucket.org/rivanvx/beamer/wiki/Home

3. Vous les trouverez dans le dossier du package "rmarkdown". Pour savoir où est ce dossier, enter "R.home()" dans votre console R, rendez-vous à l'adresse donnée puis, dans le sous-répertoire "library/rmarkdown".
4. Rappel, il suffit d'entrer "install.packages('rmarkdown')" dans votre console R.
5. Entrez "?render" dans la console R lorsque le package est chargé, ou encore "?rmarkdown::render" si le package n'est pas chargé.

— ioslide, un lien utile: https://code.google.com/p/io-2012-slides/
— Slidy, un lien utile http://www.w3.org/Talks/Tools/Slidy2/#(1)

Les documents au format pdf requièrent l'installation de Latex. En effet, pour produire un pdf à partir d'un fichier ".rmd", un premier fichier Latex est produit. Pour le lecteur qui désire conserver le fichier Latex utilisé pour produire le pdf, il suffit d'ajouter l'option "keep_tex: true" dans les options relatives au format pdf dans le Le fichier YAML. Pour le Markdown, un fichier ".md" est généré suivant la variante de Markdown précisée dans le YAML par "variant: markdown" (voir la section suivante). Le format "Tufte handout" est une mise en page particulière, avec marges larges où sont insérées les illustrations (dont les figures), que nous devons au chercheur Edward Tufte.

2.2 Le fichier YAML

Il s'agit d'un ensemble de spécifications placées en tête de votre document ".rmd" (veuillez à ne pas introduire de commentaires avant). C'est un ensemble YAML (YAML Ain't Markup Language) est un format de représentation de données intégré dans le fichier ".rmd" entre bloc de trois tirets. C'est un ensemble de métadonnées permettant de préciser nos souhaits relatifs aux formats de sortie.Ces données sont alors soit intégrées dans dans la ligne de commande Pandoc soit dans les *templates*. Pour comprendre l'essentiel du fonctionnement du YAML, je vous conseille de regarder la page française wikipedia. La page anglaise sur le même sujet vous permet d'en apprendre davantage. Pour un aperçu assez complet des options YAML utilisables dans le fichier .rmd par défaut, rendez-vous à la dernière page du guide de référence. J'ai volontairement utilisé un grand nombre de spécification pour générer ce document, en voici une reproduction quasi-identique (sans le résumé complet et sans la majorité des commentaires que vous pouvez trouver dans le fichier source de ce document) :

```
---
title: "Utiliser Rmarkdown pour créer des documents dynamiques"
date: "21 août 2015"
author: par Kevin Cazelles
lang: french
abstract: "R Markdown [...]."
fontfamily: fourier
linestretch: 1
fontsize: 10pt
lof: yes
highlight: zenburn
# on peut également utiliser ‘‘default’’, ‘‘tango’’, ‘‘pygments’’, ‘‘kate’’,
# ‘‘monochrome’’, ‘‘espresso’’, ‘‘zenburn’’, ‘‘haddock’’, ‘‘textmate’’
output:
html_document:
    # css: ./aux/designR.css
    highlight: zenburn
    # ‘‘default’’, ‘‘tango’’, ‘‘pygments’’, ‘‘kate’’, ‘‘monochrome’’, ‘‘espresso’’, ‘‘zenburn’’, ‘‘haddock’’, ‘‘textmate’
    theme: flatly
    # ou ‘‘default’’, ‘‘cerulean’’, ‘‘journal’’, ‘‘readable’’, ‘‘spacelab’’, ‘‘united’’, ‘‘cosmo’’
    toc: yes
pdf_document:
    highlight: kate
    toc: yes
    toc_depth: 3
    fig_caption: yes
    keep_tex: yes
    latex_engine: pdflatex #xelatex
    number_section: yes
    includes:
        before_body: ./aux/license.tex
```

```
  word_document:
    fig_caption: yes
    highlight: pygments
  md_document:
    variant: markdown_strict
bibliography: ./aux/mybiblio.bib
csl: ./aux/journal-of-theoretical-biology.csl
---
```

L’indentation reflète les groupes de spécification. Ainsi, de “pdf_document:” à “word_document:”, les commandes permettent de spécifier les sorties du document au format pdf. Notez que “toc” signifie “table of content” (table des matières), “toc_depth” permet de sélectionner le niveau maximal de sous-titre affiché dans la table des matières. De même “tof” signifie “table of figure” (table des figures). L’option “number_section:true” permet d’obtenir une numérotation des différentes parties. Les commentaires sont introduits par un “#”.

3 Le “Pandoc Markdown”

Dans cette partie, je détaille les éléments de formatage du texte proposés par Pandoc Markdown. L’ensemble est très bien présenté sur le site de référence [6] et très bien résumé à la première page du guide de référence. Pour une source en français, j’ai trouvé un bon tour d’horizon nommé : “Élaboration et conversion de documents avec Markdown et Pandoc” écrit par Jean-Daniel Bonjour. Notez que certains symboles sont réservés au formatage du texte. Cependant, quand leur affichage est requit, on les fait précéder du caractère d’échappement qui est, pour Markdown, l’antislash : “\”. Par exemple, j’entre :

```
\\ \& \# \$ \[
```

Pour obtenir : \ & # $ [. J’ajoute que certains sauts de ligne sont obligatoires pour obtenir la mise en page désirée (pour les notes de bas de page par exemple).

3.1 Décoration du texte

— Pour écrire du *texte en italique*, vous avez deux possibilités :
```
*le texte à mettre en italique*
_le texte à mettre en italique_
```
— Pour écrire du **texte en gras**, encore deux possibilités :
```
**le texte à mettre en gras**
__le texte à mettre en gras__
```
— Pour écrire du ***texte en gras et italique***, utilisez :
```
**le _texte en italique et en gras_**
```
— Pour obtenir un ~~texte rayé~~, entrez :
```
~~texte rayé~~
```
— Pour écrire un élément en $^{\text{exposant}}$, utilisez :
```
^texte en exposant^
```
— Pour écrire un élément en $_{\text{indice}}$, tapez :
```
~texte en indice~
```

Attention, exposants et indices ne fonctionneront pas si l’élément à formater n’est pas exempt de tout espace. Notez qu’il n’y a pas de balises pour le soulignement du texte. Si vous souhaitez l’obtenir pour le HTML, vous pouvez utilisez la balise suivante:

6. Le site de documentation est totalement écrit en Markdown et reprend ce que nous pouvons lire sur le site de Pandoc et que je reprend mais en français!

```
<u>texte souligné</u>
```

Pour le pdf vous pouvez utiliser:

```
\underline{texte souligné}
```

Si vous utilisez l’une ou l’autre de ces balises, votre formatage sera moins général.

3.2 Les titres

Le plus simple est d’utiliser un nombre croissant de “#”" pour descendre dans l’arborescence des titres :

```
# Un titre d'ordre 1
## Un titre d'ordre 2
### Un titre d'ordre 3
```

Il est aussi possible d’utiliser une série de “=” en dessous des titre de premier niveau et une ligne de “-” en dessous des titres de niveau 2. Cette option à la qualité de permettre de repérer facilement les titres dans le code source.

```
Un titre d'ordre 1
==================
Un titre d'ordre 2
------------------
### Un titre d'ordre 3
```

3.3 Les listes

Les listes sont très intuitives en Markdown, alors qu’elles requièrent des balises un peu lourdes que ce soit en Latex ou en HTML. Dans les exemples donnés, il faut toujours séparer le texte principal de la liste par des sauts de ligne.

3.3.1 Listes non numérotées

Pour obtenir une liste non numérotée j’entre :

```
* truc 1,
* truc 2,
* truc 3.
```

ou bien :

```
+ truc 1,
+ truc 2,
+ truc 3.
```

ou encore :

```
- truc 1,
- truc 2,
- truc 3.
```

et même :

```
+ truc 1,
* truc 2,
- truc 3.
```

Dans tous les cas, j'obtiens :

- truc 1,
- truc 2,
- truc 3.

En utilisation une indentation de 4 espaces (ou une tabulation), j'obtiens facilement des listes hiérarchisées, ainsi quand j'utilise :

```
- truc 1,
    + machin 1
        - chose 1
        - chose 2
    + machin 2
- truc 2,
- truc 3.
```

j'obtiens :

- truc 1,
  - machin 1
    - chose 1
    - chose 2
  - machin 2
- truc 2,
- truc 3.

En ajoutant au moins deux espaces à la fin des éléments d'une liste, chaque élément de la liste est formaté comme un paragraphe :

- truc 1,
- truc 2,
- truc 3.

Pour alterner des listes avec du texte ou du code, il faut utiliser des sauts de lignes avec l'indentation adéquate. Ainsi, avec les lignes suivantes :

```
- élément 1 :

    Un petit texte qui pourrait expliciter ce qu'est l'élément 1.

- machin 2:

          for (i in 1:2) print(i)
```

j'obtiens :

- élément 1 :
  Un petit texte qui pourrait expliciter ce qu'est l'élément 1.
- machin 2:
  for (i in 1:2) print(i)

3.3.2 Les listes non numérotées

Pour une liste numérotée, c'est aussi très simple, en entrant par exemple si je rentre :

```
1. machin 1,
2. machin 2,
3. machin 3.
```

J'obtiens:

1. machin 1,
2. machin 2,
3. machin 3.

Si les nombres ne sont pas écrits manière ordonnée, cela ne changera pas le résultat. Néanmoins, le premier nombre détermine le point de la liste. En écrivant :

```
3. machin 1,
3. machin 2,
3. machin 3,
5. machin 4.
```

j'obtiens :

3. machin 1,
4. machin 2,
5. machin 3,
6. machin 4.

Pour ne pas se soucier des numéros, il existe un style par défaut :

```
#. machin 1,
#. machin 2,
#. machin 3.
```

on retrouve bien la première liste numérotée :

1. machin 1,
2. machin 2,
3. machin 3.

Plusieurs styles de numérotation sont disponibles, p. ex. :

```
#) élément 1
#) élément 2
#) élément 3
(1) truc 1
(2) truc 2
(5) truc 3
      i. machin 1
      i. machin 2
      i. machin 3
```

nous donne :

1) élément 1
2) élément 2
3) élément 3

(1) truc 1
(2) truc 2
(3) truc 3

i. machin 1
ii. machin 2
iii. machin 3

Nous avons aussi la possibilité de mélanger les niveaux numérotés et les niveaux non-numérotés :

```
1. machin 1,
    1. machin 1.1,
    2. machin 1.2,
2. machin 2,
    - machin 2.1,
    - machin 2.2,
    - machin 3,
3. machin 4,
4. machin 5.
```

ce qui donne :

1. machin 1,
    1. machin 1.1,
    2. machin 1.2,
2. machin 2,
    — machin 2.1,
    — machin 2.2,

— machin 3,

3. machin 4,
4. machin 5.

Enfin, il possible de mettre manuellement fin à une liste en introduisant un commentaire entre les listes à séparer :

```
(1) truc 1
(2) truc 2
(3) truc 2b

<!-- end -->

(1) truc 3
(2) truc 4
```

ces lignes sont rendues ainsi :

(1) truc 1
(2) truc 2
(3) truc 2b
(4) truc 3
(5) truc 4

3.4 Blocs de citation

Pour utiliser un bloc de citation (la balise “blockquote” en HTML), il suffit d’utiliser “>” avant la citation. Ainsi,

```
> La citation est ajoutée comme ceci, elle nous donne une indentation adéquate
pour une mise en page agréable dont le style peut être facilement travailler
en html grâce au CSS.
```

devient :

> La citation est ajoutée comme ceci, elle nous donne une indentation adéquate pour une mise en page agréable dont le style peut être facilement travailler en html grâce au CSS.

Pour ajouter une hiérarchie dans les citations, on entre :

```
> La citation de départ
>
>> une hiérarchie dans la citation
```

ce qui donne :

> La citation de départ
>
> > une hiérarchie dans la citation

3.5 Blocs de code.

Pour l’ensemble des exemples que je présente sous forme de lignes de code, j’utilise un environnement simple qui utilise une police d’écriture à chasse fixe et affiche tous les caractères tel qu’il sont entrés (les balises ne sont pas interprétés). Pour cela, j’ajoute simplement 4 espaces (ou une tabulation) au début de chaque ligne pour que se soit considérer comme du code. L’indentation d’un bloc de code se fait toujours avec 4 espaces (ou une tabulation) de plus que le texte, les listes ou encore les citations.

Il est possible d’ajouter des morceaux de code colorés selon la nature des éléments de la syntaxe d’un langage donné. Cette fonctionnalité est apportée par le Markdown extra [7]. Les morceaux de codes peuvent être écrits entre série de ~ et une accolade est nécessaire pour préciser le langage. Pour présenter un morceau de code C, par exemple :

```
~~~~~~~~~~~~~~~{.c}
// Commentaire
int c,d,g ;
c=10 ;
d=4 ;
int func(int a,int b) { return a*b; }
g = func(c,d)
printf("%i",g)
~~~~~~~~~~~~~~~
```

devient :

7. Une ressource intéressante est le site Doxygen de l’université de Nouvelle Angleterre.

```
// Commentaire
int c,d,g;
c=10;
d=4;
int func(int a,int b) { return a*b; }
g = func(c,d);
printf("%i",g);
```

Il est possible d’inclure des lignes de tilde dans le morceaux de code. Pour cela il faut que les lignes de tildes en compte moins que celles qui délimitent le morceau de code.

3.6 Les barres horizontales

Une ligne barre horizontale peut être ajoutée comme suit :

```
-------------
```

---

ou encore :

```
*  *  *  *  *
```

3.7 Les tables

La création facilitée de tableaux est l’une des extensions bien utile de Pandoc Markdown. Il existe plusieurs extensions pour faire des tableaux. Pour avoir des détails sur le sujet, je vous recommande la page écrite par Jean-Daniel Bonjour. Je présente ci-dessous un exemple de tableau de style “pipe table”:

```
| Aligné à gauche | Aligné au centre | Par défaut | Aligné à droite
| :-------  | :-------: | ------  | -------:
| truc 1.1  | truc 2.1  | **_truc 3.1_** | truc 4.1
| truc 1.2  | truc 2.2  | ~~truc 3.2~~   | truc 4.2
| truc 1.3  | truc 2.3  | *truc 3.3*     | truc 4.1

: La légende associé au tableau.
```

TABLE 1: La légende associé au tableau.

Aligné à gauche	Aligné au centre	Par défaut	Aligné à droite
truc 1.1	truc 2.1	***truc 3.1***	truc 4.1
truc 1.2	truc 2.2	~~truc 3.2~~	truc 4.2
truc 1.3	truc 2.3	*truc 3.3*	truc 4.1

Si vous préférez les obtenir grâce à une interface de type “WYSIWYG” (*What you see is what you get*), vous pouvez

utiliser ce générateur de table.

3.8 Mathématiques

3.8.1 Symboles mathématiques

Pour utiliser les symboles mathématiques dans le texte, les commandes associées doivent être placées entre deux "$". Bien sur, il faut connaître les combinaisons de caractère associées aux différents symboles. Ce sont les même que celles proposées par Latex et qui seront utilisées par MathJax pour générées les expressions mathématiques dans le fichier HTML. Pour quelques exemples, regarder ce site, pour quelque choses de plus complet, jetez un œil ici. Voici tout de même quelques exemples :

— quelques lettres grecques :
`$\alpha$, $\beta$, $\delta$, $\lambda$, $\pi$, $\phi$, $\omega$, $\varpi$, $\vartheta$`
pour obtenir : α, β, δ, λ, π, ϕ, ω, ϖ, ϑ
— quelques symboles mathématiques :
`$\sum$, $\prod$, $\int$, $\infty$, $\lim$`
pour obtenir : $\sum$, $\prod$, $\int$, ∞, lim
— quelques combinaisons :
`$\mu \in\mathbb{R}$, $\lim_{x \rightarrow 3} f(x)$`
pour obtenir : $\mu \in \mathbb{R}$, $\lim_{x \rightarrow 3} f(x)$

3.8.2 Équations

Pour faire des équations, il faut placer l'équation entre double $$. Créons une première équation :

```
$$\frac{vache}{oiseau} = \frac{2\pi}{l}$$
```

$$\frac{vache}{oiseau} = \frac{2\pi}{l}$$

Pour utiliser des équations numérotées, il faut ajouter "(@label)" avant l'équation, je ré-utilise l'équation précédente :

```
(@eq1) $$\frac{vache}{oiseau} = \frac{2\pi}{l}$$
```

(1) $$\frac{vache}{oiseau} = \frac{2\pi}{l}$$

Je rajoute une seconde équation qui utilise les balises Latex pour créer un système :

```
(@eq2) $$\begin{array}{ccc}
x^2+y^2 &=& z^2 \\
xy &=& z
\end{array}$$
```

(2) $$\begin{array}{ccc}
x^2+y^2 &=& z^2 \\
xy &=& z
\end{array}$$

La référence aux équations se fait en utilisant "(@label)" dans le texte. Ainsi, en écrivant "(@eq1)", j'appelle l'équation (1). De même je peux faire référence à l'équation (2) en utilisant "(@eq2)".

3.9 Les images

Pour insérer une image, deux solutions nous sont offertes :

1. la combinaison dite “inline” : !+[nom de l’image (inclus dans la légende)]+(adresse)+{#label}. Par exemple,

   ```
   ![Le logo de R](./images/Rlogo.png)
   ```

FIGURE 1 – Le logo de R

2. la combinaison dite “reference” : !+[nom de l’image (inclus dans la légende)]+[id] et ailleurs dans le document le détail [id]+:+adresse. Par exemple

   ```
   [img2]: ./images/Markdown.png

   ![Le logo sur la page du Markdown original][img2]{#fig:logo}
   ```

{#fig:logo}

Il existe deux possibilités pour faire des références à une figure. Malheureusement aucune n’est idéale:

1- On peut injecter une balise \label{lelabel} et faire utiliser **??**, mais ce n’est valable pour le pdf (pour un exemple voir la section “Les figures produites avec R”) 2- On peut appeler un filtre particulier de pandoc : pandoc-fignos, cela demande une installation à la main (mais cela peut vite changer),

3.10 Références

3.10.1 Liens hypertextes

Les liens hypertextes sont utilisés sous la forme [groupe de mots sur lequel cliquer]+(adresse du lien). Pour créer un lien vers la page Markdown de Wikipedia, j’utilise :

```
[Markdown](https://fr.wikipedia.org/wiki/Markdown)
```

et voila le lien vers la page Markdown de Wikipedia.

3.10.2 Notes de bas de page

On peut produire une note de bas de page en plaçant la balise “[^id]” là où la note doit être insérée. Pour préciser le texte qui y est associé, on utilise: “[^id]:texte associé” où l'on souhaite dans le document, il faut simplement que les notes de bas pages (rassemblées ou non) soit séparé du reste du texte par un saut de ligne. Personnellement, je préfère rassembler les notes d'une section à la fin de la section en question. Par exemple :

```
Un bout de texte avec une note[^note1] et une autre [^note2].

[^note1]: à la fin d'une section par exemple.
[^note2]: ou encore, à la fin du document.
```

Un bout de texte avec une note [8] et une autre [9].

Notez qu'il faut séparer le texte des notes de bas de pages du texte principal d'un saut de ligne.

3.10.3 Références à une section

La référence à une section se fait à l'aide de deux crochets. Dans le premier crochet, on trouve le texte associé au lien et dans le second, le nom de la partie à laquelle on fait référence :

```
Référence à la [section sur les liens hypertextes][Liens hypertextes]
```

Référence à la section sur les liens hypertextes

On peut également utiliser le nom de la section dans le premier crochet et rien dans le second :

```
Rappelez vous la section [Liens hypertextes][]
```

Rappelez vous la section Liens hypertextes

3.10.4 Références bibliographiques

Un des points forts de Pandoc est la possibilité de gérer de manière très efficace votre bibliographie. Un grand nombre de fichiers de bibliographie sont bien gérés [10], dont les fichiers bibtex. Pour plus de renseignements, visitez la page du site de R Markdown consacrée.

L'exemple qui suit est basé sur le fichier bibtex (mybiblio.bib) disponible dans le dossier “aux” qui est donnée avec ce document et que je donne ci-dessous :

```
@article{Lande1979,
    author = {Lande, R},
    journal = {Evolution (N. Y).},
    number = {1},
    pages = {402--416},
    title = {Quantitative Genetic Analysis of Multivariate Evolution,
    Applied   to    Brain : Body Size Allometry Russell Lande},
    volume = {33},
    year = {1979}
```

8. la première
9. la seconde
10. Les logiciels de gestion de bibliographie génèrent ces fichiers.

```
}
@article{Oreskes1994,
    author = {Oreskes, Naomi and Shrader-Frechette, Kristin and Belitz,   Kenneth},
    journal = {Science (80-. ).},
    number = {5147},
    pages = {641--646},
    title = {Verification, Validation, and Confirmation of Numerical
    Models in the Earth Sciences},
    volume = {263},
    year = {1994}
}
@article{Knauff2014,
    author = {Knauff, Markus and Nejasmic, Jelica},
    issn = {1932-6203},
    journal = {PLoS One},
    number = {12},
    pages = {e115069},
    title = {{An Efficiency Comparison of Document Preparation Systems
     Used in  Academic Research and Development}},
    volume = {9},
    year = {2014}
}
```

Ce fichier est spécifié dans le YAML : “bibliography: mybiblio.bib” (voir la section “Le fichier YAML”). Le style de citation peut être spécifié, entre autres, à l'aide d'un fichier CSL, lui aussi spécifié dans le YAML et fourni dans le dossier “aux” : “csl: ./aux/journal-of-theoretical-biology.csl”. Dans le texte, on peut appeler une référence en utilisant la forme sans parenthèse ou avec parenthèse. Pour la forme sans parenthèse, il suffit d'utiliser “@id” où “id” est l'identifiant donné par le fichier “mybiblio.bib”. Pour une référence sans parenthèse :

```
Selon @Oreskes1994, la modélisation [...].
```

1. Selon Oreskes et al. (1994), la modélisation [...].

Pour les citations entre parenthèses, on procède ainsi :

```
Dans la littérature, [...] [@Oreskes1994; @Lande1979; @Knauff2014].
```

2. Dans la littérature, [...] (Knauff and Nejasmic, 2014; Lande, 1979; Oreskes et al., 1994).

Il est important de savoir que la liste des références est mise à la fin du document, nous ajoutons donc #Références (ou # bibliographie) à la fin du document pour lui donner un titre. Une autre remarque pour vous dire qu'il existe de nombreux fichiers “csl” (acronyme pour *Citation Style Langage*) sur le site de Zotero.

4 Intégration de R dans le document

L'intérêt du package “rmarkdown” est d'étendre la syntaxe Pandoc Markdown avec les fonctionnalités du package knitr pour insérer du code R ainsi que les sorties associées (sorties console et figures). Nous obtenons ainsi un document dynamique en ce sens que si les données associées et/ou le code R changent, le document évolue aussi.

4.1 Les morceaux de code (*code chunks*)

4.1.1 Deux injections de code possible

Il y a deux types de morceaux de code. En premier lieu, ceux qui sont insérés directement dans le texte (“inline”). Pour cela, il faut inclure le texte sous la forme : ‘r+ commande R + ‘. Je peux, par exemple, demander l’heure et la date à R en utilisant la function *Sys.time()* : 2015-08-21 00:50:35.

Ensuite, il y a le code que nous souhaitons séparer du reste du texte et dont on souhaite obtenir le résulat. Les morceaux de code sont alors écrits :

````
```{r + spécifications}
ligne de code 1
ligne de code 2
...
ligne de code n
```
````

Les commentaires sont introduits, comme dans R, sous la forme de lignes de codes commençant par un “#”. Débutons avec un exemple simple qui inclut un commentaire et une addition :

````
```{r addition}
# une addition avec R.
2+3
```
````

Notez que “addition” qui suit “r” dans l’accolade est l’identifiant du morceau de code. Les lignes précédentes me donnent :

```r
# une addition avec R.
2+3
```

```
## [1] 5
```

J’obtiens ainsi le code de R dans un environnement adéquate (voir la coloration du code) avec la sortie console associée, en l’occurrence, le résultat de l’addition. Je présente dans la suite un certain nombre d’options qui donne une large flexibilité dans la création des morceaux de code. Pour avoir accès à plus de précisions, reportez-vous au guide de référence ou au site de knitr. L’apparence des morceaux de code peut être facilement modifiée dans le YAML, avec le paramètre “highlight” (voir plus haut) qui peut être spécifié pour les différents formats.

4.1.2 Le paramètre “results”

Il est aussi possible de changer l’apparence des sorties console grâce au paramètre “results” dans l’accolade.

— result=‘asis’ pour que les résultats soient affichées avec la police du document texte principal :

````
```{r addition2, results='asis'}
# une addition avec R.
2+3
```
````

```r
# une addition avec R.
2+3
```

[1] 5

— result='hide' :

```
``{r addition3, results='hide'}
# une addition avec R.
2+3
```
```

```
# Commentaire : une addition avec R.
2+3
```

— Il existe aussi les options "results='hide' " et "results='hold' ". La première n'affiche pas les sorties console de R (sauf les messages spéciaux comme les messages d'erreur). La seconde permet d'afficher l'ensemble des sorties après le morceau de code. Par exemple, sans cette option, en utilisant le code suivant' :
"{r} a <- 2 print(a) b <- 3 print(b) 2+3 "' j'obtiens :

```
a <- 2
print(a)
```

```
## [1] 2
```

```
b <- 3
print(b)
```

```
## [1] 3
```

```
2+3
```

```
## [1] 5
```

alors qu'en ajoutant "results='hold' " :

```
a <- 2
print(a)
b <- b
print(b)
2+3
```

```
## [1] 2
## [1] 3
## [1] 5
```

4.1.3 Le paramètre "comment"

Les symboles devant les résultats de R peuvent être choisis en utilisant le paramètre "comment" :

```
``{r addition4, comment=">>", background='#FFFF00'}
# Commentaire : une addition avec R.
2+3
```
```

```r
# une addition avec R.
2+3
```

```
>> [1] 5
```

4.1.4 Les paramètres “eval” et “echo”

Nous pouvons être amené à présenter du code qu’il ne faut pas exécuter, il existe pour cela le paramètre “eval”. Ainsi, avec

```
``{r, eval=FALSE}
install.packages("rmarkdown")
```
```

l’installation du package “rmarkdown” n’est pas exécutée mais bien affichée :

```r
install.packages("rmarkdown")
```

Inversement, il est possible, grâce au paramètre “echo”, de ne pas afficher un code, mais ce qu’il génère, les lignes suivantes :

```
``{r, echo=FALSE}
cat("Mais quel code a été utilisé?")
```
```

deviennent :

```
## Mais quel code a été utilisé?
```

Encore une fois, je n’essaye pas d’être exhausti. Pour plus de précision, il y a le guide de référence ou le site de knitr. Vous y apprendrez, entre autres, qu’il est possible de créer des dépendances entre morceaux de code, ce que je n’aborde pas dans le document.

4.2 Les tableaux depuis R

Dans cette partie, j’utilise directement les morceaux de codes R sans présenter leur forme dans le document “.rmd”. Avec le package “knitr”, il est possible d’intégrer dans le texte une table créée sous R. Pour cela, il suffit de construire un objet R, le plus cohérent est de créer un “data.frame” avec R (éventuellement à partir d’un fichier importé), par exemple :

```r
experience <- paste0("exp",rep(1:3,each=5))
replicat <- rep(letters[1:5],3)
var1 <- 20*runif(15)
var2 <- var1+rnorm(15)
table1 <- data.frame(experience,replicat,var1,var2)
```

La fonction *kable()* du package “knitr” nous permet d’obtenir notre object *table1* en différents formats (dont “latex”, “html”, “markdown”).

```
knitr::kable(table1, caption="Table créée à partir du data.frame *table1*")
```

TABLE 2: Table créée à partir du data.frame *table1*

experience	replicat	var1	var2
exp1	a	14.226015	14.2850252
exp1	b	8.544408	9.0407947
exp1	c	5.187918	6.6099257
exp1	d	6.724766	6.6817903
exp1	e	0.915414	0.8012633
exp2	a	8.531672	8.2191216
exp2	b	11.914925	11.2426993
exp2	c	12.436396	11.8746123
exp2	d	15.053177	14.9803066
exp2	e	11.624848	13.4042969
exp3	a	8.313187	9.0549419
exp3	b	7.650244	7.8058431
exp3	c	6.998615	6.5548835
exp3	d	5.941371	7.6413846
exp3	e	18.031434	17.4319476

Pour en savoir plus, reportez-vous à la documentation de cette fonction.

4.3 Les figures produites avec R

Avec le package “rmarkdown”, il est très facile d’insérer les figures produites avec R dans un document, le plus simple est de regrader ce que nous génère l’appel à un plot simple.

```
plot(table1$var1,table1$var2)
```

J’obtiens la figure demandée qui est d’assez grande dimension et numérotée “Figure 3”. Pour formater une figure obtenue avec R, il existe des options pour contrôler, entre autres, sa taille, son alignement et sa légende (taille et alignement ne sont pas supportés pour la sortie Word). Pour visualiser l’ensemble des options disponibles, je vous invite à regarder la page 3 du guide de référence. Je vais changer la taille des figures, l’alignement et ajouter une légende

```
``{r, fig.cap="une figure avec une légende",
fig.height=4, fig.width=4,    fig.align='left'}
plot(table1$var1,table1$var2)
```
```

```
plot(table1$var1,table1$var2)
```

On peut alors profiter des facilités graphiques de R pour créer des figures élégantes. Je peux, par exemple, personnaliser la figure précédente avec quelques lignes supplémentaires.

```
par(bty="l", font=2)
plot(table1$var1,table1$var2, pch=15:19, col=rep(c(8,4,6),each=5), xlab="Mon axe des abscisses",
    ylab="Mon axe des ordonnées")
  legend("bottomright", letters[1:5], pch=15:19, bty="n")
```
```

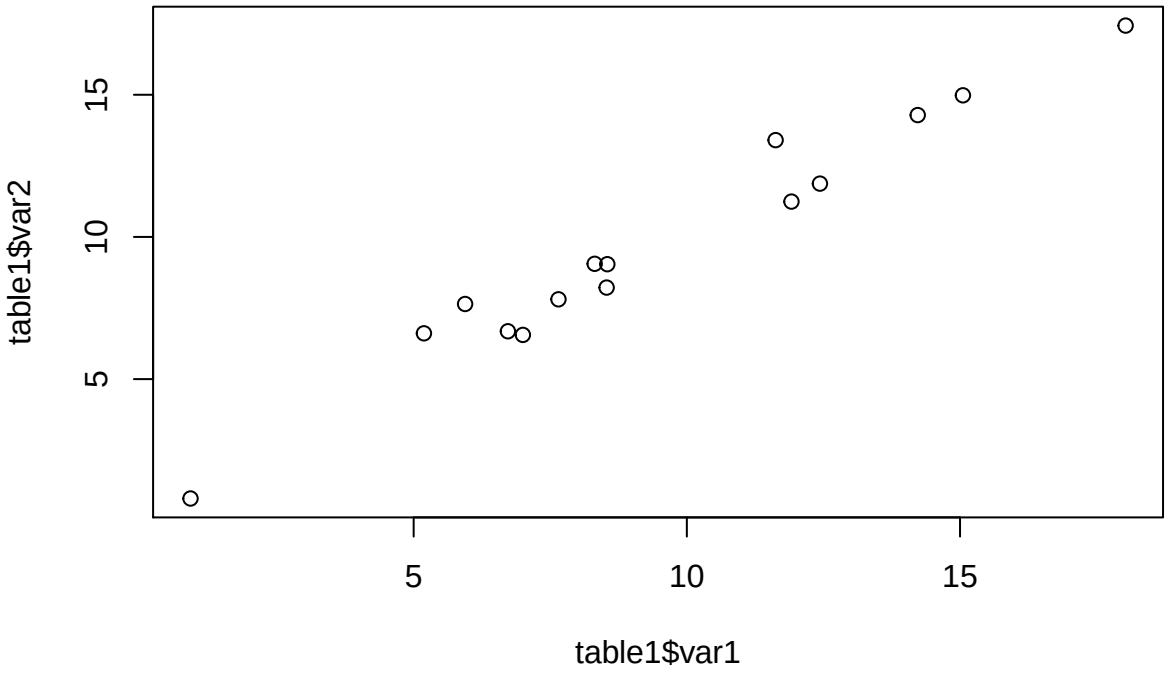

FIGURE 2 –

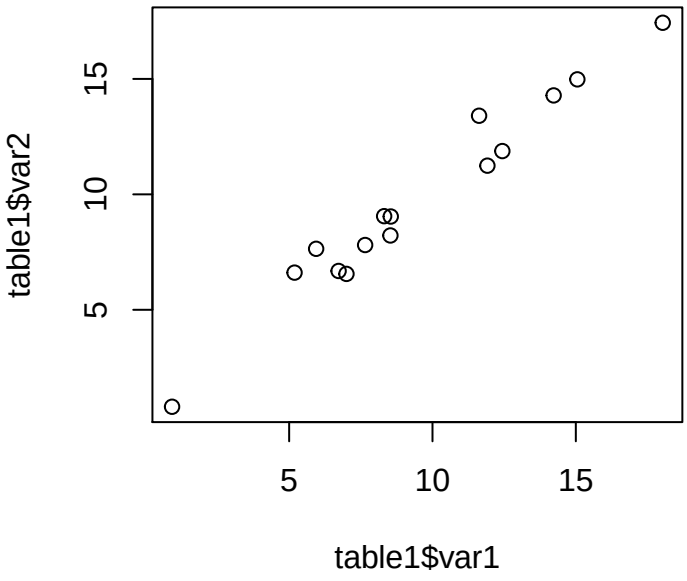

FIGURE 3 – une figure avec une légende

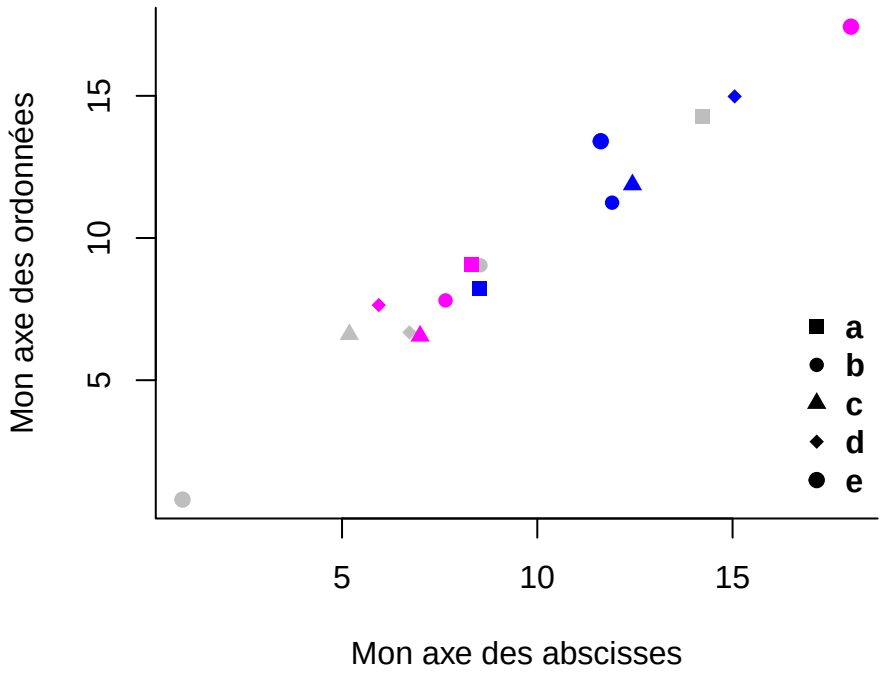

FIGURE 4 – Figure associé à la table 1 [...]. Le gris pour l'expérience 1, bleu pour la seconde et violet pour la troisième.

J'ai ajouté à la fin de la légende "\label{figcol}", je peux donc l'appeler en utilisant "\ref{figcol}" mai ce n'est valable que pour la sortie pdf. De mon point de vue, on peut considérée qu'on a une extension graphique en ligne de commande un peu moins verbeuse que PGF/TikZ.

```
kcircle <- function(rad=1,centre=c(0,0),from=0,to=2*pi,dt=0.001,...){
    pt <- seq(from, to, dt)
    polygon(centre[1]+rad*cos(pt),centre[2]+rad*sin(pt),...)
}
plot(c(-1.5,1.5), c(-1.5,1.5), asp=1, type="n", ann=FALSE,
    axes=FALSE, xaxs="i", yaxs="i")
kcircle(col="#AA000088")
for (i in 1:6) kcircle(1,c(cos(i*pi/3),sin(i*pi/3)), col="#0000AA88")
```

4.4 Une petite application.

Je souhaite faire un modèle linéaire avec la variable explicative *var1* et la variable à expliquer *var2* de notre *table1*. Nous créons alors un petit document sachant que les données évolueront. Je vais faire une inclusion donc utiliser du code R dans mon document.

```
mod1<-lm(var2~var1, data=table1)
summary(mod1)
```

```
##
## Call:
## lm(formula = var2 ~ var1, data = table1)
##
## Residuals:
```

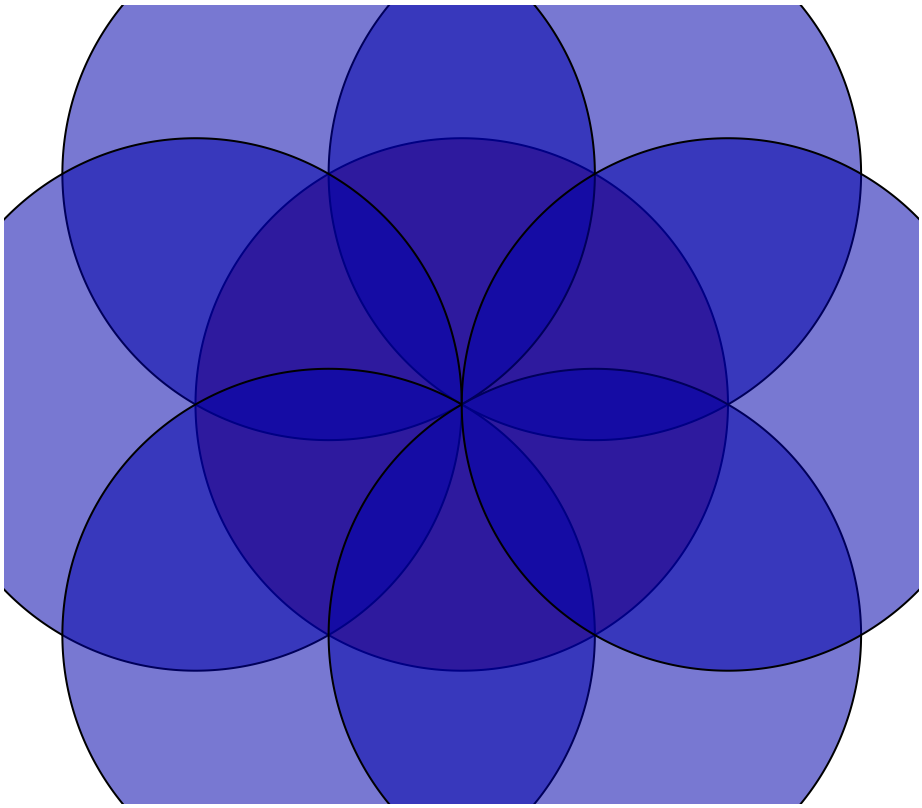

FIGURE 5 – Ma rosace

```
##     Min      1Q  Median      3Q     Max
## -0.8543 -0.6132 -0.1875  0.3219  1.6707
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.79414    0.51252   1.549    0.145
## var1         0.94104    0.04937  19.061 6.96e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.812 on 13 degrees of freedom
## Multiple R-squared:  0.9655, Adjusted R-squared:  0.9628
## F-statistic: 363.3 on 1 and 13 DF,  p-value: 6.962e-11
```

```
par(bty="l", mfrow=c(2,2))
plot(mod1)
```

Quatre lignes plus loin,… j'ai fini !

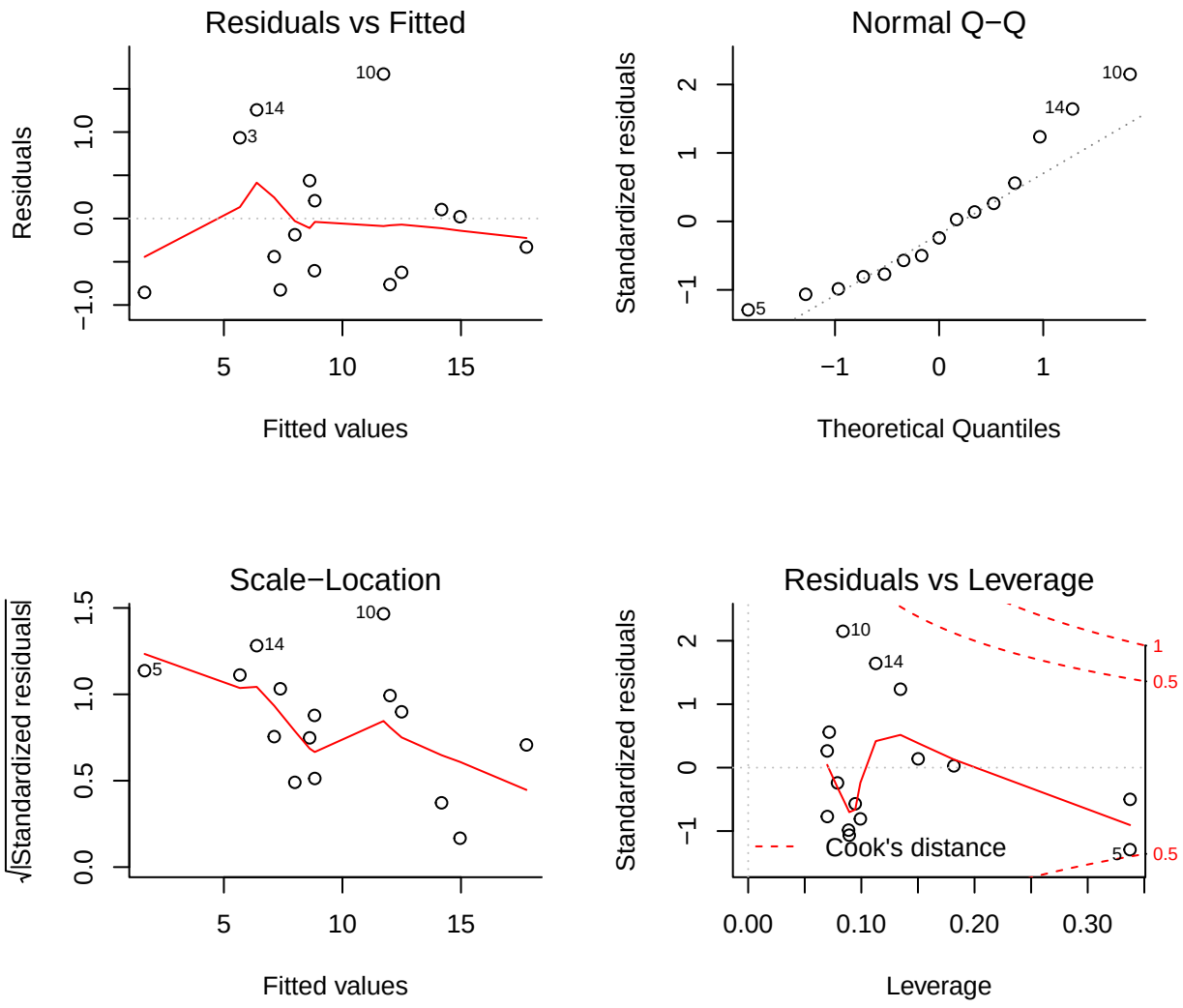

FIGURE 6 – Mon modèle linéaire

Références

Knauff, M., Nejasmic, J., 2014. An Efficiency Comparison of Document Preparation Systems Used in Academic Research and Development. PLoS One 9, e115069.

Lande, R., 1979. Quantitative Genetic Analysis of Multivariate Evolution , Applied to Brain : Body Size Allometry Russell Lande. Evolution (N. Y). 33, 402–416.

Oreskes, N., Shrader-Frechette, K., Belitz, K., 1994. Verification, Validation, and Confirmation of Numerical Models in the Earth Sciences. Science (80-.). 263, 641–646.