Fiche TD avec le logiciel R : tdr78

Comment rédiger un rapport avec la commande `Sweave()` de R

J.R. Lobry

Cette fiche donne quelques indications pour rédiger un rapport intégrant des analyses statistiques et des graphiques produits par R.

Table des matières

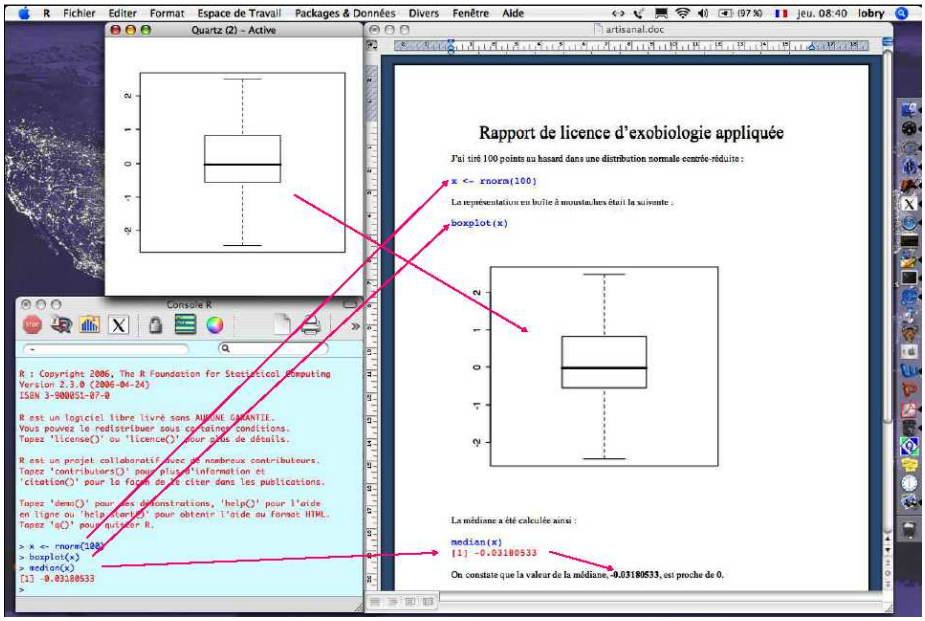
1 Introduction

Il existe deux grandes approches pour rédiger un rapport faisant appel à des analyses et à des graphiques produits sous R.

1. Une première approche, que nous qualifierons d'artisanale, sans aucun sens péjoratif, est destinée à la production d'un nombre très restreint de rapports tous très courts. Cette approche est adaptée si vous n'envisagez de produire que des rapports de façon sporadique et que vous n'êtes pas intéressé par une carrière scientifique au sens très large du terme[1].
2. Une deuxième approche, que nous qualifierons de professionnelle, est plus adaptée à la production de nombreux rapports, ou d'un rapport de taille conséquente. Elle consiste à intégrer directement le code R dans le rapport et à utiliser R pour produire les graphiques et les analyses à intégrer directement dans le document final (oui c'est possible !). Ceci garanti la parfaite *reproductibilité* de vos résultats.

2 Approche artisanale : copier/coller manuel

Cette approche consiste simplement à ouvrir simultanément R et votre traitement de texte favori pour copier/coller les résultats qui vous intéressent :



Voici quelques indications utiles pour obtenir un résultat acceptable :

[1] Contrairement à ce que l'on pourrait penser, de nombreux domaines des sciences dites, à tort, molles sont extrêmement exigeants pour ce qui est de la reproductibilité des résultats des analyses statistiques produites. Cette exigence n'est pas l'apanage de la recherche académique, elle concerne tous les secteurs où le *contrôle de la qualité* a un intérêt stratégique.

- Pour les graphiques, commencez par re-dimensionner la fenêtre graphique dans R jusqu'à ce que vous soyez satisfait du résultat. Copier/coller ensuite le graphique dans votre traitement de texte, puis re-dimensionnez le, de façon proportionnelle en x et y pour ne pas introduire de distorsions, pour obtenir la taille finale désirée.
- Pour les commandes R entrées dans la console, copiez/collez les dans le traitement de texte puis supprimez le caractère d'invite de commande (par défaut le caractère `'>'`) de la console R. Ceci vous permettra ultérieurement de facilement copier/coller dans l'autre sens, de votre traitement de texte vers R[2]. Définissez un style pour mettre en évidence qu'il s'agit de commandes R, par exemple en utilisant une police de caractères `non proportionnelle`, éventuellement de `couleur rouge` pour signifier qu'il s'agit de commandes R en entrée.
- Pour les résultats obtenus dans la console, définissez un style similaire à celui des entrées, mais changez la couleur, par exemple en `couleur bleue`, pour mettre en évidence qu'il s'agit de résultats, et non d'entrées.
- Pour intégrer des résultats numériques dans le corps du texte, il vous faudra faire un copier/coller de plus, veillez à maintenir la cohérence de l'ensemble !

Cette approche permet de générer facilement de petits documents de qualité, mais elle trouve très rapidement ses limites car il est extrêmement fastidieux de maintenir la cohérence entre les différents éléments du rapport. Supposez par exemple que vous ayez envie de modifier un graphique, pour par exemple lui rajouter un titre. Il vous faudra alors générer le graphique modifié sous R, mais comme vous avez oublié de noter quelque part la dimension du périphérique utilisé dans la version précédente, vous allez perdre du temps pour retrouver un résultat acceptable. Puis il faudra copier/coller le graphique dans le traitement de texte, et de nouveau, le re-dimensionner à la main (et encore une fois vous n'avez pas noté ce que vous aviez fait la dernière fois, encore du temps perdu à tâtonner pour retrouver quelque chose d'à peu près similaire). Mais ce n'est pas fini : le code R n'est plus cohérent avec le graphique, il faut une nouvelle opération manuelle de copier/coller, et de remise en forme dans le traitement de texte, pour assurer la cohérence. Et, bien entendu, si vous avez intégré des résultats numériques dans le corps du texte, il faudra une troisième opération manuelle de copier/coller pour mettre tout le document en cohérence. On arrive très rapidement aux limites de l'exercice.

3 Approche professionnelle : utilisation de `Sweave()`

3.1 Qu'est ce que `Sweave()` ?

La fonction `Sweave()` est une commande standard de R qui permet d'automatiser la production de documents en intégrant directement le code source R dans le document, la cohérence globale est alors automatiquement assurée. Comme `Sweave()` est une commande standard, il n'y a rien de particulier à installer si vous avez déjà R d'installé. Cependant, ceux qui travaillent sous Windows doivent être avertis de la note importante suivante :

[2]Vous pouvez le faire directement au niveau de la console en entrant la commande `options(prompt = " ")`.

Sous Windows : ne pas installer R dans le dossier `"Program Files"`, cela pose des problèmes à LaTeX pour gérer les blancs dans les chemins d'accès aux fichiers.

En effet, `Sweave()` va générer un fichier intermédiaire LaTeX qui comportera juste avant `\begin{document}` une directive du type[3] :

```
\usepackage{/Library/Frameworks/R.framework/Resources/share/texmf/Sweave}
\begin{document}
```

pour utiliser le fichier `Sweave.sty` qui est distribué avec R. Si R a été installé dans le dossier `"Program Files"`, cela posera un problème à LaTeX pour retrouver ce fichier parce qu'il y aura une espace dans le chemin d'accès. D'une façon générale, il est beaucoup plus simple de travailler sous des systèmes d'exploitations de type unix que sous Windows dès que l'on veut utiliser R de façon professionnelle. Ce n'est pas que cela soit impossible, le présent document est compilable sous Windows, mais il y a des analyses coût-bénéfice à prendre en compte. Ces dernières sont assez bien résumées de façon humoristique dans le folklore de R :

```
library(fortunes)
fortune("Knoppix")
```

```
Benjamin Lloyd-Hughes: Has anyone had any joy getting the rgdal package to compile
under windows?
Roger Bivand: The closest anyone has got so far is Hisaji Ono, who used MSYS
(http://www.mingw.org/) to build PROJ.4 and GDAL (GDAL depends on PROJ.4, PROJ.4
needs a PATH to metadata files for projection and transformation), and then
hand-pasted the paths to the GDAL headers and library into src/Makevars, running
Rcmd INSTALL rgdal at the Windows command prompt as usual. All of this can be
repeated, but is not portable, and does not suit the very valuable standard binary
package build system for Windows. Roughly: [points 1 to 5 etc omitted]
Barry Rowlingson: At some point the complexity of installing things like this for
Windows will cross the complexity of installing Linux... (PS excepting live-Linux
installs like Knoppix)
   -- Benjamin Lloyd-Hughes, Roger Bivand and Barry Rowlingson
      R-help (August 2004)
```

Dans la suite de ce document, nous supposerons que vous travaillez sous un système de type unix. Il en existe des distributions libres.

3.2 Pré-requis

Vous n'avez pas besoin de connaître LaTeX pour utiliser `Sweave()`, mais vous avez besoin d'avoir les outils nécessaires pour transformer un fichier source LaTeX en un document PDF, par exemple pdfLaTeX, qui est fourni avec la plupart des distributions LaTeX. Vérifiez que vous êtes capable de compiler le petit fichier (`mini.tex`) source LaTeX suivant :

mini.tex

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\begin{document}
\begin{Huge}Bonjour, ô mon$\delta$e !\end{Huge}
\end{document}
```

Pour ce faire suivez les étapes suivantes :

[3] Le chemin d'accès au fichier `Sweave.sty` sera codé en dur si vous activez l'option `stylepath` avec par exemple `Sweave("mini2.rnw", stylepath = TRUE)`.

1. Allez sur le site `http://pbil.univ-lyon1.fr/R/donnees/` pour enregistrer une copie du fichier `mini.tex` dans votre dossier de travail. Si vous n'avez pas de dossier de travail, créez en un.
2. Ouvrez le fichier `mini.tex` présent dans votre dossier de travail avec votre éditeur de texte favori. Le codage des caractères est en latin1 (ISO-8859-1), il peut être nécessaire de l'indiquer au moment de l'ouverture du fichier.
3. Vérifiez que dans votre éditeur de texte que le caractère 'ô' est bien présent dans la quatrième ligne : `\begin{Huge}Bonjour, ô mon$\delta$e !\end{Huge}`. Si ce n'est pas le cas, c'est sans doute que votre éditeur de texte n'a pas ouvert le fichier en latin1. Essayez encore.
4. Lancez la compilation en entrant la commande `pdflatex mini.tex` dans un terminal[4].

Vous devez obtenir un document PDF d'une page au format A4, numérotée 1 en bas, et sur le haut de laquelle est écrit en gros caractères :

Bonjour, ô monδe !

Ce petit test vous permet de vérifier deux choses :

1. Vous êtes capable de compiler un fichier source LaTeX qui contient des caractères accentués, comme ici le ô.
2. Le caractère grec delta minuscule, δ, est plaisant à lire. En effet, par rapport à la version initiale de 1986, le rendu du caractère δ a été fortement amélioré en 1992 (voir `http://www-cs-faculty.stanford.edu/~uno/cm.html`). Si le rendu du caractère δ n'est pas bon (*cf* ci-contre) , c'est sans doute que votre version de LaTeX est légèrement obsolète et qu'il faut absolument faire une mise à jour.

δ laid

δ beau

3.3 Vérifier que tout marche bien

R contient en standard un fichier d'exemple pour `Sweave()` qui s'appelle `Sweave-test-1.Rnw`. Lancez les exemples de `Sweave()` avec :

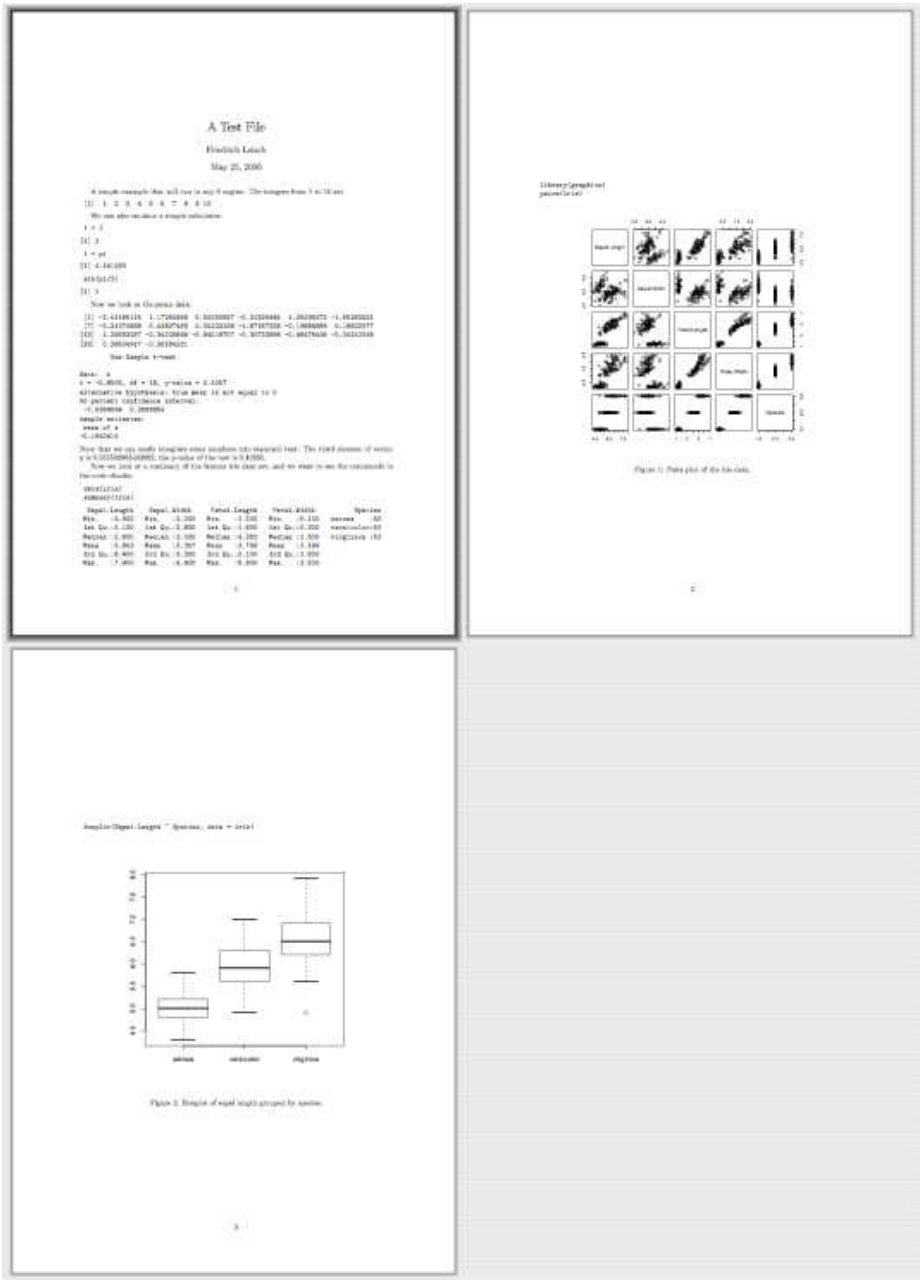
[4]Remarque : on peut aussi le faire directement sans sortir de R avec `system("pdflatex mini.tex")`

```
 example(Sweave, ask = FALSE)
Sweave testfile <- system.file("Sweave", "Sweave-test-1.Rnw", package = "utils")
Sweave ## enforce par(ask=FALSE)
Sweave options(device.ask.default=FALSE)

Sweave ## create a LaTeX file
Sweave Sweave(testfile)
Writing to file Sweave-test-1.tex
Processing code chunks ...
 1 : print term verbatim
 2 : term hide
 3 : echo print term verbatim
 4 : term verbatim
 5 : echo term verbatim
 6 : echo term verbatim eps pdf
 7 : echo term verbatim eps pdf

Vous pouvez maintenant lancer LaTeX sur 'Sweave-test-1.tex'

Sweave ## This can be compiled to PDF by
Sweave ## Not run: tools::texi2dvi("Sweave-test-1.tex", pdf=TRUE)
Sweave ## or outside R by
Sweave ## R CMD texi2dvi Sweave-test-1.tex
Sweave ## which sets the appropriate TEXINPUTS path.
Sweave
Sweave ## create an S source file from the code chunks
Sweave Stangle(testfile)
Writing to file Sweave-test-1.R

Sweave ## which can be sourced, e.g.
Sweave source("Sweave-test-1.R")
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 [1] -0.448385261 -1.153393786  0.004526230  0.941152965  0.709870066 -0.506996953
 [7]  0.664921651 -0.279653623 -1.761071346 -1.872803751  0.838177695  0.085461445
[13]  1.198015636  0.520078181 -2.030519006  3.437191494  0.227157076  0.235975436
[19] -0.872275630  0.791817401

        One Sample t-test

data:  x
t = 0.1295, df = 19, p-value = 0.8983
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.5528859  0.6258105
sample estimates:
 mean of x
0.03646230


Sweave ## Don't show:
Sweave if(!interactive()) unlink("Sweave-test-1*")
```

Si tout s'est bien passé, vous devez avoir maintenant dans votre dossier courant un fichier `Sweave-test-1.tex`, ainsi qu'un certain nombre de fichiers graphiques (au format eps et au format pdf) et un fichier de commandes R :

```
 list.files(pattern = "Sweave-test-1")
[1] "Sweave-test-1-006.eps" "Sweave-test-1-006.pdf" "Sweave-test-1-007.eps"
[4] "Sweave-test-1-007.pdf" "Sweave-test-1.R"       "Sweave-test-1.tex"
```

Utilisez pdfLaTeX pour compiler le fichier `Sweave-test-1.tex`, vous devez obtenir un fichier au format PDF de trois pages ressemblant à ceci :



Si vous êtes arrivé à produire ce fichier, c'est que vous avez tous les outils nécessaires pour utiliser `Sweave()` pour produire un rapport.

3.4 Intégration d'entrées et sorties de la console

Enregistrez une copie du fichier `mini2.rnw` dans votre dossier de travail, disponible au même endroit à `http://pbil.univ-lyon1.fr/R/donnees/`.

mini2.rnw

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\begin{document}
Essayons de calculer 2 + 2 :
<<essai2plus2>>=
2 + 2
@
\end{document}
```

L'extension `rnw` désigne par convention un fichier destiné à être traité par la commande `Sweave()` de R. Lancez la commande `Sweave()` en lui donnant comme argument le nom de ce fichier :

```
> Sweave("mini2.rnw")
Writing to file mini2.tex
Processing code chunks ...
 1 : echo term verbatim (label=essai2plus2)
Vous pouvez maintenant lancer LaTeX sur 'mini2.tex'
```

Vous pouvez alors compiler le fichier intermédiaire `mini2.tex` qui a été généré par `Sweave()` pour générer un document PDF contenant le texte suivant :

Essayons de calculer 2 + 2 :

```
> 2 + 2

[1] 4
```

Le fragments de code R à intégrer dans le document final (ici `2 + 2`) doit être encadré au minimum par `<<>>=` au début et `@` à la fin, et il ne doit pas il y avoir de caractères en début de ligne. Le résultat (ici `[1] 4`) sera automatiquement intégré dans le document final. Ici, nous avons en plus donné un nom au fragment de code R en utilisant `<<essai2plus2>>=` pour pouvoir le repérer facilement lors de la phase de compilation avec `Sweave()`.

Pour y voir plus clair, on décide de supprimer le caractère d'invite de commande de la console R (`>`) et de colorier les entrées et sorties de façon différente. Récupérez le fichier `mini3.rnw` :

mini3.rnw

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage{color}
\usepackage{pdfcolmk}

\begin{document}

\DefineVerbatimEnvironment{Sinput}{Verbatim}{formatcom = {\color[rgb]{0, 0, 0.56}}}
\DefineVerbatimEnvironment{Soutput}{Verbatim}{formatcom = {\color[rgb]{0.56, 0, 0}}}

<<options, echo = FALSE>>=
options(prompt = " ", continue = " ", width = 85)
@

Essayons de calculer 2 + 2 en couleur :
<<essai2plus2>>=
```

```
2 + 2
@
\end{document}
```

Lancez `Sweave()` sur ce fichier :

```
 Sweave("mini3.rnw")
Writing to file mini3.tex
Processing code chunks ...
 1 : term verbatim (label=options)
 2 : echo term verbatim (label=essai2plus2)
Vous pouvez maintenant lancer LaTeX sur 'mini3.tex'
```

Notez qu'il y a maintenant deux fragments de code :

1. Le premier qui s'appelle `options` et qui sert en autres à supprimer le caractère d'invite de commande.
2. Le deuxième qui s'appelle `essai2plus2` et qui calcule 2 + 2 comme précédemment.

Compilez le fichier `mini3.tex`, vous devez obtenir le résultat suivant :

Essayons de calculer 2 + 2 en couleur :

```
2 + 2
```

```
[1] 4
```

Notez que le premier fragment de code (`options`) n'apparaît pas dans le document final, c'est parce que nous l'avons commencé avec
`<<options, echo = FALSE>>=`,
et que `echo = FALSE` signifie que l'on ne veut pas que le code apparaisse dans le document final.

Les autres modifications que nous avons apportées s'adressent à LaTeX, nous avons demandé à utiliser les bibliothèques `color` et `pdfcolmk` pour pouvoir avoir des couleurs dans le document final. Nous avons aussi redéfini les styles `Sinput` et `Soutput` pour formater les entrées et sorties de code R.

3.5 Incorporation de résultats dans le corps du texte

Il est possible avec la commande `\Sexpr{}` d'insérer dans du texte la valeur d'une variable. Par exemple, le fichier source `mini4.rnw` :

mini4.rnw

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage{color}
\usepackage{pdfcolmk}

\begin{document}

\DefineVerbatimEnvironment{Sinput}{Verbatim}{formatcom = {\color[rgb]{0, 0, 0.56}}}
\DefineVerbatimEnvironment{Soutput}{Verbatim}{formatcom = {\color[rgb]{0.56, 0, 0}}}

<<options, echo = FALSE>>=
options(prompt = " ", continue = " ", width = 85)
```

Le volume V d'une sphÃ¨re de rayon r est donnÃ© par :

$$V(r) = \frac{4}{3}\pi r^3$$

```
V <- function(r) 4/3 * pi * r^3
V(1)
```

```
[1] 4.18879
```

Le volume d'une sphÃ¨re de rayon unitÃ© est donc de 4.18879020478639.

FIG. 1 – **Le syndrome de l'encodage de la mort qui tue**. Si dans votre document terminal vous obtenez des caractères étranges en lieu et place des caractères accentués, comme ci-dessus, c'est qu'un des acteurs impliqué dans sa production n'est pas au courant du fait qu'il faut utiliser l'encodage latin1. Qui est le coupable ? Il y a trois acteurs impliqués. Le premier c'est votre éditeur de texte favori, mais il ne peut pas être incriminé puisque vous avez déjà réussi à editer un texte avec des accents (*cf* pré-requis). Le deuxième, LaTeX, ne peut pas non plus être accusé puisque vous avez déjà réussi à produire un texte avec des accents (*cf* pré-requis). Reste R. Quand vous utilisez la commande `Sweave()`, celle-ci va lire le fichier `*.rnw` et écrire le fichier `*.tex`. Il faut aussi lui dire d'utiliser l'encodage latin1. Pour ce faire il suffit d'entrer la commande `options(encoding = "latin1")` dans R avant de lancer `Sweave()`.

```
@

Le volume $V$ d'une sphère de rayon $r$ est donné par :
$$
V(r) = \frac{4}{3}\pi r^3
$$
<<volsphere>>=
V <- function(r)  4/3*pi*r^3
V(1)
@
Le volume d'une sphère de rayon unité est donc de \Sexpr{V(1)}.
\end{document}
```

donne le résultat suivant :

```
 Sweave("mini4.rnw")
Writing to file mini4.tex
Processing code chunks ...
 1 : term verbatim (label=options)
 2 : echo term verbatim (label=volsphere)
Vous pouvez maintenant lancer LaTeX sur 'mini4.tex'
```

Le volume V d'une sphère de rayon r est donné par :

$$V(r) = \frac{4}{3}\pi r^3$$

```
V <- function(r) 4/3 * pi * r^3
V(1)
```

```
[1] 4.18879
```

Le volume d'une sphère de rayon unité est donc de 4.18879020478639.

Si tout s'est bien passé, tout ne devrait pas s'être bien passé, vous devriez normalement souffrir ici du syndrome de l'encodage la mort qui tue. Corrigez le problème si besoin est (*cf* figure 1).

Vous pouvez utiliser la fonction `round()` pour arrondir les résultats, par exemple `\Sexpr{round(V(1), 2)}` donnerait 4.19 dans le document final. Ceci permet donc d'insérer une, et une seule, valeur obtenue par le calcul dans R directement dans le document final. Mais il est également possible d'insérer dans le document final des tableaux, donc un grand nombre de valeurs, grace au package `xtable`. Nous ne donnerons qu'un exemple simple. Le fichier source suivant :

mini5.rnw

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}

\begin{document}
<<options, echo = FALSE>>=
library(xtable) # pour pouvoir produire des tables LaTeX dans R
data(crimtab)   # pour charger le jeu de données (voir ?crimtab)
colnames(crimtab) <- as.numeric(colnames(crimtab))/2.54 # taille en pouces
rownames(crimtab) <- as.numeric(rownames(crimtab))*10   # majeur en mmm
crimtab[which(crimtab == 0, arr.ind = TRUE)] <- NA      # pour virer les 0
crimtab <- crimtab[nrow(crimtab):1, ] # pour inverser l'ordre des lignes
matable <- xtable(x = crimtab, label = "Student",
  caption = "Les données utilisées par William Sealy Gosset,
  plus connu sous le nom de plume de 'Student' et du test statistique éponyme,
  en 1908 (\\textit{Biometrika} \\textbf{6}, 1-25). La collecte des
  données a été faite par MacDonell en 1902 (\\textit{Biometrika}
  \\textbf{1}, 177-227). Il s'agit d'une table de contingence obtenue
  en discrétisant la taille (exprimée ici en pouces, soit 2.54 cm, et
  portée en colonne) et la longueur du majeur de la main gauche
  (exprimée ici en mm et portée en ligne) de 3000 criminels adultes
  de sexe mâle écroués au pays de Galle et en Angleterre à la fin
  du XIX$\\mathrm{^e}$ siècle. Voir \\texttt{?crimtab} pour plus
  d'informations.")
# Notez les doubles \\ nécessaires dans R, c'est la "double escape rule"
print(matable, file = "Student.tex", size = "tiny", NA.string = ".")
# On veut des '.' au lieu des des NA
@
\input{Student.tex}

Les fameuses données utilisées par Student sont présentées dans la table \ref{Student}.

\end{document}
```

permet d'obtenir ceci :

```
 Sweave("mini5.rnw")
Writing to file mini5.tex
Processing code chunks ...
 1 : term verbatim (label=options)
Vous pouvez maintenant lancer LaTeX sur 'mini5.tex'
```

	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77
135	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.
134	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
133	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	1	.	.	.
132	.	.	.	.	.	.	.	.	.	.	1	1	.	1	.	3	.	.	.	.	.	.
131	.	.	.	.	.	.	.	.	.	.	.	1	1	.	.	.	.	.	.	.	.	.
130	.	.	.	.	.	.	.	.	.	.	3	.	1	.	1	.	2	1	.	.	.	.
129	.	.	.	.	.	.	.	.	.	.	.	1	2	2	.	1	1	.	.	.	.	.
128	.	.	.	.	.	.	.	.	.	.	1	2	3	1	8	5	3	1	1	.	.	.
127	.	.	.	.	.	.	.	.	.	1	1	7	5	5	8	2	2	.	.	.	.	.
126	.	.	.	.	.	.	.	.	.	1	.	3	5	7	8	6	3	1	1	.	.	.
125	.	.	.	.	.	.	1	.	.	1	3	12	11	8	6	8	.	2	.	.	.	.
124	.	.	.	.	.	1	1	1	2	7	12	4	7	7	1	.	.	1	.	.	.	.
123	.	.	.	.	.	.	.	4	8	10	13	20	23	6	5	.	.	.	.	.	.	.
122	.	.	.	.	.	1	2	5	6	23	17	16	11	8	1	1	.	.	.	.	.	.
121	.	.	.	.	.	.	2	4	13	22	28	15	27	10	4	1	.	.	.	.	.	.
120	.	.	.	.	.	1	4	8	19	42	39	22	16	8	2	2	.	.	.	.	.	.
119	.	.	.	1	1	2	12	10	27	32	35	19	10	9	3	1	.	.	.	.	.	.
118	.	.	.	1	.	2	11	15	35	41	34	29	10	5	1	.	.	.	.	.	.	.
117	.	.	.	.	2	9	17	30	37	48	45	24	9	9	2	.	.	.	.	.	.	.
116	.	.	.	2	1	4	13	37	39	48	38	27	12	2	2	.	.	1	.	.	.	.
115	.	.	.	.	5	11	17	33	57	38	34	25	11	2	.	.	.	.	.	.	.	.
114	.	.	.	3	4	9	29	56	58	26	22	10	11	.	.	.	.	.	.	.	.	.
113	.	.	1	.	5	10	24	26	39	26	24	7	2	.	.	.	.	.	.	.	.	.
112	.	.	3	2	7	21	30	38	29	27	20	4	1	.	.	.	.	.	.	.	.	1
111	.	.	3	3	12	22	26	24	26	24	7	4	1	.	.	.	.	.	.	.	.	.
110	.	.	2	6	12	15	31	37	27	17	10	6	.	.	.	.	.	.	.	.	.	.
109	.	.	.	2	6	14	24	27	14	10	4	1	.	.	.	.	.	.	.	.	.	.
108	.	.	2	5	6	14	27	10	7	1	2	1	.	.	.	.	.	.	.	.	.	.
107	.	1	2	4	9	14	16	15	7	3	1	2	.	.	.	.	.	.	.	.	.	.
106	.	.	1	4	5	9	14	6	3	1	.	.	1	.	.	.	.	.	.	.	.	.
105	.	.	1	3	7	6	4	3	1	3	1	.	1	.	.	.	.	.	.	.	.	.
104	.	1	1	2	3	3	4	3	3	.	.	.	.	.	.	.	.	.	.	.	.	.
103	.	1	1	3	2	2	3	5	.	.	.	.	.	.	.	.	.	.	.	.	.	.
102	.	.	2	2	2	1	.	2	.	1	.	.	.	.	.	.	.	.	.	.	.	.
101	.	.	.	1	3	1	.	1	1	.	.	.	.	.	.	.	.	.	.	.	.	.
100	1	.	.	1	2	.	2	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.
99	.	.	1	.	1	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
98	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
97	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
96	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
95	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
94	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Table 1: Les données utilisées par William Sealy Gosset, plus connu sous le nom de plume de 'Student' et du test statistique éponyme, en 1908 (*Biometrika* **6**, 1-25). La collecte des données a été faite par MacDonell en 1902 (*Biometrika* **1**, 177-227). Il s'agit d'une table de contingence obtenue en discrétisant la taille (exprimée ici en pouces, soit 2.54 cm, et portée en colonne) et la longueur du majeur de la main gauche (exprimée ici en mm et portée en ligne) de 3000 criminels adultes de sexe mâle écroués au pays de Galle et en Angleterre à la fin du XIX^{e} siècle. Voir `?crimtab` pour plus d'informations.

Les fameuses données utilisées par Student sont présentées dans la table 1.

Il se peut que dans votre document final vous ayez `table??` au lieu de `table 1`. C'est normal, nous avons fait référence à la table dans le texte avec `\ref{Student}` pour ne pas avoir à nous préoccuper de la numérotation des tables, c'est le travail de LaTeX, pas le notre. Mais pour pouvoir faire son travail correctement, LaTeX a besoin de compiler le document une première fois pour générer les tables, puis une deuxième fois pour résoudre les références. Il nous suffit donc d'invoquer LaTeX une deuxième fois et les références seront résolues.

3.6 Inclusion de graphiques

`Sweave()` permet d'inclure automatiquement des graphiques dans le document final (yes!). Commencez par créer dans votre dossier de travail courant un dossier nommé `figs` pour rassembler toutes les figures au même endroit et éviter d'encombrer votre répertoire avec trop de fichiers[5].

Le fichier source suivant :

mini6.rnw

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage{color, pdfcolmk}
\begin{document}
```

[5]Mais il est aussi intéressant de désobéir à ces directives. À votre avis, si le dossier `figs` n'existe pas, il y aura-t-il un problème au moment de la compilation sous R ou bien un problème au moment de la compilation avec LaTeX ?

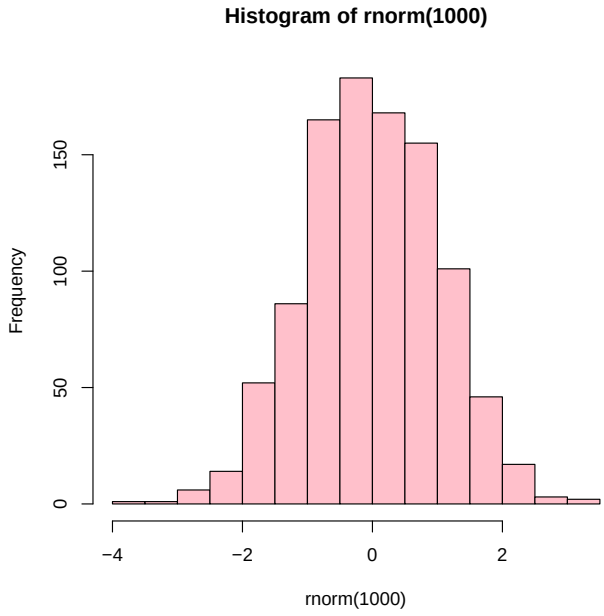
```
\DefineVerbatimEnvironment{Sinput}{Verbatim}{formatcom = {\color[rgb]{0, 0, 0.56}}}
\SweaveOpts{prefix.string = figs/essai, eps = FALSE, pdf = TRUE}
\setkeys{Gin}{width=0.8\textwidth}
Essayons de faire un histogramme :
<<essaifig, echo = T, fig = T>>=
hist(rnorm(1000), col = "pink")
@
\end{document}
```

devrait vous permettre d'obtenir ceci :

```
 Sweave("mini6.rnw")
Writing to file mini6.tex
Processing code chunks ...
 1 : echo term verbatim pdf (label=essaifig)
Vous pouvez maintenant lancer LaTeX sur 'mini6.tex'
```

Essayons de faire un histogramme :

```
hist(rnorm(1000), col = "pink")
```



Quelques commentaires sur les instructions utilisées ici. La ligne `\SweaveOpts{prefix.string = figs/essai, eps = FALSE, pdf = TRUE}` sert à dire que nous voulons ranger les fichiers contenant les figures dans le dossier `figs` et que leur nom commence par `essai-`, vous devez normalement avoir dans ce dossier un fichier s'appelant `essai-essaifig.pdf` :

```
 list.files(path = "figs", pattern = "essaifig")
[1] "essai-essaifig.pdf"
```

le nom du fichier a été complété en ajoutant le nom du fragment de code

(`essaifig`), puis l'extension `.pdf` pour signifier qu'il s'agit d'un fichier au format PDF. Notez que nous avons demandé à `Sweave()` de ne générer que des fichiers PDF avec l'option `eps = FALSE, pdf = TRUE`, parce que nous utilisons ici pdfLaTeX et que nous n'avons pas l'utilité des fichiers au format EPS.

Le fragment de code commence ici avec la ligne

```
<<essaifig, echo = T, fig = T>>=
```

qui signifie que le fragment de code produit une figure (`fig = T`).

> Une erreur fréquente consiste à commencer un fragment de code avec l'option `fig = T` mais que ce fragment de code ne produise en fait pas de figure. Le fichier PDF généré sera alors vide, ce qui conduira à une erreur quand pdfLaTeX essayera de l'inclure pour générer le document final.

Enfin, la ligne

```
\setkeys{Gin}{width=0.8\textwidth}
```

sert à contrôler la taille du graphique dans le document final. On signifie à LaTeX que l'on veut que le graphique occupe en largeur 80 % de la largeur d'une ligne courante.

3.7 Contrôle de la taille des graphiques

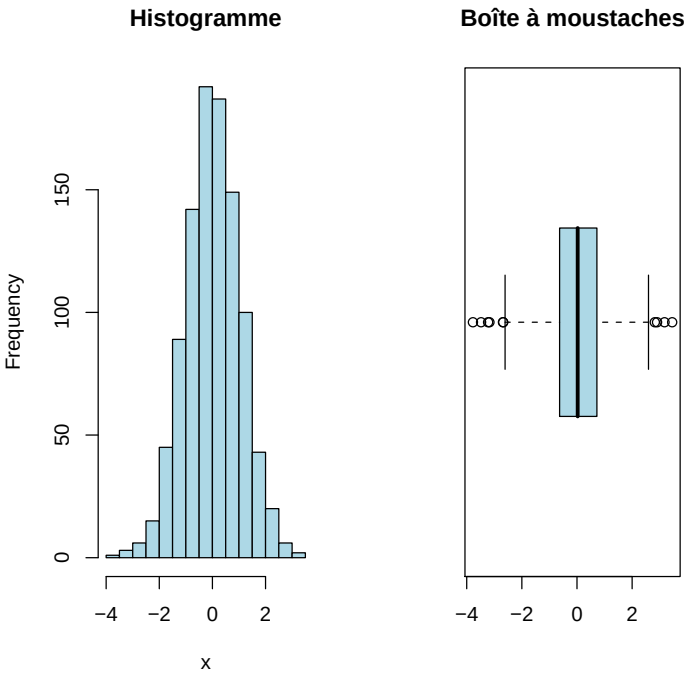
Voici une façon de procéder pour contrôler de façon fine et reproductible la dimension de vos graphiques. Commencez par mettre au point dans le mode interactif sous R votre graphique. Le plus simple pour cela est de créer un fichier source de code R appelé par exemple `mafigure.R` que vous exécutez avec `source("mafigure.R")`. Par exemple avec le fichier suivant :

mafigure.R

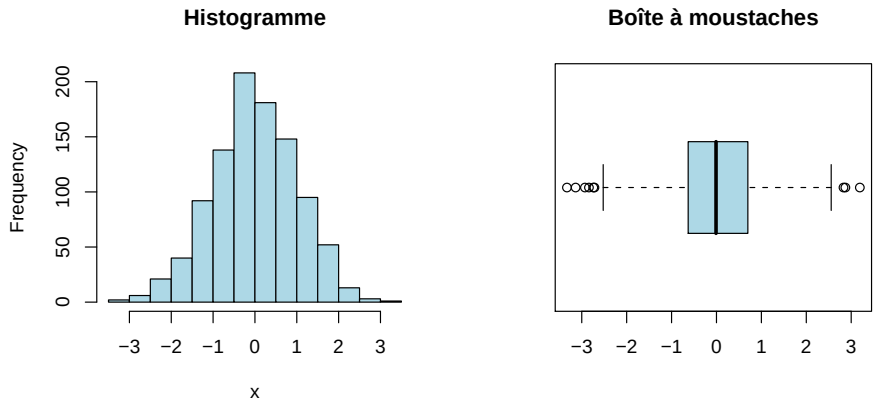
```
par(mfrow = c(1,2))
x <- rnorm(1000)
hist(x, col = "lightblue", main = "Histogramme")
boxplot(x, col = "lightblue", horizontal = TRUE, main = "Boîte à moustaches")
```

vous devez obtenir le résultat suivant :

```
source("mafigure.R")
```



Les dimensions générales du graphiques ne sont pas très adaptées, on préférerait avoir quelque chose de moins haut et de plus large. Quelque chose dans ce goût :



Toujours dans le mode interactif sous R, modifiez à la souris les dimensions de la fenêtre graphique, peaufinez le code source dans le fichier `mafigure.R`, jusqu'à ce que vous soyez satisfait du résultat. Utilisez alors la commande `par("din")` pour connaître les dimensions de la fenêtre graphique.

```
par("din")
[1] 6.993274 6.994751
```

La première valeur correspond à la largeur, la deuxième à la hauteur. Supposons que vous soyez satisfaits avec une largeur de 8 et une hauteur de 4. Le source suivant :

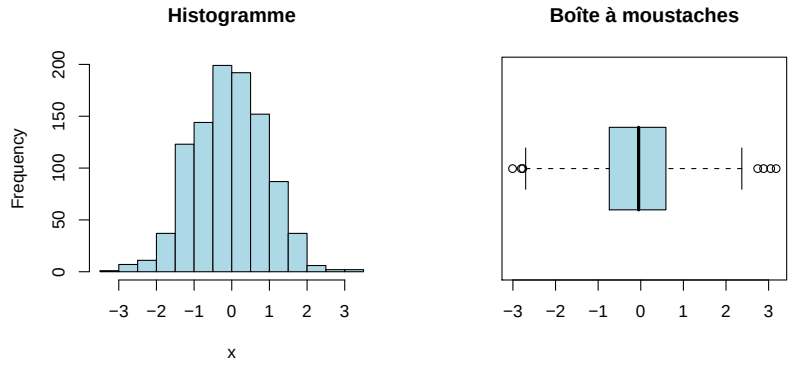
mini7.rnw

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\begin{document}
\SweaveOpts{prefix.string = figs/essai, eps = FALSE, pdf = TRUE}
\setkeys{Gin}{width=\textwidth}
Je veux une figure de 8 unités de large par 4 unités de haut
et qui tienne toute la largeur de la page dans le document final.

<<essaifig, echo = F, fig = T, width = 8, height = 4>>=
par(mfrow = c(1,2))
x <- rnorm(1000)
hist(x, col = "lightblue", main = "Histogramme")
boxplot(x, col = "lightblue", horizontal = TRUE, main = "Boîte à moustaches")
@
\end{document}
```

devrait vous permettre d'obtenir ceci :

```
 Sweave("mini7.rnw")
Writing to file mini7.tex
Processing code chunks ...
 1 : term verbatim pdf (label=essaifig)
Vous pouvez maintenant lancer LaTeX sur 'mini7.tex'
```

Je veux une figure de 8 unités de large par 4 unités de haut et qui tienne toute la largeur de la page dans le document final.



4 Conclusion

4.1 Pour aller plus loin

Vous avez l'essentiel des éléments pour pouvoir produire un rapport intégrant des résultats obtenus avec R. Vous trouverez sur le web des éléments d'information complémentaires sur `Sweave()`. Quelques pistes pour approfondir :

– La FAQ de `Sweave()` à `http://www.ci.tuwien.ac.at/~leisch/Sweave/FAQ.html` qui contient des liens vers la documentation (en anglais). Une tentative de traduction suit ci-après.

– L'article [2] de Friedrich Leisch, l'auteur de `Sweave()` (en anglais)
– Une démonstration de l'utilisation de `Sweave()` à `http://www.stat.umn.edu/~charlie/Sweave/` (en anglais)
– google(Sweave)
– Sous R faire `RSiteSearch("Sweave")`.
– La mise en page des entrées et sorties est faite par LaTeX et le paquet `fancyvrb` [5]. Nous avons déjà vu comment modifier le style des entrées et sorties en modifiant les environnements `Sinput` et `Soutput`. Ces deux environnements sont encapsulés dans l'environnement `Schunk`, qui est lui-même modifiable, par exemple, celui qui a été utilisé dans ce document est donné ci-dessous.

Schunk.tex

```
%
% Essai de modification de Schunk
%
\definecolor{gris10}{gray}{0.90}
\fvset{listparameters={\setlength{\topsep}{0pt}}}
\makeatletter\renewenvironment{Schunk}{
   \vspace{\topsep}
   \noindent\begin{lrbox}{\@tempboxa}\begin{minipage}{\columnwidth}}{\end{minipage}\end{lrbox}
   \fcolorbox{black}{gris10}{\usebox{\@tempboxa}}
   }{
   \vspace{\topsep}
}
\makeatother
```

4.2 À propos d'odfWeave

Il existe un paquet R nomé `odfWeave` [1] qui permet de générer des documents texte au format ODF[6] au lieu de fichiers au format PDF avec `Sweave`. Le fonctionnement est très similaire à celui de `Sweave`. Après installation du paquet, une commande du type :

```
odfWeave("mini2.odt", "mini2out.odt")
```

traitera les fragments de code R du fichier `mini2.odt` pour produire le fichier `mini2out.odt`. Vous trouverez à `http://pbil.univ-lyon1.fr/R/donnees/` les fichiers d'exemples suivants permettant de faire l'équivalent de ce que nous avons vu avec `Sweave` :

`mini2.odt` Calcul de 2 + 2.

`mini3.odt` Calcul de 2 + 2 en couleur.

`mini4.odt` Inclusion de résultat simple dans le corps du texte.

`mini5.odt` Génération d'une table à partir de données.

`mini6.odt` Inclusion d'une figure.

`mini7.odt` Contrôle de la taille des figures incluses.

La principale différence par rapport à `Sweave` est que le document final étant au format ODF, il est très facile de le modifier avec un traitement de texte comme `OpenOffice`. Le gros inconvénient est que l'on n'a plus à sa disposition tous les outils de LaTeX.

[6] Open Document Format, c'est le format utilisé par OpenOffice.

5 Traduction de la FAQ de `Sweave()`

Cette section est une traduction de la foire aux questions (FAQ) de Friedrich Leisch dans sa version du 4 janvier 2006. Pour une version plus à jour se référer au document source (en anglais).

5.1 Où trouver le manuel et des informations à propos de `Sweave` ?

La dernière version du manuel est toujours disponible sur la page de base de `Sweave` à l'adresse : `http://www.ci.tuwien.ac.at/~leisch/Sweave` Vous trouverez également sur ce site plusieurs fichiers d'exemples, ainsi que des fragments de code utiles de la FAQ. Il y de plus plusieurs articles à propos de `Sweave` comme l'article dans *CompStat* [2] et deux articles de *R News* [3, 4].

5.2 Comment faire pour que `Emacs` reconnaisse automatiquement des fichiers au format `Sweave` ?

Les versions récentes de ESS (Emacs speaks statistics, `http://ess.R-project.org`) reconnaissent automatiquement les fichiers avec l'extension `.Rnw` comme étant des fichiers `Sweave` et activent les modes adéquats. Suivre les instructions sur la page de base de ESS pour installer ESS sur votre ordinateur.

5.3 Comment exécuter `Sweave` à partir d'un interpréteur de lignes de commandes

Il peut être utile (par exemple pour écrire des fichiers de type `makefile`) de lancer `Sweave` directement à partir d'un interpréteur de lignes de commandes plutôt que de lancer manuellement R puis d'utiliser la fonction `Sweave()`. Ceci peut être réalisé facilement avec un script du type :

```
#!/bin/sh
echo "library(\"utils\"); Sweave(\"$1\")" | R --no-save --no-restore
```

Une solution plus élaborée, qui enchaîne en plus la compilation LaTeX, a été écrite par Gregor Gorjanc et est disponible à : `http://www.bfro.uni-lj.si/MR/ggorjan/software/shell/Sweave.sh`.

5.4 Pourquoi est ce que LaTeXne trouve pas mes figures EPS et PDF quand il y a un point dans le nom des fichiers ?

Sweave utilise pour gérer les fichiers graphiques le paquet LaTeX standard `graphicx`. Ce paquet utilise automatiquement des fichiers EPS pour du LaTeXstandard et des fichiers PDF pour PDFLaTeX si le nom du fichier graphique à inclure ne comporte pas d'extension (c'est à dire pas de point). C'est pour cela que vous risquez d'avoir des problèmes pour gérer les graphiques si le nom de votre fichier `Sweave` contient des points surnuméraires : `toto.Rnw` est OK, mais `toto.bis.Rnw` ne l'est pas.

5.5 Pourquoi `Sweave` créé par défaut à la fois des fichiers EPS et PDF ?

Le paquet LaTeX `graphicx` à besoin de fichier EPS pour du LaTeXpur, mais de fichiers PDF pour PDFLaTeX (ce dernier sachant également gérer des fichiers PNG et JPEG). `Sweave` génère automatiquement des fichiers EPS et PDF pour que l'utilisateur puisse utiliser librement `latex` ou bien `pdflatex` pour produire son document final.

5.6 Pourquoi les fragments de code sans figure donnent une erreur LaTeX ?

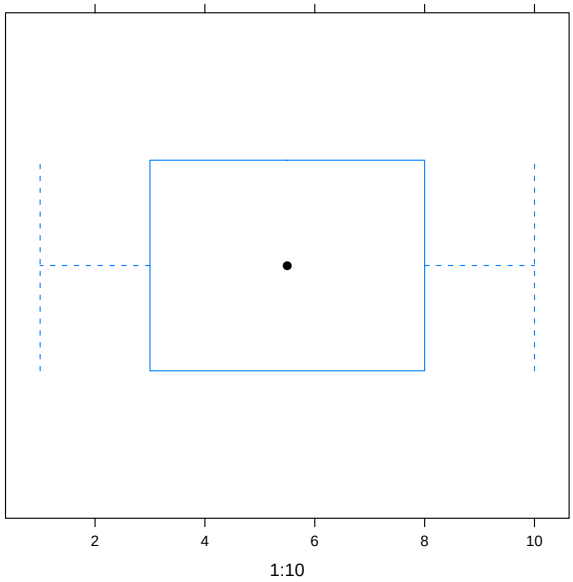
Quand un fragment de code avec `fig=true` ne fait appel à aucune fonction graphique de R, des fichiers EPS et PDF non valides sont produits. `Sweave` ne peut pas deviner si un fragment de code produit effectivement une figure ou non, il va essayer d'inclure le fichier graphique invalide, ce qui conduira inéluctablement à une erreur lors de la compilation LaTeX.

5.7 Pourquoi les graphiques `lattice` ne marchent pas ?

Les commandes dans le paquet R `lattice` ont un comportement différent des commandes graphiques standard : les commandes `lattice` retournent un objet de la classe `trellis`, et leur affichage effectif est effectué par la méthode `print` de cette classe. Il suffit d'encapsuler les appels aux fonctions dans un `print()`, par exemple :

```
<<fig=TRUE>>=
library(lattice)
print(bwplot(1:10))
@
```

```
library(lattice)
print(bwplot(1:10))
```



Les futures versions de `Sweave` auront peut être des moyens plus automatiques de traiter les graphiques du paquet `lattice`.

5.8 Comment faire pour avoir des graphiques `lattice` en noir et blanc ?

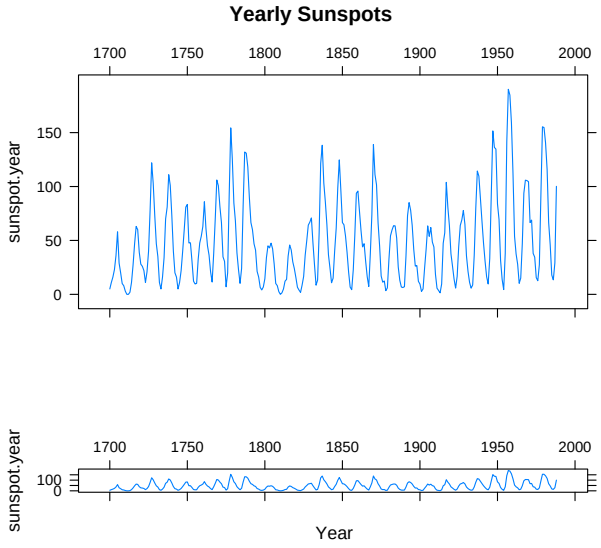
Quelle est la façon la plus élégante de produire des graphiques `lattice` en noir et blanc en conjonction avec `Sweave` ? Comment faire pour que cette option soit globale pour toutes les figures du document ?

Une réponse de Deepayan Sarkar : je fais quelque chose dans ce goût dans l'initilisation du document :

```
<<...>>
library(lattice)
ltheme <- canonical.theme(color = FALSE)     ## in-built B&W theme
ltheme$strip.background$col <- "transparent" ## change strip bg
lattice.options(default.theme = ltheme)      ## set as default
```

Essai :

```
library(lattice)
ltheme <- canonical.theme(color = FALSE)
ltheme$strip.background$col <- "transparent"
lattice.options(default.theme = ltheme)
data(sunspot.year)
plot <- xyplot(sunspot.year ~ 1700:1988, xlab = "", type = "l",
    scales = list(x = list(alternating = 2)), main = "Yearly Sunspots")
print(plot, position = c(0, 0.3, 1, 0.9), more = TRUE)
print(update(plot, aspect = "xy", main = "", xlab = "Year"), position = c(0,
    0, 1, 0.3))
```



5.9 Comment insérer plusieurs figures à partir d'un seul fragment de code ?

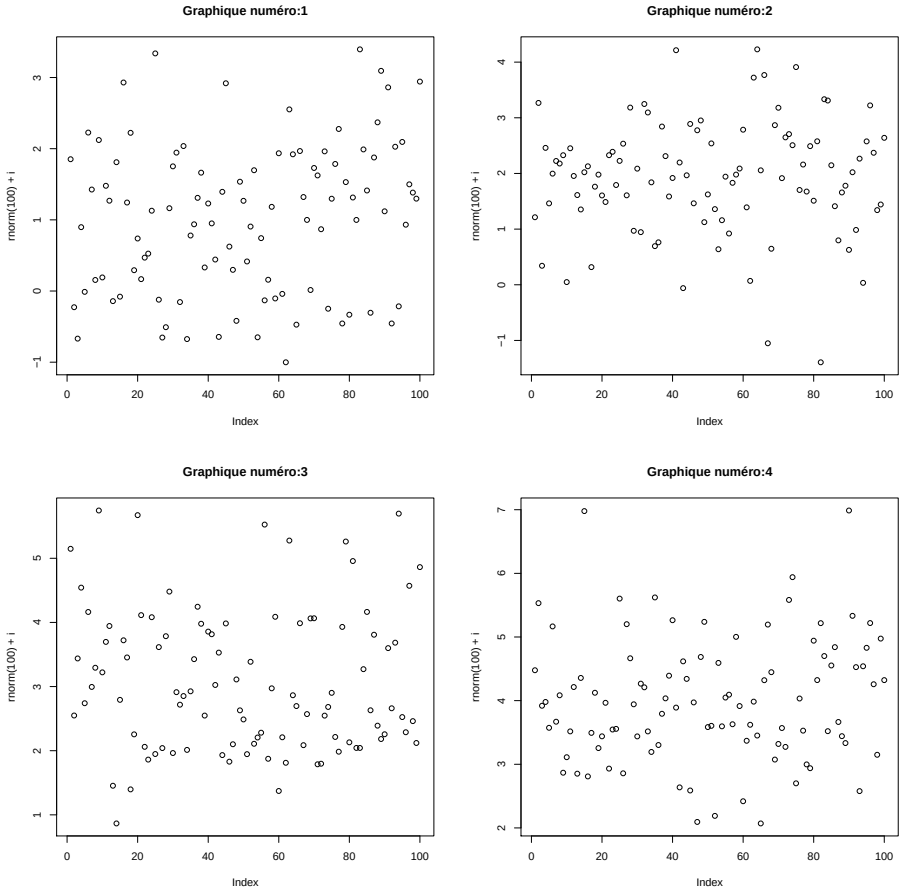
Imaginions que vous vouliez produire plusieurs figures dans une boucle du type :

```
<<fig=TRUE>>=
for(i in 1:4) plot(rnorm(100)+i)
@
```

Ceci **ne** pourra **pas** marcher parce que `Sweave` n'autorise **qu'une seule figure** par fragment de code. La raison en est que `Sweave` ouvre un périphérique graphique avant chaque fragment de code et le ferme juste après. Mais si vous avez besoin d'insérer tout une série de graphiques vous pouvez y arriver en générant du LaTeX directement, par exemple :

```
<<results=tex,echo=FALSE,fig=F>>=
for(i in 1:4){
    fichier <- paste("figs/monfichier", i, ".pdf", sep = "")
    pdf(file = fichier)
    plot(rnorm(100) + i, main = paste("Graphique numéro", i, sep = ":"))
    dev.off()
    cat("\\noindent\\includegraphics[width=0.5\\textwidth]{", fichier, "}\n", sep="")
}
@
```

Essayons :



5.10 Comment ranger tous les fichiers graphiques produits par `Sweave` dans un sous-dossier plutôt qu'au même niveau que le fichier `Sweave` ?

Après : `\SweaveOpts{prefix.string=figs/toto}`, `Sweave` rangera toutes les figures dans le dossier `figs` et leur nom commenceront par `toto` (au lieu du nom du fichier `Sweave`). Le dossier `figs` doit exister avant de lancer la fonction `Sweave()`.

5.11 Comment changer les paramètres graphiques `par()` pour plusieurs fragments de code ?

Comme chaque fragment de code graphique ouvre un nouveau périphérique EPS ou PDF, un appel à `par()` n'aura d'effet que pour le fragment de code courant. Si vous voulez changer les les valeurs par défaut, il est plus facile de définir une fonction pour factoriser le code plutôt que de répéter les mêmes appels à `par()` dans chaque fragment de code graphique.

L'effet de :

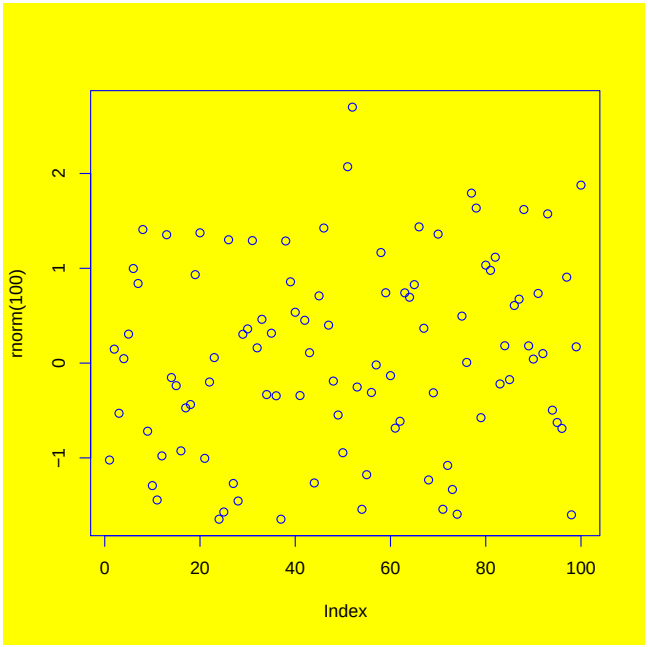
```
<<...>>=
```

```
opar <- par(no.readonly = TRUE)
options(SweaveHooks = list(fig = function() par(bg = "yellow", fg = "blue")))
@
```

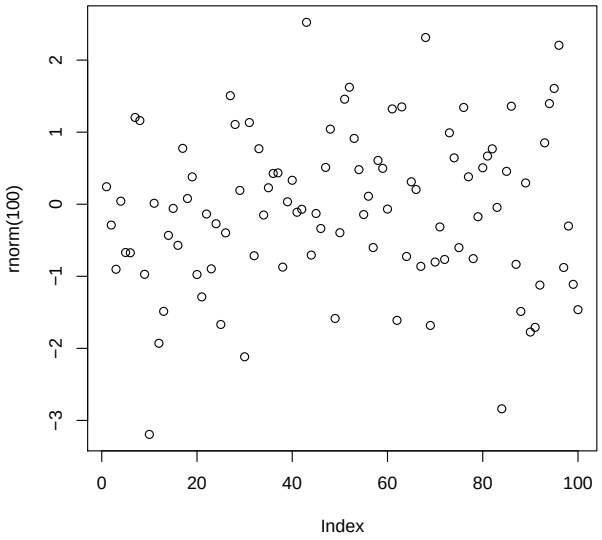
redéfinit les paramètres graphiques `bg` et `fg` pour tous les fragments de code suivants. N'oubliez de restaurer les valeurs par défaut à la fin du fichier `Sweave` sauf si vous voulez les utiliser comme option globale pour tous les documents `Sweave`.

Essayons :

```
options(SweaveHooks = list(fig = function() par(bg = "yellow", fg = "blue")))
plot(rnorm(100))
```




```
options(SweaveHooks = list(fig = function() par(opar)))
plot(rnorm(100))
```



5.12 La compilation LaTeX ne marche pas sous Windows

Si vous pouvez créer le fichier `.tex` en lançant la fonction `Sweave()` dans R, mais que vous n'arrivez pas à transformer le fichier `.tex` en `.dvi` ou `.pdf`, c'est sûrement à cause d'une espace dans le chemin d'accès de votre installation de R. Si le chemin d'accès à R une espace cela peut poser des problèmes parce que les programmes comme `tex` ou `latex` ne gèrent pas bien les espaces dans les chemins d'accès. Deux solutions possibles :

1. Installer R à un endroit dont le chemin d'accès ne possède pas d'espace.
2. Copier le fichier `Sweave.sty` dans un dossier accessible par LaTeXou dans le dossier qui contient votre fichier `Sweave` et ajoutez une directive `\usepackage{Sweave}` dans le préambule de votre fichier `Sweave`.

5.13 Comment changer le style des entrée et sorties de fragment de code ?

`Sweave` utilise le paquet LaTeX `fancyvrb` pour formater toutes les entrées et sorties de code. Ce paquet est très puissant et flexible pour contrôler finement de l'aspect du texte dans un environnement verbatim. Si vous voulez changer l'aspect par défaut, lisez la documentation du paquet `francyvrb` et modifiez les définitions des environnements `Sinput` et `Soutput`.

5.14 Comment changer la longueur des lignes des entrées et sorties de fragments de code

`Sweave` utilise la façon standard de spécifier la longueur des lignes de code avec `options(width)`. Par exemple, après `options(width = 40)` les lignes de

code seront formatées pour avoir, dans la mesure du possible, au plus 40 caractères.

5.15 Peut on utiliser `Sweave` pour des fichiers HTML ?

Le paquet `R2HTML` contient un pilote pour utiliser `Sweave` en conjonction avec du HTML plutôt que du LaTeX.

5.16 Après avoir chargé le paquet `R2HTML`, `Sweave` ne marche plus !

Le paquet `R2HTML` définit un pilote `Sweave` pour les fichiers au format HTML, et après cela la syntaxe pour l'HTML vient avant celle de la syntaxe par défaut dans liste des syntaxes explorées.

`options(SweaveSyntax=`SweaveSyntaxNoweb")"

ou bien l'appel de la fonction `Sweave()` par :

`Sweave(..., syntax=`SweaveSyntaxNoweb")"

force l'utilisation de la syntaxe par défaut, même après avoir chargé le paquet `R2HTML`.

5.17 Pourquoi `Sweave` enlève tous mes commentaires du code ? Pourquoi est ce qu'il ne respecte pas mes sauts à la ligne ?

Afin de savoir où insérer les sorties dans le code, `Sweave()` exécute tous les fragments de code via l'analyseur syntaxique de R. Les fragments de code en entrée que vous voyez sont le résultat de l'évaluation du code via les fonctions `parse()` et `deparse()`, qui enlèvent tous les commentaires et reformattent les retours à la ligne. L'option `keep.source=T` vous permet de neutraliser ce comportement, mais vous êtes alors entièrement responsable de la mise en page du code. Il y a essentiellement deux cas où il est intéressant de le faire :

1. Pour conserver les commentaires du code R dans le document final.
2. Pour éviter de déborder à droite dans le document final quand vous avez des arguments du type `main = "Une_très_longue_chaîne_de_caractères"` qui sont mal gérés dans le code.

Références

[1] Max Kuhn and Steve Weaston. *odfWeave : Sweave processing of Open Document Format (ODF) files*, 2009. R package version 0.7.10.

[2] F. Leisch. Sweave : Dynamic generation of statistical reports using literate data analysis. *Proceedings in Computational Statistics*, Compstat 2002 :575–580, 2002.

[3] Friedrich Leisch. Sweave, part I : Mixing R and LaTeX. *R News*, 2(3) :28–31, December 2002.

[4] Friedrich Leisch. Sweave, part II : Package vignettes. *R News*, 3(2) :21–24, October 2003.

[5] T. Van Zandt. *The 'fancyvrb' Package. Fancy Verbatims in LaTeX*, 1998. http://ctan.tug.org/tex-archive/macros/latex/contrib/fancyvrb.