

Économétrie — TD 9

Simulation de Monte Carlo

Pierre Beaucoral

Introduction générale

1.1 Origines de la méthode

« The Monte Carlo method ... is an invention of statistical sampling for the solution of mathematical problems for which direct methods are not feasible. » @metropolis1949.

- Le nom **Monte Carlo** fait référence au **casino de Monaco**, symbole du **hasard**.
- Mise en œuvre pour la première fois dans les **années 1940**, durant le **projet Manhattan** en physique nucléaire.
- Objectif initial : approximer des **intégrales complexes** ou des **probabilités** impossibles à calculer analytiquement.

[Article original Metropolis & Ulam \(1949\)](#)

1.2 Idée en économétrie

Une simulation de Monte Carlo est une **expérience de pensée codée sur ordinateur** :

1. **Spécifier un modèle théorique connu** (par exemple une régression linéaire).
2. **Fixer les « vrais » paramètres** et la distribution de l'erreur (normale, χ^2 , etc.).
3. **Générer artificiellement de nombreux échantillons** de données.
4. **Estimer** sur chacun de ces échantillons le même modèle que l'on veut étudier.
5. **Observer empiriquement** la distribution des estimateurs : biais, variance, forme, robustesse des tests.

💡 Tip

La simulation de Monte Carlo agit comme un **laboratoire virtuel** : elle permet de vérifier et d'illustrer les propriétés théoriques des estimateurs quand la démonstration analytique est complexe ou quand on veut observer leur comportement "en pratique".

2. Mise en œuvre pratique

2.1 Étapes générales d'une simulation

- Fixer la taille d'échantillon n et le nombre de répliques R .
- Définir le modèle :

$$y_i = \alpha + \beta x_i + \varepsilon_i$$

avec par exemple $\varepsilon_i \sim \mathcal{N}(0, 1)$.

- Pour chaque réplication $r = 1, \dots, R$:
 1. Générer les x_i et les erreurs ε_i .
 2. Calculer y_i .
 3. Estimer $(\hat{\alpha}_r, \hat{\beta}_r)$ par MCO.
- Analyser la distribution empirique des $\hat{\beta}_r$: moyenne, variance, skewness, kurtosis, comparaison à la valeur vraie β .

2.2 Commandes EViews utiles

| Action | Commande |
|-------------------------------------|---|
| Générer une normale centrée réduite | <code>series e = nrnd</code> |
| Générer une loi $\chi^2(v)$ | <code>series e = @rchisq(v)</code> |
| Créer une variable simulée | <code>series y1 = 7 + 0.4*lsize + 0.8*bed + 0.2*bath + 0.2*airco + e</code> |
| Estimer par MCO | <code>ls y1 c lsize bed bath airco</code> |
| Lancer un programme | <code>run Montecarlo.prg</code> |

3. Exemple R : animation de la convergence

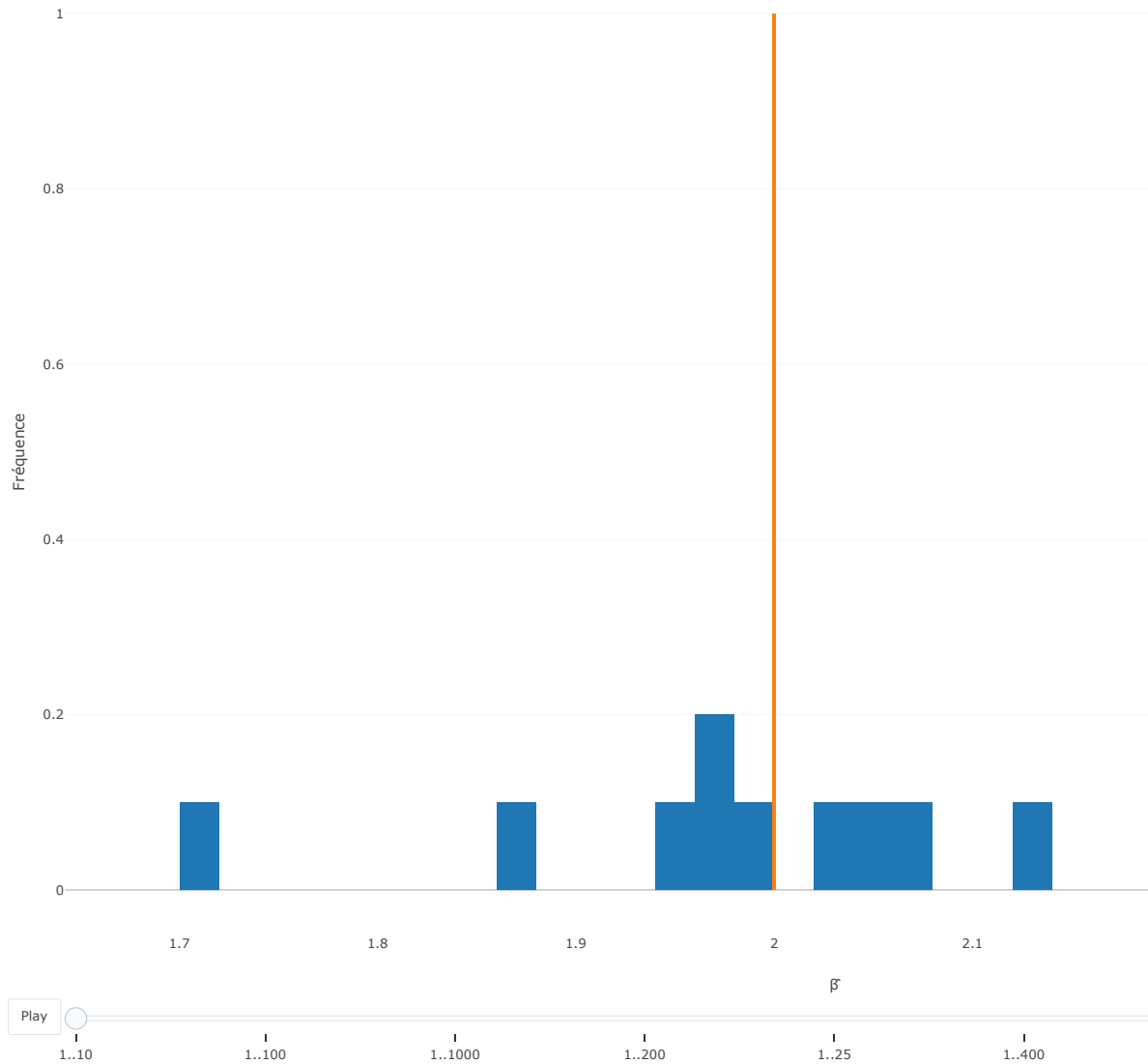


Figure 1: Histogramme cumulé des estimateurs de β (vraie valeur = 2).

Ce graphique illustre la **loi des grands nombres** : à mesure que le nombre d'itérations augmente, l'histogramme des $\hat{\beta}$ se resserre autour de la vraie valeur $\beta = 2$.

Questions du TD et réponses détaillées

Q1 — Générer y_1 et estimer

En supposant $\varepsilon \sim \mathcal{N}(0,1)$, générez une série y_1 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
```

Les coefficients estimés doivent être proches des valeurs vraies (0.4, 0.8, 0.2, 0.2) avec de légères fluctuations aléatoires.

Q2 — Répéter y_2 ... y_5

Refaire la Q1 pour y_2 à y_5 , même modèle, nouvelles erreurs.

Afficher la réponse

Même démarche en changeant le nom de la série (y_2 , y_3 ...). Comparer les $\hat{\beta}$ obtenus : on doit observer une dispersion autour des valeurs vraies, illustrant la **variabilité d'échantillonnage**.

Q3 — Programme Monte Carlo

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

```
run Montecarlo.prg
```

Puis `View → Descriptive Statistics → Histogram & Stats` sur chaque colonne de `resultat`.

Les estimateurs doivent avoir :

- une moyenne proche du vrai paramètre (absence de biais),
 - une variance reflétant la précision,
 - skewness = 0 et kurtosis = 3 si les erreurs sont normales.
-

Q4 — 1000 itérations

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

Afficher la réponse

Dans `Montecarlo.prg`, modifier :

```
!nbiter = 1000
```

et exécuter.

La moyenne des $\hat{\beta}$ se rapproche encore plus de la valeur vraie,

et la variance estimée se stabilise — illustration de la **loi des grands nombres**.

Q5 — Variance d'erreur différente

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

Afficher la réponse

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

Une variance d'erreur plus faible → des estimateurs plus précis (variance plus faible).

Avec 5000 simulations, l'évaluation des moments (moyenne, variance, skewness, kurtosis) devient plus stable.

Q6 — Erreurs non normales

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

Afficher la réponse

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

La distribution est asymétrique ($\text{skewness} > 0$). Les MCO restent **asymptotiquement sans biais**, mais les tests t/F, fondés sur la normalité, peuvent avoir un **risque de première espèce mal calibré**.

Utiliser des **erreurs-types robustes** (White, HAC) ou des tests non paramétriques pour l'inférence.

5. Points clés à retenir

- La **simulation de Monte Carlo** est un outil de validation empirique : elle confirme les propriétés théoriques des estimateurs et permet de tester la robustesse des procédures statistiques.
- Les estimateurs MCO restent sans biais même si les erreurs ne sont pas normales, mais l'inférence (tests t/F) peut devenir peu fiable en petit échantillon.
- En cas de non-normalité, recourir à des **erreurs-types robustes** ou à des méthodes d'inférence alternatives (bootstrap, tests non paramétriques).