HEIG-VD - Simulation et Optimisation

Travail Pratique 2 : Méthode de Monte-Carlo

Luc Wachter

28 mai 2019

1 Introduction

De quoi s'agit-il ? C'est pourtant clair.
Si vous ne le savez pas, c'est que vous n'êtes pas prêts.

2 Approche utilisée

Je fais ça.

3 Calcul de N

Après avoir lancé la simulation pour exécuter N_{init} expériences, il nous faut estimer le nombre d'expériences supplémentaires nécessaires pour arriver à un intervalle de confiance à 95% dont la demi-largeur ne dépasse pas Δ_{max}.

Ce nombre d'expériences n peut être calculé en utilisant la formule pour la largeur de l'intervalle de confiance.

Nous connaissons le calcul pour la largeur Δ_{I_c} de l'intervalle de confiance, et pouvons en déduire celui pour sa demi-largeur sans efforts :

$$\begin{aligned}
\Delta_{I_c} &= 2 \cdot z_{1-\frac{\alpha}{2}} \cdot \frac{s}{\sqrt{n}} \\
\Rightarrow \frac{\Delta_{I_c}}{2} &= z_{1-\frac{\alpha}{2}} \cdot \frac{s}{\sqrt{n}}
\end{aligned}$$

Il nous suffit alors d'isoler n pour déterminer la formule à utiliser dans notre programme.

$$\begin{aligned}
\frac{\Delta_{I_c}}{2} &= z_{1-\frac{\alpha}{2}} \cdot \frac{s}{\sqrt{n}} \\
\Rightarrow \frac{\Delta_{I_c}}{2 \cdot z_{1-\frac{\alpha}{2}}} &= \frac{s}{\sqrt{n}} \\
\Rightarrow \frac{\Delta_{I_c}}{2 \cdot s \cdot z_{1-\frac{\alpha}{2}}} &= \sqrt{n} \\
\Rightarrow n &= \frac{\Delta_{I_c}}{2 \cdot s \cdot z_{1-\frac{\alpha}{2}}}
\end{aligned}$$

4 Choix d'implémentation

4.1 Implémentation d'une expérience

Implémentation d'`Experiment`

```java
public double execute(Random rnd) {
    // Generate two points in the unit square and return the distance
    // between them
    return Point2D.distance(rnd.nextDouble(), rnd.nextDouble(),
```

```
                        rnd.nextDouble(), rnd.nextDouble());
}
```

4.2 Méthode principale de simulation

Méthode `simulateTillGivenCIHalfWidth`

4.3 Code client

La méthode main, quoi, avec les tests.

5 Résultats

Insérer graphiques super cool ici.

6 Conclusion

C'était **trivial**.