# Débogage, profilage et benchmarking

## Techniques avancées en programmation statistique R

Patrick Fournier

Automne 2020

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.

# Débogage

# Qu’est-ce que le débogage ?

⇝ Processus de recherche et de correction des bugs.

# Qu’est-ce que le débogage ?

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

# Qu’est-ce que le débogage ?

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

# Qu’est-ce que le débogage ?

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

  **Méthode naïve**

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

# Qu’est-ce que le débogage ?

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

  **Méthode naïve**

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

  **Méthode interactive**

  Utilisation d’un débogueur.

# Débogage naïf

⇝ Probablement la méthode la plus employée !

# Débogage naïf

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

# Débogage 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.

# Débogage 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.

Voir `exemples.ipynb`

# Débogage interactif

$\rightsquigarrow$ De base, R vient avec un débogueur respectable.

# Débogage interactif

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

# Débogage interactif

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

## Workflow

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

# Débogage interactif

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

## Workflow

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

Voir `exemples.ipynb`

# 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 ⇒ RStudio !

# Comment profiler ?

- ⇝ De base, R fournit la fonction `Rprof`.
- ⇝ Peu commode à utiliser ⇒ RStudio !
- ⇝ Voir `exemples.ipynb`

# Exemple

## Points positifs

# Exemple

## Points positifs

$\rightsquigarrow$ Speedup de $\approx 1.2$ ! Nous reviendrons sur ce point.

# Exemple

## Points positifs

- $\rightsquigarrow$ Speedup de $\approx 1.2$ ! Nous reviendrons sur ce point.
- $\rightsquigarrow$ ...

# Exemple

## Points positifs

- $\rightsquigarrow$ Speedup de $\approx 1.2$ ! Nous reviendrons sur ce point.
- $\rightsquigarrow$ ...

## Points négatifs

# Exemple

**Points positifs**

- $\rightsquigarrow$ Speedup de $\approx 1.2$ ! Nous reviendrons sur ce point.
- $\rightsquigarrow$ ...

**Points négatifs**

- $\rightsquigarrow$ Code moins lisible

# Exemple

**Points positifs**

- $\rightsquigarrow$ Speedup de $\approx 1.2$ ! Nous reviendrons sur ce point.
- $\rightsquigarrow$ ...

**Points négatifs**

- $\rightsquigarrow$ Code moins lisible ?

# Exemple

**Points positifs**

- $\rightsquigarrow$ Speedup de $\approx 1.2$ ! Nous reviendrons sur ce point.
- $\rightsquigarrow$ ...

**Points négatifs**

- $\rightsquigarrow$ Code moins lisible ?
- $\rightsquigarrow$ Plus grande utilisation de mémoire.

# Exemple

## Points positifs

- $\rightsquigarrow$ Speedup de $\approx 1.2$ ! Nous reviendrons sur ce point.
- $\rightsquigarrow$ ...

## Points négatifs

- $\rightsquigarrow$ Code moins lisible ?
- $\rightsquigarrow$ Plus grande utilisation de mémoire.
- $\rightsquigarrow$ ...

# Exemple

**Points positifs**

- $\rightsquigarrow$ Speedup de $\approx 1.2$ ! Nous reviendrons sur ce point.
- $\rightsquigarrow$ ...

**Points négatifs**

- $\rightsquigarrow$ Code moins lisible ?
- $\rightsquigarrow$ Plus grande utilisation de mémoire.
- $\rightsquigarrow$ ...

**Moralité**

# Exemple

**Points positifs**

- $\rightsquigarrow$ Speedup de $\approx 1.2$ ! Nous reviendrons sur ce point.
- $\rightsquigarrow$ ...

**Points négatifs**

- $\rightsquigarrow$ Code moins lisible ?
- $\rightsquigarrow$ Plus grande utilisation de mémoire.
- $\rightsquigarrow$ ...

**Moralité**

- $\rightsquigarrow$ Pas de réponse absolue.

# Exemple

**Points positifs**

- $\rightsquigarrow$ Speedup de $\approx 1.2$ ! Nous reviendrons sur ce point.
- $\rightsquigarrow$ ...

**Points négatifs**

- $\rightsquigarrow$ Code moins lisible ?
- $\rightsquigarrow$ Plus grande utilisation de mémoire.
- $\rightsquigarrow$ ...

**Moralité**

- $\rightsquigarrow$ Pas de réponse absolue.
- $\rightsquigarrow$ 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`**

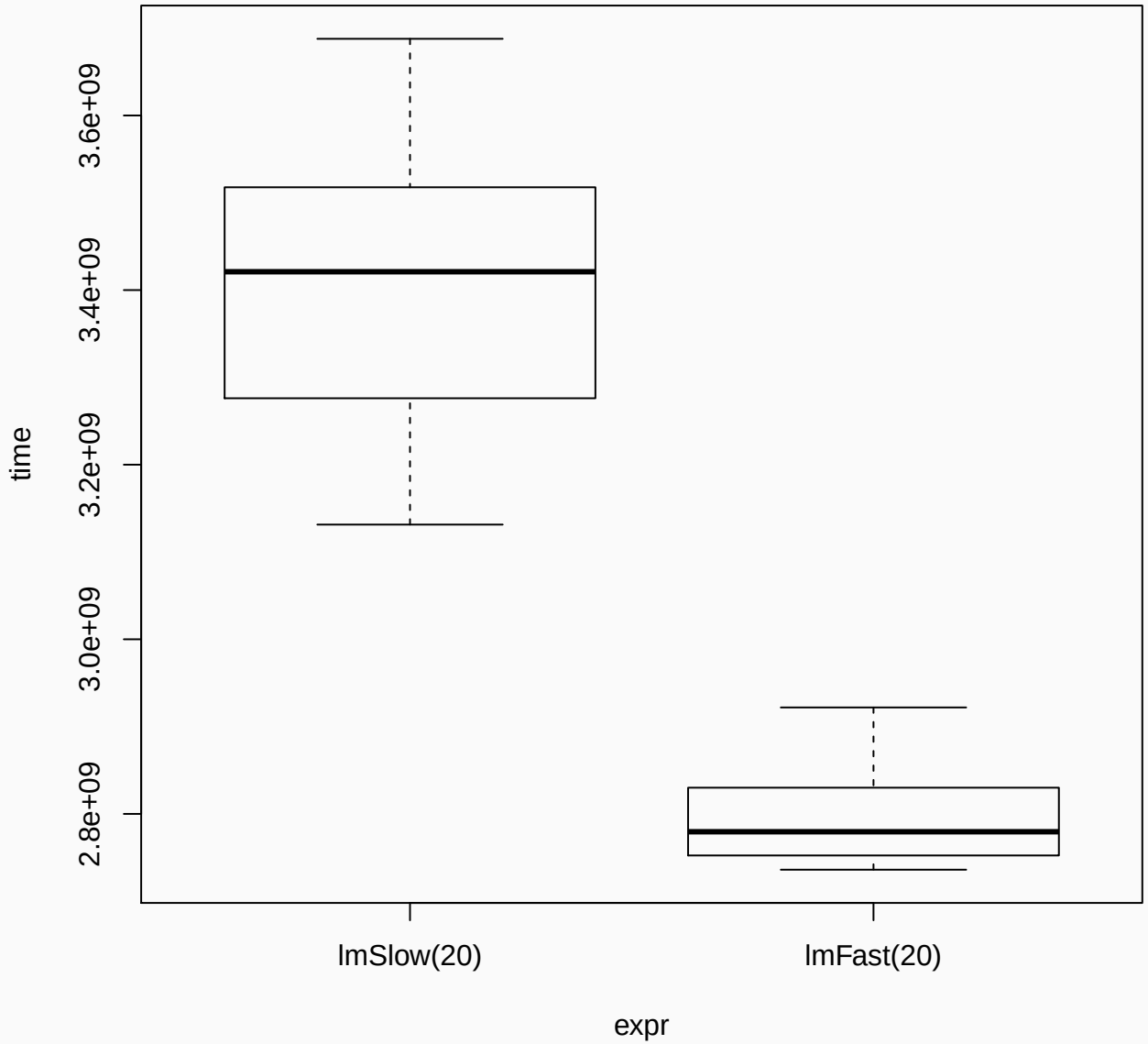
  Procède au benchmark.

- ⇝ 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 $\approx 1.2$.

# 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 $\approx 1.2$.
- $\rightsquigarrow$ Médiane : speddup de $\approx 1.15$.

# 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 $\approx 1.2$.
- $\rightsquigarrow$ Médiane : speddup de $\approx 1.15$.
- $\rightsquigarrow$ Notre estimation naïve du speedup ($\approx 1.2$) était peut-être un peu 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.