RjpWiki アーカイブス

【JIS, ISO 式四捨五入[1] (06.06.28) 】

桁数の多いデータはしばしば丸め処理 (rounding) される．よく知られたものに，切り下げ，切り上げ，そして四捨五入がある．四捨五入は切り上げ，切り下げが相半ばし，長い目で見ると公平であるといわれるが，実際は，少しだけ大きめに丸める傾向があることは，次の表から分かる．

元の数	3.0	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9
四捨五入後	3	3	3	3	3	4	4	4	4	4
差	0.0	-0.1	-0.2	-0.3	-0.4	0.5	0.4	0.3	0.2	0.1

JIS(日本工業規格) や，ISO(国際標準化機構) では，任意の桁位置における丸め処理法を次のように定めている (JIS Z 8401 方式および ISO 3110 方式)．この方式は 10 進法だけでなく，2 進法等にも適用される．R の四捨五入関数 round は IEEE 754 方式 (IEEE 規格は計算機における数値処理の基本を定める) に基づき、これは JIS, ISO 方式と一致 (簡単にいえば誤差が最小になるように丸めるということ) するが、小学校で習った四捨五入とは微妙に食い違う (つまり五入ばかりでなく五捨もあり得るの) ので注意が必要である。

(1) 一番近い丸め結果候補が 1 つだけなら，その数に丸める，

(2) 一番近い丸め結果候補が 2 つある場合は，末尾が偶数のものに丸める，

(3) 丸め処理は 1 段階で行なわなければならない．

例えば小数点以下第 1 桁に丸めるなら次のようになる．規則 (2) が適用される場合は通常の四捨五入とは異なる結果になる場合がある．

丸める数	12.223	12.251	12.25	12.35
丸め結果	12.2	12.3	12.2	12.4
適用規則	(1)	(1)	(2)	(2)

規則 (3) は，例えば 12.345 は直に 12.3 と丸めるべきであって、まず 12.35 とし、それから 12.4 としてはならないことを主張している．この規則は，丸めによる誤差が最小になる利点がある．

```
> round(122.5)  # 五捨！
[1] 122
> round(122.51) # 五入
[1] 123
> round(123.5) # 五入
[1] 124
> round(122.501) # 五入
[1] 123
```

[1] 「なんでも掲示板」の中間さんの投稿記事参照。

計算機ソフトでは，伝統的に通常の四捨五入方式を採用してきた．市販の電卓やパソコンソフトの丸め処理法は，必ずしも JIS，ISO 方式になっていないことも多く，チェックが必要であるといわれてる．(今入った情報では、Excel の丸め関数は、小学生方式、一方 VBA は IEEE 方式だそうです。)

とはいえ、実際は気にするほどの差ではない (なにかおかしいかな？)

```
> test1 <- function () {
      x <- floor(100*runif(10000000))/10
      y <- round(x)   # JIS, ISO, IEEE 式四捨五入
     mean(abs(x-y))
   }
> test1()
[1] 0.2499519
> test2 <- function () {
     x <- floor(100*runif(10000000))/10
     y <- oround(x)  # (中間さん作の) 世間一般風四捨五入
    mean(abs(x-y))
  }
> test2()
[1] 0.250041
```

絶対値をとらないで平均すると大きな (?) 差 (これで安心)。いずれにしても丸めた方が若干大きくなる傾向に注意。

```
> test1 <- function () { x <- floor(100*runif(10000000))/10
> y <- round(x); mean(x-y)}
> test1()
[1] -5.579e-05
> test2 <- function () { x <- floor(100*runif(10000000))/10
> y <- oround(x); mean(x-y)}
> test2()
[1] -0.04997603
```

中間さんの oround 関数へのコメントに答えるべくシミュレーション。最大一万円の買いものを一千万回したとして、消費税込の値段を、普通の四捨五入と、ISO 式でした場合の食い違い総和は

```
> test <- function(){
     x <- floor(10000*runif(10000000))
     x <- 1.05*x
     sum(oround(x) -x)}
> test()
[1] 249700.4  # 現状方式では25万円消費者がとられすぎ。
> test <- function(){
     x <- floor(10000*runif(10000000))
     x <- 1.05*x
    sum(round(x) -x)}
> test()
[1] 1515.3     # ISO 式！
```