ARTICLE

基般

情報教育と統計教育 No. 1

R による 1 行プログラミング

奥村晴彦
三重大学教育学部

情報教育と統計教育の接点を扱う 3 回シリーズの第 1 回として，話題の R 言語を使った情報＋統計教育を紹介する．

R による 1 行プログラミング

問　正規分布の乱数を百万個作り，配列 x に入れよ．

答　`x = rnorm(1000000)`

問　x の度数分布図を描け．

答　`hist(x)`

（ちょっと寂しいので色を付けよう）

```
hist(x, col="gray")
```

これは，R 言語を使った私の授業でウォーミングアップ的に行う小課題の一例である．次ページ**図 -1** に RStudio（R のフロントエンドの 1 つ）で上のことを実行した画面を示す．

注目すべき点は，ほとんどの作業が 1 行プログラムでできてしまうこと，そして，1 行 1 行の効果が即時に得られることである．エラーが出ても ↑ キーで履歴を遡って修正できる．

同じことを C や Java の類で行うのはとても面倒である．Excel でも（「百万」が「百」であっても）まともな度数分布図を描くには髪（かみ）ワザが必要である．R 言語では，上の程度のことなら，オマジナイ（`#include <stdio.h>` や `import matplotlib.pyplot as plt` の類）を唱えることなく，掛け値なしの 1 行プログラミングが可能である．

対話型の環境ということなら，R に限らず，多数存在する．古い人なら BASIC のダイレクトモードを思い出すであろうし，Ruby（irb など），Python（特に IPython）もそうである．これらの中で R の特徴をより詳しく説明するために，次の問題を考えてみよう．

FizzBuzz 問題：従来型解法

1 から 100 までの整数を唱えよ．ただし，3 の倍数なら Fizz，5 の倍数なら Buzz，両方の倍数なら FizzBuzz と唱えよ――これが有名な FizzBuzz 問題である．これを R で解いてみよう．おそらく C や Java を習った学生なら次のようなプログラムを考えるであろう（`%%` は剰余演算子，`cat()` は出力）：

```
for (i in 1:100) {
    if (i %% 15 == 0)
        cat("FizzBuzz\n")
    else if (i %% 3 == 0)
        cat("Fizz\n")
    else if (i %% 5 == 0)
        cat("Buzz\n")
    else
        cat(i, "\n")
}
```

R でこれを実行するには，テキストエディタで作っておいて R の入力画面にコピペするか，fizzbuzz.R といった名前で保存してファイル名を指定して実行する．RStudio なら，ソースエディタの選択された部分を実行する機能がある．OS によっては，fizzbuzz.R の先頭に

```
#!/usr/bin/env Rscript
```

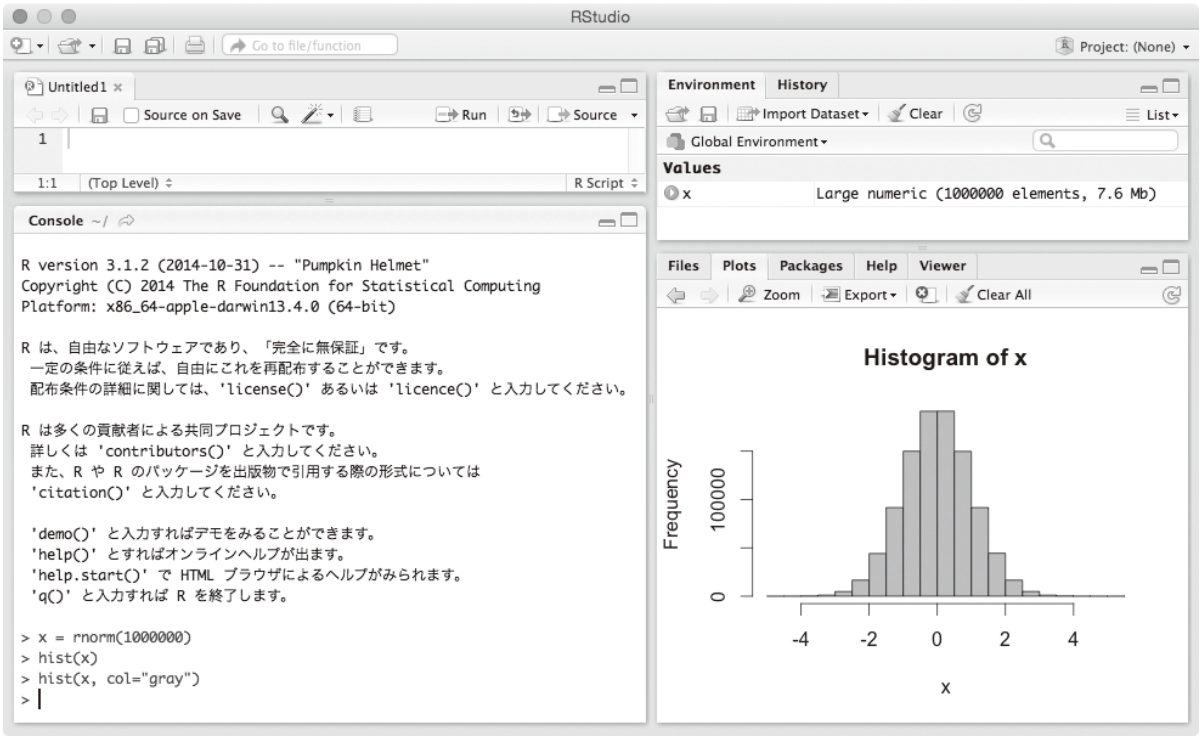

図 -1　RStudio に本稿冒頭の例を打ち込んだところ

と書いて実行許可を与えれば，単独で実行できるスクリプトになる．

FizzBuzz 問題の 1 行プログラミング

よりR的な方法は，たとえば次のように，1行プログラムを積み重ねて解くことである．ここでは対話型環境に1行ずつ打ち込むことを強調するために，Rのプロンプト > を明示的に書いている：

```
> a = 1:100
> a[seq(3,100,3)] = "Fizz"
> a[seq(5,100,5)] = "Buzz"
> a[seq(15,100,15)] = "FizzBuzz"
> a
```

1行プログラミングの特徴として，どの段階でも，上の例ではaというベクトル（配列）に，途中段階の解が入っていることである．最後の行のように，Rのコンソールにaとだけ打ち込めば，全要素が表示される．試しに1行目を打ち込んだ直後にaと打ち込めば，

```
> a = 1:100
> a
  [1]  1  2  3  4  5  6  7  8
  [9]  9 10 11 12 13 14 15 16
 [17] 17 18 19 20 21 22 23 24
...
```

のように出力される（改行位置は端末の幅に依存する）．左端の [ ] 内は各行の先頭の要素の番号（1から始まる）である．これで 1:100 は1から100までの整数のベクトルであることが分かる．

seq(*from, to, by*)は，初項・上限・公差を与えて等差数列のベクトルを作る関数である：

```
> seq(3, 100, 3)
 [1]  3 6 9 12 15 18 21...
```

したがって，

```
> a[seq(3,100,3)] = "Fizz"
```

はa[3]="Fizz";a[6]="Fizz";...と同じことで，配列aの3の倍数番目の要素に "Fizz" を代入する．

最終的に上のプログラムの出力は次のようになる（整数型が文字列型に変換されたことに注意する）：

```
> a
  [1] "1"     "2"     "Fizz"  "4"
  [5] "Buzz"  "Fizz"  "7"     "8"
...
```

明示的に1行ごとに出力したいなら，次のようにする：

```
> cat(a, sep="\n")
1
2
Fizz
...
```

このように，Rの基本データ型はベクトル(配列)であり，スカラはベクトルの特別な場合（長さ1の場合）である．ベクトルデータの作り方はいろいろあり，上の`a = 1:100`もその1つであるが，次のように要素を列挙してもよい：

```
> x = c(3.14, 2.718, 0.577)
> 初夢 = c("富士", "鷹", "茄子")
```

変数(オブジェクト)名には日本語も使える．

ベクトルをうまく使うと，かなりのforループやif文を隠ぺいでき，場合によっては高速化が可能である．

文字列処理

行ごとに収めた単語を読み込み，「全角」英数字を「半角」に直し，英字を小文字に揃え，出現頻度順に10位まで出力したい．これも1行プログラムの並びで解くことができる．

2～3行目は，本誌紙面に合わせるため，途中で改行している．Rは文法的に完結しない行は継続すると見なす．継続行のプロンプトは+である．

```
> x = readLines("words.txt")
> x = chartr("０-９Ａ-Ｚａ-ｚ",
+            "0-9A-Za-z", x)
> x = tolower(x)
> s = sort(table(x),decreasing=TRUE)
> head(s, 10)
```

`readLines()`は，ファイル全体を読み込み，各行をベクトルとして返す．`chartr()`は，Unixのtrコマンドと同様に，対応する文字に変換する．`tolower()`は大文字を小文字に変換する．これらの関数はすべてベクトルに要素ごと作用する．`table()`は度数分布表を作り，`sort()`は整列する．`head()`は，Unixの同名コマンドと同様に，先頭だけを取り出す．

上の例では使っていないが，POSIX互換またはPerl互換の正規表現も使える．

また，文字列処理ではないが，上で求めた度数分布sのエントロピーを求めるのも簡単である：

```
> p = s / sum(s)
> -sum(p * log2(p))
```

Rとは？

ここで改めてRを紹介する．

Rは，オープンソースの言語処理系である．対話形式で使われることが多い．特に統計・データ解析，統計グラフ作成に強い．追加パッケージによって機能拡張できる．公式パッケージだけで6,000種以上が公開されており，XML，JSON，Excelファイルの読み書きや，多倍長演算など，多くのことが簡単なコマンドで行える．

◆R言語の代入は <- か = か

R言語（およびその前身のS言語）の代入は，もともと`x <- 3`または`3 -> x`のように書いたが，1998年のS Version 4から`x = 3`という書き方もできるようになった．Chambersの本 *Programming with Data : A Guide to the S Language*（1998）では=が主に使われている．R言語では2001年から=も使えるようになった．現在でも<-を使う人が多い（例："Google's R Style Guide"）が，本稿では=を使っている．

<-を使う理由として，=は関数の引数リストの中では名前付き引数を指定する=と衝突することが挙げられる．たとえば`mean(x = var)`ではxは引数名と解釈される．もっとも，引数リストの中での代入は，遅延評価のため予想外の結果になることもあり，推奨されない．

Rの元となったSは，1976年から，AT&T Bell研究所の統計学者たちによって開発された．Sは商用のS-PLUSに発展した．一方，オープンソースのRは，1990年代に，Auckland大学のRoss IhakaとRobert Gentleman（2人のR）によって実装された．現在は，彼らを含むR Core Teamによって開発が続けられている[1]．

Rは内部的にはSchemeに近い関数型言語である．ChambersはRの特徴をfunctional OOP（関数型オブジェクト指向プログラミング）と呼んでいる．

ソースコードのほか，Windows，Mac，Linux用のインストーラが配布されている．Windows用・Mac用のものには専用の実行環境が付属しているが，ターミナルに「R」と打ち込んでも使える．最近はRStudio[☆1]というフロントエンドがよく使われる．Emacsからも使える（ESS）．IPython Notebookからも使える（rpy2）．

学生用のPCで自由にRやRStudioがインストールできない場合は，別のLinuxサーバでRとRStudio Serverを動かしておけば，ブラウザから使える．

ヘルプは対話型環境内で参照できる．あるいはコマンドでたとえば

```
> ?rnorm
```

と打てばrnorm()のヘルプが表示される（現状では英語）．

☆1 http://www.rstudio.com

情報と統計を融合した授業例

筆者は，かねてから情報処理と統計・データ解析とを融合した教育を模索していた[2]．2009年度からはRを使った統計学の授業の中で試行錯誤を重ねている[3]．適当な教科書がないので，教材はすべて自作し，Webで公開している[4]．授業はPC教室（Windows）で行っている．最近はR標準のユーザインタフェースとRStudioフロントエンドから好きな方を選ばせているが，ほとんどの学生がRStudioを使っている．

簡単な計算から，平均・標準偏差・相関係数などの統計量の計算，直線や曲線のあてはめ，p値や信頼区間の計算，グラフ描画，乱数を使ったシミュレーションなど，多様な実習を通じて，データ処理から統計学の基本的な考え方までを教えている．

Hello, world! を出力するのに何行も必要な実習と比べて，たった1行のプログラムでも意味のあるデータ処理ができるので，最近のデータサイエンスブームも手伝って，学生のモチベーションは高い．

参考文献

1) R Core Team：R：A Language and Environment for Statistical Computing, http://www.R-project.org
2) 奥村晴彦：情報教育と統計，情報処理学会研究報告コンピュータと教育，2008-CE-97, pp.81-88 (2008).
3) 奥村晴彦：Rを使った情報教育，情報処理学会SSS2010論文集，pp.77-80 (2010).
4) 奥村晴彦：統計・データ解析，http://oku.edu.mie-u.ac.jp/~okumura/stat/

（2015年3月31日受付）

奥村晴彦（正会員） okumura@okumuralab.org

三重大学教育学部教授（情報教育）．LHAデータ圧縮アルゴリズムの開発者，『LaTeX2e美文書作成入門』（現在第6版）の著者．