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).

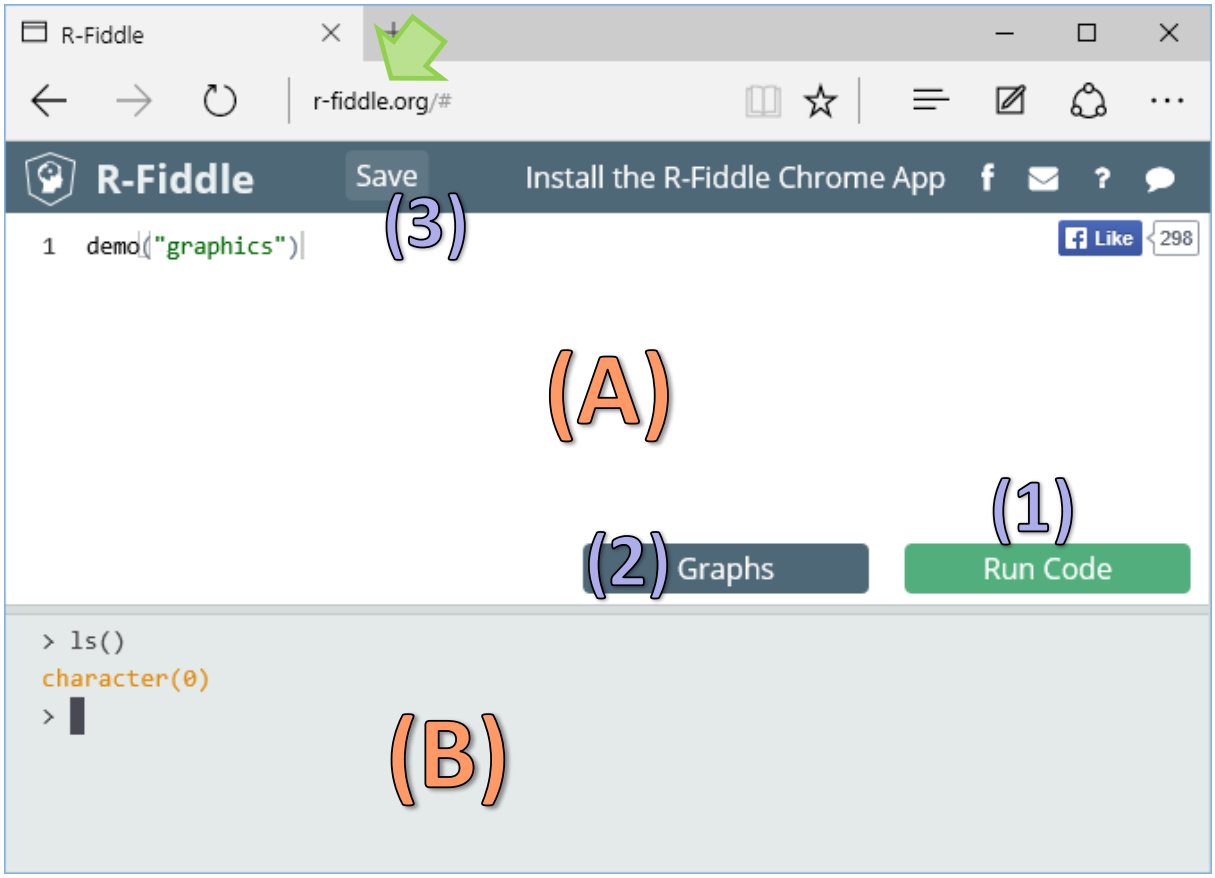
R-Fiddle se positionne plutôt comme un prototype (POC : ''proof of concept''). Il permet de lancer des petites applications pour s'exercer. Il lui manque plusieurs fonctionnalités pour être réellement opérationnel dans un environnement professionnel : il n'est pas possible de disposer d'un compte utilisateur 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 prise 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 avions 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.

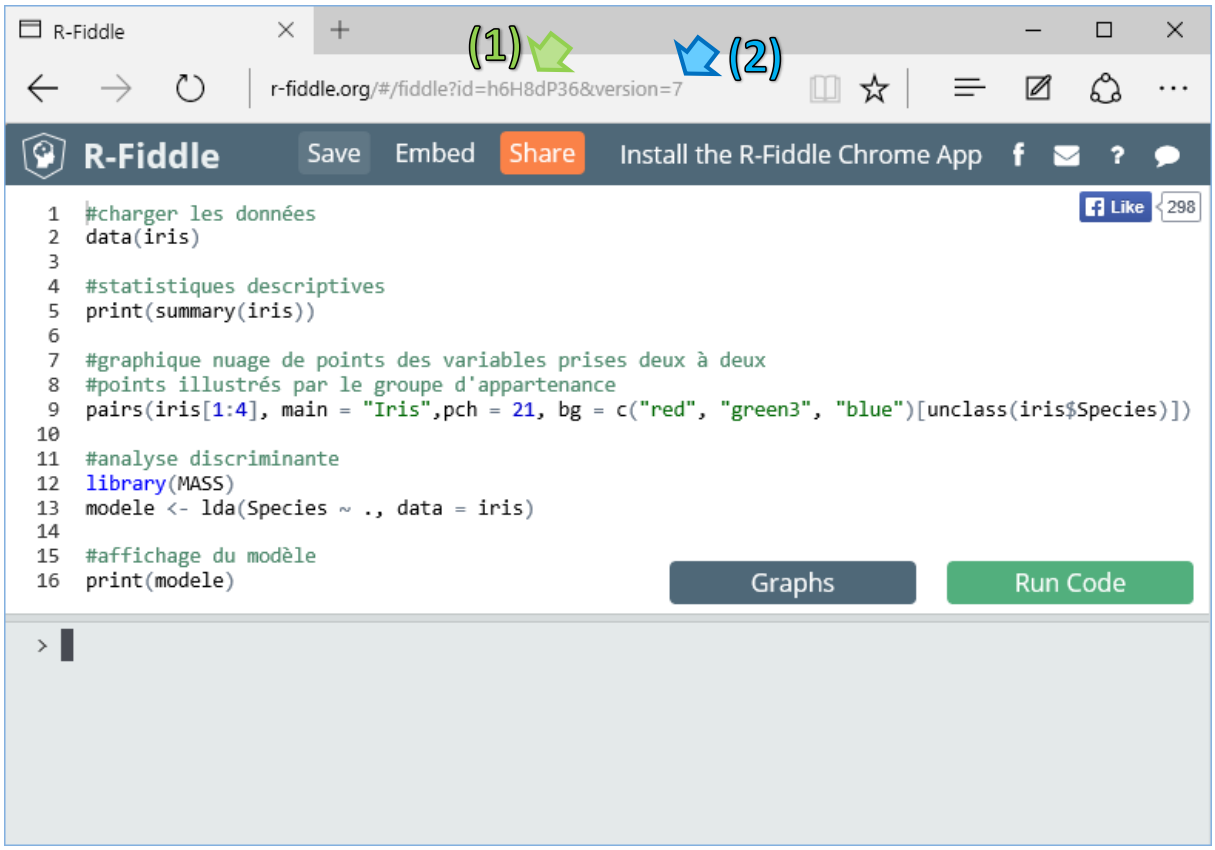
L'instruction par défaut demo("graphics") effectue une démonstration des potentialités graphiques de R lorsque nous l'exécutons avec RUN CODE.

2 Travailler sur des données intégrées dans les packages

2.1 Utilisation des packages standards

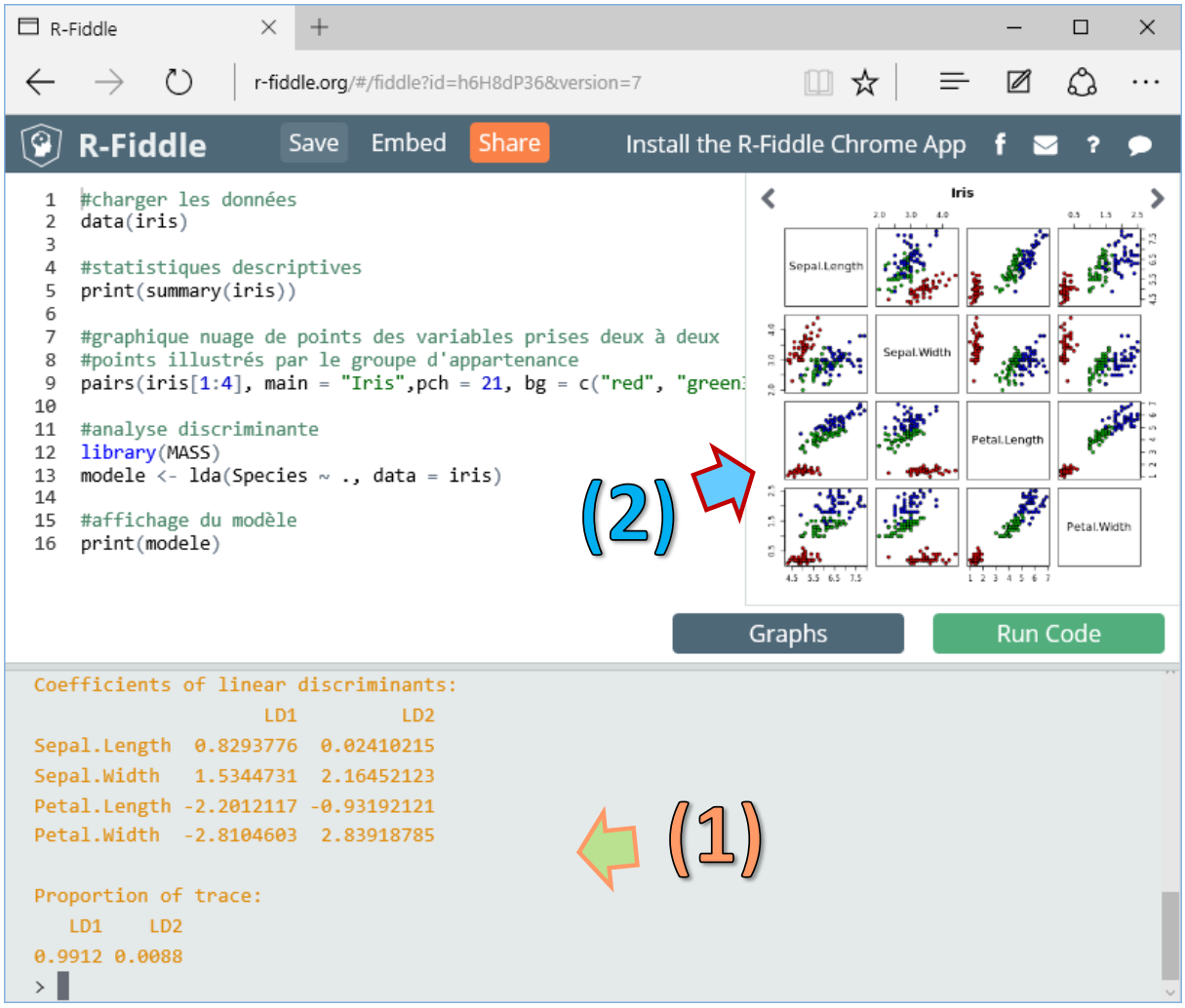
Dans un premier temps, nous souhaitons travailler sur des données intégrées dans le package « Dataset » de R. Nous pouvons ainsi fonctionner sans avoir à aborder l'épineux problème du chargement de fichier que nous étudierons dans la section suivante. Nous rédigeons le code ci-dessous. Les sorties comportent 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'incrémente 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 !



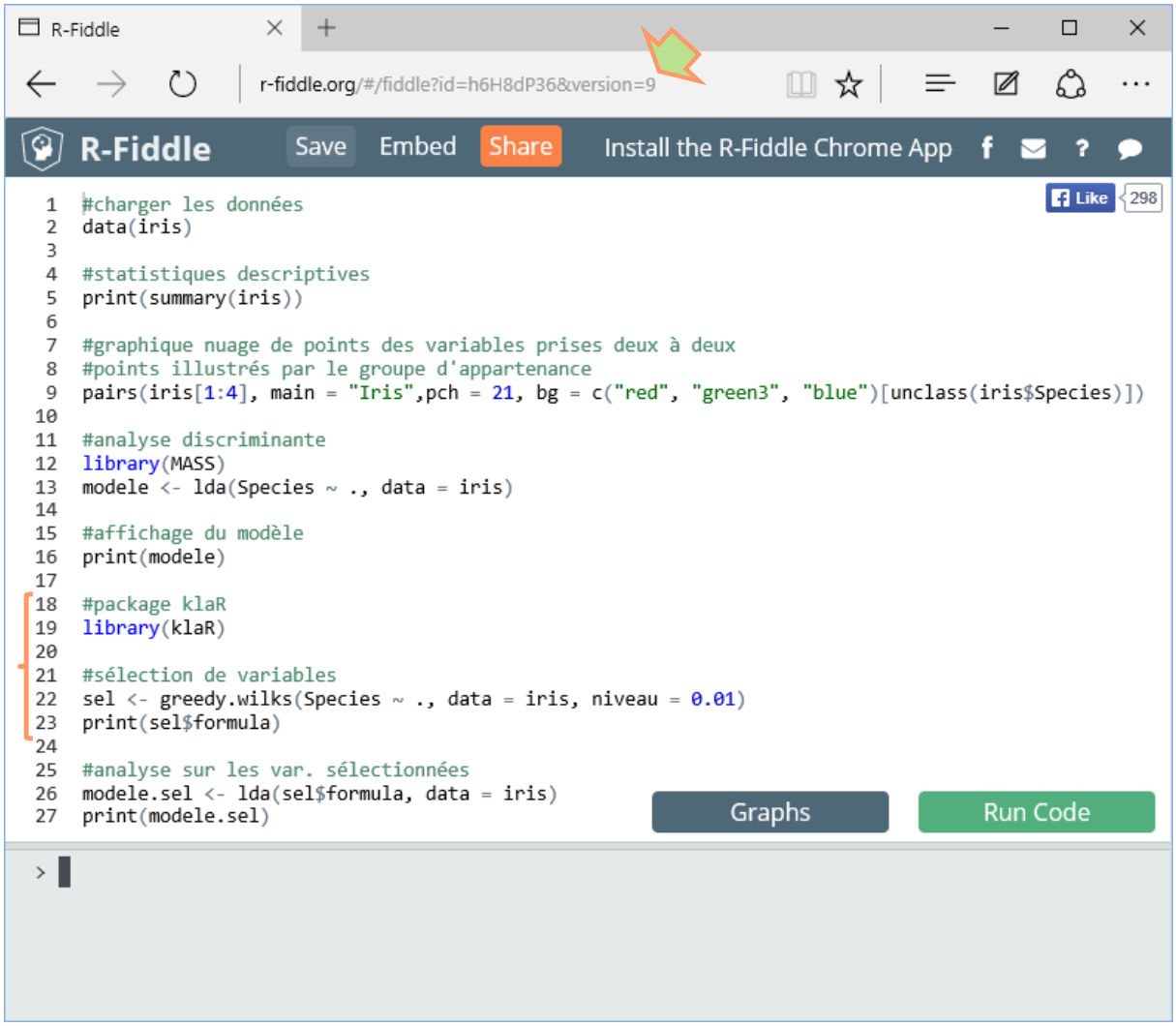
Deux nouveaux boutons apparaissent : EMBED permet d'incorporer le cadre dans une page web ; SHARE permet de partager l'adresse de notre projet.

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.



2.2 Utilisation des packages non standards

Pour évaluer l'extensibilité du dispositif, nous avons souhaité utilisé 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 (9ème version comme nous pouvons le noter dans l'URL du projet).

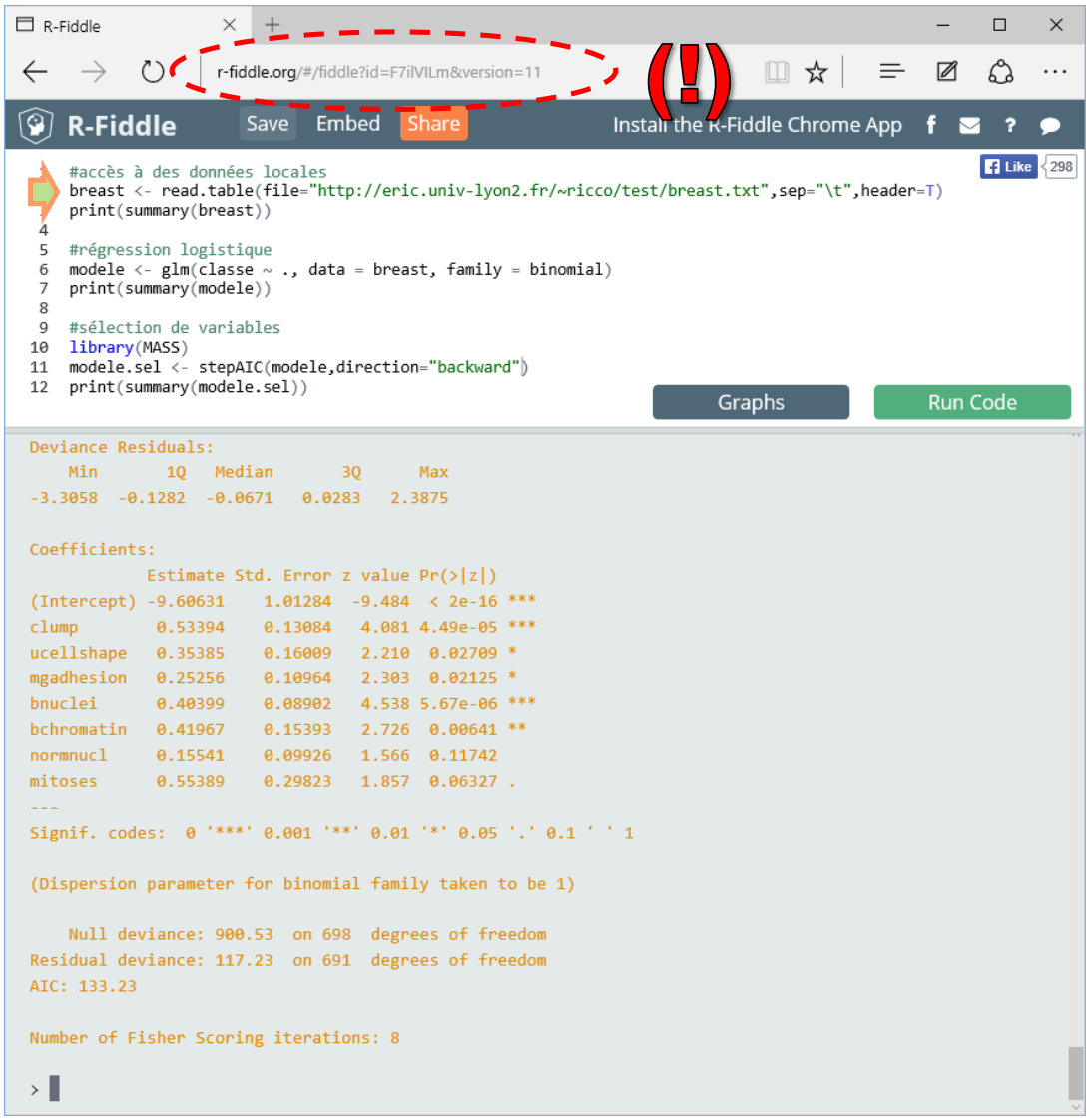


Le programme a parfaitement fonctionné. La variable SEPAL.LENGTH a été éliminée.

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.

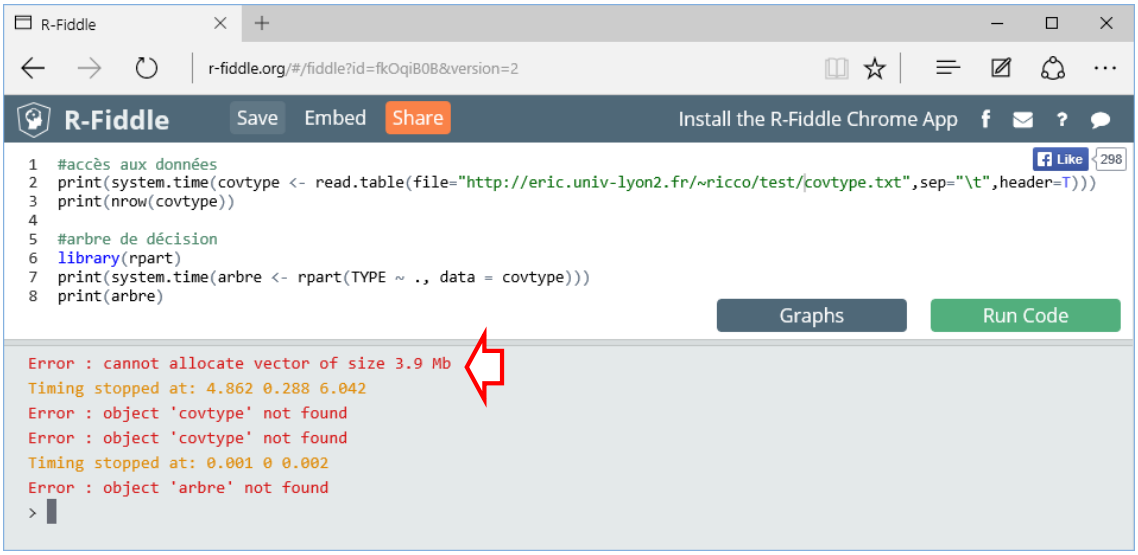


Un nouvel identifiant a été attribué à ce nouveau projet. J'en étais à la 11ème version de la sauvegarde sur cette copie d'écran. Concernant l'exécution du programme, le chargement des données a fonctionné et la régression logistique a été menée à son terme.

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.



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

4.2 Rapidité de calculs

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).