

# Exemples de code Sweave

Loic PONGER

29 mars 2012

## Table des matières

<b>1</b>	<b>Premiers morceaux de codes</b>	<b>2</b>
<b>2</b>	<b>Du code un peu plus joli</b>	<b>2</b>
<b>3</b>	<b>Du code pour des figures</b>	<b>4</b>
<b>4</b>	<b>Du code pour des tables</b>	<b>6</b>
<b>5</b>	<b>Du code avec cacheSweave</b>	<b>6</b>

## 1 Premiers morceaux de codes

Un petit morceau de code sans options :

```
> #comments
> data=rnorm(100,0,1)
> summary(data)

      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.76000 -0.80560  0.02896 -0.02693  0.60310  2.05000
```

Un petit morceau de code avec `echo=F` :

```
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-3.00100 -0.37230  0.15470  0.07807  0.66210  2.39000
```

Un petit morceau de code avec `echo=F` et `results=hide` :

On ne voit pas grand chose, évidemment!!! Il est possible d'insérer le résultats de commandes R dans du texte avec la commande `. La moyenne des valeurs aléatoires est -0.051 alors que la variance est 1.327.`

## 2 Du code un peu plus joli

Un petit morceau de code pour visualiser la longueur des lignes R :

```
> data[1:100]

 [1] -2.27118096 -0.03242496 -0.42328025  0.18477655 -0.56632534  0.35969354
 [7]  0.50184542 -2.01826492  2.41146030 -0.49813163  2.66708028  2.21386325
[13] -1.02682887  0.32135117  0.52486906 -2.82421849  0.19610056  0.22662043
[19] -0.97594709 -0.24442585 -0.91451514  1.90065235  0.73761587 -0.13270364
[25]  0.60415596 -0.83393533  1.05573291 -1.85635706  1.33970180 -0.63292367
[31]  1.70912388 -0.67991073 -1.21556923  0.43830221  0.72834104 -1.94136298
[37] -0.24947880 -2.01613274 -0.22930310 -0.88152471  1.83116550 -0.75038318
[43] -0.68660435  0.15378929  0.64731206  0.42337005 -1.75382945  0.15984790
[49] -0.23895701 -1.84667053 -0.31829751  1.60388590 -1.59979511 -1.25363138
[55]  0.18125467  0.76521910 -0.52777437  0.59345811 -0.52977719 -0.18713794
[61]  0.74123051  1.15946937  1.08358762 -0.27143918 -0.33611739 -0.75236731
[67] -0.16814150 -0.31317751  0.55428041 -0.06690778  1.42598546 -0.25262070
[73]  1.52618091 -0.89833714  0.94269324  0.05282348 -0.07145531  0.91642536
[79]  0.33681440  1.19348178  1.20841801 -0.16028563  0.98222500  3.42185759
[85]  0.19093619  1.19314832 -0.40834495 -1.61536419 -1.15968241 -0.34954346
[91] -1.37580815  0.15817889 -1.14342230 -1.02146397 -0.39218635 -2.16615822
[97] -0.58792260 -0.38918672 -1.61936264  0.96674063
```

On modifie la longueur de la ligne R en utilisant la commande `options(width=60)` dans un morceau de code ayant l'option `results=hide`. Voici les conséquences de cela

```
> data[1:100]
```

```

[1] -2.27118096 -0.03242496 -0.42328025  0.18477655
[5] -0.56632534  0.35969354  0.50184542 -2.01826492
[9]  2.41146030 -0.49813163  2.66708028  2.21386325
[13] -1.02682887  0.32135117  0.52486906 -2.82421849
[17]  0.19610056  0.22662043 -0.97594709 -0.24442585
[21] -0.91451514  1.90065235  0.73761587 -0.13270364
[25]  0.60415596 -0.83393533  1.05573291 -1.85635706
[29]  1.33970180 -0.63292367  1.70912388 -0.67991073
[33] -1.21556923  0.43830221  0.72834104 -1.94136298
[37] -0.24947880 -2.01613274 -0.22930310 -0.88152471
[41]  1.83116550 -0.75038318 -0.68660435  0.15378929
[45]  0.64731206  0.42337005 -1.75382945  0.15984790
[49] -0.23895701 -1.84667053 -0.31829751  1.60388590
[53] -1.59979511 -1.25363138  0.18125467  0.76521910
[57] -0.52777437  0.59345811 -0.52977719 -0.18713794
[61]  0.74123051  1.15946937  1.08358762 -0.27143918
[65] -0.33611739 -0.75236731 -0.16814150 -0.31317751
[69]  0.55428041 -0.06690778  1.42598546 -0.25262070
[73]  1.52618091 -0.89833714  0.94269324  0.05282348
[77] -0.07145531  0.91642536  0.33681440  1.19348178
[81]  1.20841801 -0.16028563  0.98222500  3.42185759
[85]  0.19093619  1.19314832 -0.40834495 -1.61536419
[89] -1.15968241 -0.34954346 -1.37580815  0.15817889
[93] -1.14342230 -1.02146397 -0.39218635 -2.16615822
[97] -0.58792260 -0.38918672 -1.61936264  0.96674063

```

Le code suivant permet de modifier les sorties de R ( le code peut être caché dans un véritable document) :

```
> options(prompt = "MyR> ", continue = "...")
```

Voici le résultat :

```
MyR> summary(data)
```

```

      Min.  1st Qu.  Median    Mean 3rd Qu.    Max.
-2.82400 -0.75090 -0.16420 -0.05142  0.66760  3.42200

```

Voici des commandes permettant de mettre un peu de couleurs dans les morceaux de code et un cadre autour des entrées et des sorties :

```

\DefineVerbatimEnvironment{Sinput}{Verbatim}{formatcom={\color[rgb]{0.56,0,0}},
xleftmargin=2em, frame=single}
\DefineVerbatimEnvironment{Soutput}{Verbatim}{formatcom={\color[rgb]{0,0,0.56}},
xleftmargin=2em, frame=single}

```

```
MyR> summary(data)
```

```

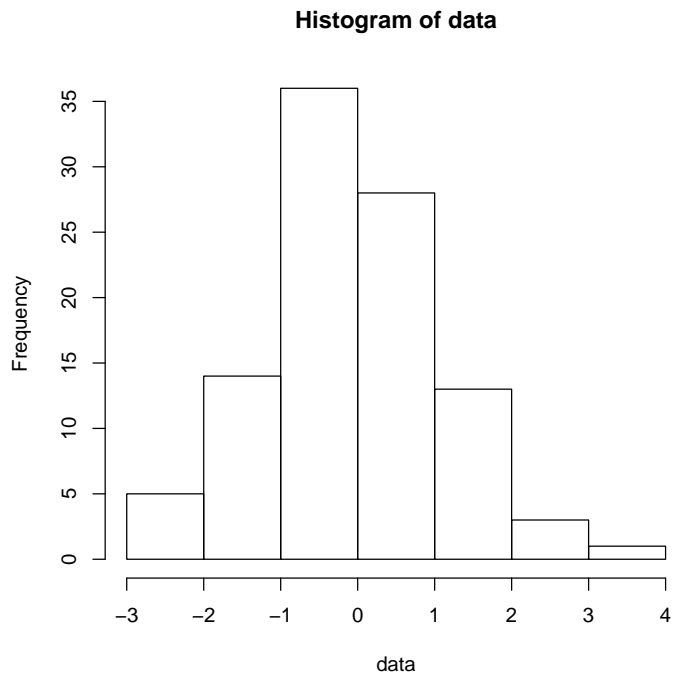
      Min.  1st Qu.  Median    Mean 3rd Qu.    Max.
-2.82400 -0.75090 -0.16420 -0.05142  0.66760  3.42200

```

### 3 Du code pour des figures

Un petit morceau de code permettant de générer un histogramme (à noter que `data` a été généré dans un morceau de code précédent :

```
MyR> hist(data)
```



```
MyR> hist(data)
```

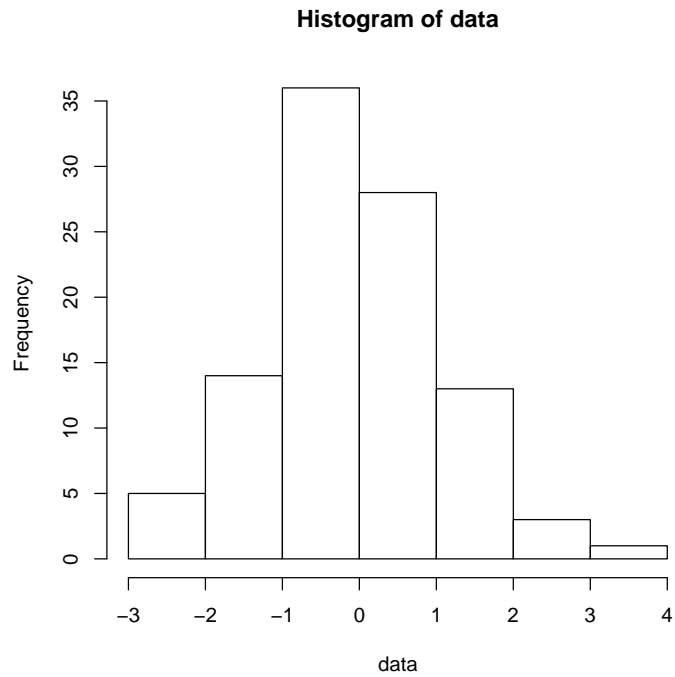


FIGURE 1 – Des données aléatoires

Un morceau de code un peu plus sexy intégrant l'histogramme dans une figure avec une légende et un label permettant d'y faire référence (cf. figure 1). A noter que la figure est "flottante" dans le document. A noter l'usage de l'option `eps=F` pour ce morceau de code. Il n'y a qu'un fichier pdf pour ce chunk.

```
MyR> hist(data)
```

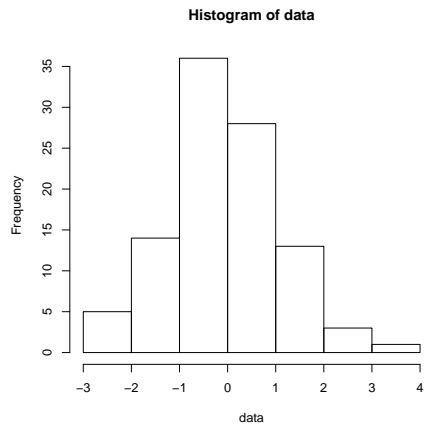


FIGURE 2 – Des données aléatoires

Le même morceau de code en essayant de modifier la taille des figures (cf. figure 2) en utilisant la commande `LATEX \setkeys{Gin}{width=0.5\textwidth}`.

A noter : la présence de figures en pdf dans le répertoire courant. Pour organiser les données, il faut utiliser la commande :

```
SweaveOpts{prefix.string=figures/myfigure}
```

## 4 Du code pour des tables

Ce petit bout de code permet de générer une "jolie" table (cf. figure 1). A noter l'utilisation de la fonction `xtable` et de l'option `results=tex`.

```
MyR> library(xtable)
MyR> aa=as.matrix(summary(data))
MyR> xtable(aa, caption = "Résumé des données étudiées.",
...           , label = "tab:resume" )
```

## 5 Du code avec cacheSweave

```
MyR> A=30
MyR> Sys.sleep(A)
```

Un petit exemple de code avec l'option `cache=T` qui évite de devoir attendre 30 secondes à chaque tour.

	x
Min.	-2.82
1st Qu.	-0.75
Median	-0.16
Mean	-0.05
3rd Qu.	0.67
Max.	3.42

TABLE 1 – Résumé des données étudiées.