

Résumé

Les étudiants en biologie de l'Université de Mons (UMONS) suivent plusieurs cours consécutifs en sciences des données (notés A, B, et C ci-dessous), leur permettant d'acquérir des compétences en analyse de données — telles que le remaniement, la visualisation, l'inférence et la modélisation — tant sur le plan théorique que pratique. Dans ce cadre, ils apprennent à programmer en R, un logiciel open source dédié à l'analyse de données. Au sein de notre plateforme pédagogique d'apprentissage de R, **Learnit::R**, nous proposons un dispositif d'évaluation en ligne qui permet aux étudiants de tester, de

manière interactive, leurs compétences en programmation. Les questionnaires, développés à l'aide du package R **{learnr}**, sont mis à disposition via un serveur **Posit Connect**. Lors des évaluations, l'accès aux ressources informatiques est strictement contrôlé grâce à **Safe Exam Browser** et **AppArmor**. Les réponses sont ensuite collectées dans une base de données **MongoDB**. Ces données permettent d'évaluer les étudiants et de fournir un retour individualisé. Elles permettent également d'analyser les difficultés rencontrées en vue d'améliorer le dispositif pédagogique.

Learnit::R plateforme d'apprentissage de R

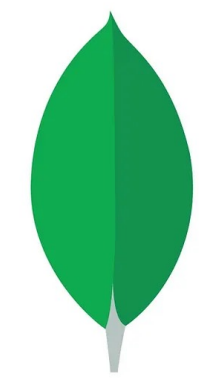
Questionnaires en R



Serveur d'applications



Base de données



mongoDB

Espace contrôlé & limité



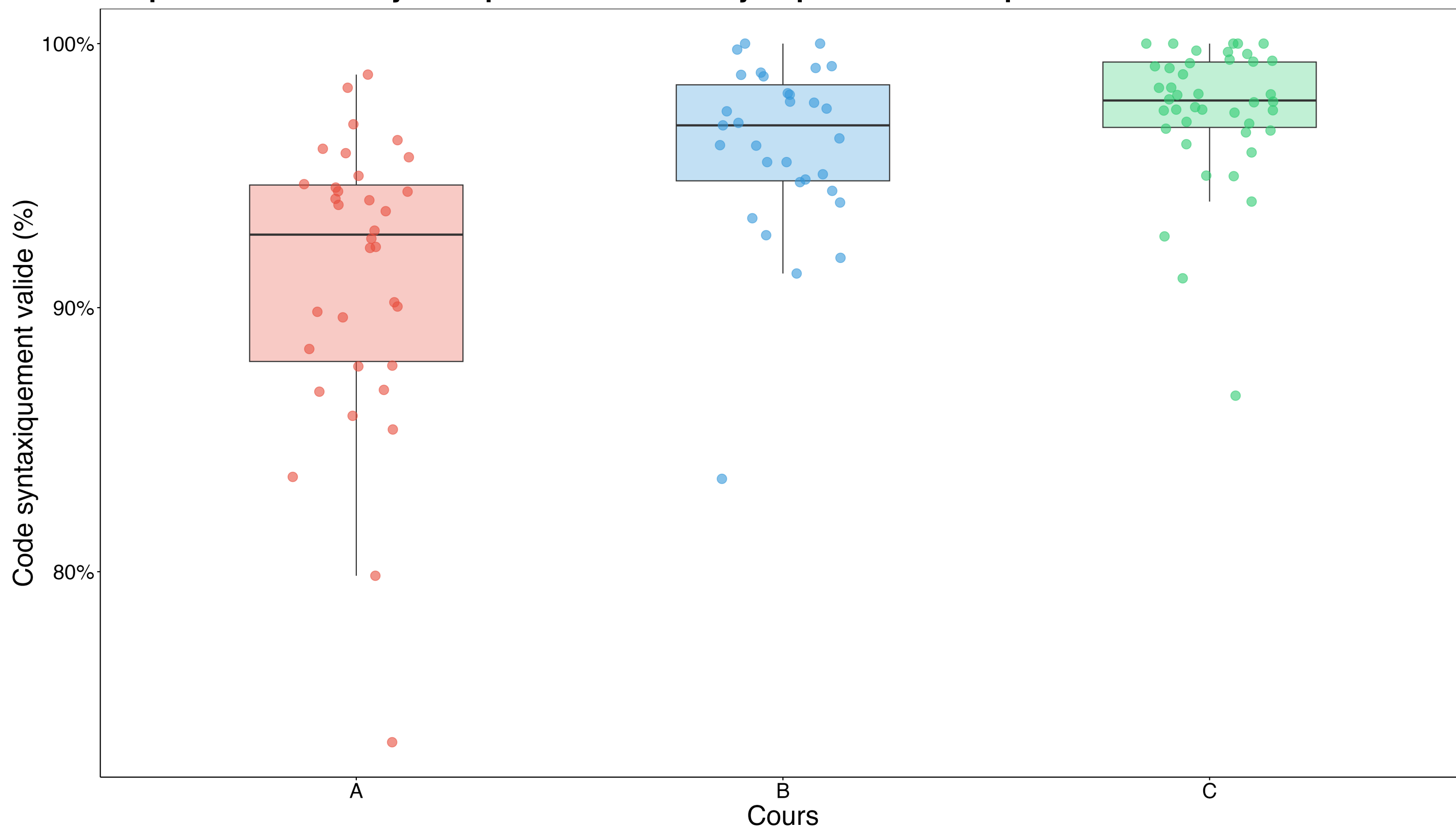
Safe Exam Browser & AppArmor

Questionnaire en ligne

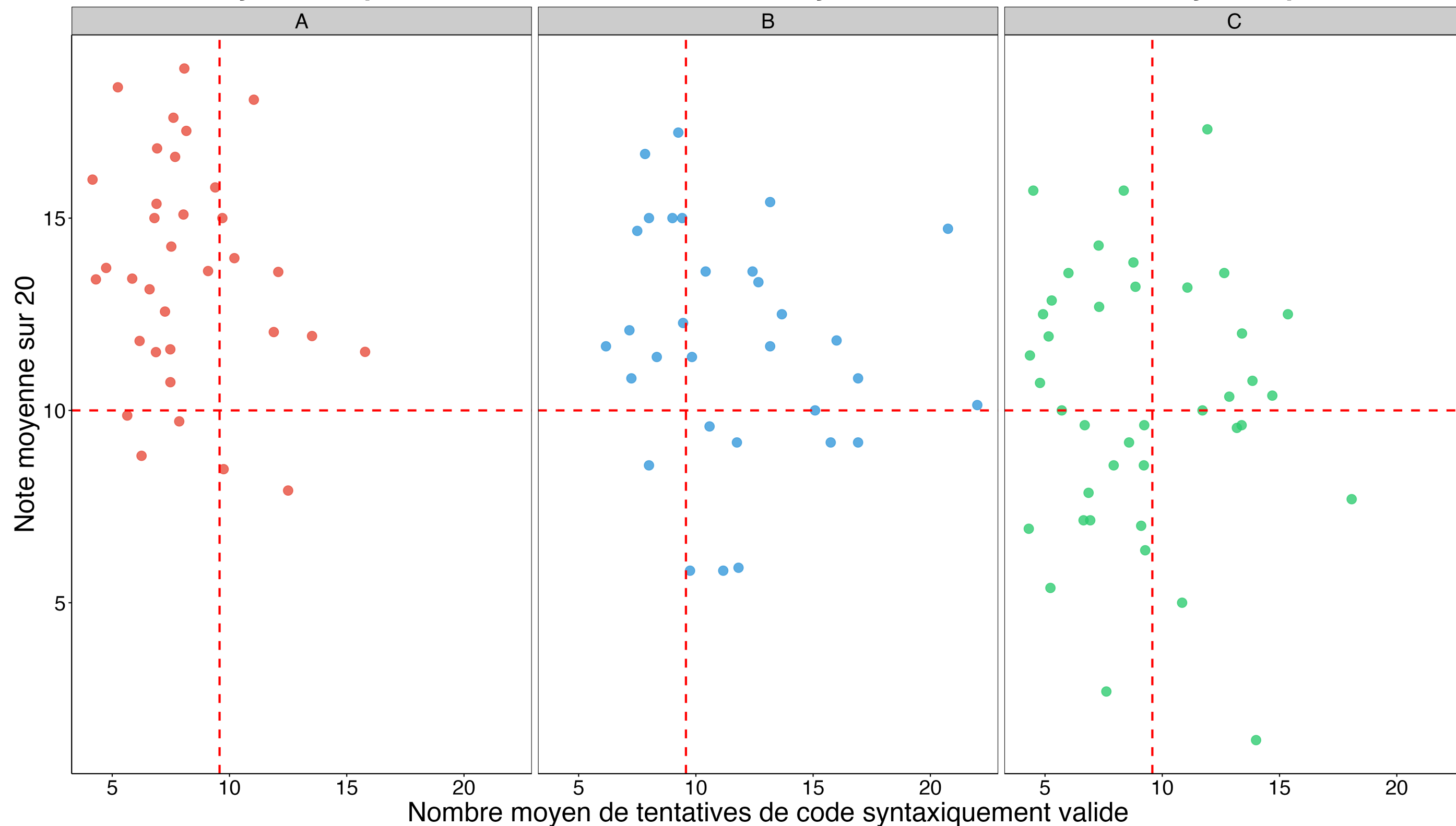
Exécuter du code

Analyse des traces d'apprentissage

Proportion de code syntaxiquement valide moyen par étudiant et par cours



Résultat moyen aux questions de code vs nombre moyen de tentatives de code syntaxiquement valide



- Amélioration dans l'écriture du code R syntaxiquement valide au fil des années (A = Bab2, B = Bab3, C = Ma1) = maîtrise du langage R ↗
- Difficulté croissante du code au cours des années

Génération d'un corrigé pour chaque étudiant

Question 1

0/2 - Par vraiment. Ils se différencient par la fonction de lien du GLM qui permet d'avoir une variable réponse à distribution non normale.

Question 2

1/2 - Tu ne tiens pas compte du nombre d'observations (donc implants)

dose	dead	implants	alive
0	0	3	3
0	2	8	6
0	0	8	8
0	1	9	8
0	1	11	10
...
0.4	0	13	13
0.4	0	13	13
0.4	1	14	13
0.4	0	15	15
0.4	2	21	19

```

try({# Code en réponse à la question:
# Calcul de la nouvelle variable
boric <- smutate(boric, "fract" = dead/implants)
# Modèle
boric_glm <- glm(data = boric, fract ~ dose, family = quasibinomial)
summary(boric_glm)
})

Call:
glm(formula = fract ~ dose, family = quasibinomial, data = boric)

Coefficients:
(Intercept) -2.3889  0.1416 -16.249 < 2e-16 ***
dose.L      0.7444  0.2588  2.877  0.00489 **
dose.0      0.6814  0.2832  2.486  0.01791 *
dose.C      0.6884  0.2827  2.484  0.02228 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasibinomial family taken to be 0.1582864)

Null deviance: 21.285 on 186 degrees of freedom
Residual deviance: 17.381 on 183 degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 5

```

Question 3

1/2 - L'item 7 devait être coché. L'item 8 ne devait pas être coché.

- Interrogation en ligne : plus de papier et interactivité pour tester le code
- Document correctif généré à partir des questions et grilles de correction
- Feedback individualisé