Penvins cup : explications et conseils

Par quel bout attaquer le problème ?
De tous les côtés à la fois. Chacune des tâches 1 à 7 du cahier des charges de la *Penvins Cup* peut être considérée comme un micro-projet de programmation. Ce découpage est parfaitement volontaire : il est évident que vous n'allez travailler à 15 ou 25 sur la même ligne de code. L'esprit de la *Penvins Cup* est celui d'un travail collaboratif dans lequel une promo s'organise pour résoudre un gros problème (et vous verrez que le script obtenu au final est assez long et complexe) en le découpant en petits problèmes abordables sur lequel travaillent autant de petits groupes. Le mot clé à retenir est "fonction". Chacune des tâches 1 à 7 peut être programmée comme une fonction, même si vous pouvez aussi tenter l'approche du *code spaghetti*, mais c'est plus difficile quand vous êtes plusieurs à trifouiller dans le plat. Les fonctions ont en effet l'avantage qu'elles sont des boites étanches donc vous n'avez pas à vous préoccuper des noms d'objets choisis par les autres, par exemple.

Qui fait quoi ?
Vous constaterez rapidement que la difficulté des tâches va crescendo. Après les avoir examinées, confiez donc les plus simples aux archi-débutants R parmi vous, qui pourront ainsi fièrement obtenir leur première victoire et prendre confiance. Les plus chevronnés s'attaqueront aux tâches plus complexes, et ils aideront les autres à moins s'arracher les cheveux. Même les plus réfractaires peuvent avoir une grande utilité : faites-leur tester vos bouts de scripts en tant qu'utilisateurs. ils apporteront un feedback bienvenu sur la clarté et l'utilité de vos messages d'erreur. Ils vous diront également volontiers qu'ils ne comprennent absolument rien au fonctionnement du script lui-même, ce qui est un signe sûr que ce dernier ne contient pas assez de commentaires.

Bref, il y a vraiment de la place pour tous les niveaux dans la *Penvins Cup*, n'acceptez pas les excuses foireuses de ceux qui refusent de participer parce qu'ils "sont trop nuls" ou "ont peur de tout casser".

Ca va vraiment nous servir, vu les délais ?
Bien que vos scripts soient à envoyer dans environ deux mois (jeudi 23 novembre avant minuit) leurs différentes petites parties seront en phase de test *bien avant*, et vous rendront donc déjà bien service pour vérifier les fichiers de données avant d'intégrer tout ça dans un Grand Script Final.

Qui va juger nos scripts ? Comment garantir que le master X n'est pas favorisé ?

D'abord, les quatre masters en compétition seront représentés dans le jury, à savoir :

1) Cécile le Lann, coresponsable du M1 IMABEE.
2) Cédric Wolf, coresponsable du M1 MODE
3) Frédéric Ysnel, coresponsable du M1 PNB.
4) votre serviteur, coresponsable du M1 EFCE.

D'autre part, le jury jugera vos scripts *à l'aveugle* : nous *ignorerons* quel master a produit quel script.

Mais alors à qui envoyer nos scripts ?
Vous transmettrez vos scripts par mail non pas aux membres du jury directement mais à Maxime Hervé (oui, le gars qui a programmé l'interface graphique ***GrapheR*** — qui représente plus de 8 000 lignes de code — "juste parce qu'il trouvait que faire des graphes dans R n'était pas toujours très pratique "). Maxime ne prendra pas part à la notation mais son rôle est essentiel : d'une part il va

rendre vos scripts anonymes (en les renommant selon son imagination fertile), et d'autre part il va s'assurer qu'ils s'ouvrent correctement dans R. Si ça coince, il tentera brièvement de voir d'où vient le problème pour déterminer si c'est réparable facilement mais évidemment il tiendra le jury au courant de toute intervention de sa part donc inutile de vous dire que si Maxime a besoin d'intervenir sur votre script pour qu'il s'ouvre dans R c'est très mal parti pour vous.

Pour des raisons évidentes il est interdit de poser des questions au jury en cours de compétition, en particulier leur envoyer un petit bout de fonction ou ceci-cela qui permettrait ensuite de reconnaître qui est qui. Si vous avez des questions sur R, posez-les à votre gourou habituel ou bien à Maxime, qui jugera si votre requête est raisonnable et vous pointera dans la bonne direction — mais sans programmer à votre place !

Comment le jury va-t-il juger nos scripts ?

Après avoir chargé votre script dans R, le jury appliquera `checkPenvins()` sur plusieurs jeux de données-test contenant volontairement des erreurs — ou pas — pour voir si votre programme se comporte correctement quelle que soit la situation. A vous de construire un programme qui ne laisse rien passer mais ne sonne pas l'alarme pour rien, et pour cela le plus efficace est évidemment de le tester face à de vrais jeux de données (vous aurez l'embarras du choix) et face à des jeux de données dans lesquels vous aurez introduit volontairement des erreurs à des emplacements connus.

Pour décider du podium final, vos scripts jugés selon trois critères :

1) **Détection pertinente des erreurs** (toutes les erreurs de nos fichiers-tests sont-elles bien détectées ? Les alertes sont-elles toutes justifiées ?). C'est évidemment le critère le plus important.

2) **Qualité des messages adressés à l'utilisateur**, c'est à dire ceux qui vont s'afficher lors du fonctionnement. Sont-ils aussi peu envahissants que possible mais clairs et explicites ? Signalent-ils bien exactement à quel endroit est l'erreur et si possible en quoi elle consiste et ce qui était attendu à la place ?

3) **Qualité des commentaires adressés au programmeur**, c'est à dire ceux qui sont inclus directement dans le script et *que l'utilisateur lambda ne lira pas*. Ces commentaires seraient utiles à un utilisateur averti qui voudrait comprendre ce que fait votre script, par exemple pour le modifier sans rien casser. Attention, le jury ne va pas examiner réellement votre programmation elle-même, il va tenter de comprendre ce que fait votre script *d'après les commentaires qui y sont inclus*.

Merci d'avance aux plus *geeks* parmi vous (il y en a forcément quelques uns) de ne *pas* tout faire tout seul dans leur coin pour frimer devant les copains (ou leur éviter de se frotter aux dures lois des `IF(machin){ceci}else{cela}`), ça tuerait toute la logique collaborative et l'aspect formateur du challenge. Par contre, ils se rendront *extrêmement utiles* en conseillant les débutants, ce qui évitera des claviers brisés et des écrans jetés par la fenêtre (c'est agaçant un bug qu'on n'arrive pas à trouver, hein ?). Ils joueront aussi un rôle clé en fin de manip quand il va s'agir d'intégrer les différentes fonctions en un tout cohérent.

Bonne programmation à tous

Denis Poinsot