La génération de nombres pseudo-aléatoires

6-601-09 Simulation Monte Carlo

Geneviève Gauthier

HEC Montréal

Mise en garde

Ce texte ne présentera pas les derniers développements concernant les générateurs de nombres pseudo-aléatoires. C'est un survol historique permettant à l'étudiant de comprendre les concepts permettant la création de ces nombres afin d'en faire la meilleure utilisation possible. En effet, les logiciels courants ont leurs générateurs et la plupart sont de bons générateurs.

Cependant, les techniques de transformation nous seront utiles lors de l'utilisation des suites à discrépences faibles.

Ce texte est fortement inspiré de Niederreiter H (1992) *Random Number Generation and Quasi-Monte Carlo Methods*, Siam, 237 pages.

Les générateurs de nombres pseudo-aléatoires classiques

Propriétés souhaitables d'une suite de nombres pseudo-aléatoires

Les nombres pseudo-aléatoires doivent pouvoir être générés à partir d'algorithmes déterministes nécessistant peu de paramètres de sorte que

- leur génération ne dépend pas de sources externes (ce qui est souvent le cas des générateurs de nombres aléatoires),
- ils sont reproductibles,
- ils n'occasionnent pas de problème d'entreposage,
- ils sont relativement rapides à calculer.

Les générateurs de nombres pseudo-aléatoires

Propriétés souhaitables d'une suite de nombres pesudo-aléatoires

- La suite de nombres pseudo-aléatoires doit aussi satisfaire certains tests statistiques.
- Évidemment, comme elle n'est pas aléatoire, elle les échouera tous pourvu que la taille de l'échantillon soit suffisamment grande. On se contentera donc qu'un échantillon de taille "raisonnable" par rapport à l'utilisation que l'on veut en faire satisfasse ces tests.
- Les algorithmes standards servant à engendrer les nombres pseudo-aléatoires sont généralement de type récursif et produisent des suites qui sont ultimement périodiques. Une bonne suite devrait avoir un long cycle.

Les tests statistiques

L'hypothèse nulle est que les observations sont indépendantes et identiquement distribuées, leur loi étant uniforme sur l'intervalle $[0, 1]$.

Les tests statistiques I

Les tests d'uniformité

- L'idée est de vérifier si la fonction de répartition échantillonnale F_N engendrée par les N premiers termes de notre suite de nombres pseudo-aléatoires,

$$F_N(x) = \frac{1}{N} \sum_{i=0}^{N-1} \mathbf{1}_{x_i \leq x},$$

est significativement différente de la fonction de répartition d'une distribution uniforme sur l'intervalle $[0, 1]$.

Les tests statistiques II

Les tests d'uniformité

- Ce sont des tests qualifiés de "goodness-of-fit". Il en existe plusieurs, déjà implémentés dans la plupart des logiciels de statistique. On peut penser, entre autres, au "two-sided Kolmogorov test" qui est basé sur la $*-$discrépance de la suite. Il est décrit brièvement dans Niederreiter, p.166. Il y a aussi le test de Kolmogorov-Smirnov, un classique en statistique.

Les tests statistiques

Test d'ajustement à une loi

- On peut aussi utiliser le test du χ^2 en dénombrant le nombre d'observations contenues dans chaque sous-intervalle d'une partition $\{J_1, ..., J_k\}$ de l'intervalle $[0, 1]$. Sous l'hypothèse nulle, la statistique

$$\sum_{j=1}^{k} \left(\frac{\sum_{i=0}^{N-1} \mathbf{1}_{x_i \in J_j} - N\lambda(J_j)}{N\lambda(J_j)} \right)^2$$

où la longueur $\lambda(J_j)$ de l'intervalle J_j devrait être petite.

- Notons que $\sum_{i=0}^{N-1} \mathbf{1}_{x_i \in J_j}$ = le nombre d'observations de notre suite qui sont dans le sous-intervalle J_j, et $N\lambda(J_j)$ = le nombre d'observations que l'on devrait théoriquement trouver dans J_j si l'hypothèse nulle est vérifiée. Cependant, ce test est peu puissant.

Les tests statistiques

Les tests de corrélation

- Les observations sont supposées être indépendantes. On peut donc vérifier si les autocovariances de différents délais sont significativement différentes de zéro.
- L'autocovariance échantillonnale de délai k est

$$\frac{1}{N-k} \sum_{i=0}^{N-k-1} (x_i - \overline{x})(x_{i+k} - \overline{x})$$

où $\overline{x} = N^{-1} \sum_{i=0}^{N-1} x_i$.

- Le problème avec ce type de test est que si les observations sont indépendantes, les autocovariances devraient être nulles mais si les autocovariances sont nulles, il est possible que les observations ne soient pas indépendantes. Ce n'est donc pas un test puissant.

Les tests statistiques

Les tests d'indépendance

- Il existe des tests plus puissants pour vérifier l'indépendance entre les observations. Cependant, ils exigent souvent que l'on spécifie le type de dépendance sous la contre-hypothèse.
- En pratique, il est possible d'utiliser le test classique d'indépendance du khi-deux (qui n'est pas très puissant). Niederreiter en propose un autre à la page 167.

Les tests statistiques

Les tests basés sur les trous

- Soit J un sous-intervalle de $[0,1]$. Si, pour un certain n, nous avons

$$x_{n+j} \notin J, j \in \{0,1,...,k-1\} \text{ et } x_{n+k} \in J,$$

alors nous disons qu'il y a un trou de longueur k.

- Sous l'hypothèse nulle, la longueur d'un trou est de loi géométrique de paramètre $\lambda(J)$.
- En pratique, on choisit un entier positif h et on dénombre le nombre de trous de longueurs $0,1,2,...,h-1$ et $\geq h$ jusqu'à ce qu'un grand nombre de trous soit obtenu. Un test du χ^2 est ensuite appliqué. (Niederieter, p.166)

Les tests statistiques

Les tests basés sur les «uns»

Un segment de la suite satisfaisant

$$x_{n-1} \geq x_n < x_{n+1} < \ldots < x_{n+k-1} \geq x_{n+k}$$

est appelé *segment à la hausse de longueur* k. Il existe un test basé sur le nombre de tels segments dans notre échantillon. (Niederieter, p.166-167)

Générateurs

«Linear congruential method»

Définition. Soit M un grand nombre entier positif et $a, c \in \{1, 2, ..., M-1\}$ tels que M et a sont relativement premiers (le plus grand commun diviseur (PGCD) de M et de a est 1). Nous construisons récursivement une suite $\{y_n : n \in \{0, 1, 2, ...\}\}$ en choisissant la valeur initiale $y_0 \in \{1, 2, ..., M-1\}$ et en générant les autres éléments de la suite selon la récursion

$$y_{n+1} = ay_n + c \mod M.$$

À partir de cette suite appelée le *générateur*, nous obtenons la suite $\{x_n : n \in \{0, 1, 2, ...\}\}$ de nombres pseudo-aléatoires

$$x_n = \frac{y_n}{M},\ n \in \{0, 1, 2, ...\}.$$

Générateurs

«Linear congruential method»

Exemple. Cet exemple illustrant la méthode emploie de petits nombres afin de faciliter les calculs.

$$y_{n+1} = ay_n + c \mod M \text{ et } x_n = \frac{y_n}{M}.$$

Posons $a = 3$, $c = 0$, $M = 11$ et $y_0 = 5$

$$\begin{aligned}
& & x_0 &= \tfrac{5}{11} \\
y_1 &= (3 \times 5) \bmod 11 = 4 & x_1 &= \tfrac{4}{11} \\
y_2 &= (3 \times 4) \bmod 11 = 1 & x_2 &= \tfrac{1}{11} \\
y_3 &= (3 \times 1) \bmod 11 = 3 & x_3 &= \tfrac{3}{11} \\
y_4 &= (3 \times 3) \bmod 11 = 9 & x_4 &= \tfrac{9}{11} \\
y_5 &= (3 \times 9) \bmod 11 = 5 & x_5 &= \tfrac{5}{11} \\
y_6 &= (3 \times 5) \bmod 11 = 4 & x_6 &= \tfrac{4}{11} \\
y_7 &= (3 \times 4) \bmod 11 = 1 & x_7 &= \tfrac{1}{11} \\
& \ldots & & \ldots
\end{aligned}$$

$$\bullet \quad \tfrac{1}{11} \quad \bullet \quad \tfrac{3}{11} \quad \tfrac{4}{11} \quad \tfrac{5}{11} \quad \bullet \quad \bullet \quad \bullet \quad \tfrac{9}{11} \quad \bullet \quad \bullet$$

Générateurs

«Linear congruential method»

Exemple (suite). Posons $a = 7$, $c = 0$, $M = 11$ et $y_0 = 5$

$$
\begin{array}{ll}
 & x_0 = \frac{5}{11} \\
y_1 = (7 \times 5) \bmod 11 = 2 & x_1 = \frac{2}{11} \\
y_2 = (7 \times 2) \bmod 11 = 3 & x_2 = \frac{3}{11} \\
y_3 = (7 \times 3) \bmod 11 = 10 & x_3 = \frac{10}{11} \\
y_4 = (7 \times 10) \bmod 11 = 4 & x_4 = \frac{4}{11} \\
y_5 = (7 \times 4) \bmod 11 = 6 & x_5 = \frac{6}{11} \\
y_6 = (7 \times 6) \bmod 11 = 9 & x_6 = \frac{9}{11} \\
y_7 = (7 \times 9) \bmod 11 = 8 & x_7 = \frac{8}{11} \\
y_8 = (7 \times 8) \bmod 11 = 1 & x_8 = \frac{1}{11} \\
y_9 = (7 \times 1) \bmod 11 = 7 & x_9 = \frac{7}{11} \\
y_{10} = (7 \times 7) \bmod 11 = 5 & x_{10} = \frac{5}{11} \\
y_{11} = (7 \times 5) \bmod 11 = 2 & x_{11} = \frac{2}{11} \\
\ldots & \ldots
\end{array}
$$

Générateurs

«Linear congruential method»

- **Remarque**. Pour tout n, $0 \leq x_n < 1$.
- **Remarque**. La suite $\{x_n : n \in \{0, 1, 2, ...\}\}$ ainsi générée est périodique et la période est nécessairement plus petite ou égale à M. Pour cette raison, il faut choisir M grand.

Générateurs

«Linear congruential method»

Théorème. La période de la suite $\{x_n : n \in \{0, 1, 2, ...\}\}$ est M si et seulement si

1. M et c sont relativement premiers (PGCD$(M, c) = 1$),
2. $a \equiv 1 \bmod p$ (c'est-à-dire qu'il existe un nombre entier k tel que $a = kp + 1$) pour tout nombre premier p qui divise M,
3. $a \equiv 1 \bmod 4$ si 4 est un diviseur de M.

(Niederreiter, p.169)

Générateurs

«Linear congruential method»

Il y a trois cas standards qui sont généralement considérés lors de l'implémentation de tels générateurs.

- **Premier cas**. M est un nombre premier, a est une racine primitive modulo M (voir définition page suivante), $c = 0$ et $y_0 \neq 0$.

$$y_{n+1} = a y_n \mod M \text{ et } x_n = \frac{y_n}{M},\ n \in \{0, 1, 2, ...\}.$$

La suite de nombres pseudo-aléatoires $\{x_n : n \in \{0, 1, 2, ...\}\}$ a une période de M. (Neidereiter, p. 169)

Générateurs I

«Linear congruential method»

- **Définition**. a est une racine primitive modulo M si

$$\left\{a^k \bmod M : k \in \{1, 2, ..., M-1\}\right\} = \{1, 2, ..., M-1\}.$$

- **Exemple**. $a = 3$ est une racine primitive modulo $M = 5$ puisque

$$\begin{aligned} & \{3 \bmod 5, 3^2 \bmod 5, 3^3 \bmod 5, 3^4 \bmod 5\} \\ = \; & \{3 \bmod 5, 9 \bmod 5, 27 \bmod 5, 81 \bmod 5\} \\ = \; & \{3, 4, 2, 1\} \\ = \; & \{1, 2, 3, 4\}. \end{aligned}$$

Générateurs II

«Linear congruential method»

- **Exemple**. $a = 4$ n'est pas une racine primitive modulo $M = 5$ puisque

$$\begin{aligned} & \left\{4 \bmod 5, 4^2 \bmod 5, 4^3 \bmod 5, 4^4 \bmod 5\right\} \\ = \; & \left\{4 \bmod 5, 16 \bmod 5, 64 \bmod 5, 256 \bmod 5\right\} \\ = \; & \{4, 1, 4, 1\} = \{1, 4\} \neq \{1, 2, 3, 4\}. \end{aligned}$$

- Il n'y a malheureusement pas de recette magique pour trouver une racine primitive modulo un nombre. Cependant, Mathematica vous en trouvera avec la fonction *PrimitiveRoot*.

Générateurs

«Linear congruential method»

- **Deuxième cas**. M est une puissance de 2, $a \equiv 5 \bmod 8$ (c'est-à-dire qu'il existe un entier k tel que $a = 8k + 5$), c est un nombre impair.

$$y_{n+1} = a y_n + c \mod M \text{ et } x_n = \frac{y_n}{M},\ n \in \{0, 1, 2, ...\}.$$

La suite de nombres pseudo-aléatoires $\{x_n : n \in \{0, 1, 2, ...\}\}$ a une période de M. (Neiderreiter, p. 169)

Générateurs

«Linear congruential method»

- **Troisième cas**. M est une puissance de 2, $a \equiv 5 \bmod 8$ (c'est-à-dire qu'il existe un entier k tel que $a = 8k + 5$), $c = 0$ et y_0 est un nombre impair.

$$y_{n+1} = a y_n + c \mod M \text{ et } x_n = \frac{y_n}{M},\ n \in \{0, 1, 2, ...\}.$$

La suite de nombres pseudo-aléatoires $\{x_n : n \in \{0, 1, 2, ...\}\}$ a une période de $M/4$. (Neiderreiter, p. 169)

Générateurs

«Linear congruential method»

Faiblesses de ce type de générateur :

- Le module M et, par conséquent, la période du générateur sont bornés à cause de la "longueur des mots" de la machine utilisée. Par exemple, avec un processeur de 32 bits, nous avons

$$\text{période} \leq M \leq 2^{32}$$

  à moins de vouloir utiliser la double précision qui augmente les temps de calcul.
- Il y a trop de régularité dans la suite à cause de la simplicité de l'algorithme.

(Neiderreiter, pages 173-174)

«Multiple-recursive linear congruential method»

Définition. Les paramètres du générateur sont

- $M =$un grand nombre premier,
- $k \in \{2, 3, 4, ...\}$ représente l'ordre de la récursion,
- $a_0, ..., a_{k-1} \in \{0, 1, 2, ..., M-1\}$ avec $a_0 \neq 0$,
- $y_0, ..., y_{k-1} \in \{0, 1, 2, ..., M-1\}$ sont les valeurs initiales telles qu'il existe au moins une valeur différente de zéro.

«Multiple-recursive linear congruential method»

- La suite $\{y_n : n \in \{0, 1, 2, ...\}\}$ où

$$y_{n+k} = \sum_{i=0}^{k-1} a_i y_{n+i} \bmod M,\ n \in \{0, 1, 2, ...\}$$

est telle que $y_n \in \{0, 1, 2, ..., M-1\}$ et a une période inférieure ou égale à $M^k - 1$ (puisque le point $(y_n, y_{n+1}, ..., y_{n+k}) = \mathbf{0}$ ne peut être atteint).

- La suite $\{x_n : n \in \{0, 1, 2, ...\}\}$ où

$$x_n = \frac{y_n}{M},\ n \in \{0, 1, 2, ...\}$$

constitue notre générateur.

«Multiple-recursive linear congruential method»

- Il a la même période que la suite $\{y_n : n \in \{0, 1, 2, ...\}\}$ et tous ses éléments sont dans l'intervalle $[0, 1]$.
- Il existe des critères permettant de déterminer les paramètres du générateur de sorte que sa période soit la plus grande possible, c'est-à-dire $M^k - 1$. Cela fait appel à des notions d'algèbre, en particulier au polynôme caractéristique de la récursion permettant d'engendrer la suite $\{y_n : n \in \{0, 1, 2, ...\}\}$. (Voir Niederreiter, p.174 pour plus de détail).
- On trouve au chapitre 7 de Neiderreiter (pages 174-175) des résultats conscernant les propriétés d'uniformité de ce type de générateurs.

«Nonlinear congruential method» I

«First order congruential method»

Définition. Soit M un grand nombre entier positif. Le générateur $\{y_n : n \in \{0, 1, 2, ...\}\}$ est construit récursivement en choisissant la valeur initiale $y_0 \in \{1, 2, ..., M-1\}$ et en générant les autres éléments de la suite selon la récursion

$$y_{n+1} = f(y_n) \mod M$$

où la fonction $f : \{0, 1, 2, ..., M-1\} \to \mathbb{N}$ est à valeurs entières. À partir de ce générateur, nous obtenons la suite $\{x_n : n \in \{0, 1, 2, ...\}\}$ de nombres pseudo-aléatoires

$$x_n = \frac{y_n}{M},\ n \in \{0, 1, 2, ...\}.$$

(Neiderreiter, ch. 8)

«Nonlinear congruential method»

«First order congruential method»

- La période de ce générateur est inférieure ou égale à M.
- Considérons donc la suite périodique $\{y_n : n \in \{0, 1, 2, ...\}\}$ d'éléments de l'ensemble $\{0, 1, ..., M-1\}$. L'application qui associe à chaque entier $n \in \{0, 1, ..., M-1\}$ un nombre $y_n \in \{0, 1, ..., M-1\}$ peut être représentée par un polynôme g de degré $d < M$. En d'autres termes, il existe un polynôme g de degré d tel que

$$y_n = g(n) \text{ pour tout } n \in \{0, 1, 2, ...\}.$$

  Le degré de g est important dans cette théorie.
- Si l'ensemble $\{y_0, y_1, ..., y_{M-1}\} = \{0, 1, ..., M-1\}$ (c'est-à-dire que la période de $\{y_n : n \in \{0, 1, 2, ...\}\}$ est M), alors g est appelé un *polynôme de permutations*. Nous nous restreignons à ce cas. (Neiderreiter, ch. 8, p. 178)

«Nonlinear congruential method» I

«First order congruential method»

Définition. Pour tout entier $s \geq 1$, un générateur $\{y_n : n \in \{0, 1, 2, ...\}\}$ satisfait le *test du treillis de dimension s* si les vecteurs

$$\left\{ \overrightarrow{\mathbf{y}}_n - \overrightarrow{\mathbf{y}}_0 : n \in \{0, 1, 2, ...\} \right\}$$

engendrent le treillis $\{0, 1, ..., M-1\}^{\times s}$ où

$$\overrightarrow{\mathbf{y}}_n = (y_n, y_{n+1}, ..., y_{n+s-1}) .$$

- Les générateurs congruentiels linéaires ne peuvent pas réussir ce test lorsque $s > 1$. C'est une de leurs faiblesses.

«Nonlinear congruential method» II

«First order congruential method»

- Les générateurs congruentiels non linéaires réussissent le test si et seulement si $s \in \{1, 2, ..., d\}$. Attention, il existe un exemple simple d'un tel type de générateur qui satisfait ce test mais qui a de très mauvaises propriétés d'uniformité. Ce test ne devrait donc être utilisé que pour éliminer de mauvais générateurs. (Neiderreiter, ch. 8, p. 178-180)

«Nonlinear congruential method»

«First order congruential method»

Quelques cas particuliers

- «*Quadratic congruential method*» proposée par Knuth en 1981.
- **Définition**. Soit $M = 2^\alpha$ avec $\alpha \geq 2$, $y_0 \in \{1, 2, ..., M-1\}$ et

$$y_{n+1} = ay_n^2 + by_n + c \mod M \text{ et } x_n = \frac{y_n}{M},\ n \in \{0, 1, 2, ...\}.$$

où $a, b, c \in \{0, 1, 2, ..., M-1\}$. (Neiderreiter, ch. 8, p.181)

- **Théorème**. La période du générateur $\{y_n : n \in \{0, 1, ...\}\}$ et de la suite de nombres pseudo-aléatoires est M si et seulement si a est pair, $b \equiv a + 1 \mod 4$ (c'est-à-dire qu'il existe un entier k tel que $b = 4k + a + 1$ et c est impair. (Neiderreiter, ch. 8, p. 180-181)

«Nonlinear congruential method»

«First order congruential method»

Quelques cas particuliers

- «*Inversive congruential method*» proposée par Eichenauer et Lehn en 1986.
- **Définition**. Soit un nombre premier M et $c \in \{0, 1, 2, ..., M-1\}$. L'entier $\overline{c} \in \{0, 1, 2, ..., M-1\}$ est tel que $\overline{c} = 0$ si $c = 0$ et, si $c > 0$,

$$c\overline{c} \equiv 1 \bmod M \ .$$

c'est-à-dire qu'il existe un entier k tel que $kM + 1 = c\overline{c}$.

- **Définition**. Soit $M \geq 5$ est un nombre premier, $y_0 \in \{1, 2, ..., M-1\}$ et

$$y_{n+1} = a\overline{y}_n + b \bmod M \text{ et } x_n = \frac{y_n}{M},\ n \in \{0, 1, 2, ...\}\ .$$

où $a, b \in \{0, 1, 2, ..., M-1\}$, $a \neq 0$. (Neiderreiter, ch. 8, p.182)

«Nonlinear congruential method»

«First order congruential method»

- Il existe des conditions sur a et b nous assurant que la période du générateur est M. Ces conditions sont énoncées au théorème 8.4 de Niederrieter.
- Ce type de générateurs réussit le *test du treillis de dimension s* pour tous les entiers $s \leq (M+1)/2$.
- D'autres tests mesurant l'uniformité de ce générateur sont décrits aux pages 183-189 de Niderreiter.

«The digital multistep method»

Tausworthe en 1965

Définition. Les paramètres du générateur sont

- $M =$un **petit** nombre premier (généralement $M = 2$),
- $k \in \{2, 3, 4, ...\}$ représente l'ordre de la récursion,
- $a_0, ..., a_{k-1} \in \{0, 1, 2, ..., M-1\}$ avec $a_0 \neq 0$,
- $y_0, ..., y_{k-1} \in \{0, 1, 2, ..., M-1\}$ sont les valeurs initiales telles qu'il existe au moins une valeur différente de zéro.

«The digital multistep method»

- La suite $\{y_n : n \in \{0, 1, 2, ...\}\}$ où

$$y_{n+k} = \sum_{i=0}^{k-1} a_i y_{n+i} \bmod M,\ n \in \{0, 1, 2, ...\}$$

  est telle que $y_n \in \{0, 1, 2, ..., M-1\}$ et a une période plus petite ou égale à $M^k - 1$ (puisque le point $(y_n, y_{n+1}, ..., y_{n+k}) = \mathbf{0}$ ne peut être atteint).
- C'est le générateur de la méthode "Multiple-recursive linear congruential".

«The digital multistep method»

- La suite de nombres pseudo-aléatoires se construit

$$x_n = \sum_{j=1}^{m} y_{nm+j-1} M^{-j},\ n \in \{0, 1, 2, ...\}$$

où $m \in \{2, ..., k\}$.

En d'autres mots, x_n s'obtient en subdivisant le générateur en blocs de longueur m puis en utilisant les nombres de ce bloc comme le développement digital en base M d'un nombre compris entre 0 et 1.
La période de cette suite de nombres pseudo-aléatoires est

$$\frac{M^k - 1}{PGCD\,(m, M^k - 1)}.$$

(Neiderreiter, ch. 9, p.191-192)

«The generalized feedback shift register (GFSR) method»

Lewis et Payne en 1973

Encore une fois, le générateur "Multiple-recursive linear congruential" est utilisé :

Rappel. Les paramètres du générateur sont

- $M =$un petit nombre premier,
- $k \in \{2, 3, 4, ...\}$ représente l'ordre de la récursion,
- $a_0, ..., a_{k-1} \in \{0, 1, 2, ..., M-1\}$ avec $a_0 \neq 0$,
- $y_0, ..., y_{k-1} \in \{0, 1, 2, ..., M-1\}$ sont les valeurs initiales telles qu'il existe au moins une valeur différente de zéro.

$$y_{n+k} = \sum_{i=0}^{k-1} a_i y_{n+i} \bmod M,\ n \in \{0, 1, 2, ...\}.$$

«The generalized feedback shift register (GFSR) method»

Lewis et Payne en 1973

- La suite de nombres pseudo-aléatoires se construit

$$x_n = \sum_{j=1}^{m} y_{n+h_j} M^{-j},\ n \in \{0, 1, 2, ...\}$$

où $m \in \{2, 3, ...\}$.et $h_1, ..., h_m \in \{0, 1, 2, ...\}$.

- Si les paramètres sont convenablement choisis, la période de cette suite de nombres pseudo-aléatoires est $M^k - 1$. (Neiderreiter, chapitre 9, p.198-199)