# knitr

## Produire des documents dynamiques

Pascal Bessonneau

Pour le Semin⟍R

05/2013

# *Sweave*

Sweave est apparu il y a quelques années comme une fonctionnalité importante de R.
L'idée est de pouvoir rédiger un document contenant les résultats et éventuellement le code correspondant.
Il correspond assez bien à l'esprit de *Litterate programming*. . .

# Principe de *Sweave*

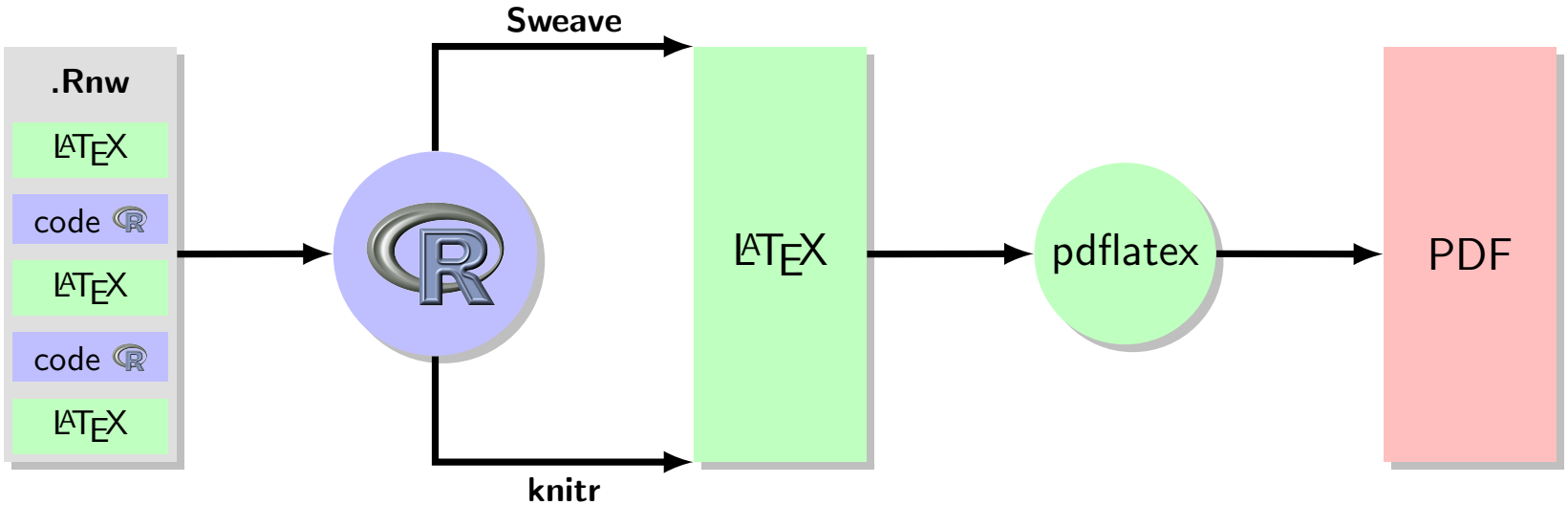
Un document unique va servir de base. Dans cet unique document, on trouve :

- le corps du texte
- le code source
- la mise en forme (éventuellement)

# *Sweave* et LaTeX

L'exemple typique est un fichier *Rnw*.
C'est un document LaTeXqui contient les codes sources R qui permettent de générer les résultats, les tableaux et les graphiques.
*Sweave* (et *knitr*) permet en deux phases de compilation de produire un PDF contenant l'ensemble d'une analyse statistique.

# *Sweave* et LaTeX

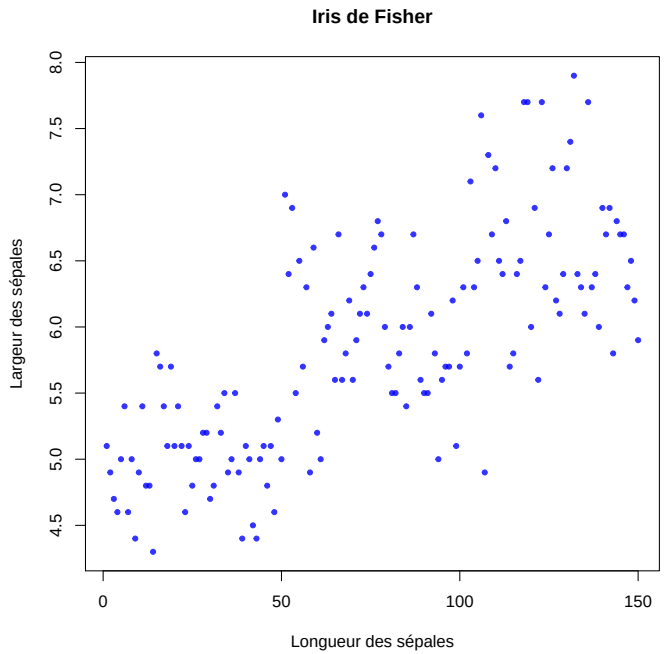


# *Sweave* et LaTeX

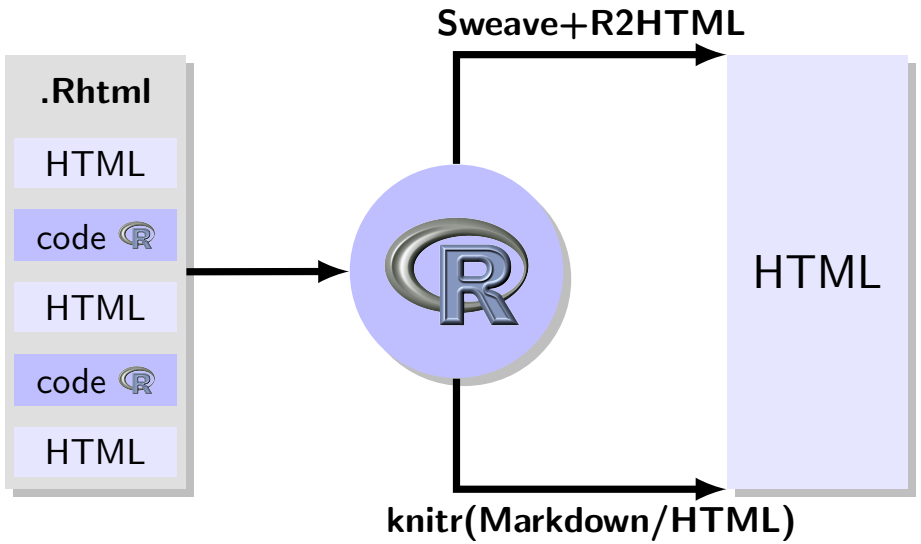
```
...
\begin{frame}[containsverbatim]
  \frametitle{\Sweave et \LaTeX}
<<plot_iris,out.height="0.5\textheight",out.width="0.5\textheight">>=
plot(
  iris$Sepal.Length, iris$Iris.Width,
  xlab="Longueur des sépales",
  ylab="Largeur des sépales",
  main="Iris de Fisher",
  col=rgb(0,0,1,0.8), pch=20
)
@
\end{frame}
...
```

# *Sweave* et LaTeX

```
plot(
  iris$Sepal.Length, iris$Iris.Width,
  xlab="Longueur des sépales",
  ylab="Largeur des sépales",
  main="Iris de Fisher",
  col=rgb(0,0,1,0.8),pch=20
)
```



# *Sweave* et *HTML*



# Rapidement...

Les développeurs ont adapté *Sweave* pour d'autres formats que LaTeX.
Ainsi le paquet *R2HTML* permet d'étendre *Sweave* à des fichiers HTML.
On trouve également un paquet *odfWeave* qui travaille au format Open Document (odf).

# En parallèle...

Si Sweave ne permet pas de réaliser des exports à l'esthétique très fouillé, de nombreux paquets permettent supplémentaires de générer du code LaTeX, HTML, ... pour certains objets de R avec une esthétique plus satisfaisante.
C'est le cas notamment des paquets *Hmisc*, *xtable*, *R2HTML*, ...
Ces paquets permettent de générer des exports de *data.frames*, des tables de régression qui sont directement inclus dans le document.

# En parallèle... (*xtable*)

| | Estimate | Std. Error | z value | Pr(>\|z\|) |
|---|---|---|---|---|
| (Intercept) | 1.3907 | 1.0901 | 1.28 | 0.2020 |
| age | -0.0432 | 0.0354 | -1.22 | 0.2219 |
| lwt | -0.0144 | 0.0067 | -2.16 | 0.0308 |
| smoke | 0.5539 | 0.3444 | 1.61 | 0.1078 |
| ptl | 0.5943 | 0.3483 | 1.71 | 0.0879 |
| ht | 1.8732 | 0.6908 | 2.71 | 0.0067 |
| ui | 0.7393 | 0.4567 | 1.62 | 0.1055 |
| ftv | 0.0234 | 0.1731 | 0.14 | 0.8923 |

TABLE : Regression logistique sur la naissance d'enfance prématurés

# En parallèle... (*xtable*)

Modèle 1 low age + lwt + smoke + ptl + ht + ui + ftv

Modèle 2 low age + lwt + ptl + ht + ui + ftv

| | Resid. Df | Resid. Dev | Df | Deviance |
|---|---|---|---|---|
| 1 | 181 | 208.75 | | |
| 2 | 182 | 211.33 | -1 | -2.58 |

TABLE : En incluant ou en excluant le tabagisme

# *knitr*

Ces packages permettaient un contrôle plus fin des exportations des résultats.
Mais *Sweave* restait assez monolithique. Par exemple :

- la mise en forme du code R n'était pas très simple à réaliser
- le dimensionnement des figures ne se faisait pas proportionnellement au dimensions du document
- il n'était pas possible de personnaliser les options des *chunks*
- des paquets comme *cacheSweave* étant présent mais séparement de *Sweave*
- ...

Pour améliorer la plasticité, faciliter la personnalisation des documents, Xie Yihui et l'aide de quelques autres ont réalisé un nouveau package *knitr*.

# *knitr*

Les formats de documents supportés par *knitr* sont :

- LaTeX
- HTML
- Markdown

Il n'est plus nécessaire de passer par des différents paquets et/ou drivers *Sweave* pour produire des documents en LaTeXet HTML. *knitr* offre une interface unifiée pour les différents langages.
le Markdown est un langage très simple qui s'adapte très bien au style *cahier de laboratoire*. Via l'utilitaire *pango*, le fichier compilé peut être transformé en de nombreux formats.

# Syntaxe de base

Le langage est très proche de *Sweave* $\pm$*R2HTML*.
Le but annoncé n'était pas de réinventer la roue mais de permettre une plus grande transparence, plasticité et utiliser les fonctionnalités de paquets essentiels comme *cacheSweave* dans un ensemble plus cohérent.

# Les fondamentaux

Par rapport à *Sweave*, *knitr* mets à disposition :

**des objets** ils définissent la syntaxe des chunks, les options par défaut, les styles, ... Ils sont accessibles et modifiables par l'utilisateurs directement par des accesseurs.

**les nouveautés** elles sont nombreuses... On notera le système de cache, la production de graphique, ...

**les *hooks*** ce sont les fonctions qui sont chargées du traitement des entrées et des sorties de R.

**les patterns** ce sont les définitions (avec des expressions régulières) de la syntaxe même des chunks

# Un exemple, *cache=TRUE*

Les résultats et les calculs réalisés au sein d'un *chunk* peuvent être stockés automtiquement. Objets, fonctions, paquets utilisés sont stockés. Un calcul long ne sera réalisé qu'une seule fois, la deuxième fois, le contenu du cache sera récupéré automatiquement ainsi que la sortie générée lors de la première compilation.
Le cache est renouvellé si le *chunk* est modifié ou si une valeur est modifiée par rapport à l'exécution précédente : la valeur calculée peut être la version de R, le nombre de lignes d'un objet, le mois courant, ...

# Un exemple, des *chunks* dans des programmes externes

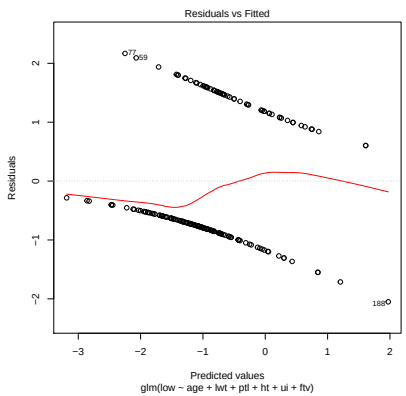
la commande *readchunk(fichier)* lit un fichier R quelconque et importe les parties du code intéressantes.
Par exemple, vous avez un script R qui contient différents essais de graphiques. Le troisième vous plaît. Il suffit de le faire précéder de cette syntaxe *## @knitr label*. La fin du chunk est indiqué par le début d'un autre chunk (quelconque, par exemple ## @knitr arretetoila).

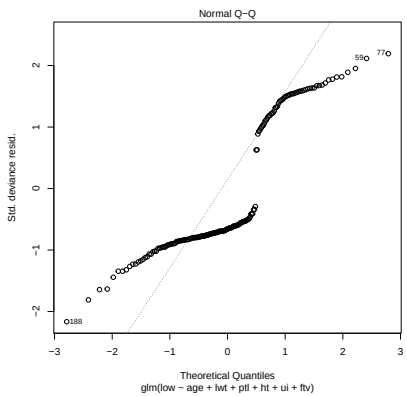
# La gestion des graphiques

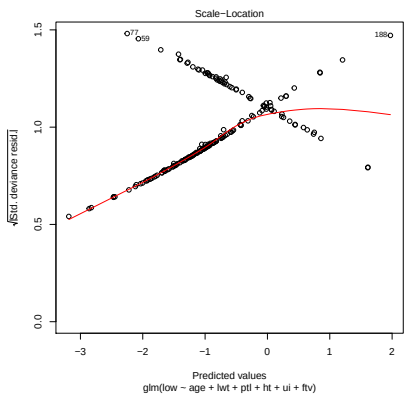
La gestion des graphiques est plus aisée. Ainsi on peux générer plusieurs graphiques au sein d'un même chunk et ne garder que le premier, le dernier, ...
Avec le paquet *animate*, il est possible de réaliser des animations dans les graphiques PDF.
Pour HTML ou Markdown, l'animation est réalisé par transformation en mp4 via *ffmpeg*. Il s'agit donc d'une vidéo.

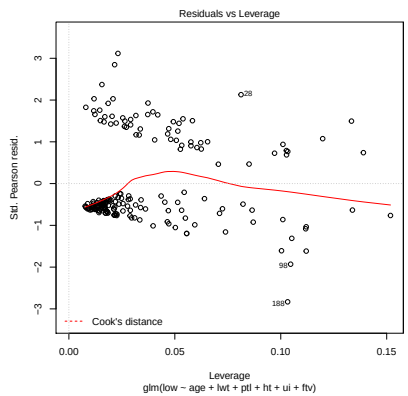
# Exemples d'une série de graphiques

```
plot(reg2, ask = F)
```



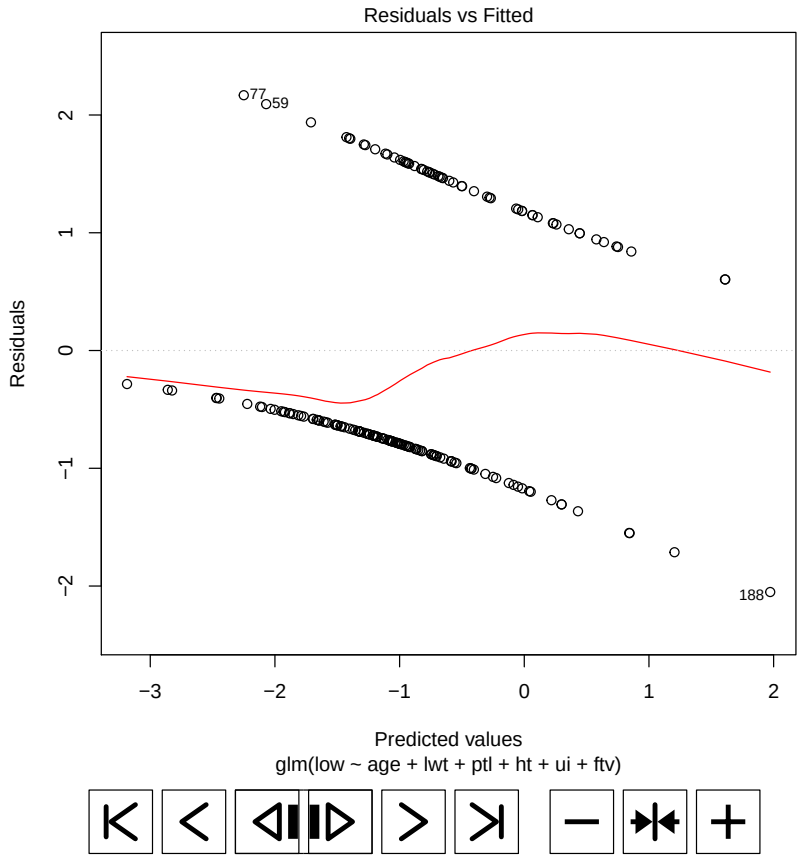



# Exemples d'une série de graphiques

```
plot(reg2, ask = F)
```

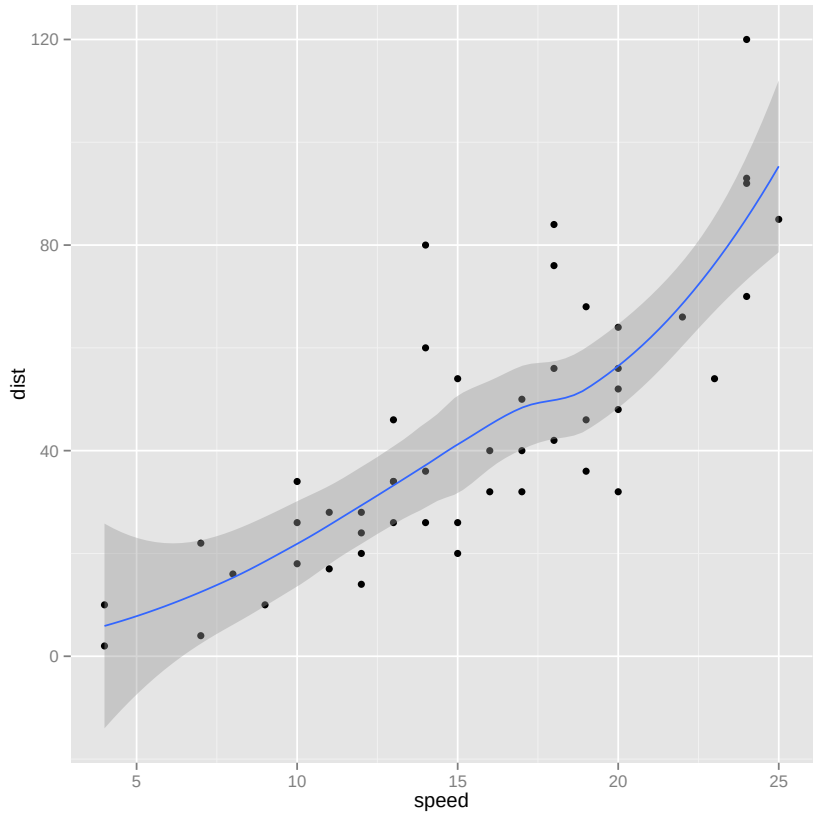


# La gestion des graphiques

Il y a environ une vingtaine de formats graphiques (*devices*). Mais il est possible d'en définir soit même. Par exemple pour personnaliser la taille des sorties.

```
<<custom-dev>>=
my_pdf = function(file, width, height) {
  pdf(file, width = width, height = height, pointsize = 10)
}
@
<<dev-demo, dev="my_pdf", fig.ext="pdf">>=
plot(rnorm(100))
@
```

# Un exemple, avec ggplot2



# Les objets par défaut

En fait des principaux objets disponibles sont :

| | |
|---|---|
| opts_knit | il contient toutes les options par défaut |
| knit_hooks | il contient les fonctions pour l'évaluation, la mise en forme |
| all_patterns | il contient la syntaxe de *knitr* pour chaque langage |

Des accesseurs permet de les manipuler : *set*, *get*, ...

# Les objets par défaut

Comme ce sont des objets R, ils sont manipulables assez facilement.
Ainsi on peut ajouter de nouvelles fonctions qui s'exécuteront en début et/ou en fin de bloc, modifier à la volée des options par défaut, ...

# Les objets par défaut

```
opts_chunk$set(fig.path = "Graphiques/beamer-",
    fig.align = "center", size = "footnotesize",
    fig.height = 7, fig.width = 7, fig.show = "hold")
```

# Les objets par défaut

```
# Exemple tiré du manuel de knitr
hook_rgl <- function(before, options, envir) {
    library(rgl)
    if (before || rgl.cur() == 0)
        return()
    name = fig_path("", options)
    par3d(windowRect = 100 + options$dpi * c(0,
        0, options$fig.width, options$fig.height))
    Sys.sleep(0.05)
    switch(options$dev, postscript = rgl.postscript(str_c(name,
        ".eps"), fmt = "eps"), pdf = rgl.postscript(str_c(name,
        ".pdf"), fmt = "pdf"), rgl.snapshot(str_c(name,
        ".png"), fmt = "png"))
    hook_plot_custom(before, options, envir)
}
```

# Les objets par défaut

```
hook_output <- knit_hooks$get("output")

knit_hooks$set(output=function(x,options)
{
    x <- gsub(".",",",x)

    hook_output(x, options)
}
```

# Les objets par défaut

Deux familles de hooks sont disponibles.
La première présentée dans l'exemple précédent est exécuté en début et en fin de bloc : les arguments sont before, options, envir.
La seconde famille de chunk est dédiée au traitement du contenu des blocs : message, warning, ... Les arguments sont différents : x, options. x contient une partie du chunk (code, sortie, ...)
Dans les deux cas, ce qui est retourné par la fonction qui sera affiché ou exécuté.
Pour une explication plus claire, la page sur les hooks du site de knitr.

# Evaluation des arguments des chunks

Les valeurs donnés aux options de chaque peuvent être des objets R.
Cela permet par exemple d'afficher/masquer un graphique ou un tableau en fonction de la date.
Ou bien de modifier la taille d'une figure, mettre en cache ou non, ...

# Evaluation des arguments des chunks

```
affiche <- TRUE
```

```
cat("Affiche")

## Affiche
```

```
affiche <- FALSE
```

```
cat("Affiche")
```

# Evaluation des arguments des chunks

```
<<plasticite1>>=
affiche <- TRUE
@
<<plasticite2,eval=affiche>>=
cat('Affiche')
@
<<plasticite3>>=
affiche <- FALSE
@
<<plasticite4,eval=affiche>>=
cat('Affiche')
@
```

# À retenir selon les niveaux

Pour ceux qui ne connaissaient pas *Sweave* et/ou *R2HTML*, il est possible en R de réaliser des documents dynamiques mélangeant votre texte avec des résultats provenant de R.

Les plus timides, tenterons le *Markdown* qui est très pratique comme cahier de laboratoire par exemple ou comme outil d'exploration des données.

# À retenir selon les niveaux

Les autres, plus aventureux, pourront générer dans leur langage favori HTML ou LaTeXdes documents de grande qualité. Même utilisation que précedemment, pour des articles ou des documents automatisés et éventuellement par lot. *knitr* offre des avantages non négligeables par rapport à *Sweave* pour une transition assez facile (d'ailleurs un convertisseur fichier *Sweave* vers fichier *knitr* est présent dans le paquet).

Les meilleurs d'entre nous (l'auteur exclu donc) pourront explorer les fonctionnalités des hooks pour produire des documents utilisant une syntaxe modifié qui pour s'insérer des CMS maison par exemple. A ce titre ils pourront s'inspirer de la fonction *knit2wp* qui permet de générer des entrées pour *WordPress*.

# Autres langages que R

La page de garde de knitr annonce une compatibilité avec d'autres langages (non testé) que R tels que awk,Python,...
Des démonstrations sont disponibles dans les démos de knitr.

# Références

Les ressources sur knitr sont très abondantes sur le web.

- Le site de knitr
- Un tutoriel intéressant par G. Williams]
- Un google groups maintenu par l'auteur de knitr
- L'incontournable StackOverFlow
- ...

A noter la sortie d'un livre par l'auteur de knitr, *Dynamic Documents With R and Knitr* chez CRC Press prévu pour juillet 2013.