Projet R-Dyndoc

R. Drouilhet

Cqls Team

Plan

Motivation initiale pour le projet dyndoc

A l'origine, (il y a plus de 15 ans)

- **Enseignement :** gain de temps lors de la rédaction des sujets d'examens et préparation de documents de cours
  $\Rightarrow$ Marre du copier-coller-modifier et du "je sais plus où c'est!"
- **Consulting statistique :** idée de proposer des outils de génération automatique de rapports en intégrant les résultats extraits de traitements R .

Puis, (il y a $\simeq$ 10 ans)

- **Programmation :** création d'un langage de type "chef d'orchestre"
  1. dédié à la manipulation de contenu de **doc**ument `latex`, `html`, ...
     $\Rightarrow$ pas besoin de manipulation de `print` et `"..."`
  2. utilisant tel quel ("as is") plusieurs langages (R , `ruby`, ...) aux caractéristiques complémentaires pour **dyn**amiser le contenu.

Plan

L'habituel "Hello code" :

```
1   [#R<] mister <- "misteR" #R code
2   [#rb<] mister = "mister" #ruby code
3   [#<] This is not output!
4      {#def]hello[#,]name[Miss][#>]
5         [hello #{name}]
6      [#}
7   [#>]
8         [from Dyn, {#hello#}
9         |from Dyn, {#hello]Mister[#}
10        |from ruby, hello :{mister}
11        |from R, hello :r{mister}]
```

Résultat:

```
from Dyn, hello Miss
from Dyn, hello Mister
from ruby, hello mister
from R, hello misteR
```

Le “Hello code” avec librairie :

Le code ci-dessous étant le contenu du fichier `helloLib.dyn`

```
1  No output inside a library
2
3  {#def]hello[#,]name[Miss][#>]
4     [Hello #{name}]
5  [#}
```

le code dyndoc suivant donne le même résultat que précédemment :

```
1  [#require]helloLib
2  [#main][#R<] mister <- ”misteR” #R code
3  [#rb<] mister = ”mister” #ruby code
4  [#>]
5        [from Dyn, {#hello#}
6        |from Dyn, {#hello]Mister[#}
7        |from ruby, hello :{mister}
8        |from R, hello :r{mister}]
```

L'habituel "Hello code" en `latex` :

```
1   [#R<] mister <- "misteR" #R code
2   [#rb<] mister = "mister" #ruby code
3   [#<] This is not output!
4      {#def]hello[#,]name[Miss][#>]
5         [hello \textbf{#{name}}]
6      [#}
7   [#>]
8         [from Dyn, {#hello#}\\
9         |from Dyn, {#hello]Mister[#}\\
10        |from ruby, hello \textit{:{mister}}\\
11        |from R, hello \underline{:r{mister}}]
```

Résultat:

```
from Dyn, hello \textbf{Miss}\\
from Dyn, hello \textbf{Mister}\\
from ruby, hello \textit{mister}\\
from R, hello \underline{misteR}
```

L'habituel "Hello code" en `latex` :

```
1   [#R<] mister <- "misteR" #R code
2   [#rb<] mister = "mister" #ruby code
3   [#<] This is not output!
4      {#def]hello[#,]name[Miss][#>]
5         [hello \textbf{#{name}}]
6      [#}
7   [#>]
8         [from Dyn, {#hello#}\\
9         |from Dyn, {#hello]Mister[#}\\
10        |from ruby, hello \textit{:{mister}}\\
11        |from R, hello \underline{:r{mister}}]
```

Résultat:

from Dyn, hello **Miss**
from Dyn, hello **Mister**
from ruby, hello *mister*
from R, hello misteR

L'habituel "Hello code" en `latex` via `textile` :

```
1   [#R<] mister <- "misteR" #R code
2   [#rb<] mister = "mister" #ruby code
3   [#<] This is not output!
4      {#def]hello[#,]name[Miss][#>]
5         [hello *#{name}*]
6      [#}
7   [#txtl>]
8         [from Dyn, {#hello#}
9         |from Dyn, {#hello]Mister[#}
10        |from ruby, hello __:{mister}__
11        |from R, hello +:r{mister}+]
```

Résultat:

```
from Dyn, hello \textbf{Miss}
from Dyn, hello \textbf{Mister}
from ruby, hello \textit{mister}
from R, hello \underline{misteR}
```

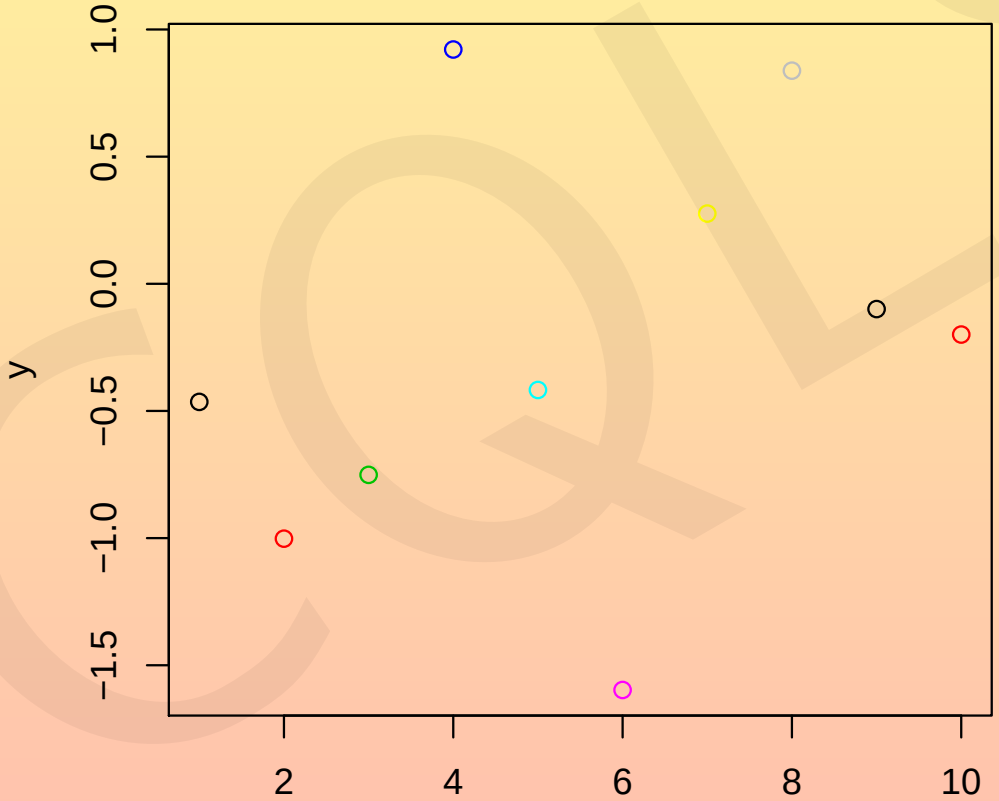
Le code dyndoc suivant :

```
{#VerbFrame]
[#>]\noindent \textbf{Session \texttt{R}}~ :
   {#beameR]
   y<-rnorm(10)
   y
   plot(1:10,y,col=1:10)
   ##fig | img=test | opt=width=7cm
   rexp(10)
   [#beameR}
[#VerbFrame}
```

permet à un simple utilisateur de générer la page beamer suivante (modulo la définition des fonctions VerbFrame et beameR en mode développeur dyndoc) :

Session R :

```
> y<-rnorm(10)
> y
 [1] -0.46427489 -1.00212281 -0.75123961  0.92142857 -0.41738740
 [6] -1.59711418  0.27624377  0.83824923 -0.09926501 -0.19942505
> plot(1:10,y,col=1:10)
```




```
> rexp(10)
 [1] 1.56596590 0.18665871 0.09234515 0.79924628 0.35327647 0.29527908
 [7] 0.62965370 1.44351636 6.38193172 0.20541198
```

Exemple Sweave (très basique)

```
1  \documentclass{article}
2  \begin{document}
3  «»=
4  a<-c(1,3,2)
5  a
6  @
7  La moyenne est \Sexpr{mean(a)}.
8  \end{document}
```

Son équivalent dyndoc :

```
1  {#rverb]
2  a<-c(1,3,2)
3  a
4  [#}
5  La moyenne est :r{mean(a)}.
```

N.B. pas d'entête latex grâce à un système de "layout".

Plan

Equivalent Sweave (par défaut) :

```
[#>]\textit{Mode séquentiel}~ :
[#R»]x <- rnorm(3)
x
[#>]
\textit{Mode imbriqué}~ :
{#rverb]
c(mean(x),sd(x))
[#}
```

Résultat:

Mode séquentiel :

```
> x <- rnorm(3)
> x
[1] -1.1696092  0.5256851 -0.9953191
```

Mode imbriqué :

```
> c(mean(x),sd(x))
[1] -0.5464144  0.9325461
```

Equivalent Sweave (echo=FALSE,results='hide') et \Sexpr :

```
1   [#R<]# No output even for the following "cat(...)"
2   a <- c("toto","titi")
3   cat(a,'\n')
4   a2 <- paste(a,1:2,sep="")
5   cat(a2,'\n')
6   a3 <- paste(a2,collapse=" & ")
7   cat(a3)
8   [#>]Extracting the R results inside dyndoc:
9   a is :r{a}
10  a2 is #r{a2}
11  a3 is :r{a3}
```

Résultat:

```
Extracting the R results inside dyndoc:
a is toto,titi
a2 is toto1,titi2
a3 is toto1 & titi2
```

Idem en ruby :

```
1   [#rb<]# No output even for the following "puts"
2   a=["toto","titi"]
3   p a
4   a2=a.each.with_index.map{|e,i| e+(i+1).to_s}
5   puts a2
6   a3=a2.join(" & ")
7   puts a3
8   [#>]Extracting the ruby results inside dyndoc:
9   a is :{a}
10  a2 is #rb{a2}
11  a3 is :rb{a3}
```

Résultat:

```
Extracting the ruby results inside dyndoc:
a is toto,titi
a2 is toto1,titi2
a3 is toto1 & titi2
```

Equivalent Sweave (echo=FALSE) :

```
[#R>]# Each output goes to the document
a <- c("toto","titi")
cat(a,'\n')
a2 <- paste(a,1:2,sep="")
cat(a2,'\n')
a3 <- paste(a2,collapse=" & ")
cat(a3)
```

Résultat:

```
toto titi
toto1 titi2
toto1 & titi2
```

Idem en ruby :

```ruby
[#rb>]# Each output goes to the document
a=["toto","titi"]
p a
a2=a.each_with_index.map{|e,i| e+(i+1).to_s}
puts a2
a3=a2.join(" & ")
puts a3
```

Résultat:

```
["toto", "titi"]
toto1
titi2
toto1 & titi2
```

Plan

Variables dyndoc :

```
1  [#>]toto[TOTO]          [#=]titi[TITI]
2  [#>]First step: (#{toto}) AND (#{titi})
3  [#>]toto+[ and TOTO2] [#+]titi[ and TITI2]
4  [#>]Second step: (#{toto}) AND (#{titi})
5  [#>]toto[NewTOTO]       [#=]titi[NewTITI]
6  [#>]Third step: (#{toto}) AND (#{titi})
```

Résultat:

```
First step: (TOTO) AND (TITI)
Second step: (TOTO and TOTO2) AND (TITI and TITI2)
Third step: (NewTOTO) AND (NewTITI)
```

Variables dyndoc pour R et ruby :

```
1  [#=]toto[TOTO]
2  [#%]Ruby variables from Dyndoc environment
3  [#=]toto@[[1,3,2]]
4  [#>] [Content of toto@: #{toto@}<\n>]
5  [#%]R variables from Dyndoc environment
6  [#=]toto$[c(1,3,2)]
7  [#>] [Content of toto$: #{toto$}<\n>]
```

Résultat:

```
Content of toto@: 1,3,2
Content of toto$: 1.0,3.0,2.0
```

Utiliser ces variables en R :

```
1  [#R<]
2  <toto@>[1]= sum(<toto@>)
3  <toto:> = tolower(<toto:>)
4  <toto$>[1] = diff(<toto$>[1:2])
5  [#>]After R,
6  toto=#{toto}
7  toto@=#{toto@}
8  toto$=#{toto$}
```

Résultat:

```
After R,
toto=toto
toto@=6,3,2
toto$=2.0,3.0,2.0
```

Utiliser ces variables en ruby :

```
1  [#rb<]
2  <toto@>[0]= 1
3  <toto:> = <toto:>.upcase
4  <toto[1]$> -= 1
5  [#>]After ruby,
6  toto=#{toto}
7  toto@=#{toto@}
8  toto$=#{toto$}
```

Résultat:

```
After ruby,
toto=TOTO
toto@=1,3,2
toto$=1.0,3.0,2.0
```

Plan

Instruction `if` (mode imbriqué) :

```
1  [#rb<]n=11
2  [#>]:{n} is {#if]n%2==1[#>]odd[#else]even[#if} and |
3  [#R<]n <- 28
4  [#>]:r{n} is {#if]:r{n%%2==1}[#>]odd[#else]even[#if}
```

Résultat:

```
11 is odd and 28 is even
```

Instruction `if` (mode séquentiel) :

```
1  [#rb<]n=14[#R<]n <- 25
2  [#>]:{n} is [#?]n%2==1[#>]odd[#?]else[#>]even[#?]end
3  [#>] and :r{n} is |
4  [#?]:r{n%%2==1}[#>]odd[#?]else[#>]even
```

Résultat:

```
14 is even and 25 is odd
```

Instruction `if` en R :

```
1  [#R>]n=31
2      if(n%%2==1) {#>]:r{n} is odd[#>}
3      else {#>]:r{n} is even[#>}
```

Résultat:

```
31 is odd
```

Instruction `if` en ruby :

```
1  [#rb>]n=33
2      if n%2==1
3          {#>]:{n} is odd[#>}
4      else
5          {#>]:{n} is even[#>}
6      end
```

Résultat:

```
33 is odd
```

Instruction case :

```
1  [#=]todo[first,third,second,first]
2  [#>]List of the translated words in French:
3  {#case]#{todo}
4  [#when]first[#>][premier ]
5  [#when]second[#>][deuxième ]
6  [#when]third[#>][troisième ]
7  [#case}
```

Résultat:

```
List of the translated words in French:
premier troisième deuxième premier
```

L'équivalent en R (même si pas trop d'intérêt, ici) :

```
1  [#=]todo[first,third,second,first]
2  [#>]List of the translated words in French:
3  [#R>]
4     for(todo in strsplit(#{=todo},",")[[1]]) {
5        switch(todo,
6        first={#>]premier [#>},
7        second={#>]deuxième [#>},
8        third={#>]troisième [#>})
9     }
```

Résultat:

```
List of the translated words in French:
premier troisième deuxième premier
```

L'équivalent en ruby (même si pas trop d'intérêt, ici) :

```
1  [#=]todo[first,third,second,first]
2  [#>]List of the translated words in French:
3  [#rb>]#{=todo}.split(",").each do |todo|
4     case todo.to_sym
5     when :first then {#>]premier [#>}
6     when :second then {#>]deuxième [#>}
7     when :third then {#>]troisième [#>}
8     end
9  end
```

Résultat:

```
List of the translated words in French:
premier troisième deuxième premier
```

Boucles directement en R :

```
1  [#>]Loop in R:
2  [#R>]
3  for(cpt in 1:4) {#>]item:r{cpt-1} [#>}
4  [#>]<\n>but also,
5  [#R>] sapply(5:8,function(cpt) {
6     cpt <- cpt -1
7     {#>]item#r{cpt} [#>}
8  })
```

Résultat:

```
Loop in R:
item0 item1 item2 item3
but also,
item4 item5 item6 item7
```

Boucles directement en ruby :

```
1   [#>]Loop in ruby:
2   [#rb>]
3   for cpt in 1..4 do
4      {#>]item:{cpt-1} [#>}
5   end
6   [#>]<\n>but also,
7   [#rb>] (5..8).each do |cpt|
8      cpt=cpt-1
9      {#>]item#rb{cpt} [#>}
10  end
```

Résultat:

```
Loop in ruby:
item0 item1 item2 item3
but also,
item4 item5 item6 item7
```

Plan

Libraries d'objets dyndoc (ici classe Data) :

```
1   {#meth]new.Data[#,] .objR[] [#,].rdata[] [#<][#}
2
3   {#meth]begin.Data
4     [#yield]default
5     [#rb>].rdata[#{=.rdata}.strip]
6     [#rb>].objR[#{=.objR}.strip]
7     [#?]#{+?.rdata}[#r<]attach('#{.rdata}')
8     [#?]#{+?.objR}[#r<]attach(#{.objR})
9   [#}
10
11  {#meth]end.Data
12    [#?]#{+?.objR}[#r<]detach(#{.objR})
13    [#?]#{+?.rdata}[#r<]detach('file:#{.rdata}')[#?]end
14    [#yield]default
15  [#}
```

Déclaration d'objets et appels de méthodes :

```
1  [#require]Import/LM/lm[#main][#<]
2  {#new]data[#of]Data[#in]lm1
3  [#,]chomage[#,]~ /cqls/data/chomage.RData[#}
4  {#new]lm1[#of]LM[#in]lm1
5  [#,]chom~ txpib+deppub+pfisc[#,]:data[#}
6  [#>]{#summary]lm1[#}
7  [#<]{#end]lm1[#}
```

Résultat:

Régression linéaire : `chom~txpib+deppub+pfisc (data=chomage)`

$\mathbf{x}^{(j)}$	$\widehat{\beta_j}(\mathbf{y}\|\underline{\mathbf{x}})$	$\widehat{\sigma}_{\widehat{\beta_j}}(\mathbf{y}\|\underline{\mathbf{x}})$	$\widehat{\delta_{\beta_j,0}}(\mathbf{y}\|\underline{\mathbf{x}})$	p-valeur bilatérale
1	-22.05345	1.96807	-11.20562	$\simeq 0$
txpib	0.01359	0.10423	0.13035	0.89716
deppub	0.27974	0.17793	1.57221	0.12639
pfisc	0.34835	0.20287	1.71711	0.09627

Rédaction abrégée d'un hypothèse de test :

```
1  [#require]Import/StatDevel/HypoTest[#main][#<]
2  {#new]pChom[#of]ParamFrame[#in]pChom[#,]p[#,]asymp
3  [#.nomPb]C[#.thetaEstR]16/200[#.ny]200[#}
4
5  {#new]pChomTest[#of]TestParam[#in]pChom[#,]:pChom
6  [#.theta0R]0.1[#.theta0Tex]10\%[#.side]<
7  [#.assertion]le taux de chômage en France serait
8  cette année inférieur à $10\%$[#}
9  [#>]{#redaction]pChomTest[#,]a[#}
```

Résultat:
Assertion d'intérêt : $\mathbf{H}_1 : p^C < 10\%$
Application numérique : puisqu'au vu des données,

$$p\text{-}valeur \overset{R}{=} \texttt{pnorm((16/200 - 0.1)/sqrt(0.1 * (1 - 0.1)/200))} \simeq 34.58\% \not< 5\%,$$

on ne peut pas plutôt penser (avec un risque de 5%) que le taux de chômage en France serait cette année inférieur à 10%.

Flag + environnement R + Paramètre nommé :

```
[#<] {#r]a<-rnorm(5) [#in]exo[#}
{#def]test[#,]opt[m][#,]l[5]
[#<]{#opt]#{opt}[#}[#>]
{#rverb][#??]m[#>]mean(a)
[#??]s[#>]sd(a)[#in]exo[#rverb}
#r{exo:a[1:#{l}]}[#def}
[#>]{#test]m[#} {#test]s[#l]3[#}
```

Résultat:

```
> mean(a)
[1] 0.3965875
```

0.7449406,-2.857718,0.731556,1.045191,2.318967

```
> mean(a)
[1] 0.3965875
> sd(a)
[1] 1.932633
```

0.7449406,-2.857718,0.731556

Intégrer knitr (et Sweave)

```
[#R>] for(i in c(5,6)) {
   {#>]{#knitr]
   |«»=
   |rnorm(:r{i})
   |@
   |[#knitr}[#>}

   if(i==6) {#>]{#knitr]
   |«fig,echo=FALSE,out.width='.3\\linewidth'»=
   |plot(1::r{i},rnorm(:r{i}),type='l',lwd=5)
   |@
   [#knitr}[#>}
}
```

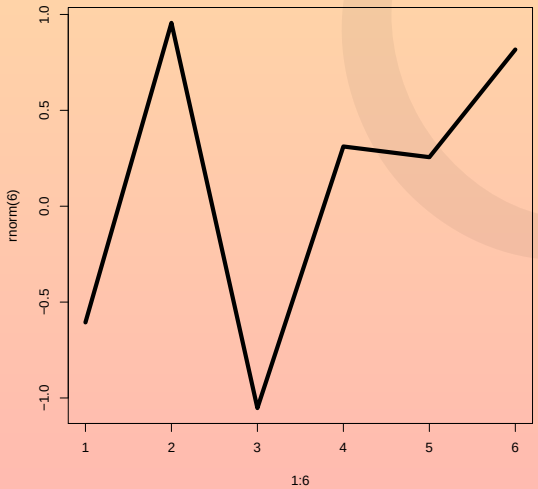
Résultat :

```
rnorm(5)
```

```
## [1]  0.98093 -0.19633  0.05886  0.17587  0.54683
```

```
rnorm(6)
```

```
## [1] -0.1426  0.1304 -1.2630 -0.3823  0.4926  0.5112
```



Plan

En résumé, dyndoc c'est

Un outil développé en `ruby` avec pour principales caractéristiques :

- **système de modèles ("templating system") :** basé sur un langage de balises (de type "domino") suffisamment exotique pour s'intègrer facilement dans tout document au format "human readable" (tel que `latex`, `html`, ...) afin d'en dynamiser le contenu.
- **langage (de script) :** permettant d'intégrer pleinement les langages `ruby`, `R` et `Mathematica` (expérimental) et éventuellement `python` ou tout autre langage interfaçable en `C`.

Principaux avantages

- *mode développeur*: créer des librairies regroupant fonctions et objets `dyndoc` (dédiés à la génération de parties textuelles relatives à certaines expertises)
- *mode utilisateur*: rassembler dans un même document `dyndoc` à la fois les dernières étapes des analyses statistiques (fait avec `R` , `ruby` et `Mathematica`) ainsi que le contenu textuel. A partir de ce document `dyndoc`, permettre de générer un ou plusieurs documents finaux dans différents formats (`latex`, `html`, ...).

Plan

Modes d'utilisation de dyndoc

- **Ligne de commande en version autonome :** (sans hors-connexion)
  `dyndoc-ruby [options] all [options] document.dyn`
- **Ligne de commande en version client-serveur :**
  `dyndoc-client all dyn://sag6/document.dyn`
  client (léger) se connectant à un serveur `dyndoc` pour traitement de petit projet (pas un livre sur R !)
- **Interface web** : (hors-connexion)
  `dyndoc-web [start|stop|status]`
  dédié à la création de documents `html5` incluant du `javascript` pour dynamiser les traitements (en particulier, graphique).
- **Interface web** : (connexion server)
  Zéro install, dédié à la création en ligne de pages web (ou documents pdf) avec possibilité de proposer directement des traitements R .

Informations complémentaires

Page web dyndoc

`http://dyndoc.upmf-grenoble.fr/Dyndoc.html`

Esprit du projet :

- *sur un plan général* : dyndoc est né d'une quête de sens, d'une mise en veille de l'égo, d'un éveil de conscience, de l'envie de partager des valeurs
- *sur un plan plus pratique* : dyndoc c'est une tentative de participer au développement d'une activité (ici, autour du reporting statistique) dans un but de regrouper une communauté (ici, de développeurs, d'utilisateurs) où la dimension sociale prévaudrait.