Économétrie — TD 9

Simulation de Monte Carlo

Pierre Beaucoral

```
library(knitr)
knit_hooks$set(optipng = hook_optipng)
```

Objectifs du TD

- Comprendre l'intérêt des **simulations de Monte Carlo** en économétrie.
- Savoir générer des variables aléatoires dans **EViews** (nrnd, (**rchisq?**), etc.).
- Étudier le comportement des estimateurs MCO (biais, variance, distribution) quand l'échantillon et la distribution des erreurs varient.

Rappel

Origines de la méthode de Monte Carlo

> « *The Monte Carlo method … is an invention of statistical sampling for the solution of mathematical problems for which direct methods are not feasible.* » Metropolis and Ulam (1949);.

- Le nom vient du **casino de Monte-Carlo**, en référence au hasard des jeux de dés.
- Première application systématique : **années 1940**, projet **Manhattan** (physique nucléaire).
- Objectif initial : estimer des **intégrales complexes** ou des **probabilités** impossibles à calculer analytiquement.

Metropolis & Ulam, 1949 — texte intégral (JASA)

Principe et rôle en économétrie

- Définir un **modèle théorique connu** (ex. : régression linéaire avec erreurs d'une loi choisie).
- **Simuler de très nombreux échantillons** à partir de ce processus générateur.
- **Estimer** sur chaque échantillon la même statistique ou le même estimateur que l'on souhaite étudier.

But :

- Observer la **distribution empirique** des estimateurs (biais, variance, forme).
- Évaluer la **robustesse des tests** (risque réel d'erreur de première espèce, puissance).
- Étudier l'impact de la **taille d'échantillon** ou de la **forme de la loi des erreurs**.

> Tip
>
> En pratique, la simulation de Monte Carlo est un **laboratoire virtuel** :
> elle permet de **vérifier** ou **illustrer** les propriétés théoriques quand la démonstration analytique est difficile ou quand on veut comprendre le comportement « en conditions réelles ».

Rappels — principe Monte Carlo

Idée : créer de nombreux échantillons artificiels selon un modèle connu, puis mesurer la *distribution empirique* des estimateurs.

Étapes :
1) Fixer un « vrai » modèle et des paramètres.
2) Simuler les erreurs (loi normale, ^{2}, etc.).
3) Générer la variable dépendante.
4) Estimer par **MCO** à chaque réplication.
5) Observer moyenne, variance, skewness, kurtosis des estimateurs.

Commandes EViews utiles

- Normale centrée réduite :

  ```
  series e = nrnd
  ```

- Khi-deux à v ddl :

  ```
  series e = @rchisq(v)
  ```

- Série simulée (exemple TD) :

  ```
  series y1 = 7 + 0.4*lsize + 0.8*bed + 0.2*bath + 0.2*airco + e
  ls y1 c lsize bed bath airco
  ```

- Lancer un programme : `run Montecarlo.prg`

Exemple interactif

Ci-dessous, on répète **R** fois une régression simple $y = \alpha + \beta x + \varepsilon$ avec $\beta = 2$ et $\varepsilon \sim N(0,1)$. Le graphique montre l'**histogramme** des $\hat{\beta}$ **cumulés** au fil des itérations (utilisez le *play/slider*).

Q1 — Générer y1 et estimer

Intitulé
En supposant $\varepsilon \sim N(0,1)$, générez une série **y1** et estimez l'équation par MCO.

Afficher la réponse

EViews:

```
series e = nrnd
series y1 = 7 + 0.4*lsize + 0.8*bed + 0.2*bath + 0.2*airco + e
ls y1 c lsize bed bath airco
```

Interprétez les coefficients et comparez-les aux valeurs vraies (0.4, 0.8, 0.2, 0.2).

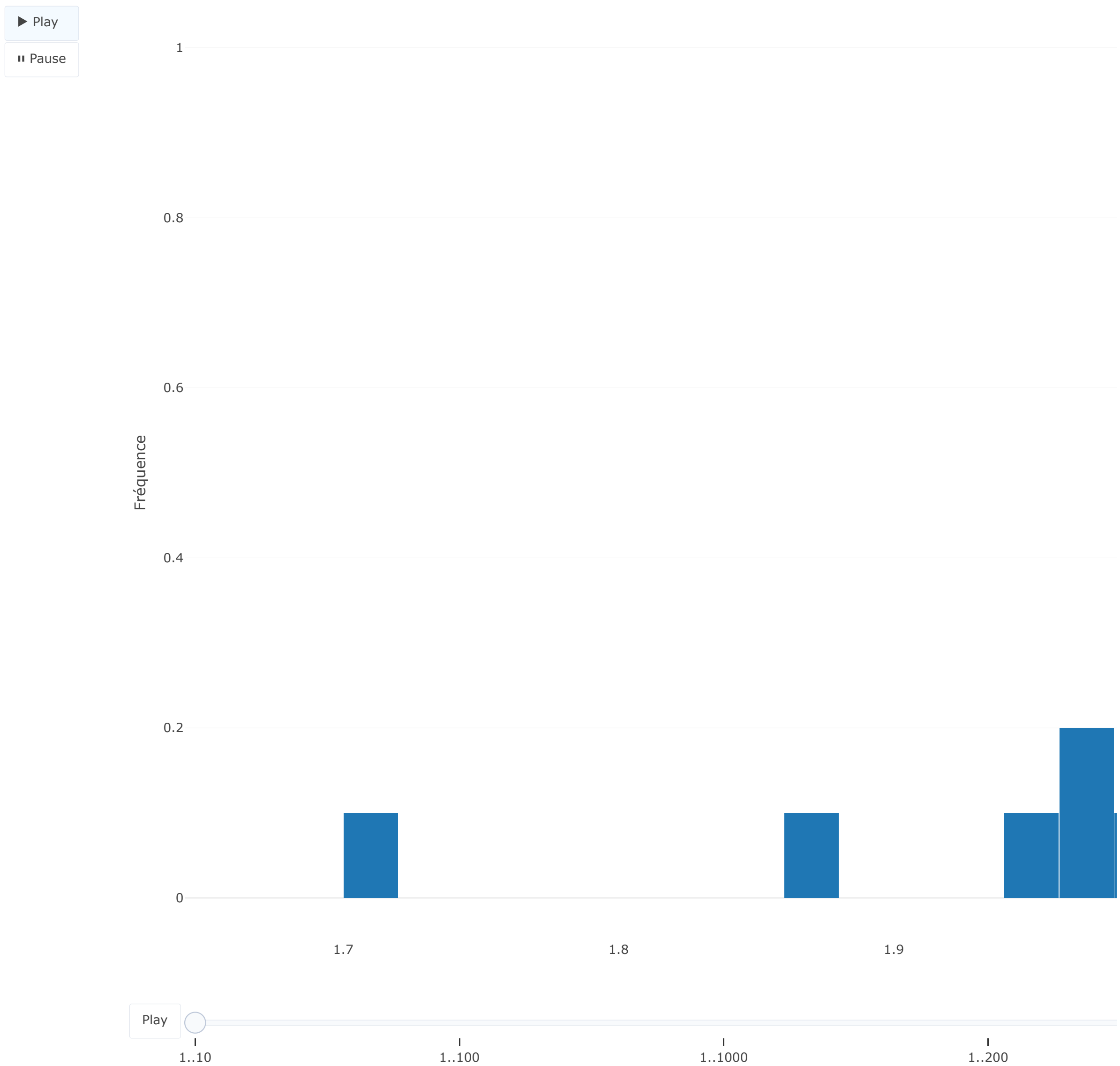

Figure 1: Distribution des estimateurs de (vraie valeur = 2). Les barres se stabilisent avec plus d'itérations.

Q2 — Répéter y2…y5

Intitulé
Refaire la Q1 pour **y2** à **y5**, même modèle, nouvelles erreurs.

Afficher la réponse

Même démarche en changeant le nom de la série :

```
series e = nrnd
series y2 = 7 + 0.4*lsize + 0.8*bed + 0.2*bath + 0.2*airco + e
...
```

Estimez chaque équation, relevez \(\hat{\beta}\) et comparez-les.

Q3 — Programme Monte Carlo

Intitulé
Ouvrez `Montecarlo.prg`, exécutez-le. Les coefficients sont enregistrés > dans la matrice **`resultat`**. Faites un histogramme et calculez > moyenne, variance, skewness, kurtosis.

Afficher la réponse

Lancer:

```
run Montecarlo.prg
```

Ensuite : `View → Descriptive Statistics → Histogram & Stats` sur chaque colonne de **`resultat`**
ou via commandes (selon script fourni). Attendez-vous à une moyenne proche du vrai paramètre, une variance qui diminue avec n, skewness 0 et kurtosis 3 si erreurs normales.

Q4 — 1000 itérations

Intitulé
Appliquer la même procédure avec **1000** itérations.

Afficher la réponse

Dans le programme, définir :

```
!nbiter = 1000
```

Puis `run Montecarlo.prg`.
La loi des grands nombres fait converger la moyenne des \(\hat{\beta}\) vers la vraie valeur, et stabilise la variance estimée.

Q5 — Variance d'erreur différente

Intitulé
Refaire 1–4 en supposant $\varepsilon \sim N(0, 0.625)$, 1000 puis 5000 simulations.

Afficher la réponse

Générer:

```
series e = sqrt(0.625)*nrnd
```

Moins de variance d'erreur $\Rightarrow$ estimateurs plus précis (variance plus faible).
Augmenter le nombre de simulations (5000) rend l'évaluation des moments plus précise.

Q6 — Erreurs non normales

Intitulé
Refaire 1–4 avec $\varepsilon \sim \chi^2(7)$, 1000 puis 5000 simulations.

Afficher la réponse

Générer :

```
series e = @rchisq(7)
```

Distribution asymétrique : les MCO restent (asymptotiquement) sans biais mais l'inférence t/F peut être mal calibrée (voir normalité des résidus, préférer erreurs-types robustes).

À retenir

- Monte Carlo permet d'observer **biais** et **variance** des estimateurs, ainsi que la convergence quand **n** et/ou le nombre d'itérations augmentent.
- La **forme** de la distribution des erreurs influence surtout l'**inférence** (tests t/F).
- Pensez à documenter vos **semences aléatoires** et **tailles d'échantillon**.

Metropolis, Nicholas, and S. Ulam. 1949. "The Monte Carlo Method." *Journal of the American Statistical Association* 44 (247): 335–41. https://doi.org/10.1080/01621459.1949.10483310.