データ解析基礎論a1 講義01

**資料の見方**：原則不要な情報は記載していないつもりです。ただし、本来は重要な情報ですが、データ解析基礎論a1~a2においては「気にしなくても良い」情報も含まれています。オンライン講義の性質上、そのニュアンスが明確でない可能性もあるため、資料内に重要度１〜３に区分します。

　重要度１：データ解析基礎論a1~a2においては「気にしなくても良い」レベル

　重要度２：知っておいて欲しいレベル

重要度３：中学＆高校の先生なら、「ここテストに出るぞ！」いうレベル

**注意点**：できるだけ正確な資料を提供するつもりですが、松香は日本での教育は中学までしか受けていないので、誤字・脱字や「てにをは」の誤用が予想されます。

1. Rの初歩的な演算　[重要度３]

1.1 四則演算＋

|  |  |
| --- | --- |
| 演算の種類 | Rの実行例 |
| 加算 例：1+2+3 | > 1+2+3 [1] 6 |
| 減算例：3-2-1 | > 3-2-1　[1] 0 |
| 乗算例：1×2×3 | > 1\*2\*3　[1] 6 |
| 除算例：$10÷2$ | > 10/2 [1] 5  |
| 除算の商例：$10÷3=3,余1$ | > 10%/%3 [1] 3 |
| 除算の余り例：$10÷3=3,余1$ | > 10%%3 [1] 1 |
| べき乗例：24 | > 2^4 [1] 16 |

余白メモ　（ただし重要度３）

* 関数：何か入力すると何かを出力してくれるコマンド
	+ 必ず「( )」を用いる (例：plot(x, y)は作図する関数)
* 引数：関数に入力するもの
	+ 例：plot(x, y)のxとyは引数と呼ばれる

1.2 論理演算（右辺と左辺の比較）

|  |  |
| --- | --- |
| 演算の種類 | Rの実行例 |
| 等号 $(≡)$ 例：AとBは等しいか | > 1==1 [1] TRUE> 1==2[1] FALSE |
| 等号否定$(\ne )$例：AとBは等しくないか | > 1 != 1[1] FALSE> 1!= 2 [1] TRUE |
| 不等号例：AはBより大きいか例：AはB以上か | > 2 > 1[1] TRUE> 1 >= 1[1] TRUE |
| 除算例：$10÷2$ | > 10/2 [1] 5  |
| 除算の商例：$10÷3=3,余1$ | > 10%/%3 [1] 3 |
| 除算の余り：$10÷3=3,余1$ | > 10%%3 [1] 1 |
| べき乗例：24 | > 2^4 [1] 16 |

1. Rでベクトルの生成　[重要度３]

2.1 combine – c

 複数のデータ(数字等)を1つのオブジェクトとして結合する関数

使用方法：

c(var1,var2,...,varN)

Rの実行例

# [1,2,3,4]をxとする

> x<-c(1,2,3,4)

> x

[1] 1 2 3 4

# x2の生成する際、最初の要素は先に生成したxとしそれに[5,6,7,8]を加える

> x2<-c(x,5,6,7,8)

> x2

[1] 1 2 3 4 5 6 7 8

# 文字が要素となる「ベクトル」も生成できる

> y=c('a0','a1','b0','b1')

> y

[1] "a0" "a1" "b0" "b1”

# ただし、1つのベクトル内では複数のデータの種類（数値、文字など）は混在できない

# 下の例では、数値であったxが文字（"で囲まれるもの）として認識されている

> z=c(x,y)

> z

[1] "1" "2" "3" "4" "a0" "a1" "b0" "b1"

2.2 sequence - seq

 等間隔の数列を作るための関数

使用方法：

seq(start, end, increment/decrement)

seq(start, end, length.out = *N*)

Rの実行例

#　間隔が決まっている場合

> x<-seq(1,4,1)

 [1] 1 2 3 4

> x<-seq(0,40,10)

[1] 0 10 20 30 40

> x<-seq(10,2,-2)

[1] 10 8 6 4 2

# ただし、間隔が１もしくは−１の場合は「：」を用いることができる

> x<-1:4

[1] 1 2 3 4
> x<-10:1

[1] 10 9 8 7 6 5 4 3 2 1

#　ベクトルの要素の数が決まっている場合

#　1から４を５等分

> x <- seq(1, 4, length.out = 5)

> x

[1] 1.00 1.75 2.50 3.25 4.00

#　0から1を５等分

> x <- seq(1, 0, length.out = 5)

> x

[1] 1.00 0.75 0.50 0.25 0.00

2.3 replication - rep

 数列の生成を繰り返しする関数

使用方法：

rep(x, times)

# １を４回繰り返す

> x<-rep(1,4)

[1] 1 1 1 1

#　[1,7,87]のベクトルを３回繰り返す

> x<-rep(c(1,7,87),3)

[1] 1 7 87 1 7 87 1 7 87

#　1:4のseqを３回繰り返す

> x<-rep(1:4,3)

[1] 1 2 3 4 1 2 3 4 1 2 3 4

2.4 実習1

以下のようなベクトルを作りましょう (実行例が次のページにあります)

* [1, 2, 3, 4, 5, 4, 3, 2, 1]
* [1, 4, 9, 16, 25]
* [2, 4, 8, 16, 32, 64, 128]

2.4.1 実習１の実行例

> j1 = c(1:5,4:1)

> j2 = (1:5)^2

> j3 = 2^(1:7)

> j1

[1] 1 2 3 4 5 4 3 2 1

> j2

[1] 1 4 9 16 25

> j3

[1] 2 4 8 16 32 64 128

1. Rで行列・データの生成　[重要度３]

3.1 行列の生成

使用方法：matrix(vector, nrow, ncol)

nrow × ncol の行列の作成(備考通常はべクトルの長さは一定なので、nrow か ncol だけで足りる。その場合どちらか一方のみを定義すればよい。Rの他のコマンド・関数も全ての引数を明示する必要でないことが多い。その場合、初期値・初期設定が適用される。)

Rの実行例

> x<-matrix(1:8, nrow=2)

 [,1] [,2] [,3] [,4]

[1,] 1 3 5 7

[2,] 2 4 6 8

# Rの初期設定では１列目から、列ごとに数値を入力する。行ごとに入力したい場合は「byrow ＝TRUE」とする。

> x<-matrix(1:8, nrow=2,byrow=T)

 [,1] [,2] [,3] [,4]

[1,] 1 2 3 4

[2,] 5 6 7 8

3.2 データフレーム

Matrixはベクトルと同じで数値と文字などが混在できない。

data.frameでは変数内（列内）では数値、文字などが混在できないが、変数間（異なる列）では数値、文字などが混在できる。

使用方法：data.frame(varName1=vector1, varName2 = vecotr2, …. )

varNameとは変数の名前で任意のものが使用できる

Rの実行例

data01<-data.frame(score = c(2,4,3,4),

 dose = c(rep(10,2),rep(100,2)),

 condition = rep(c('exp','control'),2))

> data01

 score dose condition

1 2 10 exp

2 4 10 control

3 3 100 exp

4 4 100 control

3.3 実習2

以下のようなデータフレームを作りましょう (実行例がすぐ下にあります)

|  |  |  |
| --- | --- | --- |
| subjectID | Month | Succeeded |
| 1 | april | Yes |
| 2 | april | No |
| 3 | april | Yes |
| 4 | april | No |
| 5 | april | Yes |
| 1 | may | No |
| 2 | may | Yes |
| 3 | may | No |
| 4 | may | Yes |
| 5 | may | No |

3.3.1 実習2の実行例

> dat <- data.frame(subjID = rep(1:5,2),

 Month = c(rep("april",5),rep("may",5)),

 Succeeded = rep(c("Yes","No"),5))

> dat

 subjID Month Succeeded

1 1 april Yes

2 2 april No

3 3 april Yes

4 4 april No

5 5 april Yes

6 1 may No

7 2 may Yes

8 3 may No

9 4 may Yes

10 5 may No

3.4 コマンドによるデータファイルの入力 read.csv

使用方法：read.csv(“FILE\_NAME”, header=?, row.names=N)

FILE\_NAME はディレクトリやURLも含む。

変数の名前がある場合は header=T、無い場合は header=F とする。

Csvファイルとはテキストファイルで変数をコンマで区切ったものを指します。

Rの実行例

# <http://www.matsuka.info/data_folder/>　にあるtemp\_data01.txtという名前の

#ファイルを入力す。temp\_data01.txtを確認してみてください。１行目に変数の名前があ

#ります。

dat01<-read.csv("http://www.matsuka.info/data\_folder/temp\_data01.txt",

 header=T)

> dat01

 x y z

1 11 12 13

2 21 22 23

3 31 32 33

# 各行に名前がある場合は now.names=NでN列目を観測点の名前として使うこともできる。

# <http://www.matsuka.info/data_folder/temp_data02.txt>を確認してみてください。

dat02<-read.csv("http://www.matsuka.info/data\_folder/temp\_data02.txt",

 header=T, row.name=1)

> dat02

 x y z

katsuo 11 12 13

wakame 21 22 23

tarachan 31 32 33

3.5 コマンドによるデータファイルの入力 read.table

使用方法：read.table(“FILE\_NAME”,header=?,sep=‘delim’,row.names=N)

基本的にread.csvと同じですが、区切り方法を指定できる。具体的にはsep = でdelimiter （区切りを示す文字など）を定義する(「tab」の場合は不要。tabは一般的には空のスペースなので表示されない）。例えば delimiter が「;」の場合、sep=”;”となる

# <http://www.matsuka.info/data_folder/temp_data03.txt>を確認してみてください。表示されていませんが、変数間にはtabが埋め込まれています。

> dat03<-read.table("http://www.matsuka.info/data\_folder/temp\_data03.txt",

 header=T, row.name=4)

> dat03

 x y z

sazae 11 12 13

masuo 21 22 23

tarachan 31 32 33

3.6 変数・列の参照（参照のことをindexingとも言います）

参照方法（データフレームまたは行列の場合）

　M[参照したい行、参照したい列]

> dat03

 x y z

sazae 11 12 13

masuo 21 22 23

tarachan 31 32 33

#1行、1列目を参照

> dat03[1,1]

[1] 11

# 列を問わず、すべてのn行目を参照したい場合：

M[n, 空]

#行を問わず、すべてのm列目を参照したい場合：

#２行目を参照

> dat03[2,]

 x y z

masuo 21 22 23

#１列目を参照

> dat03[,1]

[1] 11 21 31

#列に名前（変数名）がある場合

M$varName

> dat03$x

[1] 11 21 31

> dat03$y

[1] 12 22 32

> dat03$z

[1] 13 23 33

#変数名を変えたい場合

使用方法： colnames(dat03)<-c("newName1"," newName2",…)

> colnames(dat03)<-c("kinou","kyou","ashita")

> dat03

 kinou kyou ashita

sazae 11 12 13

masuo 21 22 23

tarachan 31 32 33

以上で講義１は終わりです。講義１の課題はありません。