

[R-Fiddle](#) est un environnement de programmation R accessible en ligne. Il permet de coder et de faire exécuter un programme R. Quel intérêt me direz vous puisque le logiciel [R](#) est libre et que des éditeurs performants gratuits - tels que [R-Studio](#) version Desktop - sont disponibles ?

J'y vois plusieurs raisons en ce qui me concerne. Ce type de solution est adapté à un utilisateur nomade qui change fréquemment de machine. Sous condition de disposer d'une connexion internet, il peut travailler sur un projet sans avoir à se préoccuper de l'installation de R sur des PC dont il ne dispose pas de droits administrateurs de toute façon. Le travail collaboratif est un autre contexte où ce dispositif peut se révéler particulièrement avantageux. Il nous permet de nous affranchir des transferts de fichiers toujours hasardeux (*j'ai pas reçu ton e-mail ! qui n'a pas été confronté à cette dénégation péremptoire...*) avec une gestion des versions aléatoire. Enfin, la solution nous permet de travailler sur un front-end léger, un ordinateur portable par exemple, et de déporter les calculs sur un serveur distant taillé en conséquence (dans le [cloud](#) dirait-on aujourd'hui pour faire poétique).

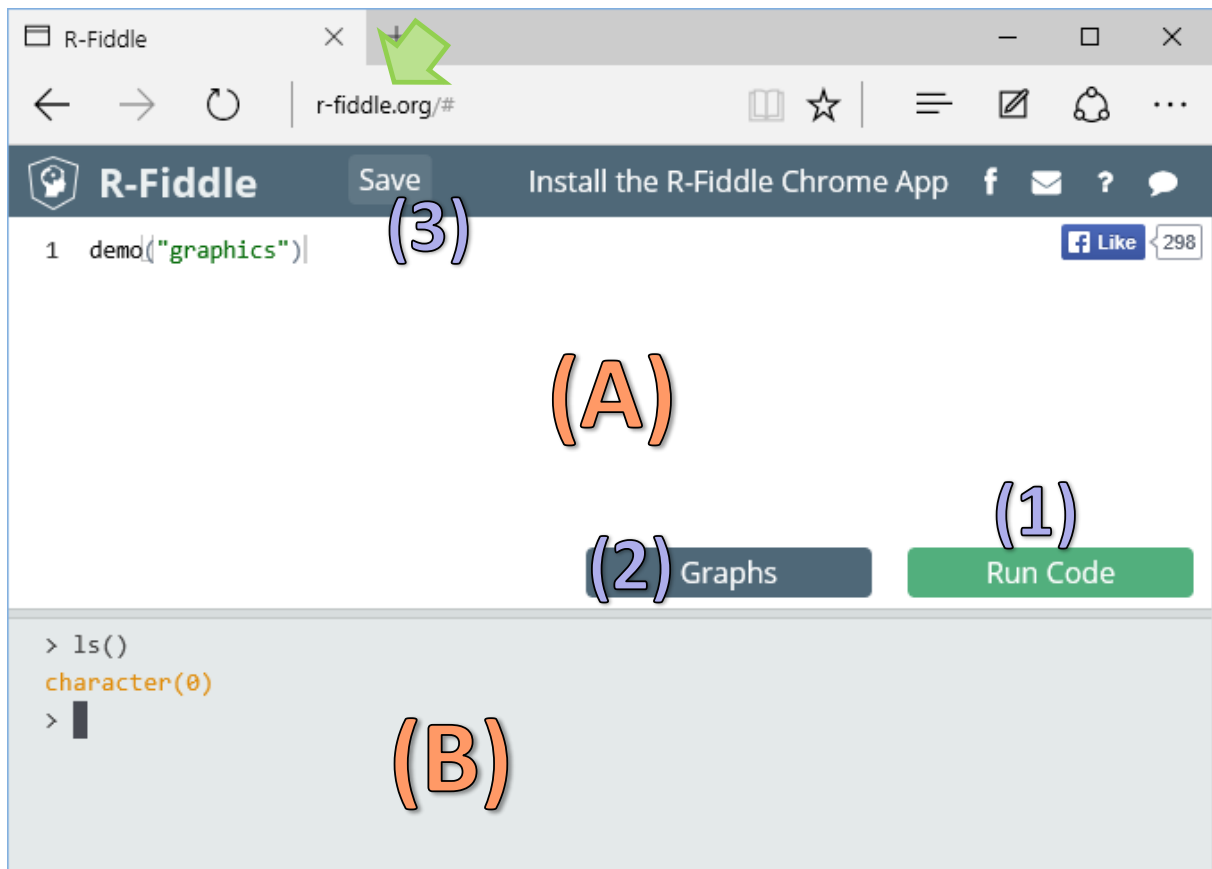
R-Fiddle se positionne plutôt comme un prototype (POC : "proof of concept"). Il permet de lancer des petites applications pour s'exercer. Il lui manque plusieurs fonctionnalités pour être réellement opérationnel dans un environnement professionnel : il n'est pas possible de disposer d'un compte utilisateur permettant de se connecter de manière privative ; il n'est pas possible de gérer un projet constitué de plusieurs programmes ; nous ne disposons pas d'un espace pour stocker les données ; une gestion des versions existe, mais elle est très fruste ; nous n'avons pas vraiment pris sur les capacités de calculs qui sont mises à notre disposition.

Il n'en reste pas moins que R-Fiddle correspond à un concept basé sur le "cloud" qui prend de plus en plus d'ampleur. Les solutions [Azure](#) de Microsoft ou [DSS](#) de Dataiku que nous avons étudiées précédemment ouvrent de nouvelles perspectives. Elle étendent les champs des possibilités en matière de management organisationnel et de gestion des ressources des projets. Au-delà des effets de mode (cloud par ci, cloud par là dirait un de mes collègues), Il nous appartient d'identifier les contextes d'utilisation où elles sont les plus pertinentes.

Dans ce tutoriel, nous étudions succinctement les fonctionnalités de R-Fiddle.

1 Présentation de R-Fiddle

Accessible via l'URL <http://www.r-fiddle.org/#/>, l'interface est fractionnée en deux cadres : (A) celui du haut permet de saisir notre code ; (B) celui du bas affiche les sorties, nous pouvons également y saisir et faire exécuter des commandes (comme pour la console R).



Nous observons plusieurs boutons de commande :

1. RUN CODE lance l'exécution du programme rédigé dans l'éditeur de code.
2. GRAPHS affiche les différents graphiques générés par notre programme.
3. SAVE permet de sauvegarder notre code.

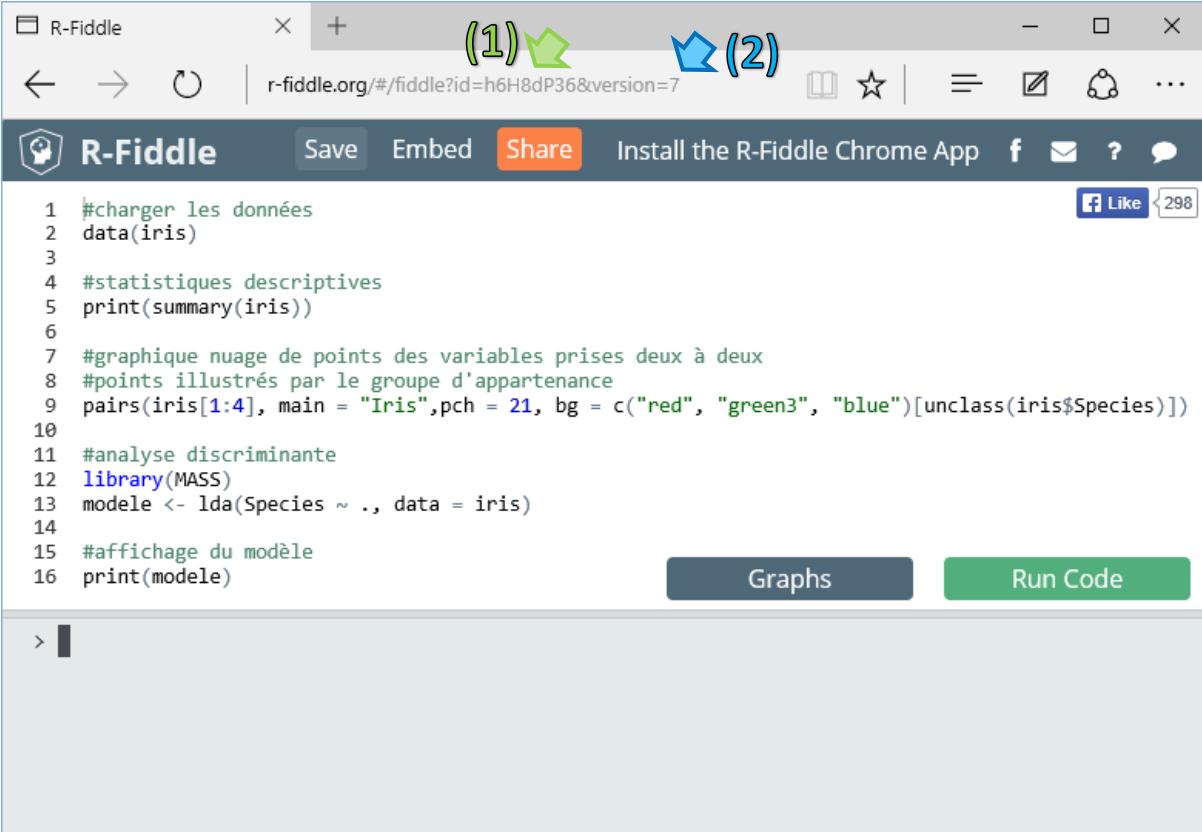
L'instruction par défaut `demo("graphics")` effectue une démonstration des potentialités graphiques de R lorsque nous l'exécutons avec RUN CODE.

2 Travailler sur des données intégrées dans les packages

2.1 Utilisation des packages standards

Dans un premier temps, nous souhaitons travailler sur des données intégrées dans le package « Dataset » de R. Nous pouvons ainsi fonctionner sans avoir à aborder l'épineux problème du chargement de fichier que nous étudierons dans la section suivante. Nous rédigeons le code ci-dessous. Les sorties comportent des graphiques. Nous actionnons le bouton SAVE pour stocker notre travail. R-Fiddle nous attribue automatiquement un identifiant (1) et un numéro de version (2) qui s'incrémentent automatiquement au fur et à mesure des sauvegardes.

Remarque importante : Nous n'avons pas de visibilité quant à la durée de vie de notre projet sur le serveur. Il faut en tenir compte si l'on veut utiliser intensivement l'outil !



The screenshot shows the R-Fiddle web interface in a browser. The address bar contains the URL `r-fiddle.org/#/fiddle?id=h6H8dP36&version=7`. The page title is "R-Fiddle". The main content area displays R code for data analysis and visualization. The code includes comments in French and R functions for loading data, printing summaries, creating scatter plots, and performing discriminant analysis. At the bottom of the code editor, there are two buttons: "Graphs" and "Run Code".

```
1 #charger les données
2 data(iris)
3
4 #statistiques descriptives
5 print(summary(iris))
6
7 #graphique nuage de points des variables prises deux à deux
8 #points illustrés par le groupe d'appartenance
9 pairs(iris[1:4], main = "Iris",pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)])
10
11 #analyse discriminante
12 library(MASS)
13 modele <- lda(Species ~ ., data = iris)
14
15 #affichage du modèle
16 print(modele)
```

Deux nouveaux boutons apparaissent : EMBED permet d'incorporer le cadre dans une page web ; SHARE permet de partager l'adresse de notre projet.

Nous actionnons le bouton RUN CODE, les résultats numériques apparaissent dans la console (1). Les sorties graphiques sont intégrées dans une fenêtre dédiée (2) que l'on peut masquer ou faire à apparaître à l'aide du bouton GRAPHS.

The screenshot shows the R-Fiddle interface with the following R code and output:

```

1 #charger les données
2 data(iris)
3
4 #statistiques descriptives
5 print(summary(iris))
6
7 #graphique nuage de points des variables prises deux à deux
8 #points illustrés par le groupe d'appartenance
9 pairs(iris[1:4], main = "Iris",pch = 21, bg = c("red", "green", "blue", "black"))
10
11 #analyse discriminante
12 library(MASS)
13 modele <- lda(Species ~ ., data = iris)
14
15 #affichage du modèle
16 print(modele)

```

Output (1):

```

Coefficients of linear discriminants:
          LD1          LD2
Sepal.Length  0.8293776  0.02410215
Sepal.Width   1.5344731  2.16452123
Petal.Length -2.2012117 -0.93192121
Petal.Width  -2.8104603  2.83918785

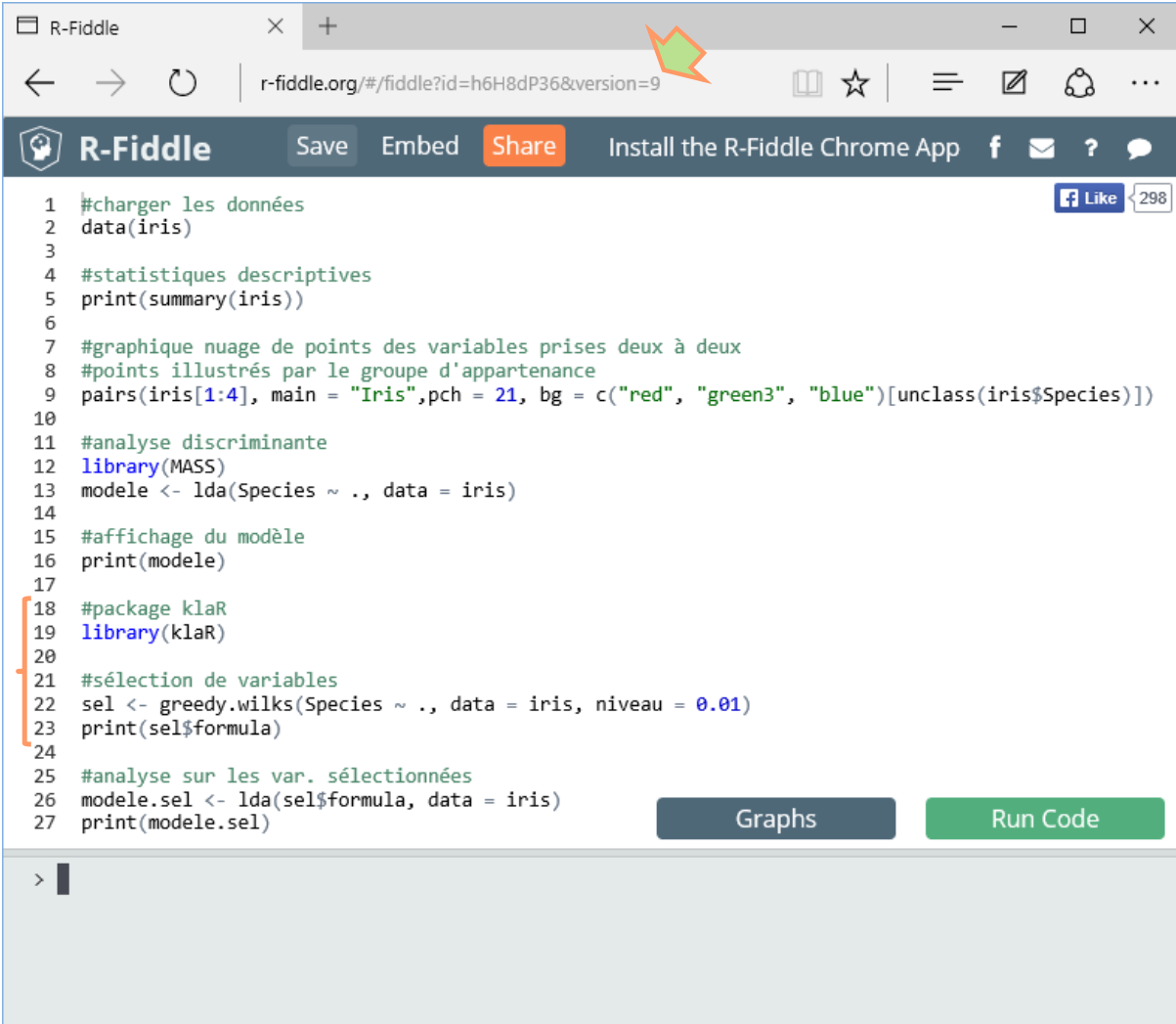
Proportion of trace:
          LD1          LD2
0.9912 0.0088

```

Graphs (2): A pairwise scatter plot titled "Iris" showing relationships between Sepal.Length, Sepal.Width, Petal.Length, and Petal.Width. The plot is a 4x4 grid of scatter plots with points colored by species (red, green, blue).

2.2 Utilisation des packages non standards

Pour évaluer l'extensibilité du dispositif, nous avons souhaité utiliser des packages qui ne sont pas intégrés dans la [distribution standard](#) de R. La librairie [KlaR](#) offre des fonctionnalités de sélection de variables, avec notamment la commande `greedy.wilks()` similaire à la procédure STEPDISC de SAS. Nous modifions notre code (9^{ème} version comme nous pouvons le noter dans l'URL du projet).



The screenshot shows a web browser window with the R-Fiddle interface. The address bar shows the URL `r-fiddle.org/#/fiddle?id=h6H8dP36&version=9`. The interface includes a header with the R-Fiddle logo, navigation buttons (Save, Embed, Share), and an option to install the Chrome app. The main area contains R code for data analysis and visualization. A red arrow points to the address bar. Below the code, there are buttons for 'Graphs' and 'Run Code'. A 'Like' button with a count of 298 is also visible.

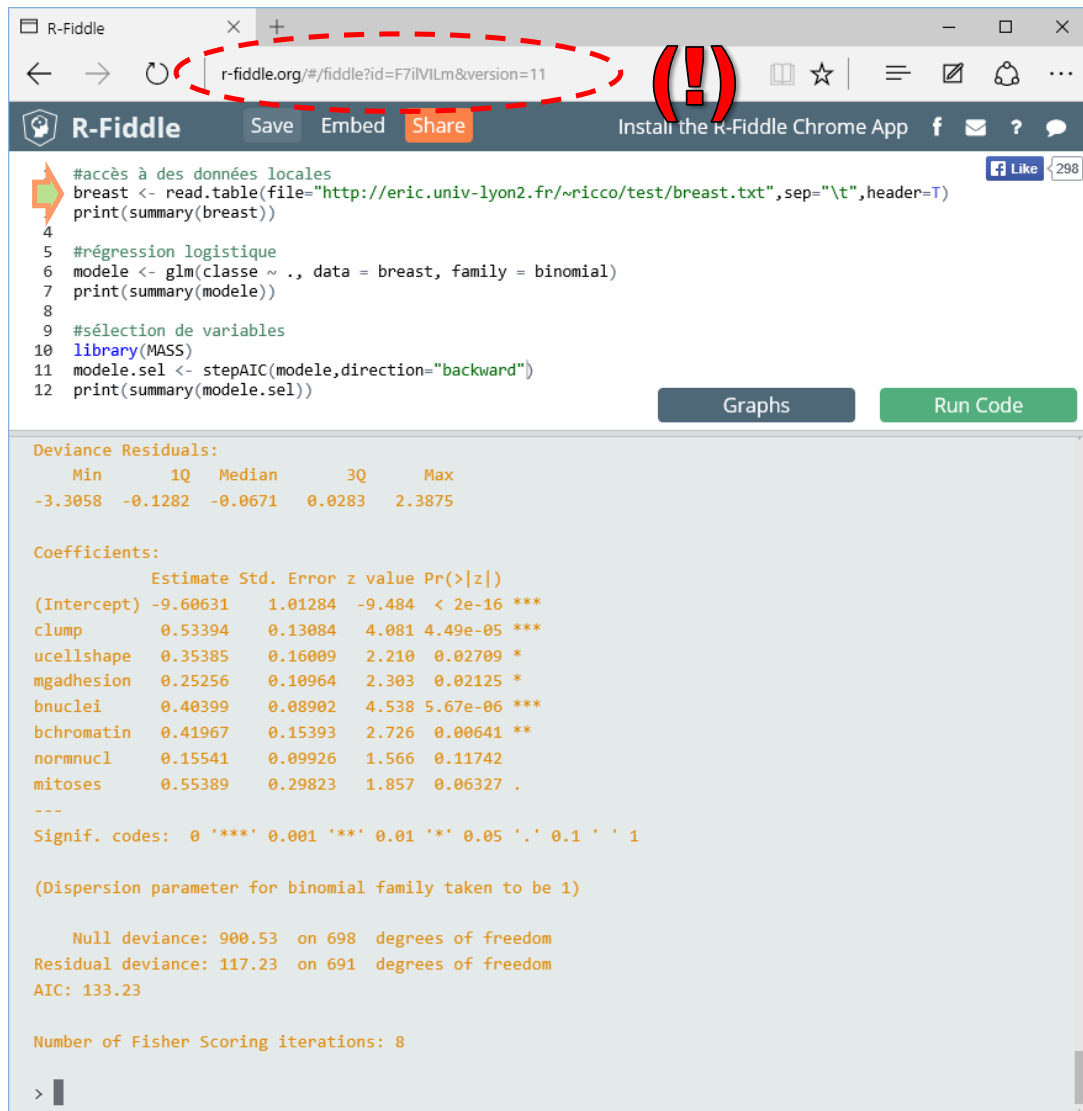
```
1 #charger les données
2 data(iris)
3
4 #statistiques descriptives
5 print(summary(iris))
6
7 #graphique nuage de points des variables prises deux à deux
8 #points illustrés par le groupe d'appartenance
9 pairs(iris[1:4], main = "Iris", pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)])
10
11 #analyse discriminante
12 library(MASS)
13 modele <- lda(Species ~ ., data = iris)
14
15 #affichage du modèle
16 print(modele)
17
18 #package klaR
19 library(klaR)
20
21 #sélection de variables
22 sel <- greedy.wilks(Species ~ ., data = iris, niveau = 0.01)
23 print(sel$formula)
24
25 #analyse sur les var. sélectionnées
26 modele.sel <- lda(sel$formula, data = iris)
27 print(modele.sel)
```

Le programme a parfaitement fonctionné. La variable `SEPAL.LENGTH` a été éliminée.

Je n'ai pas retrouvé la liste des packages disponibles. On peut espérer que la majorité des librairies intéressantes (tout dépend de ce qu'on appelle "intéressant") sont présentes.

3 Travailler avec nos propres données

L'affaire devient sérieuse si nous pouvons travailler sur nos propres données. J'ai cherché comment traiter directement un fichier situé sur ma machine. Je n'ai pas trouvé. Je ne sais pas si je dois le déplorer ou m'en féliciter. Laisser un programme externe accéder directement à notre système de fichier n'est peut être pas une bonne idée en matière de sécurité. J'ai donc chargé le fichier [BREAST](#) bien connu des data scientists sur un serveur et j'ai indiqué l'URL (protocole HTTP) dans l'instruction `read_table()` de R.



```
#accès à des données locales
breast <- read.table(file="http://eric.univ-lyon2.fr/~ricco/test/breast.txt",sep="\t",header=T)
print(summary(breast))
4
#régression logistique
6 modele <- glm(classe ~ ., data = breast, family = binomial)
7 print(summary(modele))
8
9 #sélection de variables
10 library(MASS)
11 modele.sel <- stepAIC(modele,direction="backward")
12 print(summary(modele.sel))
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-3.3058	-0.1282	-0.0671	0.0283	2.3875

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-9.60631	1.01284	-9.484	< 2e-16 ***
clump	0.53394	0.13084	4.081	4.49e-05 ***
ucellshape	0.35385	0.16009	2.210	0.02709 *
mgadhesion	0.25256	0.10964	2.303	0.02125 *
bnuclei	0.40399	0.08902	4.538	5.67e-06 ***
bchromatin	0.41967	0.15393	2.726	0.00641 **
normnucl	0.15541	0.09926	1.566	0.11742
mitoses	0.55389	0.29823	1.857	0.06327 .

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 900.53 on 698 degrees of freedom
Residual deviance: 117.23 on 691 degrees of freedom
AIC: 133.23

Number of Fisher Scoring iterations: 8

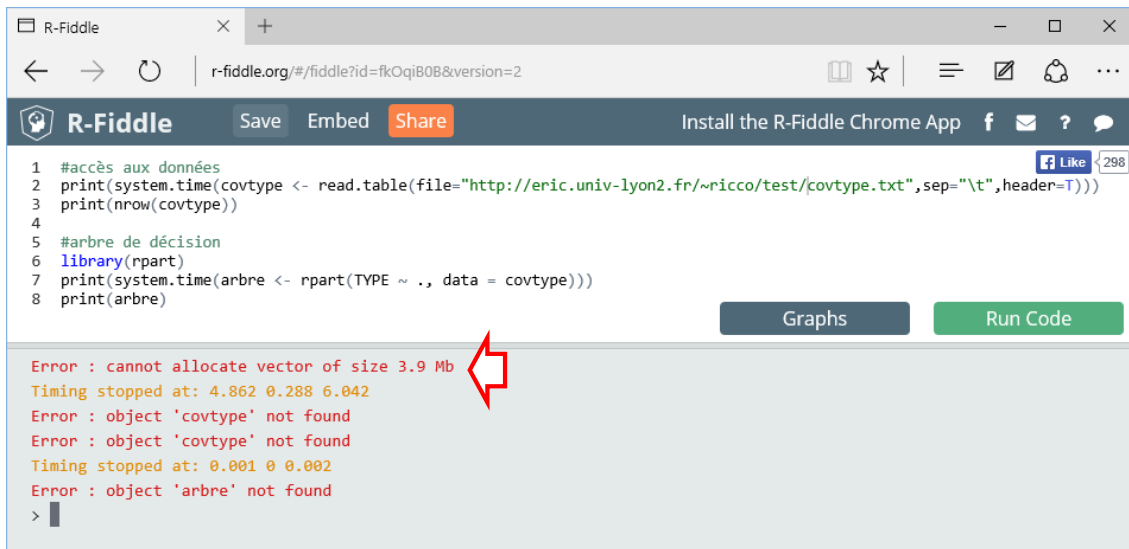
> █

Un nouvel identifiant a été attribué à ce nouveau projet. J'en étais à la 11^{ème} version de la sauvegarde sur cette copie d'écran. Concernant l'exécution du programme, le chargement des données a fonctionné et la régression logistique a été menée à son terme.

4 Performances

4.1 Gestion de la volumétrie

Pour nous faire une idée des ressources mises à notre disposition, nous mettons à contribution un fichier un peu plus volumineux dans cette section, à savoir le fichier [COVTYPE](#) avec 581012 observations et 54 variables. Nous utilisons l'algorithme d'induction d'arbres du package [rpart](#). Nous avons exécuté le même code en ligne et en local. Sur ma machine, tout a fonctionné correctement. Voyons ce qu'il en est en ligne.




```
1 #accès aux données
2 print(system.time(covtype <- read.table(file="http://eric.univ-lyon2.fr/~ricco/test/covtype.txt",sep="\t",header=T)))
3 print(nrow(covtype))
4
5 #arbre de décision
6 library(rpart)
7 print(system.time(arbre <- rpart(TYPE ~ ., data = covtype)))
8 print(arbre)
```

Error : cannot allocate vector of size 3.9 Mb
Timing stopped at: 4.862 0.288 6.042
Error : object 'covtype' not found
Error : object 'covtype' not found
Timing stopped at: 0.001 0 0.002
Error : object 'arbre' not found
> |

La sentence est immédiate. Le volume que l'on peut gérer est limité sur R-Fiddle.

4.2 Rapidité de calculs



```
1 #accès des données distantes
2 breast <- read.table(file="http://eric.univ-lyon2.fr/~ricco/test/breast.txt",sep="\t",header=T)
3 n <- nrow(breast)
4
5 #arbre de décision
6 library(rpart)
7
8 #compteur
9 tmp <- proc.time()
10
11 #generation d'une liste d'arbre
12 arbres <- list()
13 for (i in 1:1000){
14   #id échantillon
15   obs <- sample(1:n,n,replace=T)
16   #un arbre
17   one_tree <- rpart(classe ~ ., data = breast[obs,])
18   #stockage
19   arbres[[i]] <- one_tree
20 }
21
22 #affichage de contrôle
23 print(length(arbres))
24
25 #calcul durée
26 print(proc.time() - tmp)
```

[1] 1000
user system elapsed
11.650 0.017 11.667
> |

Voyons ce qu'il en est des performances de la machine distante. Nous avons exécuté le code ci-dessus. Il consiste à lancer 1000 fois la construction d'un arbre de décision sur des échantillons obtenus par tirage avec remise à partir des données BREAST.

Le temps de génération est de 7.53 sec. sur ma machine (Core i7 - 3,1 Ghz - Windows 10). Il a été de 11.67 sec. sur le serveur de R-Fiddle. L'écart n'est absolument pas rédhibitoire. La puissance proposée est tout à fait convenable compte tenu du fait que le dispositif est accessible gratuitement et qu'il n'y a jamais eu de délai d'attente lors de tous mes tests (il y a des blocages en revanche lorsque j'ai tenté de lancer plusieurs fois le même programme).

5 Conclusion

R-Fiddle est avant tout un exercice de style qui a le mérite d'attirer notre attention sur un mode opératoire des logiciels qui prend de plus en plus d'ampleur aujourd'hui. Je suis très curieux de voir comment notre domaine va évoluer dans le futur. Les outils de statistique et de data mining correspondent à des besoins et à des usages spécifiques. Il nous importe de bien cerner l'intérêt de ce nouveau mode de fonctionnement. Je constate en tous les cas que la question agite les utilisateurs de R et que plusieurs solutions - plus ou moins abouties - existent (« [Any Online R console ?](#) » sur Data Science Stack Exchange).