認知情報科学基礎実習　R Programming 2

Outline

1. 復習
2. Scriptとfunction
3. 良いと思われるプログラミングの習慣
4. 課題

1.復習　―　中央極限定理の計算機実験 [重要度２]

実習１：以下の１〜３をRで実装してください。解答例は下にあります。

[1] 1つのデータセットに一様分布に従う乱数を10個生成しその平均値を計算しましょう。

[2] [1]を10000回繰り返してください。

[3] [2]の結果を可視化してください。

ヒント

[1]はrunif()が使用できます。

[2]はforが使用できます（備考：実際にforを使わない、より効率的な方法がありますがそれは後日）

[3]はデータ解析基礎論の可視化の資料を参考にしてください。

解答・実装例

N = 10

M = 10000

means = rep(0, M)

for (i\_rep in 1:M){

dat <- runif(N)

means[i\_rep] <- mean(dat)

}

hist(means, xlab = "Estmated means", probability = T, breaks = 30, xlim =c(0,0))

x.temp = seq(0, 1, length.out = 1000)

dens <- dnorm(x.temp, mean = 0.5, sd = (1/sqrt(12\*N)))

lines(x.temp, dens, col = "red", lwd = 3)



2. Script（スクリプト）とfunction（関数）[重要度３]

Scriptは今まで皆さんがRで書いてきた（一連の）コマンドです（下の表のRCMD）。functionは一連のコマンドを１つのコマンドにまとめたものです。他のRのオブジェクトと同様にfunctionの名前(下のfun\_name)は任意に決められます。まとめ方はfunction(…){ RCMD}のようにfunctionという名Rのコマンドを用いて”{“ と “}”でRCMDを囲みます。function(…)の”…”部分は引数と呼ばれるもので関数の入力になります。

|  |  |  |
| --- | --- | --- |
|  | Script | Function |
| 形式 | RCMD | fun\_name <- function(…){  RCMD  } |
| 具体例 | X = 1  Y = 10  sumXY = X + Y | summation <- function(X,Y){  sumXY = X + Y  return(sumXY)  }  備考：return(Z)でZを関数の出力とする |
| 実行例  1)X = 1, Y = 10  2)X = 5, Y = 2  3)X = 3, Y = 1 | > X = 1  > Y = 10  > sumXY = X + Y  > sumXY  [1] 11  > X = 5  > Y = 2  > sumXY = X + Y  > sumXY  [1] 7  > X = 3  > Y = 1  > sumXY = X + Y  > sumXY  [1] 4 | > summation <- function(X,Y){  + sumXY = X + Y  + return(sumXY)  + }  > summation(X = 1, Y = 10)  [1] 11  > summation(X = 5, Y = 2)  [1] 7  > summation(X = 3, Y = 1)  [1] 4 |

ここでは非常に簡単なRCMDを用いていて、実際には関数に移行する必要ないように思いますが、実行例を見てわかるように引数となっているXとYを頻繁に変更する場合は、関数としておくと利便性が高くなると思います。

* Script
  + 再利用が「面倒」
  + 変数・定数を変更するのが「面倒」
  + すべての変数・定数がworkspaceに残る
  + Errorのチェックがしやすい（簡単なscriptでは）
* Function
  + 再利用が簡単
  + 変数・定数を引数（関数の入力：input argument）にすると変更が簡単
  + 出力のみがworkspaceに残る
  + Error のチェックには工夫が必要

備考：青色は利点、赤色は欠点を示します。

まずはscriptでプログラム（の一部）を書いて想定どおりに動くか確認することを勧める（特に慣れない内）。しかし、同様の一連のコマンドを3回以上使用するのであれば、利便性が高くなるためscriptを基にfunctionに移行することを勧める。

実習２：実習１の解答（解答例）を基にN（1回のサンプルのデータの数）,M（サンプリングの繰り返し回数）を引数とする関数を作成してください。解答例は次のページにあります。

解答例２

|  |  |
| --- | --- |
| Script | Function |
| N = 10  M = 10000  means = rep(0, M)  for (i\_rep in 1:M){  dat <- runif(N)  means[i\_rep] <- mean(dat)  }  hist(means, xlab = "Estmated means",  probability = T, breaks = 30,  xlim =c(0,0))  x.temp = seq(0, 1, length.out = 1000)  dens <- dnorm(x.temp, mean = 0.5,  sd = (1/sqrt(12\*N)))  lines(x.temp, dens, col = "red", lwd = 3)  legend("topright","theoretical desiity",  lwd =3, col = "red") | CLT\_example <- function(N, M){  means = rep(0, M)  for (i\_rep in 1:M){  dat <- runif(N)  means[i\_rep] <- mean(dat)  }  hist(means, xlab = "Estmated means",  probability = T, breaks = 30,  xlim =c(0,0))  x.temp = seq(0, 1, length.out = 1000)  dens <- dnorm(x.temp, mean = 0.5,  sd = (1/sqrt(12\*N)))  lines(x.temp, dens, col = "red", lwd = 3)  legend("topright","theoretical desiity",  lwd =3, col = "red")  } |
|  | # 実行方法  CLT\_example(50,10000) |

解答例２を見るとスクリプトと関数ではほぼ同じものです。異なるものは、重要な部分をfunction{}でくくり、NとMを関数の引数としている箇所のみです。

3. 良いと思われるプログラミングの習慣 [重要度２]

プログラムを書く前

変数や定数（パラメターなど）を全て書き出す。

言葉・式で関係や構造を示す。

擬似コードの作成する（擬似コードの例は後述します）。

プログラムを書くとき

　変数は具体的な名前を用いる。

　コメントを入れる

　・Rでは“#”を以降のコマンドは実行されないので、コメントを入れたい場合は“#”を挿入

　・本人がかいたコードでも３ヶ月後に見るとチンプンカンプンのことも多々あります。

　引数や出力をfunctionの初めにコメントしておく

　効率化と読み易さのtrade-offを見極める

　・直感的に読みやすいコードは非効率であることもあるので、見極めが必要です。

プログラムを書いた後

複数のパラメターでプログラムの挙動を確認する。

結果を可視化する。

　完成祝いをする（ただし未成年者はお酒を飲んではいけません）。

3.1 具体例　伝染病の流行モデル ―新型コロナウィルスについての考察

**

伝染病のモデル

x(t):時間tにおける身感染者

y(t):時間tにおける感染者

z(t):時間tにおける脱感染者

感染のメカニズム

・未感染者数の推移

1

2

3

4

はごく短期間の時間の変化で、その時間内で変化した（感染者）の変化量はです。

は未感染者と感染者が接触する係数で、は未感染者と感染者が接触した場合に感染する係数です。

上記の１〜３をまとめたものが４です。

・感染者数の推移

5

6

は脱感染係数です。感染者数は未感染者が感染した分（式４や式５の右辺の第１項）だけ増加します。一方で、脱感染した分だけ減少します（式５の右辺の第２項）。

・脱感染者数の推移

7

8

脱感染者数は感染者が脱感染しただけ増加します。

3.1.1. 変数・定数および変数間の関係構造

|  |  |
| --- | --- |
| 変数・定数 | 関係構造 |
| * x:　未感染者 * y:　感染者 * z:　脱感染者 * t:　時間 * dt:　時間の変化の幅 * x0,y0,z0: x,y,zの初期値 * a1:　xとyが出会う割合 * a2:　xとyが出会い、xがyに影響される割合 * beta= 流行を終える割合 | x,y,zのそれぞれの（毎dt分の）変化を以下に示す  ・xについて  　　xはa1の割合でyと遭遇する：a1\*x\*y  　　遭遇したらa2の割合でxは感染す：a2\*(a1\*x\*y)  　　減少人数　＝ a2\*(a1\*x\*y)  　　a1とa2をまとめるてalphaとするとalpha\*x\*y  ・yについて  　　増加人数（x→yになった人数）＝ a2\*(a1\*x\*y)  　　減少人数（脱感染y→z）＝beta\*y・  ・zについて  　　脱感染（yの内beta分増加）＝ beta\*y |

3.1.2. 擬似コード　― その１

|  |  |
| --- | --- |
| 擬似コード | ２〜４についての詳細 |
| 1. 初期化 2. 特定の期間繰り返す 3. xを変化させる 4. yを変化させる 5. zを変化させる 6. １へ戻る 7. 結果を可視化する | 2. xについて   * 1. xはa1の割合でyと遭遇する      1. a1\*x\*y   2. 遭遇したらa2の割合でxは感染する      1. =2\*(a1\*x\*y)   3. 減少人数　 a2\*(a1\*x\*y)   4. a1とa2をまとめるてalphaとすると   5. 減少人数　＝ lpha\*x\*y  1. yについて    1. 増加人数（x→yになった人数）alpha\*x\*y    2. 減少人数（脱感染→z）beta\*y 2. Zについて    1. 脱感染（yの内beta分増加） beta\*y |

3.1.2. 擬似コード　― その２

1. 初期化

# x0,y0,z0, それぞれ未感染者、感染者、脱感染者の初期値

# dt 時間の変化量

# T　繰り返し回数

# alpha,beta, それぞれ感染係数、脱感染係数

# {x,y,z}の推移を記録するベクトルもしくは行列

1. 時間分繰り返す
2. x(t+1)=x(t)-(alpha\*x(t)\*y(t))\*dt
3. y(t+1)＝y(t)+(alpha\*x(t)\*y(t)-beta\*y(t))\*dt
4. z(t+1)＝ z(t)+(beta\*y(t))\*dt
5. １へ戻る
6. 結果を可視化する

3.1.3. 実装例

x0 = 1e5 #　未感染者数の初期値

y0 = 10 #　感染者数の初期値

z0 = 0 #　脱感染者数の初期値

dt = 0.01 #　時間の変化量（適当に決めました）

T = 10000 #　繰り返し回数

alpha = 1e-5 #　感染係数

beta = 0.1 #　脱感染係数

x = y = z = rep(0,T) #　未感染者数、感染者数、脱感染者数の推移を記録するベクトル

x[1] = x0 #　未感染者数の初期値を記録

y[1] = y0 #　感染者数の初期値を記録

z[1] = z0 #　脱感染者数の初期値を記録

#　メインのコマンド

for(t in 1:(T-1)){

x[t+1] = x[t] - (alpha\*x[t]\*y[t])\*dt

y[t+1] = y[t] + (alpha\*x[t]\*y[t]-beta\*y[t])\*dt

z[t+1] = z[t] + (beta\*y[t])\*dt

}

#　結果を可視化

plot(x=1:T, x, type = "l", col = 2, lwd = 3, ylim = c(0,x0))

lines(x=1:T, y, col = 3, lwd = 3)

lines(x=1:T, z, col = 4, lwd = 3)

legend("right", c("not infected", "infected","no longer infected"), col=2:4,lwd=3)



4.課題

3.1.3.の実装例を基に伝染病モデルの関数を作成してください。引数は少なくともx0, y0, z0, alpha, beta, Tとしてください（その他の定数・パラメターも引数にしてもらって結構です）。

「接触の割合を８割減らすと新型コロナの問題をコントロールできる」のようなことがメディアで紹介されていました。今回のモデルを用いて比較・検証してみましょう（alphaを0.2倍にする）。備考：3.1.3の実装例のパラメターは適当に決めましたし、モデルとしては単純なので、このモデルやシミュレーションの結果は余り参考にしないでください。