

で「ブーツ」

計算機統計学への密やかなお誘い

三中 信宏

minaka@affrc.go.jp

<http://cse.niaes.affrc.go.jp/minaka/>

計算統計学の1つの手法である「ブートストラップ法」をRで実行するソースプログラムをつくってみました。この手法は、系統推定論の業界でも、系統樹の統計学的な誤差を評価するために幅広く用いられています。下記は「リクツをわかってもらおう」ためのプログラムですので、冗長なところもあるのですが、ひとつずつ入力すると、ブートストラップ法やジャックナイフ法などの標本再抽出法が「いったい何をしているのか」がわかるように書いてみました。余裕のある人は、Rを用いて実習してみてください。※「Rでブーツ」というタイトルは、下記を最初にhtml化していただいた久保拓弥（北大）さんのネーミングを拝借しました。

ブートストラップ法：経験的密度関数・信頼区間・誤差評価

「ブートストラップ」とは計算統計学のひとつの手法で、データからの無作為再抽出を繰り返すことにより、推定量（たとえば標本平均のような）の誤差（バラツキ）を評価したり、密度関数を経験的に推定したりする。特定のパラメトリックな確率分布（たとえば正規分布のような）を前提とせず、むしろデータそのものから推定量の確率分布を導き、誤差評価をしたり、信頼区間をつくらうというノンパラメトリックな「精神」の発露である。

以下では、Rを用いて、この「ブートストラップ精神」なるものを体験してみよう。

●数列テストデータによるブートストラップ原理の説明

```
> data <- c(0,1,2,3,4,5,6,7,8,9)
```

```
# 数列 {0,1,2,...,9} をテストデータとして data に格納
```

```
> var(data) # data の分散
```

```
[1] 9.166667
```

ここで **var(data)** は分散の不偏推定値（平方和 / 自由度）であることに注意. 下記に検算結果を記す.

```
> deviation <- numeric(10)
  # deviation を長さ 10 の数値ベクトルとする
> for (i in 1:10) deviation[i] <- data[i] - mean(data)
  # 偏差 = データ - 平均を求め deviation に格納
> ms <- (sum(deviation^2))/(10-1)
  # 分散 = 平方和 / 自由度, すなわち分散の不偏推定値を ms に格納
> ms # var(data) との一致を確認
[1] 9.166667
> var(data)/10 # data の標本平均の分散
[1] 0.9166667

> sample(data, 10, replace=T)
  # data から重複を許して 10 標本を無作為再抽出 (ブートストラップ)
  # 「> sample(data, 10, replace=F)」だと無作為並べ替え (permutation) になる.
[1] 7 7 5 5 3 7 4 8 9 6
  # 以下, 再抽出を繰り返してみる
> sample(data, 10, replace=T)
[1] 2 1 7 3 0 2 4 8 5 0
> sample(data, 10, replace=T)
[1] 4 6 7 1 4 7 7 8 4 5
> sample(data, 10, replace=T)
[1] 9 8 1 7 5 0 5 7 5 7
> sample(data, 10, replace=T)
[1] 3 2 0 8 7 1 2 4 7 7
> sample(data, 10, replace=T)
[1] 6 7 1 4 9 5 8 5 5 9

> mean(data) # 元データ (data) の標本平均
[1] 4.5
> mean(sample(data, 10, replace=T)) # ブートストラップ 1 の平均
[1] 4.8
> mean(sample(data, 10, replace=T)) # ブートストラップ 2 の平均
[1] 5.7
```

```

> mean(sample(data, 10, replace=T))      # 以下, 同じ操作.
[1] 3.5
> mean(sample(data, 10, replace=T))
[1] 5.4

> bootsrtap <- numeric(10)
# bootstrap を長さ 10 の数値ベクトルとしてオブジェクト定義
> for (i in 1:10) bootstrap[i] <- mean(sample(data, 10, replace=T))
# 10 回のブーツストラップ反復による平均値を bootstrap に格納
> bootstrap
# 計算結果の表示
[1] 3.4 5.0 5.2 4.4 4.3 4.6 4.8 3.8 5.3 3.8
> mean(bootstrap)      # 平均
[1] 4.46
> var(bootstrap)      # 分散 (上で求めた分散推定値よりもかなり小さい)
[1] 0.4115556

> bootstrap <- numeric(1000)
> for (i in 1:1000) bootstrap[i] <- mean(sample(data, 10, replace=T))
# ブーツストラップ反復の回数を 1000 回にしてみる
> var(bootstrap)      # 分散 (100 回反復のときよりも大きい)
[1] 0.7411788

```

●正規乱数をデータとする事例——標本平均のブーツストラップ誤差評価

まずはじめに, 正規母集団 (正規分布に従う母集団) からの無作為標本抽出の反復を行ない, 次いで, ある無作為標本集団を擬似的母集団と仮定したときのブーツストラップ再抽出の反復を実行する

```

> data <- rnorm(1000, mean=0, sd=1)
# 標準正規分布 N(0,1) からの 1000 無作為標本を data に格納

> data # data の中身
[1] 0.235074605 -1.356050398 0.267868209 0.087841559 1.012879979
////////////////////////////////////
[996] 1.788910285 -1.358704507 0.035108314 0.708958886 0.359765422

```

```
> mean(data)
```

```
[1] 0.01827712 # data の標本平均
```

ここで正規乱数をこの正規母集団 $N(0,1)$ から反復抽出するという計算を実行しよう。下記では、正規乱数を抽出するごとに、そのヒストグラムと元の分布のグラフを連続的に表示する。

```
> x <- seq(-3, 3, 0.05) # 変数 x として区間 [-3, 3] の範囲の 0.05 刻みの値を入力。
```

```
> random.norm <- rnorm(1000, mean=0, sd=1);
```

```
hist(random.norm, freq=F);
```

```
curve(dnorm(x, mean=0, sd=0.8), add=T)
```

```
# 正規乱数抽出+ヒストグラム描画+正規分布グラフ描画 (上書き)
```

この抽出を反復することにより、無作為標本のばらつき方が体感できる。たとえば、この正規母集団 $N(0,1)$ からの 100 回の無作為抽出により得られた標本平均の分散は下記のようにループ計算できる。

```
> multisample <- numeric(100)
```

```
> for (i in 1:100) multisample[i] <- mean(rnorm(1000, mean=0, sd=1))
```

```
> var(multisample) # 得られた 100 標本平均の分散の不偏推定値
```

つぎに、この $N(0,1)$ からのある無作為抽出によって得られた正規乱数の集団 (**data**) を擬似的に母集団とみなし、そこからブートストラップによって二次的な標本をリサンプリングする計算を実行する。

```
> sample(data, 1000, replace=T)
```

```
# data から重複を許して 1000 個の標本を無作為再抽出 (ブートストラップ)
```

```
[1] 0.254607843 0.613191045 -1.417716944 -0.008924308 0.223065333
```

```
////////////////////////////////////
```

```
[996] 0.474348840 2.061109996 1.757300192 -0.410549785 -0.715952569
```

```
> set.seed(101)
```

```
# 擬似乱数の seed を「101」に設定
```

```
> m <- 1000
```

```
# ブートストラップ反復回数を m=1000 に設定
```

```
> replicate1000 <- numeric(m)
```

```
# "replicate1000" を長さ 1000 の数値ベクトルとしてオブジェクト定義
```

```
> for (i in 1:m) replicate1000[i] <- mean(sample(data, m, replace=T))
```

```
# data からの m 回ブートストラップを実行し、結果を replicate1000 に格納
```

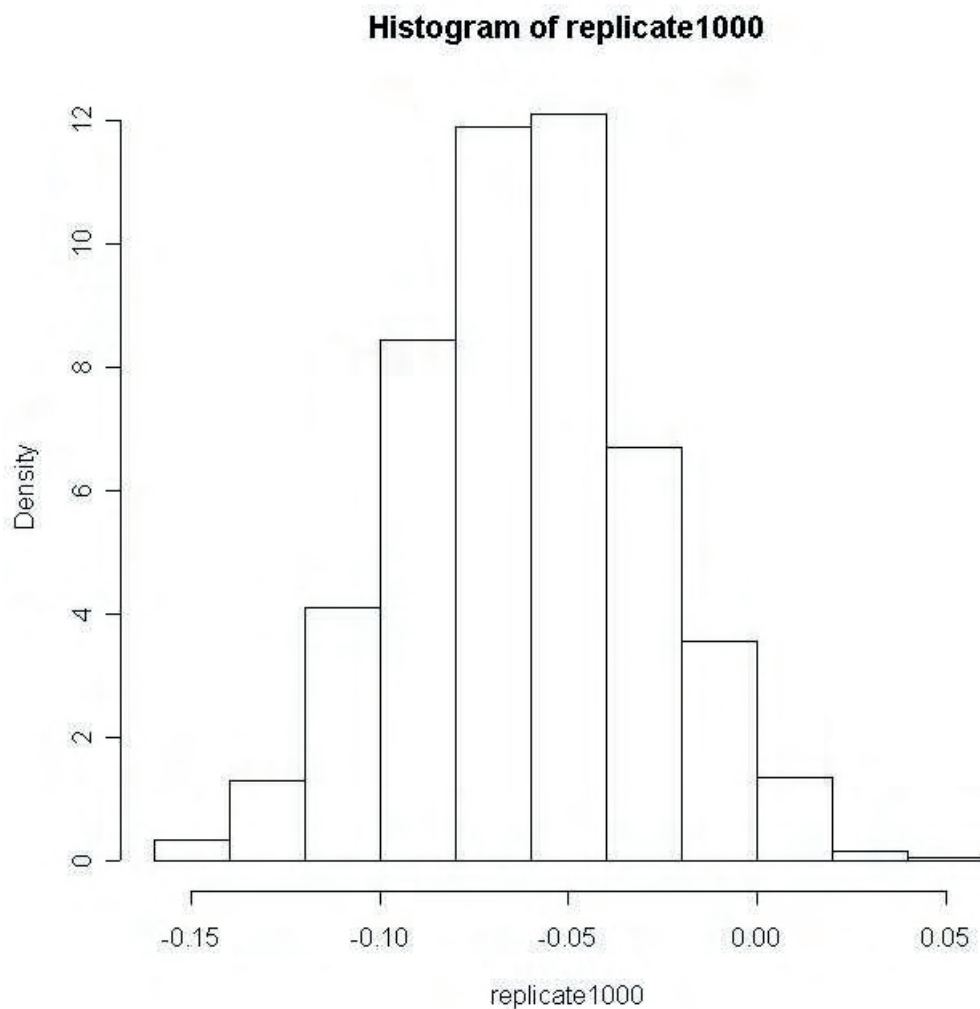
```
> mean(replicate1000)
```

```
[1] 0.01801577 # ブートストラップ平均
```

```
> var(replicate1000)
```

```
[1] 0.0009630001    # ブーツストラップ分散
> sd(replicate1000)
[1] 0.03103224     # ブーツストラップ標準偏差

> m <- 500        # ブーツストラップ反復を m=500 として実行
> replicate500 <- numeric(m)
> for (i in 1:m) replicate500[i] <- mean(sample(data, m, replace=T))
> mean(replicate500)
[1] 0.01753351
> var(replicate500)
[1] 0.002081382
> sd(replicate500)
[1] 0.04562217
```



```

> m <- 100          # ブーツストラップ反復を m=100 として実行
> replicate100 <- numeric(m)
> for (i in 1:m) replicate100[i] <- mean(sample(data, m, replace=T))
> mean(replicate100)
[1] 0.0378861
> var(replicate100)
[1] 0.00949725
> sd(replicate100)
[1] 0.09745383

```

m=100, 500, 1000 の結果をヒストグラム表示

```

> hist(replicate100, freq=F)
> hist(replicate500, density=25, freq=F, add=T)
> hist(replicate1000, density=25, angle=135, freq=F, add=T)

```

●ヒストグラムからの確率密度関数の推定と信頼区間の計算

ブーツストラップによって求められた推定量（ここでは標本平均）のバラツキをヒストグラムとして描画できたならば、それに基づいて確率密度関数を推定することが可能である。得られた経験的密度関数を用いて、推定量の信頼区間を設定することもできる。この密度関数の経験的推定には、ある基底関数（kernel）を用いたヒストグラムの平滑化（smoothing）という手法が用いられている。天下り式の確率密度関数をデータに当てはめるのではなく、むしろデータから逆に密度関数そのものを推定しようというスタンスを取る。

```

> library(MASS)
# ライブラリー MASS をインストールする。
# これは "Modern Applied Statistics with S" のためのライブラリー

> data <- rnorm(1000, mean=0, sd=1)
# 標準正規分布から 1000 乱数をデータとして data に格納

> set.seed(101)
# 擬似乱数シードを 101 に設定

> m <- 1000
# ブーツストラップ反復回数は m=1000 回

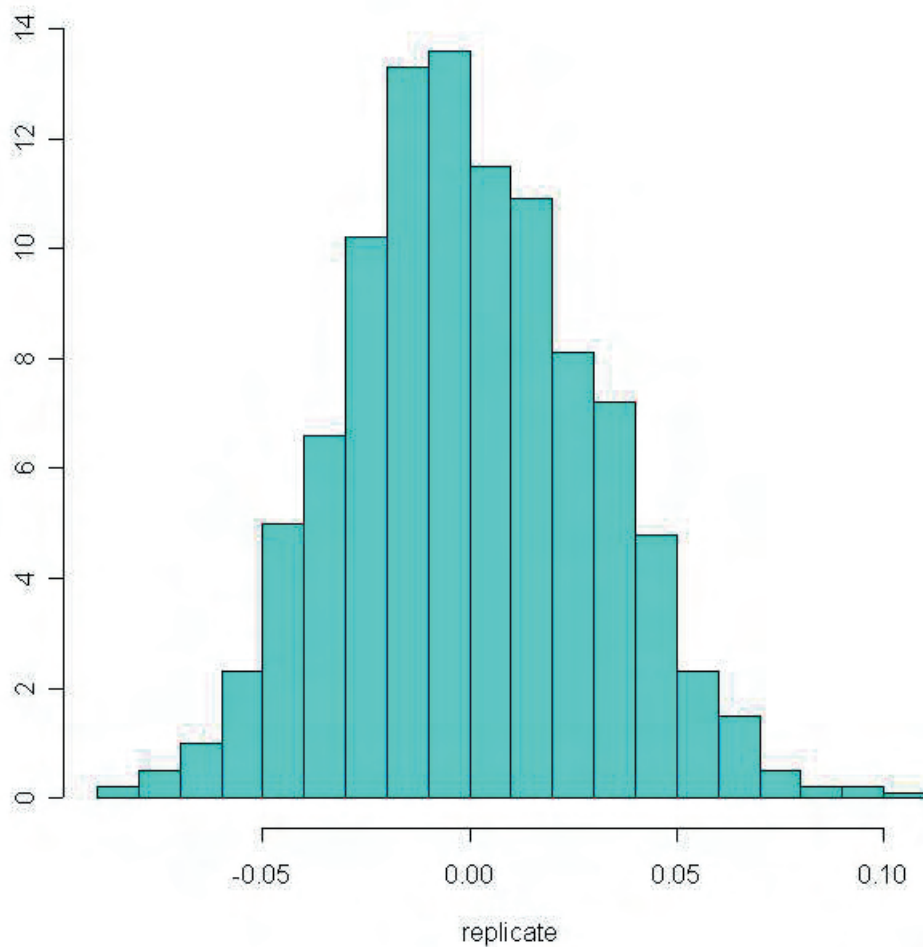
> replicate <- numeric(m)
# replicate は長さ m のベクトル

> for (i in 1:m) replicate[i] <- mean(sample(data, m, replace=T))

```

```
# data からのブーッストラッヅ計算
> mean(replicate) # 平均
[1] -0.008360919
> var(replicate) # 分散
[1] 0.001120811
> sd(replicate) # 標準偏差
[1] 0.03347851

> truehist(replicate, h=0.01)
# 得られた 1000 個のブーッストラッヅ値をヒストグラム表示
> lines(density(replicate, n=256))
# このヒストグラムから経験的密度関数を推定する
```



```
> quantile(replicate, p=c(0.025, 0.975))
```

```
# 推定された密度関数から 5% 信頼区間を求める.  
# 下側 2.5% 点と上側 2.5% 点は下記のように表示される.
```

```
2.5%    97.5%  
-0.07269752 0.05757690
```

```
> root <- sqrt(abs(replicate))
```

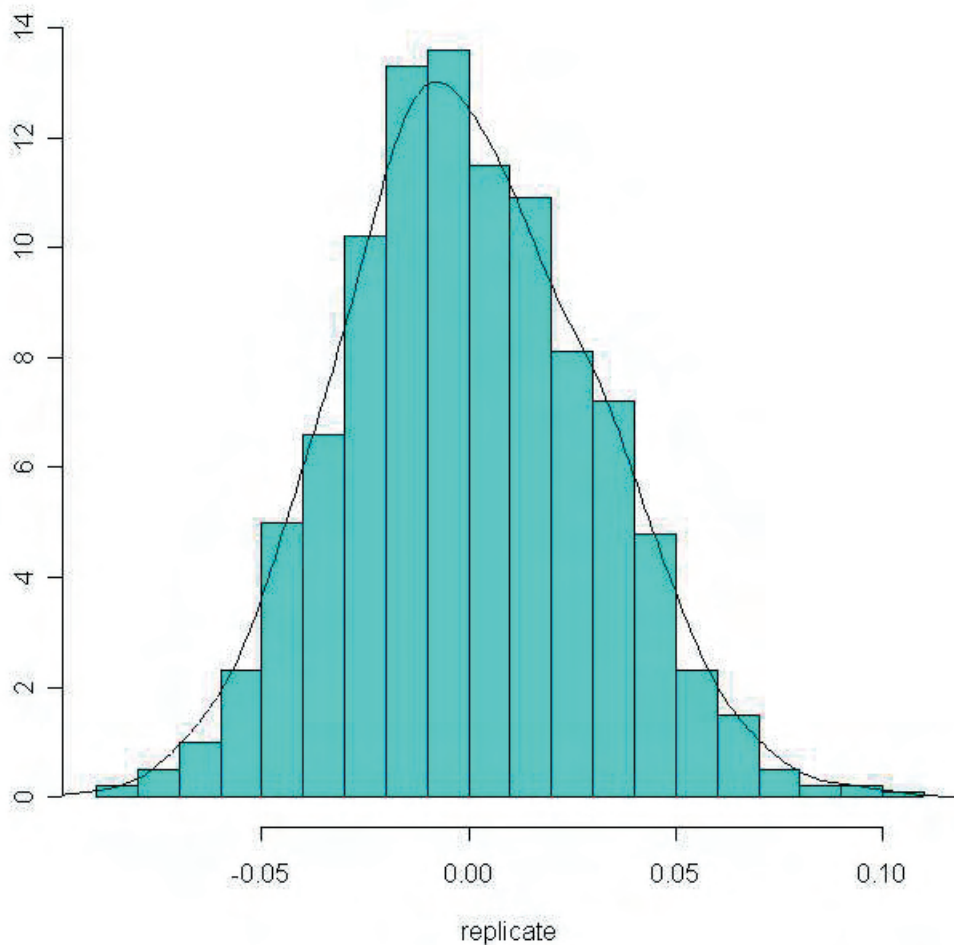
```
# 元の変量の「平方根」を「root」と定義する. この「root」の確率分布をブートストラップ  
# 用いて経験的に構築する. 「root」の密度関数は解析的には求まるかもしれない.
```

```
> truehist(root, h=0.01)
```

```
# 得られた 1000 個の「root」のブートストラップ値をヒストグラム表示
```

```
> lines(density(root, n=256))
```

```
# このヒストグラムから「root」の経験的密度関数を推定する
```




```
> logsine <- log(sin(replicate^2))
```

```
# 元の変数の関数「log(sin(replicate^3))」を「logsine」と定義する. この  
「logsine」の確率分布をブートストラップ用いて経験的に構築する. 「logsine」の密度  
関数を解析的に求めるのは困難である.
```

```
> truehist(logsine)
```

```
# 得られた 1000 個の「logsine」のブートストラップ値をヒストグラム表示
```

```
> lines(density(logsine, n=256))
```

```
# このヒストグラムから「logsine」の経験的密度関数を推定する
```

コマンド「density」を用いたこの密度関数推定は、ブートストラップのみにかぎらず、もっと広く利用できる。

●コマンド「boot」の利用

Rにはブートストラップに特化したコマンド「boot」がライブラリに用意されている。上述の操作は、この「boot」を用いることにより、格段に単純化される。たとえば、数列データを例とする、ブートストラップ計算は下記の通り：

```
> library(boot)
```

```
# ライブラリ「boot」をインストールする
```

```
> data <- c(0,1,2,3,4,5,6,7,8,9)
```

```
# 数列 {0,1,2,...,9} を data に格納
```

```
> set.seed(101)
```

```
# 乱数シードを「101」に設定
```

```
> boot(data, function(x, i)mean(x[i]), R=1000)
```

```
# ブートストラップの設定
```

1) 「data」:

再抽出対象となるデータ名. ここでは "data".

2) 「function(x, i)mean(x[i])」:

誤差評価を行なう統計量の関数指定. ここでは標本平均.

3) 「R」:

ブートストラップ反復回数. ここでは 1000 回.

以下は、計算結果：

```
ORDINARY NONPARAMETRIC BOOTSTRAP
```

Call:

```
boot(data = data, statistic = function(x, i) mean(x[i]), R = 1000)
```

Bootstrap Statistics :

```
original bias std. error  
t1* 4.5 -0.0101 0.9023204
```

「ブートストラップ統計量」として出力されるのは、元データの平均 (original)・ブートストラップ反復から得られた平均の original からのバイアス (bias)・ブートストラップから得られた標本平均の標準誤差 (std. error) である。

なお「boot」は他にも多くの機能をもっており、上述のような、ノンパラメトリック・ブートストラップだけでなく、パラメトリック・ブートストラップ (モンテ・カルロ法的一种) も実行できる。詳細は help(boot) を参照していただきたい。

ジャックナイフ法による誤差評価

ジャックナイフによる誤差評価は、データからの重複を許さない (replace=F) 再抽出の反復に基づく。したがって、反復回数を m, 再抽出数を k (< m) で指定したとき、

```
> replicate <- numeric(m)  
> for (i in 1:m) replicate[i] <- mean(sample(data, k, replace=F))  
> var(replicate)
```

によって、標本平均の誤差を求めることができる。たとえば、下記のように：

```
> data <- c(0,1,2,3,4,5,6,7,8,9)  
# データ読みこみ  
> m <- 10 # 再抽出回数  
> k <- 9 # 除去数 1 の delete-one jackknife  
> replicate <- numeric(m)  
> for (i in 1:m) replicate[i] <- mean(sample(data, k, replace=F))  
> var(replicate)  
[1] 0.08079561
```

```
> m <- 10    # 再抽出回数
> k <- 5      # 半数除去の delete-half jackknife
> replicate <- numeric(m)
> for (i in 1:m) replicate[i] <- mean(sample(data, k, replace=F))
> var(replicate)
[1] 0.6204444
### EOF ###
```