# {autoimport}: gérer l'enfer des imports

Dan Chaltiel*

**Résumé (max 300 mots)**

Toute personne ayant déjà développé un package le sait, gérer les dépendances est au mieux ennuyant, et au pire franchement pénible. Pour la plupart des packages, les dépendances sont gérées par le package `{roxygen2}` selon deux possibilités. La première est d'utiliser le tag `@import`, qui importe un namespace entier. Cette soluton est est rapide mais expose à des conflits de noms souvent problématiques. La deuxième, ma préférée, est d'utiliser le tag `@importFrom` qui importe chaque fonction indépendamment. Le problème alors, c'est que quand on importe beaucoup de fonctions, gérer tous ces tags devient vite une tâche très chronophage et pas vraiment valorisante. Heureusement, cette tache est automatisable, et `{autoimport}` essaie de s'y attelerg ! Ainsi, en appelant `autoimport()`, toutes les fonctions d'un package sont analysées pour générer une liste des dépendances externes. En fonction des fichiers `NAMESPACE` et `DESCRIPTION`, les bons tags `@importFrom` sont générés et peuvent être insérés facilement dans le code du package. Quand le choix n'est pas univoque, l'utilisateur peut choisir manuellement via la console. Pour plus de sécurité, on utilisera tout de même la fonction `import_review()`, qui propose une interface shiny utilisant `{diffviewer}` et permet de passer en revue toutes les modifications proposées avant de les accepter.

**Mots-clefs** : Package – Documentation – roxygen2

*Institut Gustave Roussy, dan.chaltiel@gustaveroussy.fr