专题一：蒙特卡洛仿真

黄嘉平

2026.3.15

1. 准备工作

为了给今后的学习做准备，也为了熟悉如何在 R 中安装和调用程序包（package），我们首先安装 **tidyverse**。

1. 第一步要选择合适的镜像服务器[1]（mirror server）。在 RStudio 的顶部菜单中选择 Tools > Global Options…，然后在新出现的窗口左侧找到 Packages 项，在 Primary CRAN Repository 一栏点击 Change…，并从列表中选择位于国内的任意选项，例如 China (Shenzhen) [https] - Southern University of Science and Technology (SUSTech)，点击 OK，最后点击 Apply。
2. 从顶部菜单中选择 Tools > Install Packages…，输入 tidyverse，并勾选下面的 Install dependencies，最后点击 Install。
3. 等待安装程序结束后，在 Console 中输入 **library(tidyverse)**，如果运行结果和下面展示的一致（版本号有可能发生变化），就是成功调用了 **tidyverse**。

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.2.0     ✔ readr     2.2.0
✔ forcats   1.0.1     ✔ stringr   1.6.0
✔ ggplot2   4.0.2     ✔ tibble    3.3.1
✔ lubridate 1.9.5     ✔ tidyr     1.3.2
✔ purrr     1.2.1
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package to force all conflicts to become errors
```

2. 伪随机数

蒙特卡洛仿真（Monte Carlo simulation）指利用计算机生成的伪随机数（pseudo-random number）进行数据生成的方法。首先需要明确的是，随机现象的生成机制是无法明确定义的，否则它就不是随机。而计算机能够给出的任何结果都有明确的计算过程。因此，利用计算机得出的任何数值都不是随机的。但是我们可以生成很难预测的近似随机的数值，称为伪随机数。

最早的伪随机数生成法是线性同余法（linear congruential generator），它是根据下面的递归方程

[1] R 的官方程序包发布网站称为 CRAN，在世界各地都有同步的服务器，称为镜像。选择地理位置接近的镜像可以避免因连接超时造成的安装失败。

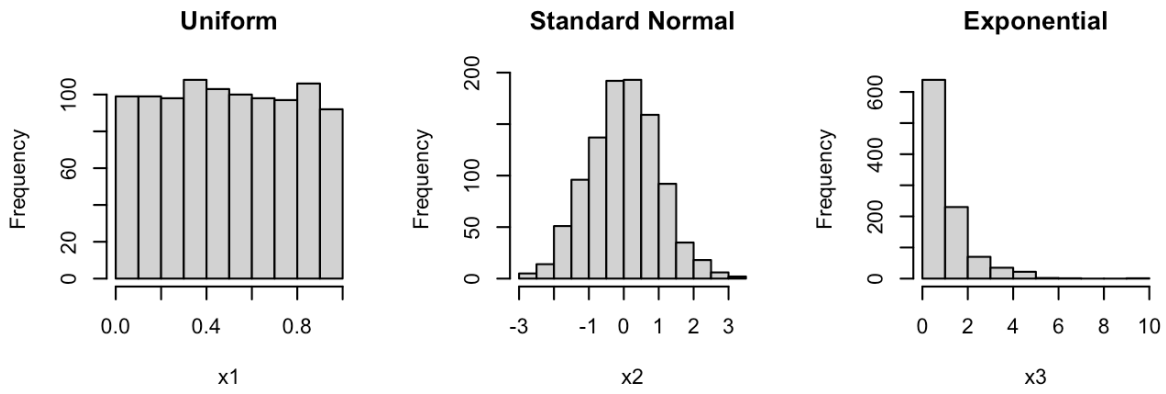
$$x_{n+1} = (ax_n + c) \mod m$$

生成数列 $\{x_n\}$，其中 $0 < a < m, 0 < c < m, 0 < x_0 < m$。当 (a, c, m) 满足一定条件时，$\{x_n\}$ 的周期为 m 且不受 x_0 取值的影响。通常需要取很大的 m 以增加预测的难度，例如 Microsoft Visual C++ 中 $(a, c, m) = \left(214013, 2531011, 2^{31}\right)$。

如今的伪随机数生成器原理更为复杂，各种现代编程语言大多采用 Mersenne-Twister 法，它生成的数列周期为 $2^{19937} - 1$，且计算速度快。如果想了解更多信息可以参考 Bilibili 上的视频 https://www.bilibili.com/video/BV1XW411s7mK/。

R 中生成伪随机数的函数是 **`r???()`**，其中 **`???`** 可以替换为具体的分布名称。例如，均匀分布对应 **`runif()`**，正态分布对应 **`rnorm()`**，t 分布对应 **`rt()`** 等。下面的程序中依次生成了服从均匀分布、正态分布和指数分布的伪随机数，并画出各自的直方图。

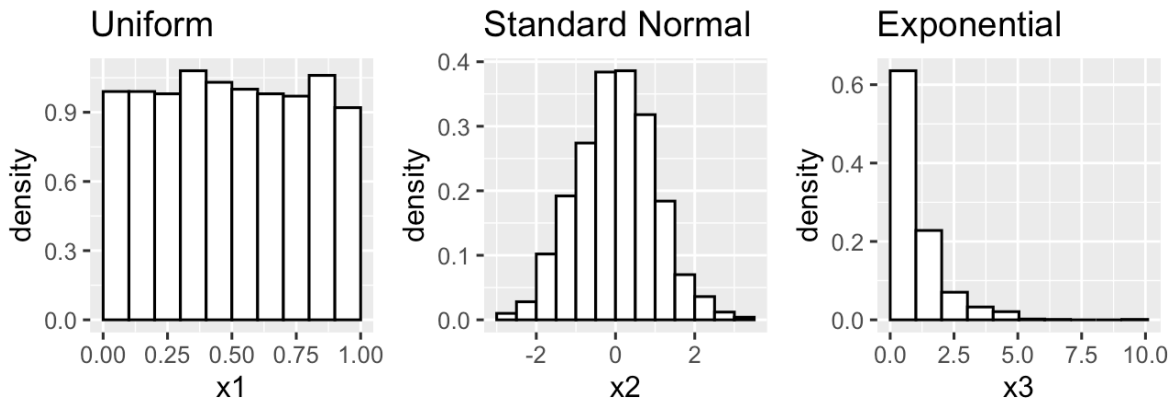
```
set.seed(123)
x1 <- runif(1000)
x2 <- rnorm(1000)
x3 <- rexp(1000, rate = 1)
par(mfrow = c(1,3))
hist(x1, main="Uniform")
hist(x2, main="Standard Normal")
hist(x3, main="Exponential")
```



第一行的 **`set.seed()`** 函数指定了生成伪随机数所需的种子值。如果不指定种子值，则每次运行的结果都不相同，虽然能够体验随机效果，但无法进行重复验证。换句话说，如果知道伪随机数生成机制以及种子值，就可以完美预测所生成的数值。种子值可以是任意整数。

以上是利用 R 标准功能（一般称为 base R）进行数据生成和绘图的例子。下面我们用 **`tidyverse`** 中的 tibble 数据结构和 **`ggplot2`** 程序包提供的绘图功能进行相同的操作。Tibble 是对 base R 中的 data.frame 数据结构的改良，是 tidyverse 中各程序包的通用数据保存形式。**`ggplot2`** 则提供了更加丰富美观的绘图功能。我们还用到了 **`patchwork`** 程序包（需安装后调用），它提供了将多个图表合并为一个的功能。

```r
library(patchwork)
set.seed(123)
rn <- tibble(x1 = runif(1000), x2 = rnorm(1000),
  x3 = rexp(1000, rate = 1))
p1 <- ggplot(rn, aes(x1, after_stat(density))) +
  geom_histogram(bins = 10, breaks = seq(0, 1, 0.1),
    color="black", fill="white") +
  labs(title = "Uniform")
p2 <- ggplot(rn, aes(x2, after_stat(density))) +
  geom_histogram(binwidth = 0.5, center = 0.25, color="black",
    fill="white") +
  labs(title = "Standard Normal")
p3 <- ggplot(rn, aes(x3, after_stat(density))) +
  geom_histogram(bins = 10, boundary = 0, color="black",
    fill="white") +
  labs(title = "Exponential")
p1 + p2 + p3
```



3. 样本均值的抽样分布

首先我们练习生成样本均值的抽样分布。假设随机变量 X 服从标准正态分布 $N(0,1)$。通过生成 n 个随机观测值，我们可以计算样本量为 n 的样本均值。令 $n=100$，生成样本并计算样本均值的代码为

```r
set.seed(12345)
n <- 100
sample <- rnorm(n, 0, 1)
sample_mean <- mean(sample)
sample_mean
```

```
[1] 0.2451972
```

从结果可知，该样本的均值为 0.2451972，和理论均值 0 相差甚远。这种差异源自随机样本的不确定性。我们知道样本均值 $\overline{X} = \sum_{i=1}^{n} X_i$ 也是随机变量，因此也存在概率分布，称为抽样分布。如果我们生成 k 个随机样本，每个样本都包含 n 个观测值，则可计算 k 个样本均值，并观察它们的分布。令 $k = 1000$，计算样本均值的代码如下。

```
set.seed(12345)
k <- 1000
sample_means <- map_dbl(1:k, \(x) mean(rnorm(n, 0, 1)))
head(sample_means) # 输出结果中的前六个值
[1]  0.24519720  0.04523311 -0.04621158  0.21527589 -0.04719075  0.06415998
```

这里我们用到了 **tidyverse** 中的 **purrr** 程序包中包含的 **map_dbl()** 函数，它是 **map()** 函数的一个变种，可以重复或迭代进行多个相同操作。**map()** 函数的基本语法是 **map(x, f)**，其中 **x** 是一个向量或列表，**f** 是一个以 **x** 的要素为输入变量的函数。**map()** 的作用是依次将 **x** 的要素代入 **f** 进行计算，最后将计算结果保存在一个和 **x** 同维度的列表里。下面的表达式可以帮助我们理解 **map()** 的作用。

$$\text{map}(x, f) = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_k) \end{pmatrix}$$

在命令 **map_dbl(1:k, \(x) mean(rnorm(n, 0, 1)))** 中，对应 **x** 的是数列 **1:k** $= (1, 2, \ldots, k)$，函数 **f** 是 **mean(rnorm(n, 0, 1))**，其中并没有出现 **x** 的要素。因此，这个命令单纯就是将 **mean(rnorm(n, 0, 1))** 重复 **k** 遍。关于 **map()** 函数的更多信息可通过 **?map** 获取。

map() 函数输出的结果是列表，而 **map_dbl()** 的输出结果则是数值向量，除此之外二者的功能相同。这里我们选择更方便操作的 **map_dbl()**。从最后的计算结果可以看出，每个数值都是一个样本均值。利用这 1000 个样本均值，我们可以计算它们的均值和方差：

```
sample_means |> mean()
[1] 0.0008166914
```

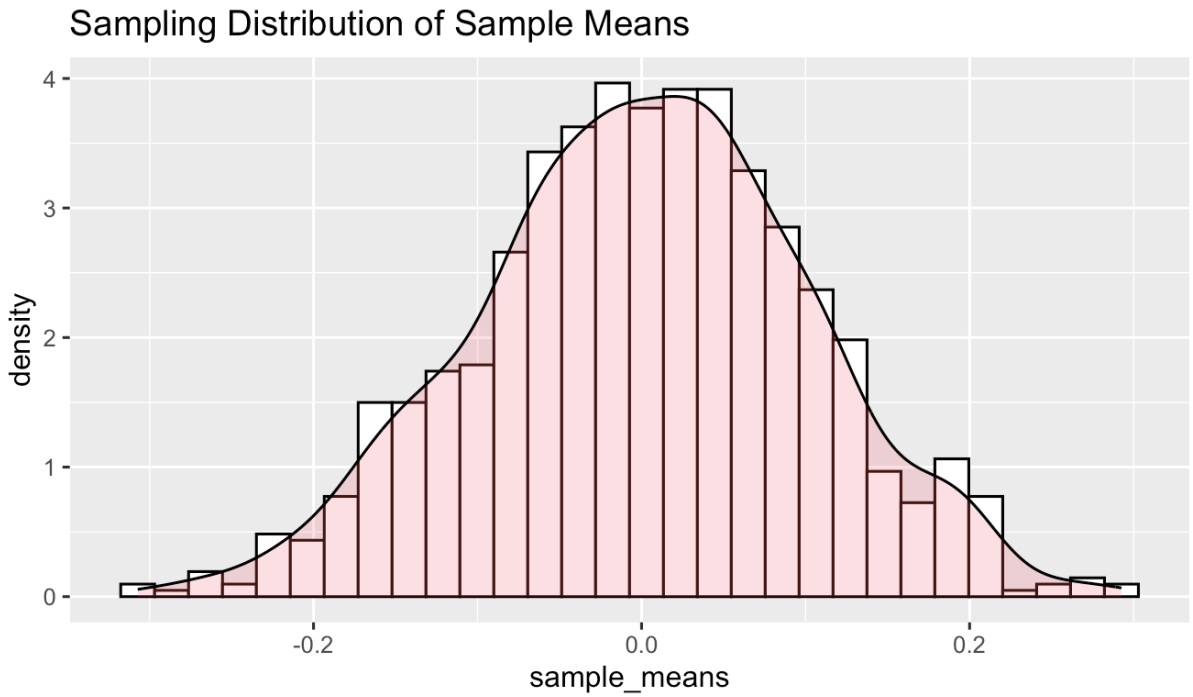
```
sample_means |> var()
[1] 0.01014769
```

上面两行代码中出现了符号 **|>**，这是管道操作符（pipe operator），可以将变量传递给后面的函数。**sample_means |> mean()** 和 **mean(sample_means)** 的含义相同。根据统计学理论，我们知道服从正态分布 $N(\mu, \sigma^2)$ 的随机变量 X 的样本均值 $\overline{X}$ 的抽样分布也是正态分布，其期望值为 $E(\overline{X}) = \mu = 0$，方差为 $\text{Var}(\overline{X}) = \sigma^2/n = 0.01$，这和计算结果基本相符。

接下来我们绘制这 1000 个样本均值的直方图。

```
sample_means |> tibble() |>
  ggplot(aes(sample_means)) +
    geom_histogram(aes(y = after_stat(density)),
                   color="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666") +
    labs(title = "Sampling Distribution of Sample Means")
```

这段代码的第一行将数值向量 **sample_means** 传递给函数 **tibble()**，其作用是生成一个 tibble 类型的数据。而转换成 tibble 数据的目的是将其带入后面的 **ggplot()** 命令进行绘图，包含直方图和核密度拟合的结果。最终得到的分布图如下。



从图中也可以基本确认样本均值的抽样分布是正态分布。

4. 验证大数定律

弱大数定律：如果 X_i 为 i.i.d. 且 $E[X_i] < \infty$，则当 $n \to \infty$ 时，

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{p} E[X_i]$$

上式中的 $\xrightarrow{p}$ 意为依概率收敛，代表随着样本量的增大，随机变量的分布将越来越窄，最后收敛于极限值。下面我们通过仿真验证大数定律。

1. 确定样本量 n 的增加序列，例如 20, 50, 100, 200, 500, 1000, 2000, 5000。

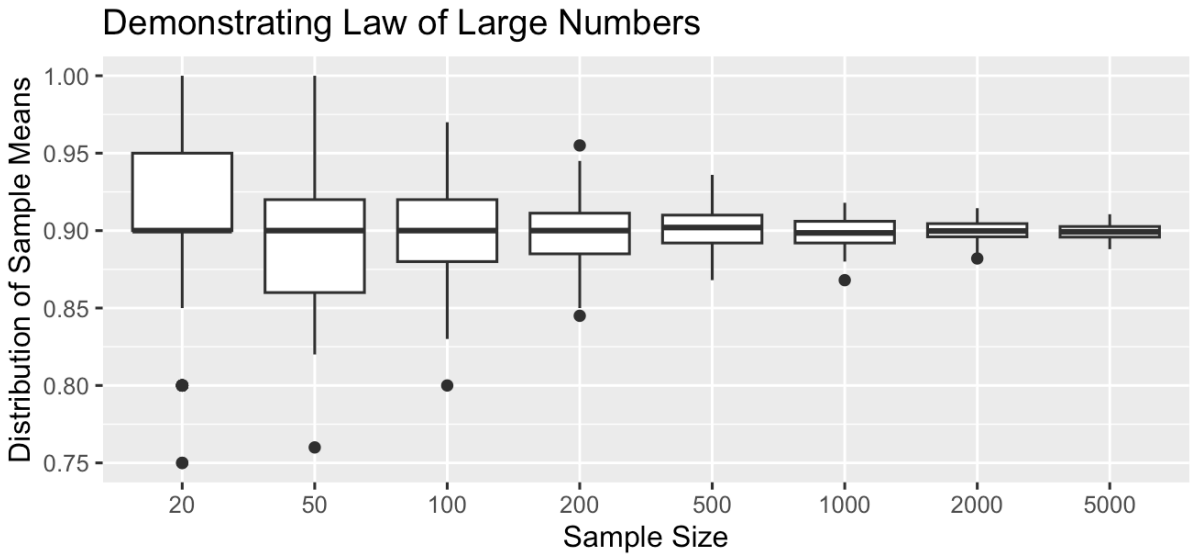
2. 确定用于计算样本均值分布的随机样本数 k，例如 100。
3. 选择 X_i 的分布函数，这里我们用 $p = 0.9$ 的 Bernoulli 分布。针对每一个随机样本计算样本均值 $\overline{X}_n$ 并保存结果。
4. 将样本均值的分布可视化，并观察它们和 n 之间的关系。每一个 n 对应 k 组样本，因此可以用计算出的 k 个样本均值画出箱线图。

```r
set.seed(999)
n_values <- c(20, 50, 100, 200, 500, 1000, 2000, 5000)
k_value <- 100    # 固定 n 时的随机样本数
mean_values <- tibble(
  n = rep(n_values, each = k_value),
  k = rep(1:k_value, length(n_values))
)   # 这一步是准备好保存样本均值的 tibble 数据

sample_means <- map2_dbl(mean_values$n,
                         mean_values$k,
                         \(x, y) mean(rbinom(x, size = 1, p = 0.9)))
    # 将计算出的样本均值依次保存

mean_values <- add_column(mean_values, sample_means)
    # 将计算出的样本均值添加到 mean_values 数据中
mean_values <- mutate(mean_values, n = as.character(n))
    # 将 n 的值改为字符类型，这一步是为了方便后面绘图

mean_values |>
  ggplot(aes(reorder(n,as.numeric(n)), sample_means)) +
  geom_boxplot() +
  labs(
    title = "Demonstrating Law of Large Numbers",
    x = "Sample Size",
    y = "Distribution of Sample Means"
  )
```



5. 验证中心极限定理

Lindeberg-Lévy 中心极限定理：如果 X_i 为 i.i.d. 且 $E\left[X_i^2\right] < \infty$，则当 $n \to \infty$ 时，

$$\frac{\sqrt{n}}{\sigma}\left(\overline{X}_n - \mu\right) \xrightarrow{d} Z \sim N(0,1)$$

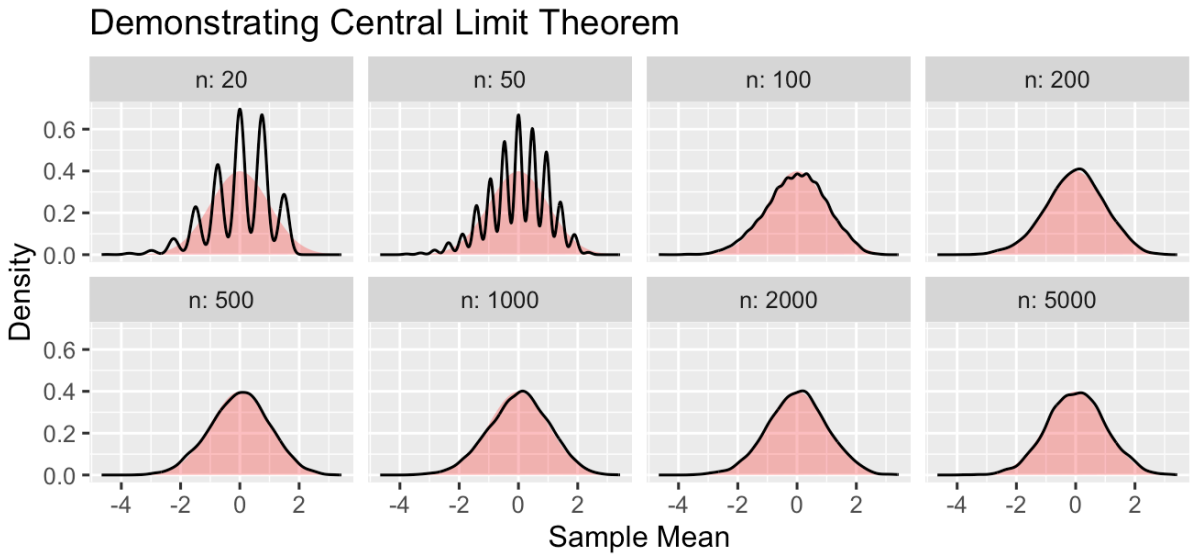
此处 $\mu = E[X_i]$, $\sigma^2 = E\left[(X_i - \mu)^2\right]$。符号 $\xrightarrow{d}$ 意为依分布收敛，代表随着样本量的增大，随机变量的分布将越来越接近极限分布。中心极限定理也可以表达为 $\overline{X}_n \xrightarrow{d} N\left(\mu, \sigma^2/n\right)$。下面我们验证中心极限定理。

1. 确定样本量 n 的增加序列，例如 20, 50, 100, 200, 500, 1000, 2000, 5000。
2. 确定用于计算样本均值分布的随机样本数 k，为了更好地描绘分布，我们选择 $k = 5000$。
3. 选择 X_i 的分布函数，这里我们依旧用 $p = 0.9$ 的 Bernoulli 分布。针对每一个随机样本计算 $\sqrt{n}\left(\overline{X}_n - \mu\right)/\sigma$ 并保存结果。
4. 将样本均值的分布可视化，并对照标准正态分布。这里用样本密度函数的拟合图像与理论标准正态密度函数进行比较。

```
set.seed(999)
n_values <- c(20, 50, 100, 200, 500, 1000, 2000, 5000)
k_value <- 5000     # 固定 n 时的随机样本数
mean_values <- tibble(
  n = rep(n_values, each = k_value),
  k = rep(1:k_value, length(n_values))
)   # 这一步是准备好保存样本均值的 tibble 数据
```

```r
sample_means <- map2_dbl(mean_values$n,
                         mean_values$k,
                         \(x, y) (mean(rbinom(x, size = 1, p = 0.9)) - 0.9) * sqrt(x) / sqrt(0.9 * 0.1))
    # Bernoulli 分布的期望值是 p, 方差是 p(1-p)
mean_values <- add_column(mean_values, sample_means)

mean_values |>
  ggplot(aes(sample_means)) +
  stat_function(fun = dnorm, geom = "polygon", fill = "red", alpha = 0.3) +
  geom_density() +
  facet_wrap(vars(n), nrow = 2, labeller = "label_both") +
  labs(
    title = "Demonstrating Central Limit Theorem",
    x = "Sample Mean",
    y = "Density"
  )
```



课后练习

假设我们要利用样本 $\{X_i\}$ 对总体方差 σ^2 进行估计。通过学习统计学，我们知道样本方差

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2$$

是总体方差的非偏估计量。用蒙特卡洛仿真确认这一点。