Debogage, profilage et benchmarking

Techniques avancées en programmation statistique R

Patrick Fournier

Automne 2019

Université du Québec à Montréal

Introduction

Introduction

- ⇝ Opérations facilitées par les capacités d'introspection de R.

Introduction

- ⇝ Opérations facilitées par les capacités d'introspection de R.
- ⇝ Possibilité de les mener interactivement dans la console.

Introduction

- ⇝ Opérations facilitées par les capacités d'introspection de R.
- ⇝ Possibilité de les mener interactivement dans la console.
- ⇝ Pour des problèmes minimalement complexes, difficile d'évaluer une solution à priori.

Introduction

- ⇝ Opérations facilitées par les capacités d'introspection de R.
- ⇝ Possibilité de les mener interactivement dans la console.
- ⇝ Pour des problèmes minimalement complexes, difficile d'évaluer une solution à priori.
- ⇝ Il ne faut pas hésiter à implémenter et à comparer différentes solutions.

Debogage

Qu’est-ce que le debugage ?

⇝ Processus de recherche et de correction des bugs.

Qu’est-ce que le debugage ?

- ⇝ Processus de recherche et de correction des bugs.
- ⇝ De ce point de vue, tout le monde a déjà débogué !

Qu’est-ce que le debugage ?

- ⇝ Processus de recherche et de correction des bugs.
- ⇝ De ce point de vue, tout le monde a déjà débogué !
- ⇝ Multiples manières de deboguer.

Qu’est-ce que le debugage ?

- ⇝ Processus de recherche et de correction des bugs.
- ⇝ De ce point de vue, tout le monde a déjà débogué !
- ⇝ Multiples manières de deboguer.

  Méthode naïve

  À la main, avec `print` et modifications directes du code.

Qu’est-ce que le debugage ?

- ⇝ Processus de recherche et de correction des bugs.
- ⇝ De ce point de vue, tout le monde a déjà débogué !
- ⇝ Multiples manières de deboguer.

  Méthode naïve

  À la main, avec `print` et modifications directes du code.

  Méthode interactive

  Utilisation d’un débogueur.

Déboguage naïf

- ⇝ Probablement la méthode la plus employée !

Déboguage naïf

- ⇝ Probablement la méthode la plus employée !
- ⇝ Convient pour de courtes fonctions/bugs simples.

Déboguage naïf

- ⇝ Probablement la méthode la plus employée !
- ⇝ Convient pour de courtes fonctions/bugs simples.

Workflow

1 : Ajouter des `print` au code.
2 : Exécuter le code augmenté.
3 : Modifier le code.
4 : Répéter.

Déboguage interactif

⇝ De base, R vient avec un débogueur respectable.

Déboguage interactif

- ⇝ De base, R vient avec un débogueur respectable.
- ⇝ RStudio enrichit grandement l'expérience de déboguage [1].

Déboguage interactif

- ⇝ De base, R vient avec un débogueur respectable.
- ⇝ RStudio enrichit grandement l'expérience de déboguage [1].

Workflow

1 : Ajouter des breakpoints au code.
2 : Exécuter du code interactivement.
3 : Corriger le code.

Exemple

```
1  cutVector <- function(vec, m)
2      list(n = length(vec) / m) %$%
3          {lapply(0:(m - 1),
4                  function(x) vec[seq(n * x + 1, n * (x + 1))])}
5
6  dist0_bad <- function(m, n = 1e4){
7      dat <- sample(0:9, size = n * m, replace = TRUE) %>%
8          cutVector(m)
9
10     lapply(dat, function(v) sapply(v, identical, y = 0)) %>%
11         sapply(mean)
12 }
```

Exemple

```r
dist0 <- function(m, n = 1e4){
    dat <- sample(0:9, size = n * m, replace = TRUE) %>%
        cutVector(m) %>%
        lapply(as.numeric)

    lapply(dat, function(v) sapply(v, identical, y = 0)) %>%
        sapply(mean)
}
```

Profilage

Qu’est-ce que le profilage ?

$\rightsquigarrow$ Analyse d’un programme visant à mesurer sa complexité.

Qu’est-ce que le profilage ?

- ⇝ Analyse d’un programme visant à mesurer sa complexité.
- ⇝ Pour cette première approche, nous nous restreignons à la complexité temporelle (temps d’exécution).

Qu’est-ce que le profilage ?

- ⇝ Analyse d’un programme visant à mesurer sa complexité.
- ⇝ Pour cette première approche, nous nous restreignons à la complexité temporelle (temps d’exécution).
- ⇝ Sachez toutefois que d’autres métriques peuvent être d’intérêt, par exemple

Qu’est-ce que le profilage ?

- ⇝ Analyse d’un programme visant à mesurer sa complexité.
- ⇝ Pour cette première approche, nous nous restreignons à la complexité temporelle (temps d’exécution).
- ⇝ Sachez toutefois que d’autres métriques peuvent être d’intérêt, par exemple
  - ⇝ complexité spatiale (mémoire utilisée),

Qu’est-ce que le profilage ?

- ⇝ Analyse d’un programme visant à mesurer sa complexité.
- ⇝ Pour cette première approche, nous nous restreignons à la complexité temporelle (temps d’exécution).
- ⇝ Sachez toutefois que d’autres métriques peuvent être d’intérêt, par exemple
  - ⇝ complexité spatiale (mémoire utilisée),
  - ⇝ utilisation d’une instruction précise,

Qu’est-ce que le profilage ?

- ⇝ Analyse d’un programme visant à mesurer sa complexité.
- ⇝ Pour cette première approche, nous nous restreignons à la complexité temporelle (temps d’exécution).
- ⇝ Sachez toutefois que d’autres métriques peuvent être d’intérêt, par exemple
  - ⇝ complexité spatiale (mémoire utilisée),
  - ⇝ utilisation d’une instruction précise,
  - ⇝ fréquence des appels à des fonctions,

Qu’est-ce que le profilage ?

- ⇝ Analyse d’un programme visant à mesurer sa complexité.
- ⇝ Pour cette première approche, nous nous restreignons à la complexité temporelle (temps d’exécution).
- ⇝ Sachez toutefois que d’autres métriques peuvent être d’intérêt, par exemple
  - ⇝ complexité spatiale (mémoire utilisée),
  - ⇝ utilisation d’une instruction précise,
  - ⇝ fréquence des appels à des fonctions,
  - ⇝ . . .

Qu’est-ce que le profilage ?

- ⇝ Analyse d’un programme visant à mesurer sa complexité.
- ⇝ Pour cette première approche, nous nous restreignons à la complexité temporelle (temps d’exécution).
- ⇝ Sachez toutefois que d’autres métriques peuvent être d’intérêt, par exemple
  - ⇝ complexité spatiale (mémoire utilisée),
  - ⇝ utilisation d’une instruction précise,
  - ⇝ fréquence des appels à des fonctions,
  - ⇝ . . .
- ⇝ Méthodes et objectifs distincts du benchmarking (prochaine section).

Quand profiler ?

$\rightsquigarrow$ Possibilité de profiler $\neq$ nécessité de profiler.

Quand profiler ?

- ⇝ Possibilité de profiler ≠ nécessité de profiler.
- ⇝ Le profilage *devrait être la première étape* de l'optimisation d'un programme.

Quand profiler ?

- ⇝ Possibilité de profiler ≠ nécessité de profiler.
- ⇝ Le profilage *devrait être la première étape* de l'optimisation d'un programme.
- ⇝ Un programme ne devrait être optimisé que si cela est nécessaire.

Quand profiler ?

- ⇝ Possibilité de profiler ≠ nécessité de profiler.
- ⇝ Le profilage *devrait être la première étape* de l'optimisation d'un programme.
- ⇝ Un programme ne devrait être optimisé que si cela est nécessaire.
- ⇝ Donald E. Knuth : "Premature optimization is the root of all evil." [2] Le processus d'optimisation

Quand profiler ?

- ⇝ Possibilité de profiler ≠ nécessité de profiler.
- ⇝ Le profilage *devrait être la première étape* de l'optimisation d'un programme.
- ⇝ Un programme ne devrait être optimisé que si cela est nécessaire.
- ⇝ Donald E. Knuth : "Premature optimization is the root of all evil." [2] Le processus d'optimisation
  - ⇝ prend du temps,

Quand profiler ?

- ⇝ Possibilité de profiler ≠ nécessité de profiler.
- ⇝ Le profilage *devrait être la première étape* de l'optimisation d'un programme.
- ⇝ Un programme ne devrait être optimisé que si cela est nécessaire.
- ⇝ Donald E. Knuth : "Premature optimization is the root of all evil." [2] Le processus d'optimisation
  - ⇝ prend du temps,
  - ⇝ est prompt à l'erreur et

Quand profiler ?

- ⇝ Possibilité de profiler ≠ nécessité de profiler.
- ⇝ Le profilage *devrait être la première étape* de l'optimisation d'un programme.
- ⇝ Un programme ne devrait être optimisé que si cela est nécessaire.
- ⇝ Donald E. Knuth : "Premature optimization is the root of all evil." [2] Le processus d'optimisation
  - ⇝ prend du temps,
  - ⇝ est prompt à l'erreur et
  - ⇝ rend (souvent) le code plus difficile à comprendre.

Comment profiler ?

⇝ De base, R fournit la fonction `Rprof`.

Comment profiler ?

- ⇝ De base, R fournit la fonction `Rprof`.
- ⇝ Peu commode à utiliser ⇒ package `proftools`.

Comment profiler ?

- ⇝ De base, R fournit la fonction `Rprof`.
- ⇝ Peu commode à utiliser ⇒ package `proftools`.

  `profileExpr`

  Profiler une expression.

Comment profiler ?

- ⇝ De base, R fournit la fonction `Rprof`.
- ⇝ Peu commode à utiliser ⇒ package `proftools`.

`profileExpr`
: Profiler une expression.

`filterProfileData`
: Filtrer les résultats du profilage.

Comment profiler ?

- ⇝ De base, R fournit la fonction `Rprof`.
- ⇝ Peu commode à utiliser ⇒ package `proftools`.

`profileExpr`
: Profiler une expression.

`filterProfileData`
: Filtrer les résultats du profilage.

`hotPaths`
: Donne la fréquence d'exécution des parties d'un programme.

Comment profiler ?

- ⇝ De base, R fournit la fonction `Rprof`.
- ⇝ Peu commode à utiliser ⇒ package `proftools`.

`profileExpr`
: Profiler une expression.

`filterProfileData`
: Filtrer les résultats du profilage.

`hotPaths`
: Donne la fréquence d'exécution des parties d'un programme.

`funSummary`
: Donne la fréquence d'exécution des fonctions.

Exemple

```
lmSlow <- function(m){
    res <- list()

    for (kk in 1:m){
        random_values <- rnorm(1e6)
        X <- matrix(random_values, ncol = 20)
        y <- rnorm(5e4)
        reg <- lm(y ~ X)
        b <- coef(r)
        res %<>% append(reg)
    }
    res
}
```

Exemple

```
> system.time(pro_slow <- profileExpr(lmSlow(20)))
   user  system elapsed
  6.832   3.107   3.490
> hotPaths(pro_slow %>% filterProfileData(focus = "lmSlow",
↪  maxdepth = 2))
 path                                total.pct self.pct
 lmSlow                              100.00     0.00
 . rnorm (examples.R!InnVue:5)        71.89    71.89
 . lm (examples.R!InnVue:8)           24.86    24.86
 . matrix (examples.R!InnVue:6)        2.29     2.29
 . rnorm (examples.R!InnVue:7)         0.96     0.96
```

Le $^3/_4$ du temps d'exécution est passé à générer des $\mathcal{N}$!

Exemple

```
1  lmFast <- function(m){
2      Xs <- rnorm(1e6 * m) %>%       ## Génération des données.
3          cutVector(m) %>%           ## 1 entrée = 1 simulation.
4          lapply(matrix, ncol = 20) ## vecteur -> matrice.
5
6      ys <- rnorm(5e4 * m) %>%
7          cutVector(m)
8
9      mapply(function(X, y) lm(y ~ X) %>% coef, Xs, ys)
10 }
```

Exemple

```
> system.time(pro_fast <- profileExpr(lmFast(20)))
   user  system elapsed
  4.849   1.607   2.831
> funSummary(pro_fast %>% filterProfileData(focus = "rnorm"))
      total.pct gc.pct self.pct gcself.pct
rnorm        20      0       20           0
```

Seulement 1/5 du temps est consacré à la génération de $\mathcal{N}$.

Exemple

Points positifs

Exemple

Points positifs

- ⇝ Speedup de 1.23 ! Nous reviendrons sur ce point.

Exemple

Points positifs

- $\rightsquigarrow$ Speedup de 1.23 ! Nous reviendrons sur ce point.
- $\rightsquigarrow$ ...

Exemple

Points positifs

- $\rightsquigarrow$ Speedup de 1.23 ! Nous reviendrons sur ce point.
- $\rightsquigarrow$ ...

Points négatifs

Exemple

Points positifs

- ⇝ Speedup de 1.23 ! Nous reviendrons sur ce point.
- ⇝ . . .

Points négatifs

- ⇝ Code moins lisible

Exemple

Points positifs

- ⇝ Speedup de 1.23 ! Nous reviendrons sur ce point.
- ⇝ . . .

Points négatifs

- ⇝ Code moins lisible ?

Exemple

Points positifs

- ⇝ Speedup de 1.23 ! Nous reviendrons sur ce point.
- ⇝ . . .

Points négatifs

- ⇝ Code moins lisible ?
- ⇝ Plus grande utilisation de mémoire.

Exemple

Points positifs

- ⇝ Speedup de 1.23 ! Nous reviendrons sur ce point.
- ⇝ . . .

Points négatifs

- ⇝ Code moins lisible ?
- ⇝ Plus grande utilisation de mémoire.
- ⇝ . . .

Exemple

Points positifs

- ⇝ Speedup de 1.23 ! Nous reviendrons sur ce point.
- ⇝ . . .

Points négatifs

- ⇝ Code moins lisible ?
- ⇝ Plus grande utilisation de mémoire.
- ⇝ . . .

Moralité

Exemple

Points positifs

- ⇝ Speedup de 1.23 ! Nous reviendrons sur ce point.
- ⇝ . . .

Points négatifs

- ⇝ Code moins lisible ?
- ⇝ Plus grande utilisation de mémoire.
- ⇝ . . .

Moralité

- ⇝ Pas de réponse absolue.

Exemple

Points positifs

- ⇝ Speedup de 1.23 ! Nous reviendrons sur ce point.
- ⇝ . . .

Points négatifs

- ⇝ Code moins lisible ?
- ⇝ Plus grande utilisation de mémoire.
- ⇝ . . .

Moralité

- ⇝ Pas de réponse absolue.
- ⇝ On ne peut juger d'une solution/approche que lorsqu'elle est mise en contexte.

Benchmarking

Qu’est-ce que le benchmarking ?

- ⇝ Comparaison de plusieurs programmes/approches/méthodes/...

Qu’est-ce que le benchmarking ?

- ⇝ Comparaison de plusieurs programmes/approches/méthodes/...
- ⇝ Tout comme le profilage, peuvent être comparés en fonction de différents critères.

Qu’est-ce que le benchmarking ?

- ⇝ Comparaison de plusieurs programmes/approches/méthodes/...
- ⇝ Tout comme le profilage, peuvent être comparés en fonction de différents critères.
- ⇝ Tout comme pour le profilage, nous nous limiterons aux aspects temporels.

Qu’est-ce que le benchmarking ?

- ⇝ Comparaison de plusieurs programmes/approches/méthodes/...
- ⇝ Tout comme le profilage, peuvent être comparés en fonction de différents critères.
- ⇝ Tout comme pour le profilage, nous nous limiterons aux aspects temporels.
- ⇝ Vise à *choisir* un programme dans un ensemble plutôt qu’à *améliorer* un programme.

Qu’est-ce que le benchmarking ?

- ⇝ Comparaison de plusieurs programmes/approches/méthodes/...
- ⇝ Tout comme le profilage, peuvent être comparés en fonction de différents critères.
- ⇝ Tout comme pour le profilage, nous nous limiterons aux aspects temporels.
- ⇝ Vise à *choisir* un programme dans un ensemble plutôt qu’à *améliorer* un programme.
- ⇝ Exemple : étude de simulation.

Comment benchmarker ?

Méthode naïve : `system.time`

Points positifs

Comment benchmarker ?

Méthode naïve : `system.time`

Points positifs

- ⇝ Très facile à mettre en place.

Comment benchmarker ?

Méthode naïve : `system.time`

Points positifs

- ⇝ Très facile à mettre en place.
- ⇝ Très peu d'overhead.

Comment benchmarker ?

Méthode naïve : `system.time`

Points positifs

- ⇝ Très facile à mettre en place.
- ⇝ Très peu d'overhead.
- ⇝ Facile à interpréter.

Comment benchmarker ?

Méthode naïve : `system.time`

Points positifs

- ⇝ Très facile à mettre en place.
- ⇝ Très peu d'overhead.
- ⇝ Facile à interpréter.

Points négatifs

Comment benchmarker ?

Méthode naïve : `system.time`

Points positifs

- ⇝ Très facile à mettre en place.
- ⇝ Très peu d'overhead.
- ⇝ Facile à interpréter.

Points négatifs

- ⇝ *Ne prend pas en compte l'erreur aléatoire associée à l'exécution.*

Comment benchmarker ?

Méthode naïve : `system.time`

Points positifs

- ⇝ Très facile à mettre en place.
- ⇝ Très peu d'overhead.
- ⇝ Facile à interpréter.

Points négatifs

- ⇝ *Ne prend pas en compte l'erreur aléatoire associée à l'exécution.*

Ce dernier point est suffisamment grave pour qu'on se tourne vers une autre solution pour tout benchmarking sérieux.

Comment benchmarker ?

Package `microbenchmark`

- ⇝ Facile à utiliser. Une seule fonction

Comment benchmarker ?

Package `microbenchmark`

⇝ Facile à utiliser. Une seule fonction

`microbenchmark`

Procède au benchmark.

Comment benchmarker ?

Package `microbenchmark`

- ⇝ Facile à utiliser. Une seule fonction

  `microbenchmark`

  Procède au benchmark.

- ⇝ Exécute un nombre fixé de fois une expression.

Comment benchmarker ?

Package `microbenchmark`

- ⇝ Facile à utiliser. Une seule fonction

  `microbenchmark`

  Procède au benchmark.

- ⇝ Exécute un nombre fixé de fois une expression.
- ⇝ Fourni des statistiques sommaires sur l'exécution.

Comment benchmarker ?

Package `microbenchmark`

- ⇝ Facile à utiliser. Une seule fonction

  `microbenchmark`

  Procède au benchmark.

- ⇝ Exécute un nombre fixé de fois une expression.
- ⇝ Fourni des statistiques sommaires sur l'exécution.
- ⇝ Format `data.frame` ⇒ possibilité

Comment benchmarker ?

Package `microbenchmark`

- ⇝ Facile à utiliser. Une seule fonction

  `microbenchmark`

  Procède au benchmark.

- ⇝ Exécute un nombre fixé de fois une expression.
- ⇝ Fourni des statistiques sommaires sur l'exécution.
- ⇝ Format `data.frame` ⇒ possibilité
  - ⇝ de représentation graphique et

Comment benchmarker ?

Package `microbenchmark`

- ⇝ Facile à utiliser. Une seule fonction

  `microbenchmark`

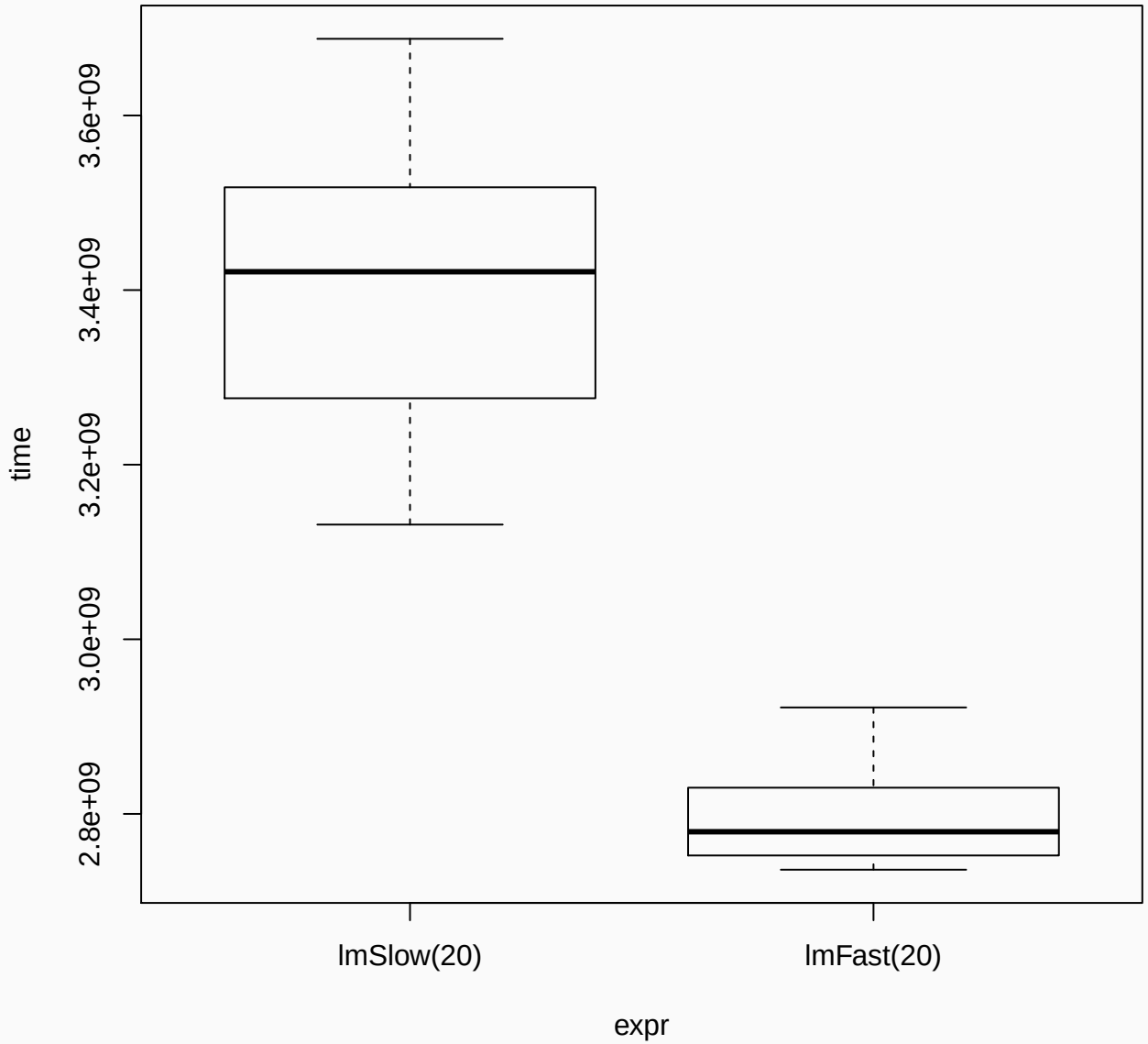
  Procède au benchmark.

- ⇝ Exécute un nombre fixé de fois une expression.
- ⇝ Fourni des statistiques sommaires sur l'exécution.
- ⇝ Format `data.frame` ⇒ possibilité
  - ⇝ de représentation graphique et
  - ⇝ d'analyses statistiques plus poussées.

Exemple

```
> mb_slow <- microbenchmark(lmSlow(20), times = 20)
> mb_fast <- microbenchmark(lmFast(20), times = 20)
> mb_slow
Unit: seconds
        expr  min   lq mean median   uq  max neval
 lmSlow(20) 3.13 3.28 3.41   3.42 3.52 3.69    20
> mb_fast
Unit: seconds
        expr  min   lq mean median   uq  max neval
 lmFast(20) 2.74 2.75  2.8   2.78 2.83 2.92    20
> plot(rbind(mb_slow, mb_fast))
```

Exemple



Exemple

Quelle statistique utiliser ?

Exemple

Quelle statistique utiliser ?

$\rightsquigarrow$ Mesure du temps d'exécution $\Rightarrow$ distribution asymétrique.

Exemple

Quelle statistique utiliser ?

- $\rightsquigarrow$ Mesure du temps d'exécution $\Rightarrow$ distribution asymétrique.
- $\rightsquigarrow$ La moyenne $\neq$ bon estimateur de tendance centrale.

Exemple

Quelle statistique utiliser ?

- $\rightsquigarrow$ Mesure du temps d'exécution $\Rightarrow$ distribution asymétrique.
- $\rightsquigarrow$ La moyenne $\neq$ bon estimateur de tendance centrale.
- $\rightsquigarrow$ Utiliser plutôt un estimateur robuste comme la médiane ou le minimum.

Exemple

Quelle statistique utiliser ?

- $\rightsquigarrow$ Mesure du temps d'exécution $\Rightarrow$ distribution asymétrique.
- $\rightsquigarrow$ La moyenne $\neq$ bon estimateur de tendance centrale.
- $\rightsquigarrow$ Utiliser plutôt un estimateur robuste comme la médiane ou le minimum.

Retour sur l'exemple

Exemple

Quelle statistique utiliser ?

- $\rightsquigarrow$ Mesure du temps d'exécution $\Rightarrow$ distribution asymétrique.
- $\rightsquigarrow$ La moyenne $\neq$ bon estimateur de tendance centrale.
- $\rightsquigarrow$ Utiliser plutôt un estimateur robuste comme la médiane ou le minimum.

Retour sur l'exemple

- $\rightsquigarrow$ Minimum : speedup de 1.23.

Exemple

Quelle statistique utiliser ?

- $\rightsquigarrow$ Mesure du temps d'exécution $\Rightarrow$ distribution asymétrique.
- $\rightsquigarrow$ La moyenne $\neq$ bon estimateur de tendance centrale.
- $\rightsquigarrow$ Utiliser plutôt un estimateur robuste comme la médiane ou le minimum.

Retour sur l'exemple

- $\rightsquigarrow$ Minimum : speedup de 1.23.
- $\rightsquigarrow$ Médiane : speddup de 1.14.

Exemple

Quelle statistique utiliser ?

- $\rightsquigarrow$ Mesure du temps d'exécution $\Rightarrow$ distribution asymétrique.
- $\rightsquigarrow$ La moyenne $\neq$ bon estimateur de tendance centrale.
- $\rightsquigarrow$ Utiliser plutôt un estimateur robuste comme la médiane ou le minimum.

Retour sur l'exemple

- $\rightsquigarrow$ Minimum : speedup de 1.23.
- $\rightsquigarrow$ Médiane : speddup de 1.14.
- $\rightsquigarrow$ Notre estimation naïve du speedup (1.23) était peut-être optimiste...

Références

[1] *Debugging with RStudio*. en-US. URL : http://support.rstudio.com/hc/en-us/articles/205612627-Debugging-with-RStudio.

[2] Donald Ervin KNUTH. *The art of computer programming*. Addison-Wesley series in computer science and information processing. Reading, Massachusetts : Addison-Wesley Publishing Company, 1973. ISBN : 9780201038033 9780201038040 9780134397603.