April 16, 2008 V0R1 A
June 24, 2008 V0R2 B
June 5, 2012 V0R3 C

情報リテラシー (あるいは同演習)

第 1 回 今年度は S 言語

1 はじめに

この科目の狙いは「基礎プログラミング」を終えた学生を対象に、既修得の言語とは異なる言語を用いて、設計はほとんど与えられるという状況でのコーディングをさらに演習する機会を与えることにある。

コーディングの得手不得手、プログラミング言語の好き嫌いはともかく、研究室、職場などでは予め合意されているプログラミング言語を用いることになり、選択の余地はない。この講義も、毎年、同じ言語を用いる予定はなく、数年後とに変更していく予定である。[1]

プログラムを創ることについて、この学科には以下の科目が用意されている。まず 1 年次の後学期に「基礎プログラミング」で、昼間コースは C、夜間主コースは Pascal と、それぞれの言語を用い、プログラムのコーディングの基礎を練習する機会を与えている。2 年次の前学期にはこの科目が開講されている。ここまでが、コーディングの基礎に相当する。効率のよいプログラムを設計するのに必要な知識を学べる科目として、来学期以降には、昼間コースは 2 年次後学期の「アルゴリズムとデータ構造」と 3 年次後学期の「ソフトウェア工学」、夜間主コースは 2 年次後学期の「数理解析」と 3 年次前学期の「アルゴリズムとデータ構造」と 3 年次後学期の「ソフトウェア工学」がある。

ちょうど二科目目として、ここでは主に、数値解析の問題を課題として、演習を進めて行く。

2 S 言語を対象に選定した理由

1. あまり意識しなくても使えるが、オブジェクト指向言語である。
2. `Maple` と同様に変数の型宣言が不要なこと。ローカル変数の宣言も不要なところが `Maple` より楽。
3. 文法が `MAPLE` よりも `C` に近く、`Perl` にも通じる雰囲気がある。
4. 自宅でも使える実行環境 (R) が利用可能。`http://cran.r-project.org/` からダウンロード可能。
5. データの分析やシミュレーションに向いている。

3 S 言語と R の関係

S 言語はベル研で開発されているプログラミング言語で、R はその派生という関係にある。S-PLUS は、S 言語に対して、Insightful 社が独自の機能拡張を行った商用パッケージの名称である。R は S 言語の文法に沿って書かれたプログラムを実行できる、処理環境の名称だが、R 言語と呼ばれてしまうことも少なくない。本テキストでは、プログラミング言語は S、実行環境は R と呼び分けることにする。

文法は同一のため、S 言語や S-PLUS について書かれた書籍は、R を用いる上でも参考になる。ただし用意されているライブラリ (R ではパッケージと呼ぶ) は、S、S-PLUS、R とで異なることが多く、S-PLUS について書かれた統計分析やデータマイニングの書籍や文書を参考にしながら、R を用いる際には、差異があることを忘れてはいけない。

[1] 数回の変更の後に、元の言語に戻る可能性はある。演習の狙いから、無償で使用できる言語を選択していく予定。

4 Rの起動

スタートメニューから「R」を探し、その中の「R 2.15.0」もしくはそれより古いバージョンの数字を伴っているアイコンを選ぶ。すると、RGui という名前のウィンドウが起動し、その中に R Console というサブウィンドウが表示される。幾つかのメッセージの下に表示されているのが次のアイコンである。

```
R version 2.14.1 (2011-12-22)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

 >
```

コマンド入力のプロンプトは>となっている。複数行に渡る命令を入力するときは、都合の良いところでそのまま改行コードを挿入すると、プロンプトが+に変わる。

例えば、後に出てくる

```
> A <- matrix(c(1,2,0,0,2,-1,1,0,3),ncol=3,nrow=3,byrow=T)
```

というコードは、幾つかの場所で改行を挟むことができる。見やすくするには例えば

```
> A <- matrix(c(1,2, 0,
+               0,2,-1,
+               1,0, 3),
+             ncol=3,nrow=3,byrow=T)
```

などとする。プロンプトは>も+もどちらも自動的に挿入されるので、メモ帳や Word などにコードを書いてから、コピーして R Console に貼り付ける場合には、+を行頭に記してはいけない。[2]

行末記号に; を用いることも可能だが、省略しても良い。

S 言語と R 言語ができることは数多く、特にデータの扱いや分析のための機能が数多く備えられている。しかしこの講義では主に、アルゴリズムのプロトタイプの実装のための処理系として S や R を取り上げたい。似たような目的でよく用いられる処理系には Matlab、Gauss、Maple、Mathematica などがある。これらに共通して

[2] Windows 版には、コピーとコマンドのみのペースト機能がある。

いるのは、それぞれのソフトウェアの限界まではだが、一旦、処理系の限界に到達すると、C や Fortran といった伝統的なプログラミング言語に立ち返って、一からプログラムを書き直さなければならない点である。そのため、いつでもプロトタイプを書いている、という心構えは持っておいた方が良い。

5 S 言語の基礎

S 言語と R 言語の基本オブジェクトには真偽、実数、複素数、文字列、ベクトル、リストなどがある。代入演算子は<-の二文字を用いる。例えば変数 x に 3 を代入するなら

```
> x<-3
```

といった具合に。これで実数値を持つ数値オブジェクトが定義されたことは、

```
> is.numeric(x)
```

の結果から確認できる。

文字列は'' で囲む。

```
> y<-"こんにちは"
```

これも文字列オブジェクトかどうかを確認すると

```
> is.character(y)
```

の返り値が TRUE になるはずである。

ベクトルとは真偽ベクトル、実数ベクトル、複素数ベクトル、文字列ベクトルなどと、全要素が四種類の基本オブジェクトと同じ属性を持つ、一次元配列のことであり、c() という関数を用いて、次のように定義する。

```
> x <- c(TRUE,FALSE,FALSE,TRUE)
> y <- c(9,7,3,2)
> year <- 1990:1993
> names <- c("payal", "shraddha", "kritika", "itida")
```

全てを入力した後で、

```
> x
> y
> year
> names
```

と実行すると、どのようなベクトルが定義されたか、が見て取れる。この一次元配列は様々なところで現れ、S 言語のプログラムを書く上での特徴的な存在と言える。c() は省略できることもあるが、引数は必ず、同じオブジェクト型を持たなければならない。例えば次のコードは、エラーを返さず、すべてを文字列型と認識して、処理していく。

```
> ex <- c(TRUE,9,1990,"payal")
```

ベクトルの生成には、この他に

```
> paste("学生",1:22)
> seq(5,1,by=-2)
> (3:1)*2-1
```

などがある。

ベクトルの各要素にアクセスするには、要素を参照する記法 `[]` をオブジェクトに加えて、

```
> x[1]
> y[length(y)]
> year[c(2,4)]
> names[year==1992]
```

などとする。

> Q1. 1ドルを年率6%で mヶ月運用すると `(1+6/1200)^m` になる、という式があるときに、毎年末の元利合計額を6年間に渡って、求めてみなさい。また17%だとどうなるか。

もう一つの特徴がリストである。リストは、複数のオブジェクトをラベル付きで収めることができる。リストの宣言には、関数 `list()` を用いる。

```
> person <- list(name="payal", x=TRUE, y=9, year=1990)
```

リストの要素は、オブジェクト型が異なっていても問題ない。各要素を取り出すには、

```
> person$name
> person$year
```

などとする。リストの全要素の名前を取り出すには、

```
> names(person)
```

と関数 `names()` を用いる。

他に行列、データ・フレームも基本的なオブジェクトである。

ところでS言語では、幾つかのアルファベットには特別な意味があり、それらを変数として用いることは避けるのが良いとされている。特に `c` と `t` は避けた方がよく、`T` と `F` は何かを代入すると `R` 全体の挙動が変わってしまう。次のそれぞれを実行し、何が表示されるかを試してみよ。

```
> c
> t
> D
> F
> T
> time
> pi
```

6 S言語の演算の優先順位

上に行くほど、結合度が高い。優先順位を無視したいときには、括弧を明示的につける。

表 1: 優先順位

`::, :::`	リストの成分	高い
`$,`	リストの成分	
`[ [[`	部分、要素	
`^`	べき乗	
`- +`	正負 (単項演算)	
`:`	等差数列	
`%名前%`	特殊演算	
`* /`	乗除算	
`+ -`	加減算	
`< > <= >= == !=`	比較演算	
`!`	否定 (単項論理演算)	
`& | && ||`	論理演算	
`<- <<- = -> ->>`	代入	
`?`	ヘルプ	

```
> 1:3-1
> 1:(3-1)
> 1:-2
> 3:0-2
> 3:(0-2)
```

7 S 言語の線形代数

列ベクトルと行列は次のような命令で定義する。

```
> x <- matrix(c(1,2,3),ncol=1,nrow=3)
> A <- matrix(c(1,2,0,0,2,-1,1,0,3),ncol=3,nrow=3,byrow=T)
```

x は 3 行 1 列の行列 (=要素が 3 の列ベクトル) で要素は最初から順に 1, 2, 3 であるとしているのが一行目で、数式で書けば

$$\mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

に相当する。二行目は

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 2 & -1 \\ 1 & 0 & 3 \end{pmatrix}$$

である。二行目の一番最後の `byrow=T` を除くと、

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 2 & 0 \\ 0 & -1 & 3 \end{pmatrix}$$

となるので、c() の中身の記述順序に合わせて byrow=T としたり、byrow=F としたりする。

統計計算と線形代数は切っても切り離せない関係にあり、線形代数の計算のための機能は、拡張ライブラリ等を呼び出したり、独自にコードを書く必要は無く、予め実装されている。

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} + \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}$$

の計算には単に、

```
> x <- matrix(c(1,2,3),ncol=1,nrow=3)
> y <- matrix(c(2,2,2),ncol=1,nrow=3)
> x+y
```

とするだけで良い。引き算も同様に、

```
> x-y
```

で済む。ベクトルや行列のスカラー倍は単に

```
> 2*y
> 3*A
```

とすれば良い。

内積

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}' \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix}$$

の計算には、片方のベクトルを転置する必要があり、そのために先ほど変数として用いることを禁じた t を用いる。更に行列同士の掛け算には特別な記号

```
> t(x)%*%y
```

で得られる。これと

```
> x%*%t(y)
```

の違いも気づいて欲しい。

行列とベクトルの積も

```
> A%*%x
```

で得られる。

行列の固有値と固有ベクトルは

```
> eigen(A)
```

と関数 eigen() を用いて得る。

これら以外にも特異値分解 svd()、QR 分解 qr()、連立方程式の解 solve()、逆行列 solve() などの関数が用意されている。

8 グラフィックスの初歩

R で関数プロットを描くには、curve() と plot() の二つの関数が利用できる。

```
> curve(expr=sin(x), from=-10, to=10, col="blue")
> curve(expr=cos(x), from=-10, to=10, col="red", add=TRUE)
```

curve() という関数は最初の引数 expr がプロットしたい関数、二つめの引数 from はプロット範囲の始点、三つ目の引数 to はプロット範囲の終点である。add=TRUE を引数に加えると、既に表示されている関数グラフに、プロットを追記する。また同じ事は

```
> plot(sin, from=-10, to=10, col="blue")
> plot(cos, from=-10, to=10, col="red", add=TRUE)
```

としても実現される。

9 フロー制御

9.1 for

単純な繰り返しには for を用いる。

例えば、1 から 100 まで足す命令は、

```
> x <- 1
> for ( i in c(2:100) ) {
    x <- x + i
  }
> x
```

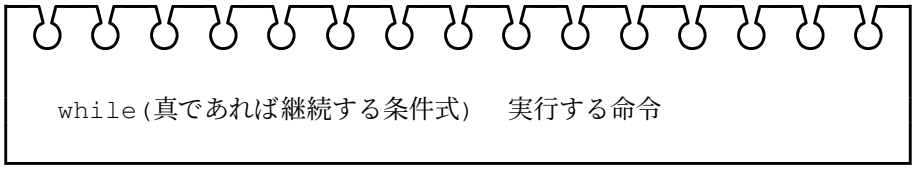
とする。ここで 2:100 は、2 から 100 まで 1 ずつ増加する等差数列の意味で、ループは変数 i が順に 2 から 100 までの値を取りながら進む。

Fibonacci 数列は次のように計算できる。

```
> x <- 1;  y <- 1;
> for ( i in c(2:100) ) {
    z <- x + y
    x <- y
    y <- z
  }
> z
```

9.2 while

for 以外に tt while でのループ制御も可能である。



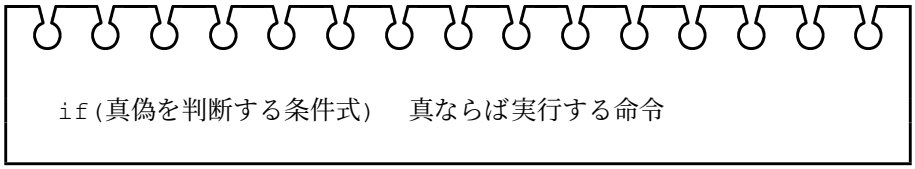
while(真であれば継続する条件式)　実行する命令

たとえば 10000 を超える最初の Fiboncci 数列の項を得るには Fibonacci 数列は次のように計算できる。

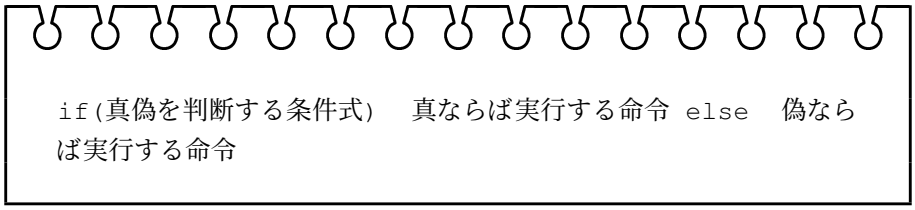
```
> x <- 1;  y <- 1;
> while ( z<=10000 ) {
    z <- x + y
    x <- y
    y <- z
  }
> z
```

と、z が 10000 以下の場合にループを継続させる。

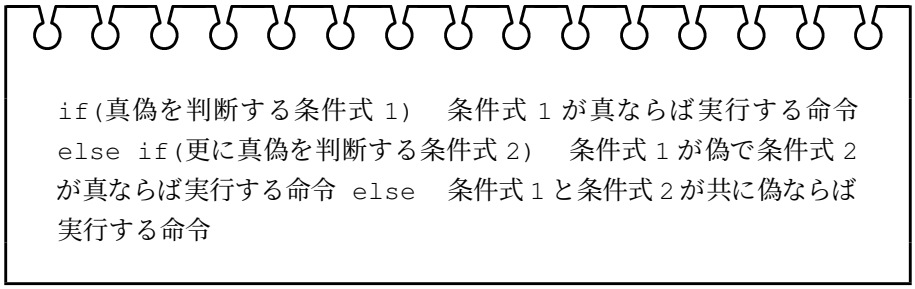
9.3 if



if(真偽を判断する条件式)　真ならば実行する命令



if(真偽を判断する条件式)　真ならば実行する命令 else　偽ならば実行する命令



if(真偽を判断する条件式 1)　条件式 1 が真ならば実行する命令 else if(更に真偽を判断する条件式 2)　条件式 1 が偽で条件式 2 が真ならば実行する命令 else　条件式 1 と条件式 2 が共に偽ならば実行する命令

10　酔歩問題

今回は、確率的シミュレーションの導入として、酔歩問題を取り上げる。元の言葉は「Random Walk」で、日本語の本や文献でもランダムウォークと記されることも少なくない。直訳すると乱歩となりそうだが、酔歩とされた理由は次の通り。

近頃はあまり見掛けないが、人がお酒を飲み過ぎるとひどく酔う。体内にアルコールが入ると、胃から 2 割ほど、残りが小腸から吸収されて、肝臓で分解される。肝臓では、アルコール脱水素酵素によってアセトアルデヒドに変えられ、これが酔いの原因となる。アセトアルデヒドは更に、アセトアルデヒド脱水素酵素により酢酸（アセテート）になり、これが血液に運ばれて体外への排出ルートに乗る。

アセトアルデヒドがある状態では、気分の高揚・消沈、眠気、言語機能の低下、頭痛、平衡感覚の消失、記憶の欠落など、様々な障害が発生する。その状態の人は、酔っぱらい、と呼ばれる。アルコールが飲めない人は、アセトアルデヒドの分解能力が弱いか無い。それでも飲むと強くなったように酔いが軽くなるのは、アセトアルデヒド脱水素酵素が増えたり、分解能力が上がるのではなく、

話が逸れたが酔歩問題は、歩平衡感覚が低下している酔っぱらいの話。次に述べる、ある意味で完全な酔っぱらい、を考える。

空間 酔っぱらいは原点を出発し、二次元平面上を歩く。

歩み 酔っぱらいが足を踏み出す方向は、現在向いている方向とは全く無関係で、しかも完全にランダム。

歩幅 1 .

数式で定義すれば、$\mathcal{R}^2$ 上の確率ベクトル $\mathbf{X}_1$, $\mathbf{X}_2$, ... が独立に同一の分布に従い、$\mathrm{E}[X_i] < \infty$ を満たすとき、$\mathcal{S}_0 = \mathbf{0}$, $\mathcal{S}_n = \sum_{i=1}^{n} \mathbf{X}_i$ と置いて、$\{\mathcal{S}_n : n \geq 0\}$ を酔歩と呼ぶ、となる。

もう少し、具体的に表現すると、次のとおり。

空間 二次元座標を用意し、原点 $(0, 0)$ から出発する。

歩み 毎歩の足を踏み出す角度 θ は、$[0, 2\pi)$ の一様分布に従う。

歩幅 歩幅は 1 なので $\mathbf{x}_{i+1} = \mathbf{x}_i + (\cos\theta, \sin\theta)$。

そして、今日のお題二つめ。

Q2. 酔歩

Q2.1 最初の 1000 歩分の酔歩の軌跡を、何枚か描き、特徴を述べよ。

Q2.2 原点から出発して 6000 歩（1 分に 100 歩として 1 時間分）歩いてから、原点からの距離を計算するプログラムを書き、1000 回実行した結果の距離のヒストグラムを描きなさい。また、2 時間分と 3 時間分のシミュレーションもそれぞれ 1000 回ずつ実行し、結果を比較せよ。

11　ヒント

シミュレーションをやり直すたびに、座標の初期化は忘れないこと。

```
> x <- 0
> y <- 0
```

歩みをすべて記録しておくには、R でも最初に配列は定義しないで良い。まずは、次のようにする。

```
> step.x <- 0
> step.y <- 0
> x <- 0
> y <- 0
```

$[0, 2\pi)$ の一様分布に従う疑似乱数は、次の命令で発生できる。

```
> theta <- runif(1,0,2*pi)
```

歩幅が 1 なので、三角関数で毎歩を積み上げていけば良い。

```
> x <- x+cos(theta)
> y <- y+sin(theta)
```

歩みを記録していくには、次のようにベクトルを宣言し直せば良い。

```
> step.x <- c(step.x, x)
> step.y <- c(step.y, y)
```

これを 1000 回繰り返すコードを書けば、1000 歩分の記録が得られる。グラフに表示するには、

```
> plot(step.x, step.y)
```

とすれば良い。タイトル等を加えるには、

```
> plot(step.x, step.y, xlab="X", ylab="Y",
+      sub="Random Walk")
```

で得られる。

12 参考書の紹介

以下の四冊を挙げておく。

1. 「S 言語 I」 R. A. Becker, J. M. Chambers, and A. R. Wilks 著、渋谷政昭, 柴田里程訳、共立出版、1992.
2. 「S によるデータ解析」 渋谷政昭、柴田里程著、共立出版、1992.
3. 「S と統計モデル」 J. M. Chambers and T. J. Hastie 編、柴田里程訳、共立出版、1994.
4. 「工学のためのデータサイエンス入門」間瀬茂、神保雅一、鎌倉稔成、金藤浩司著、数理工学社、2004.

出版から 15 年近く経った現在でも「S 言語 I」は S 全般について、初心者に向けても網羅的に書かれている良い書籍である。特に R についての網羅的な参考書が存在しないので、この書籍は特に際だつ。入門書を一冊購入するとしたら、「S によるデータ解析」は講義で用いた教材をまとめてあり、適当な難易度に留まっている。

A S言語とR言語の生い立ち

Rの位置づけは若干難しく、まずはじめにSありき、であった。

- ``S: An Interactive Environment for Data Analysis and Graphics''という1984年出版の書籍の翻訳が日本で「Sシステム I」「Sシステム II」として二巻組の書籍として出版されたのが、1987年のこと。
- ``The New S Language: A Programming Environment for Data Analysis and Graphics''という1988年出版の書籍の翻訳が日本で「S言語 I」「S言語 II」という二巻組の書籍として出版されたのが、1991年のこと。
- ``Statistical Models in S''という1992年出版の書籍の翻訳が「Sと統計モデル」として出版されたのが、1994年のこと。

これらのうち、「S言語 I」「S言語 II」「Sと統計モデル」の三冊に記述されている文法とライブラリが1990年代半ばまでのS言語である。この時点でS言語は、統計計算、シミュレーション、データ分析に必要な大抵の機能を備え、一応の完成を見ているが、SAS、SPSSといった今でもポピュラーな他のデータ分析ソフトウェアと比較して、GUIが貧相という弱点を持っていた。1990年代前半に、S言語の文法に基づいて、S言語を発展させてきたAT&Tのベル研究所とは独立に、同じ文法を持つ処理系の開発が始まっていた。それが、Rである。S言語の方はその後、インタフェースの強化とオブジェクト指向性が更に強まり、2004年に``Programming with Data: A Guide to the S Language''という書籍とともに大幅に改訂が加えられて、現在に至っている。

Rは言語といっても、Sのプログラムの多くが何の変更もなく動作するので、S言語の異なる処理系で、方言を持つもの、と考えても良いかもしれない。SとRはそれほどまでに似通っている。細かい仕様は異なるし、ソフトウェアとしての内部構造は全く異なるが、動作がかなり似ている。初期にはRはGNU Sと呼ばれていたほどである。両者が一番異なるのは、その「価格」だろう。

S及びその拡張版であるS-PLUSは常に最新の機能がソフトウェアの専門家によって付け加えられており、コードも専門のプログラマによりメンテナンスされていて、最近はデータマイニング規模のデータサイズに対応する機能も加わるなど、発展が続けられているが、少なくとも無料ではない。翻ってRは、統計の専門家が多く参画していて、ソフトウェア構造をあまり急激に改変して要らぬバグを混入させたりしないように慎重にメンテナンスされているが、無料である。この違いは大きい。

無料ならば、コンピュータがあれば、一次配布サイトのhttp://www.r-project.org/から最新版を入手して、どこにでもインストールして使用することができる。最近はRプロジェクトもたくさんの研究者を惹き付けることに成功し、GUIも含めて活発に拡張が行われるようになったが、文法やメモリ管理などコアな部分は未だに慎重に扱われている。

私は1990年頃にS言語から出発し、1994年頃にS-PLUSも使うようになり、2000年頃にRを使うようになったので、SでもS-PLUSでもRでも動くコードしか書けない。そのため、処理系としてはRを用いるが、タイトルにSとRとした。ただし以下に記す内容は、R言語独自の機能を少なからず含むため、Sで動作することを必ずしも保証しないことはお断りしておく。