Compte-rendu du TP
“Evaluation empirique d’un programme”

Nom, prénom et parcours de chaque étudiant

17 janvier 2016

Table des matières

Vous devrez rendre ce rapport complété au format PDF exclusivement, en l'uploadant dans la colonne RenduTPEvalPerf de l'UE M2COM dans Tomuss (`http://tomuss.univ-lyon1.fr`), avant le 26 janvier à 23h59. N'envoyez pas d'email. Ne rendez pas le code source Latex. Ne rendez pas non plus d'autres fichiers (données brutes sous Excel, figures, code C, etc). Les figures pertinentes et les lignes de code que vous avez modifiées devront être incluses dans le rapport PDF. La taille maximale de ce rapport est de 15 pages.

1 Présentation sommaire du programme à évaluer

1.1 Données à fournir en entrée

1.2 Données fournies en sortie

1.3 Algorithme

Décrivez en français ce que vous avez compris de l'algorithme. Expliquez entre autres ce que représentent n, m, et k, et en quoi l'algorithme est stochastique. Vous pouvez en plus écrire l'algorithme sous forme de pseudocode (bonus). Voir `https://en.wikibooks.org/wiki/LaTeX/Algorithms#Typesetting_using_the_algorithmic_package`pour plus d'informations.

Algorithm 1 <your caption for this algorithm>

```
if i ≥ maxval then
    i ← 0
else
    if i + k ≤ maxval then
        i ← i + k
    end if
end if
```

2 Premières mesures du temps d'exécution

2.1 Paramètres utilisés dans toute cette section

Indiquez ici les valeurs de m et k avec lesquelles vous avez travaillé pour cette section. Indiquez également le ou les fichiers d'entrée utilisé(s) et la ou les valeur(s) de n correspondantes. Indiquez également votre environnement de travail : système d'exploitation, compilateur (version), options de compilation, processeur, mémoire vive, etc.

Pour chacune des sous-sections suivantes, indiquez le nombre de runs effectués pour soutenir vos résultats.

2.2 Mise en évidence de la stochasticité de l'algorithme

Montrez la variabilité obtenue dans les temps d'exécution en incluant un histogramme du temps CPU dans votre rapport (Figure 4). Quantifiez cette variabilité en indiquant l'écart-type de ce temps d'exécution (sans oublier de préciser le nombre de runs sur lequel cet écart-type a été calculé).

Puis décrivez la manipulation que vous avez effectuée pour montrer qu'une grande part de cette variabilité provient de la stochasticité de l'algorithme.

2.3 Influence de la charge système

En vous appuyant sur le cours, expliquez pourquoi le temps CPU, bien que fait pour être le plus indépendant possible de la charge système, peut tout de même être influencé par la charge système en pratique. Décrivez l'expérience que vous avez effectuée pour tester cet effet et synthétisez-en les résultats, en appuyant vos affirmations sur des chiffres.

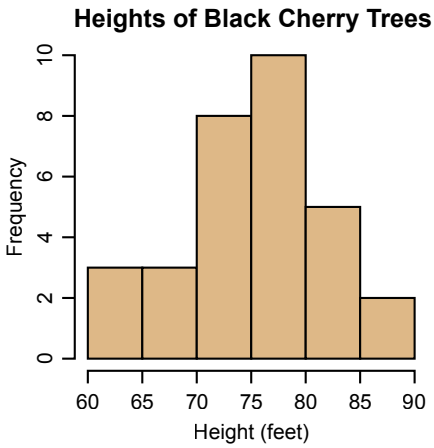

FIGURE 1 – Black cherry tree histogram. Source : Wikimedia Commons.

2.4 Influence des options de compilation

Quantifiez le gain de performance obtenu en passant d'une compilation en mode debug à une compilation en mode release (option d'optimisation) : indiquez le pourcentage d'amélioration ainsi que la façon dont vous l'avez calculé (formule, mais aussi nombre de runs, etc).

3 Opération dominante

3.1 Paramètres utilisés dans toute cette section

3.2 Identification de l'opération dominante

Indiquez le ou les outils de profilage utilisés pour trouver l'opération dominante. Pour appuyer votre réponse, vous pouvez copier-coller un extrait du rapport de profilage, ou inclure une capture d'écran comme figure.

3.3 Comptage des appels à l'opération dominante

Indiquez ici les lignes de code que vous avez modifiées pour compter les appels à l'opération dominante.

```
1 /* A remplacer par les passages que vous avez modifiés */
2 int i = 0;
3 float v = 5.9;
```

4 Analyse de l'expérience fournie par R. Thion

Rappelez le ou les objectifs de cette expérience (voir la partie 4 du sujet de TP) : à quelle(s) question(s) va-t-on répondre ?

4.1 Plan d'expérience

Rappelez ici le plan de l'expérience qui vous a été fournie :

— Variables mesurées
— Paramètres que l'on a fait varier, et les valeurs choisies (sans oublier les fichiers d'entrée).
— Mode de combinaison des valeurs de paramètres (par exemple : full factorial design, si toutes les combinaisons possibles ont été testées)

— Nombre de runs effectués pour chaque combinaison de valeurs de paramètre
— Environnement de test

4.2 Analyse des résultats

Synthétisez les résultats obtenus à l'aide de trois figures maximum. Commentez-les et indiquez dans le texte principal les coefficients de détermination des régressions linéaires (R^2), les coefficients de corrélation et leurs p-values, etc. Concluez sur les questions posées au début de cette section.

5 Conception et réalisation de votre expérience

Indiquez ici le ou les objectifs de votre propre expérience (voir la partie 4 du sujet de TP, éventuellement enrichie de vos propres questions additionnelles).

5.1 Plan d'expérience

Décrivez ici votre plan d'expérience :
— Variables mesurées
— Paramètres que vous avez fait varier, et les valeurs choisies (justifiez). On peut considérer ici que les fichiers d'entrée font partie des paramètres, donc inpliquez lesquels vous avez choisis et expliquez pourquoi.
— Mode de combinaison des valeurs de paramètres (par exemple : full factorial design, si toutes les combinaisons possibles ont été testées)
— Nombre de runs effectués pour chaque combinaison de valeurs de paramètre
— Environnement de test

Si vous avez utilisé un outil de type "cahier de laboratoire" (papier ou numérique), indiquez-le ici, en incluant une figure (scan ou copie d'écran).

5.2 Analyse des résultats

Synthétisez les résultats obtenus à l'aide de trois figures maximum, et commentez-les. Les relations obtenues correspondent-elles à ce qu'on attend étant donné l'algorithme ? Quelle est l'influence de chaque paramètre ? Obtenez-vous les mêmes résultats qu'avec l'expérience fournie ?

6 Annexe : Code réalisé pour le tutoriel sur les tests d'hypothèse

```
#########################
# Cas ou H0 est vraie
#########################

# On va considerer deux populations qui ont exactement
# les memes parametres : les deux sont uniformement
# distribuees entre 0 et 1. Les deux populations ont
# donc la meme moyenne, en l'occurrence 0.5.
# Puis nous allons faire une experience :
# - tirer un petit echantillon dans chaque population
# - calculer la moyenne observee dans l'echantillon 1,
#   puis celle observee dans l'echantillon 2. On
#   n'obtiendra pas exactement 0.5 ni pour l'une ni pour
#   l'autre.
# - calculer la difference entre les deux moyennes
#   d'echantillons. Appelons d cette difference.

tirerEchantillonPop1H0 <- function(n) {
  vraieMoyenne <- 0.5
  plageDeDispersion <- 1.0
  min <- vraieMoyenne - plageDeDispersion/2.0
  max <- vraieMoyenne + plageDeDispersion/2.0
  ech <- runif(n, min, max)
  return(ech)
}
```

```
26
27 tirerEchantillonPop2H0 <- function(n) {
28   vraieMoyenne <- 0.5
29   plageDeDispersion <- 1.0
30   min <- vraieMoyenne - plageDeDispersion/2.0
31   max <- vraieMoyenne + plageDeDispersion/2.0
32   ech <- runif(n, min, max)
33   return(ech)
34 }
35
36 # Ecrire ici le code permettant de faire une "experience"
37 # c'est a dire tirer un echantillon de taille 20 dans
38 # chaque population et mesurer la difference d observee
39 # entre les 2 moyennes d'echantillon.
40 pop1 <- tirerEchantillonPop1H0(20)
41 pop2 <- tirerEchantillonPop2H0(20)
42
43 mean(pop1)
44 mean(pop2)
45 abs(mean(pop1) - mean(pop2))
46 abs(mean(tirerEchantillonPop1H0(20)) - mean(tirerEchantillonPop2H0(20)))
47
48 # Si 1000 etudiants font independamment cette experience,
49 # chaque etudiant aura des echantillons differents et donc
50 # chacun aura une valeur differente pour d. Si nous
51 # mettons toutes ces 1000 valeurs ensemble dans un vecteur
52 # (appele diffMoyH0), quelle sera la moyenne de ce vecteur ?
53
54 diffMoyH0 = c()
55
56 for(i in  1:1000)
57 {
58   diffMoyH0 = c(diffMoyH0, abs(mean(tirerEchantillonPop1H0(20)) - mean(tirerEchantillonPop2H0
         (20))))
59 }
60
61 # On affiche la moyenne totale.
62 mean(diffMoyH0)
63
64 # On affiche les 1000 valeurs sur un histogramme.
65 plot(diffMoyH0)
66 abline(h=mean(diffMoyH0), col="red")
```

On obtient une moyenne de 0,07.

```
1
2 ########################
3 # Cas ou H1 est vraie
4 ########################
5
6 # On va voir comment se comporte notre indicateur quand
7 # les deux populations n'ont pas la meme moyenne
8 # (mais ont tout de meme la meme dispersion)
9 # eg . pop1 comprise entre 0.0 et 1.0
10 #   et pop2 comprise entre 0.2 et 1.2
11
12 tirerEchantillonPop1H1 <- function(n) {
13 vraieMoyenne <- 0.5
14 plageDeDispersion <- 1.0
15 min <- vraieMoyenne - plageDeDispersion/2.0
16 max <- vraieMoyenne + plageDeDispersion/2.0
17 ech <- runif(n, min, max)
18 return(ech)
19 }
20
21 tirerEchantillonPop2H1 <- function(n) {
22 vraieMoyenne <- 0.7
23 plageDeDispersion <- 1.0
24 min <- vraieMoyenne - plageDeDispersion/2.0
25 max <- vraieMoyenne + plageDeDispersion/2.0
26 ech <- runif(n, min, max)
27 return(ech)
28 }
29
```

```r

# Ecrire ici le code pour faire les 1000 experiences
# comme precedemment, en nommant cette fois le vecteur
# diffMoyH1. Calculez sa moyenne et tracez son histogramme.

diffMoyH1 = c()

for(i in  1:1000)
{
  diffMoyH1 = c(diffMoyH1, abs(mean(tirerEchantillonPop1H1(20)) - mean(tirerEchantillonPop2H1(20))))
}

# On affiche la moyenne totale.
mean(diffMoyH1)

# On affiche les 1000 valeurs sur un histogramme.
plot(diffMoyH1)
abline(h=mean(diffMoyH1), col="red")
```

On obtient une moyenne de 0,2.

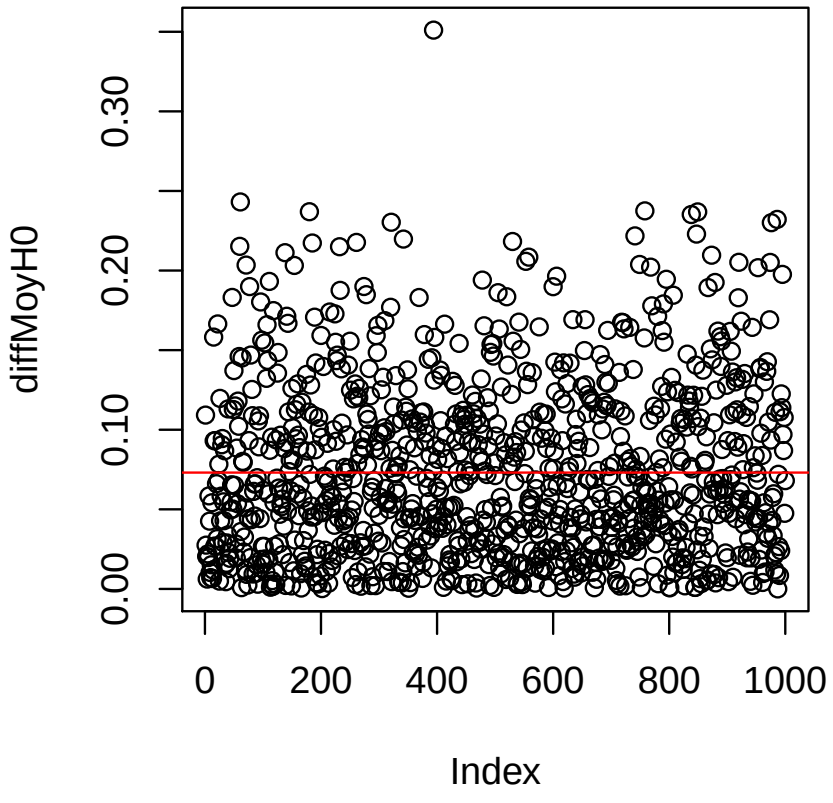

FIGURE 2 – Black cherry tree histogram. Source : Wikimedia Commons.

Placons nous maintenant dans le cas ou l'on ait fait une seule experience.

Ecrire le code pour evaluer la plausibilite de H0 : "les deux echantillons proviennent de deux populations qui ont la meme moyenne".

```r
cMean <- function(v1, v2)
```

```
{
  return(abs(mean(v1) - mean(v2)))
}
sameMean <- function(v1, v2)
{
  return(cMean(v1, v2) < 0.1)
}

sameMean(tirerEchantillonPop1H0(20), tirerEchantillonPop2H0(20))
sameMean(tirerEchantillonPop1H1(20), tirerEchantillonPop2H1(20))

x1 <- c(0.97050525, 0.13734516, 0.80793033, 0.05207726, 0.62629180, 0.93485856,
0.58220744, 0.65935145, 0.76467195, 0.73512414, 0.45139560, 0.93225380,
0.53790595, 0.99845675, 0.31035081, 0.43082815, 0.15475353, 0.42647652,
0.65676067, 0.74186048)
x2 <- c(0.33565036, 0.28830545, 0.51556544, 0.93223089, 0.29192576, 0.43505823,
0.63127002, 0.86082799, 0.56533392, 0.19083212, 0.13087779, 0.09849703,
0.98921291, 0.91480756, 0.78556552, 0.33859160, 0.88482223, 0.76701274,
0.24190609, 0.46251866)

cMean(x1, x2)
sameMean(x1, x2)
```

Ici, sameMean(x1, x2) retourne "TRUE", ce qui signifie que les deux échantillons ont une moyenne proche l'une de l'autre.

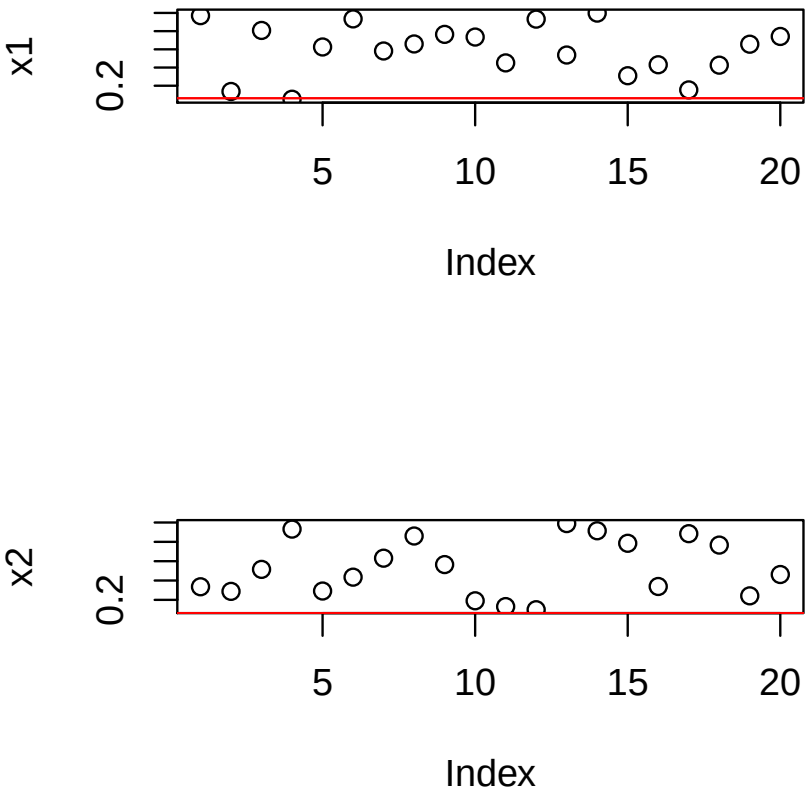

FIGURE 3 – Black cherry tree histogram. Source : Wikimedia Commons.

```
x1 <- c(0.41236444, 0.28422821, 0.15093798, 0.05885328, 0.25514435, 0.63026931,
```

```
0.56325462, 0.76304859, 0.56523993, 0.92535660, 0.10898729, 0.51579642,
0.07223967, 0.53483839, 0.52516575, 0.20250815, 0.89634680, 0.53879059,
0.58736912, 0.53945749)
x2 <- c(1.0809923, 0.5772333, 0.7252340, 1.1529082, 1.0924642, 0.6046166, 0.9495800,
0.3019857, 0.7701195, 1.0746508, 0.3928894, 1.0885017, 0.5101510, 1.1871599,
0.7953318, 0.5711237, 0.5505642, 0.9854085, 0.5105643, 0.9601635)

cMean(x1, x2)
sameMean(x1, x2)
```

Ici, sameMean(x1, x2) retourne "FALSE", ce qui signifie que les deux échantillons ont une moyenne bien différente l'une de l'autre.

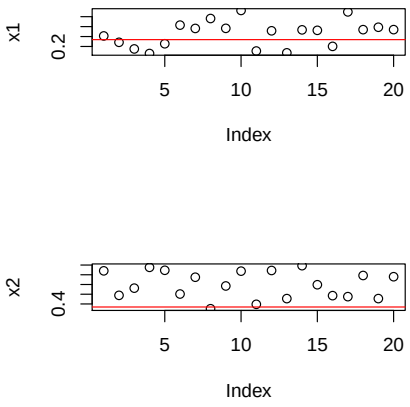

FIGURE 4 – Black cherry tree histogram. Source : Wikimedia Commons.

Incorporez également les histogrammes obtenus sous forme de figures. Indiquez vos conclusions sur les deux paires d'échantillons.