Sécurisation des analyses statistiques avec R : retour d'expérience

Julien Dugas* Aurore Philibert†

Résumé (max 300 mots)

Comment maintenir dans le temps des scripts R et applications Shiny ?

C'est la question que nous nous sommes posés au GEVES (Groupe d'Étude et de contrôle des Variétés Et des Semences) car cette maintenance crée de nombreuses contraintes au quotidien :

- chaque script R ou application Shiny doit être lancé avec la bonne version de R et de packages (et toujours la même pour chaque utilisateur) via des applications métiers spécifiques développées dans d'autres technologies
- une mise à jour des versions de R et de packages doit se faire au fur et à mesure, par analyse, en éliminant les effets de bords pour garantir la stabilité des résultats
- plusieurs utilisateurs doivent pouvoir utiliser la même application Shiny simultanément
- le choix des méthodes statistiques et de la manière de les retranscrire en R doit être facilement transmissible d'un statisticien à un autre
- les scripts R doivent être centralisés et gérés de la même manière
- les évolutions des analyses et les nouveaux développements doivent être tracés et suivis

Dans cette présentation vous apprendrez comment créer un environnement technique et fonctionnel (Docker, Shiny Proxy, Git, tests unitaires, ...) permettant de palier à toutes ces contraintes. Nous détaillerons les choix techniques et leurs raisons, les différentes étapes suivies, les compétences nécessaires, les problématiques rencontrées et les solutions choisies.

Mots-clefs (3 à 5) : Maintenance - Reproductibilité - Plateforme - Biostatistique

Développement

En tant qu'organisme réglementaire, le GEVES doit garantir la sécurité de ses systèmes d'information, de ses données mais aussi de ses analyses statistiques. Au sein du pôle Biostatistique du GEVES nous avons travaillé sur la sécurisation de toutes les analyses statistiques que nous avons développées en R. Ses scripts ayant la particularité de devoir être maintenus dans le temps.

Pour cela nous avions comme cible les états suivants :

- Quand un script R est lancé, tous les utilisateurs doivent utiliser la même version du script
- Quand un script R est lancé, c'est bien la version de R et des packages avec lesquels ce script fonctionne qui doit être utilisée
- Il doit être possible de retrouver la version de script R utilisé pour produire un précédent résultat

*GEVES, Pôle Biostatistique, julien.dugas@geves.fr
†GEVES, Pôle Biostatistique, aurore.philibert@geves.fr

- Une montée de version de R et de packages doit être faite régulièrement pour chaque script R, indépendamment
- La maintenance d'un script R doit pouvoir être repris en charge rapidement par un nouveau statisticien ou un statisticien ne travaillant pas sur ce script en temps normal.

En ce qui concerne la centralisation des scripts, le sujet de les stocker sur gitlab et de pouvoir les lancer via un seul et même serveur est rapidement apparu. Mais la difficulté de pouvoir lancer chaque script R avec SA propre version de R et de packages nous a amené à nous intéresser à la technologie Docker qui permet ce cloisonnement entre script. Malheureusement aucune personne dans les pôles informatiques n'étaient connaisseurs de cette technologie. Nous avons donc travaillé de concert avec le pôle Systèmes et Réseaux du GEVES pour comprendre et installer cette technologie.

Nos scripts étant, en partie, mis à disposition aux utilisateurs via des applicatifs métiers (développés en .net par le pôle Bases et Développement du GEVES), nous avons collaboré avec ce pôle pour trouver la manière la plus appropriée de faire les lancements de scripts R. Pour cela une API a été développée, interagissant avec l'API Docker. Les échanges de données ont du être retravaillés pour que l'application métier puisse envoyer les données d'entrées au script R et ensuite récupérer les données de sorties. Le format des fichiers d'entrées et de sorties a été défini comme du json pour faciliter ces échanges avec un protocole standard (ou normalisé). Cette solution permet de garantir que si deux utilisateurs lancent le même traitement (mêmes paramètres et mêmes données d'entrées) ils obtiendront exactement le même résultat, quel que soit leur équipement informatique, leur lieu géographique, etc.

Une base de données spécifique a été créée pour permettre le suivi des versions de scripts au fil du temps avec à un temps donné un seul script d'"actif". Tous les lancements d'analyse sont ainsi tracés ce qui permet le débuggage et la récupération de la version de script R utilisée sur un précédent lancement.

En ce qui concerne les scripts R non mis à disposition via des applicatifs métiers nous avons décidé de les mettre à disposition des utilisateurs via des applications web développées en Shiny. Afin de permettre à plusieurs utilisateurs d'utiliser la même application en parallèle nous avons opté pour la solution Shiny Proxy. De même que pour Docker il nous a fallu apprendre à installer et gérer cette nouvelle technologie.

La maintenance dans le temps des scripts R nous a amené à chercher comment procéder aux "montées de versions" de R et des packages. Nous nous sommes intéressés aux tests unitaires, très répandus dans le monde du développement informatique. Ces tests unitaires permettent, par une étape préalable de création de ces tests, de lancer automatiquement le script avec une version de R et de packages plus récentes. Les tests vérifient automatiquement que les résultats sont identiques sur un jeu de données tests. Et si un test ne fonctionne pas alors il est clairement indiqué où une modification est nécessaire. Cette solution demande du temps lors de l'implémentation mais rend très rapide ces changements de version de R par la suite.

Enfin, nous devons être capable de garantir la connaissance des analyses de chaque script R développé. Une vignette est maintenant associée à chaque analyse décrivant les modifications/sélections de données éventuelles ainsi que les analyses statistiques utilisées. Dans cette vignette un lien est fait entre les analyses et la structuration du script R permettant à une nouvelle personne de facilement prendre en main le suivi de ce script.

En conclusion, nous avons réussi à développer une plateforme de biostatistique permettant de sécuriser les analyses statistiques et de les maintenir au cours du temps. Ce développement nous a pris une année entière entre le démarrage et la mise en production. Nous avons eu besoin de nous former à de nombreuses nouvelles technologies (Docker, Shiny Proxy, API Docker, ...) et de solliciter les compétences informatiques de nos collègues. Avec nos collègues il a fallu apprendre à communiquer, les vocabulaires et notions n'étant pas les mêmes entre un statisticien et un informaticien, et à s'adapter au fur et à mesure des tests et des avancées que nous avons faîtes. L'avantage d'avoir pu le développer en interne est que cela nous a permis de monter en compétences et d'être capable de maintenir dans le temps cette plateforme que nous avons appelé le Black Pearl.

Références