```
################################
################################
####### [LIST] リスト ############
################################
################################

#--------------------------
# [LIST.1] リストの生成
#--------------------------

# c関数で生成されるアトミックベクトルは、
# 全て同じ型であることが前提
a = c(13,TRUE,"hello") #異なる型のものを入れると強制的に同一の型へと変換
#[1] "13"    "TRUE"  "hello"

# aの型は「character」
typeof(a)
#[1] "character"

# list関数でつくられるリストであれば、
# 異なる型を1つの変数にグループとしてまとめることができる。
a = list(13,TRUE,"hello")
#[[1]]
#[1] 13
#[[2]]
#[1] TRUE
#[[3]]
#[1] "hello"

# aの型とクラスはともに「リスト」
typeof(a)
#[1] "list"
class(a)
#[1] "list"

# 個々の要素の型はtypeof([[n]])でわかる。
# （R.2で詳しく説明します）
typeof(a[[1]]);typeof(a[[2]]);typeof(a[[3]])
#[1] "double"
#[1] "logical"
#[1] "character"



#--------------------------------
# [LIST.2] リスト[[n]]とリスト[n]
#--------------------------------

a = list(13,TRUE,"hello")

# リストのn番目の要素はリスト[[n]]で表します。
a[[1]];a[[2]];a[[3]]
#[1] 13
#[1] TRUE
#[1] "hello"

# リストのn番目の要素がベクトルの場合、
# m番目のサブ要素は[[n]][m]で表します。
```

```
a[[1]][1];a[[2]][1];a[[3]][1]
#[1] 11
#[1] TRUE
#[1] "hello"

# リスト[n]は、リスト[[n]]を要素として持つ「サイズ1」のリストを返します。
a[1]
#[[1]]
#[1] 13

# 以下の違いに注意。
typeof(a[[1]])
#[1] "double"
typeof(a[1])
#[1] "list"
typeof(a[1][[1]])
#[1] "double"


# リストの要素が全てアトミックベクトルの例
a = list(11:13,c(T,F,F,F),"good-bye")
#[[1]]
#[1] 11 12 13
#[[2]]
#[1]  TRUE FALSE FALSE FALSE
#[[3]]
#[1] "good-bye"

a[[1]][2];a[[2]][4]
#[1] 12
#[1] FALSE

#aの三番目の要素はアトミックベクトル
a[[3]]
#[1] "good-bye"

# a[3]は、a[[3]]を要素に持つサイズ1のリスト（復習）
a3 = a[3]
#[[1]]
#[1] "good-bye"

typeof(a3);length(a3)
#[1] "list"
#[1] 1



# リストはリストを要素に加えることができます。
a = list(list(T,c("hello","good-bye")),c(T,F,F,F),101:200)

# リストのリストは[[p]][[q]]..という形で掘っていく
a[[1]] #aの第一要素もリスト構造
#[[1]]
#[1] TRUE
#[[2]]
#[1] "hello"    "good-bye"

# aの第１要素の第２要素はアトミックベクトル
```

```
a[[1]][[2]]
#[1] "hello"    "good-bye"

# "good-bye"は
# a（リスト）の第１要素（リスト）の第２要素（ベクトル）の第２要素
a[[1]][[2]][2]
#[1] "good-bye"



# くどいですが、再び、リストとアトミックベクトルの違いの復習

a = c("hello","good-bye")
b = list("hello","good-bye")

# "hello"の参照は？
a[1];b[[1]][1]
#[1] "hello"
#[1] "hello"

# "good-bye"の参照は？
a[2];b[[2]][1]
#[1] "good-bye"
#[1] "good-bye"



#---------------------------
# [LIST.3] $記法（名前属性）
#---------------------------


# リストの各要素には名前属性をつけることができます。

chubu = list(name = c("AICHI","GIFU","MIE","SHIZUOKA"), #県の名前
             pop = c(716,211,186,379), #人口
             order = c(4,17,22,10), #人口順位
             capital = c("nagoya","gifu","mie","shizuoka"), #県庁所在地
             shinkansen = c(T,T,F,T)) #新幹線通過の有無

# 人口にアクセスするには次の2通り存在する
chubu[[2]] #二重括弧記法
#[1] 716 211 186 379
chubu$pop #$記法
#[1] 716 211 186 379

#愛知県の県庁所在地は、
chubu$capital[1]
#[1] "nagoya"

#岐阜県の人口順位は
chubu$order[2]
#[1] 17

#三重県の新幹線通過の有無は、
chubu$shinkansen[3]
#[1] FALSE
```

```
#静岡県の人口は
chubu$pop[4]
#[1] 379



#-------------------------------------
#（LIST.4）リストを出力する関数の例
#-------------------------------------


#先週の授業の例
#n個のサイコロを振って、総和を得るプログラム
roll.n = function(n){
  dice = 1:6
  dice2 = sample(dice, size=n, replace = TRUE)
  # dice2 = sample(dice,n,T) #こちらでも可
  sum(dice2)
}

# サイコロを3回振った総和の1000サンプルのベクトルです。
sample = replicate(1000,roll.n(3))
#[1] 11 11 10  9 12 10 10 12 10  9  9 13 13 10 12 14
#[17] 16  5  8 12 10 12  5 15  9  6  3 16  9 10  5  9
#省略
#[977] 10  6 12  9 11  8 15 11 11  6  8  9 11 14 11 11
#[993] 10 17  8 10 12  8 13  7

# ヒストグラムを計算し、Plotsパネルにグラフを出力します。
hi = hist(sample)

# hiはhistogramクラスのリストです。
typeof(hi);class(hi)
#[1] "list"
#[1] "histogram"

# 右上のEnvironmentパネルで「hi」を開くと
# 6つの名前属性（$breraks,$counts,$density,$mids,$xname,$equidist）
# の存在が確認できます。

# 総和サンプルの一覧
hi$breaks
#[1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
hi[[1]] #リストなので二重括弧を使うこともできます。
#[1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18

#個々のサンプル数（例です）
hi$counts #hi[[2]]でも同じです。
#[1]  12  37  44  69 108 118 119 128 122  99  58  43  22
#[14]  18   3



#---------------------------------
#（LIST.5）論理値記法
#---------------------------------

# 簡単なベクトルを例に論理値サーチの方法を見ていきます。
```

```
# まず1から20までの整数列をつくります。
n = 1:20
#[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

# 同じサイズの論理値のベクトルを使うと特定の要素だけ取り出すことができます。
# こちらは奇数部分だけを取り出す例
n[c(T,F,T,F,T,F,T,F,T,F,T,F,T,F,T,F,T,F,T,F)]
#[1]  1  3  5  7  9 11 13 15 17 19
n[c(T,F)] #リサイクル規則
#[1]  1  3  5  7  9 11 13 15 17 19
n[c(T,F,F)] #リサイクル規則
#[1]  1  4  7 10 13 16 19

# c(T,F,..)というベクトルは以下の方法でも作れます。
n %% 2 == 1
#[1]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
FALSE  TRUE FALSE
#[15]  TRUE FALSE  TRUE FALSE  TRUE FALSE

# これを使えば、よりスマートに、特定の要素を取り出すことが可能です。
n[n %% 2 == 1]
#[1]  1  3  5  7  9 11 13 15 17 19
n[n %% 3 == 1]
#[1]  1  4  7 10 13 16 19

# 1000から100個のサンプルを抽出し、そのうち10で割り切れるもののみを取り出す。
n = sample(1000,100)
n[n %% 10 == 0]
#[1] 610 920 890 200 470 940



#ここからリストに戻ります。
roll.n = function(n){
  dice = 1:6
  dice2 = sample(dice,n,T)
  sum(dice2)
}
sample = replicate(1000,roll.n(3))
hi = hist(sample)

# hi$XXXはベクトルのため、同じ探索方法が適用できます。

# 総和10のサンプル数は？
# （後述の論理値サーチを参照）
hi$counts[hi$breaks==10]
# [1] 128

# 総和10以上の各サンプル数は？
hi$counts[hi$breaks>=10]
#[1] 128 122  99  58  43  22  18   3  NA

# 総和10以上のサンプル数の総和は、
sum(hi$counts[hi$breaks>=10]) #欠損値があるとNAになります
#[1] NA

# NAを無視して総和を計算（na.rmは、NAをremoveと覚える）
sum(hi$counts[hi$breaks>=10],na.rm=T)
```

```
#[1] 493

# 総和10以上の確率密度は？
sum(hi$counts[hi$breaks>=10],na.rm=T)/sum(hi$counts)
#[1] 0.493



####################################
####################################
####### [DF] データフレーム #########
####################################
####################################


#-----------------------------------
# [DF.1] データフレームの生成
#-----------------------------------

# いよいよデータフレームを扱います。
# データフレームはリストの一種ですが、
# よりデータ分析に特化したリストと考えることができます。

# （LIST.3）で作った中部地方のリストのコンストラクタの関数部分を、
# listからdata.frameに変更します。

chubu = data.frame(name = c("AICHI","GIFU","MIE","SHIZUOKA"), #県の名前
             pop = c(716,211,186,379), #人口
             order = c(4,17,22,10), #人口順位
             capital = c("nagoya","gifu","mie","shizuoka"), #県庁所在地
             shinkansen = c(T,T,F,T)) #新幹線通過の有無

# chubuの中身を見ると、データが綺麗に整列されています。
chubu
#      name pop order  capital shinkansen
#1    AICHI 716     4   nagoya       TRUE
#2     GIFU 211    17     gifu       TRUE
#3      MIE 186    22      mie      FALSE
#4 SHIZUOKA 379    10 shizuoka       TRUE

# 型はリストのまま、クラスがdata.frameとなる。
typeof(chubu);class(chubu)
#[1] "list"
#[1] "data.frame"

# DF[n]は、n番目の要素をデータフレーム形式で返します。
chubu[2]
#pop
#1 716
#2 211
#3 186
#4 379

chubu[4]
#capital
#1   nagoya
#2     gifu
#3      mie
```

```
#4 shizuoka


# データフレームは、全ての要素がアトミックベクトルで
# かつ、それらの要素数が同一でないとエラーが返されます。
# 例えば、以下の場合、shinkansenの要素数のみが3で揃わないためエラーとなります。
chubu = data.frame(name = c("AICHI","GIFU","MIE","SHIZUOKA"), #県の名前
                   pop = c(716,211,186,379), #人口
                   order = c(4,17,22,10), #人口順位
                   capital = c("nagoya","gifu","mie","shizuoka"), #県庁所在
地
                   shinkansen = c(T,T,F)) #新幹線通過の有無
#Error in data.frame(name = c("AICHI", "GIFU", "MIE", "SHIZUOKA"), pop =
c(716,
#:arguments imply differing number of rows: 4, 3







#----------------------------
# [DF.2] ij記法 （DF[i,j]）
#----------------------------

chubu = data.frame(name = c("AICHI","GIFU","MIE","SHIZUOKA"), #県の名前
                   pop = c(716,211,186,379), #人口
                   order = c(4,17,22,10), #人口順位
                   capital = c("nagoya","gifu","mie","shizuoka"), #県庁所在
地
                   shinkansen = c(T,T,F,T)) #新幹線通過の有無
chubu
#      name pop order  capital shinkansen
#1    AICHI 716     4   nagoya       TRUE
#2     GIFU 211    17     gifu       TRUE
#3      MIE 186    22      mie      FALSE
#4 SHIZUOKA 379    10 shizuoka       TRUE

# DF[i,j]で、データフレームを行列と見立てた時の
# i行目j列目の要素を取り出すことができます。

# 三重県の人口順位（3行3列目）
chubu[3,3] #対応するベクトル要素を返します。
#[1] 22

# 名前属性を使った$記法も、リスト要素の記法も使えます。
chubu$order[3];chubu[[3]][3]
#[1] 22
#[1] 22

# ij記法の拡張
# jの要素数が複数の場合、データフレームを返します！！
chubu[3,c(1,3)] #三重の県名と人口順位（3行の1列目と3列目）
#  name order
#3  MIE    22

chubu[1:3,c(1,5)] #静岡以外の名前と新幹線情報
#   name shinkansen
```

```
#1 AICHI       TRUE
#2  GIFU       TRUE
#3   MIE      FALSE

chubu[-4,c(1,5)] #上と同じです。（4行目を除外）
#   name shinkansen
#1 AICHI       TRUE
#2  GIFU       TRUE
#3   MIE      FALSE

chubu[3,] #三重県の全て
#  name pop order capital shinkansen
#3  MIE 186    22     mie      FALSE

chubu[,4] #全ての県の県庁所在地
#[1] "nagoya"   "gifu"     "mie"      "shizuoka"
# jの要素数が1つなので、ベクトルが返ります。

# !! 1つの行を選択すると,データフレームを返す一方で、
# !! 1つの列を選択すると,アトミックベクトルを返すことに注意

# 順番の入れ替え
chubu[c(2,1,3,4),]
#      name pop order  capital shinkansen
#2     GIFU 211    17     gifu       TRUE
#1    AICHI 716     4   nagoya       TRUE
#3      MIE 186    22      mie      FALSE
#4 SHIZUOKA 379    10 shizuoka       TRUE

# 列指定で順番の入れ替え
chubu[c(2,1,3,4),2] #人口を愛知・岐阜・三重・静岡の順で
#[1] 211 716 186 379
# 列のサイズが1なので、やはりベクトルが返ります。

# 行も列も順番の入れ替え
chubu[c(2,1,3,4),c(4,1)]
#capital     name
#2     gifu     GIFU
#1   nagoya    AICHI
#3      mie      MIE
#4 shizuoka SHIZUOKA
# 列のサイズが2なので、データフレームで返します。



#---------------------------------
# [DF.3] 名前による指定、$記法
#---------------------------------

chubu = data.frame(name = c("AICHI","GIFU","MIE","SHIZUOKA"), #県の名前
                   pop = c(716,211,186,379), #人口
                   order = c(4,17,22,10), #人口順位
                   capital = c("nagoya","gifu","mie","shizuoka"), #県庁所在
地
                   shinkansen = c(T,T,F,T)) #新幹線通過の有無

# 列の名前属性を、ij記法のjに直接に指定することができます。
```

```
# jのサイズが複数の場合、データフレームとして返ります。
chubu[1,c("name","pop")] #愛知県の名前と人口
#   name pop
#1 AICHI 716
chubu[1,c("pop","name")] #名前の順番は無関係
#   name pop
#1 AICHI 716
chubu[,c("name","pop")] #全ての県の名前と人口
#      name pop
#1    AICHI 716
#2     GIFU 211
#3      MIE 186
#4 SHIZUOKA 379

# jのサイズが1の場合、ベクトルが返ることに注意。
chubu[,c("pop")] #全ての県の人口
#[1] 716 211 186 379



# データフレームはリストの一種ですので、
# [LIST.3]で紹介した$記法を適用できます。

chubu$capital #全ての県の県庁所在地、ベクトルを返します。
#[1] "nagoya"   "gifu"     "mie"      "shizuoka"

chubu$shinkansen #全ての県の新幹線状況（ベクトル）
#[1]  TRUE  TRUE FALSE  TRUE

chubu$pop #全ての県の人口（ベクトル）
#[1] 716 211 186 379

sum(chubu$pop) #総和
#[1] 1492

mean(chubu$pop) #平均
#[1] 373

# ほとんどの関数は、引数にアトミックベクトルをとりますが、
# リストを引数にとることはできません。
# そのため、データフレームのデータをベクトルに変換する
# $記法は、 非常に重宝します。


mean(chubu[2]) #エラー（chubu[2]はベクトルではなくリスト!!）
#[1] NA
#Warning message:
#  In mean.default(chubu[2]) :
#  argument is not numeric or logical: returning NA

mean(list(c(716,211,186,379))) #同じくエラー
#[1] NA
#Warning message:
#  In mean.default(chubu[2]) :
#  argument is not numeric or logical: returning NA

# 二重格好でベクトルに変換すれば、一般的な関数の引数にとれます。
mean(chubu[[2]])
```

```
#[1] 373
```

```
#-------------------------
# [DF.4] 論理値による探索
#-------------------------

chubu = data.frame(name = c("AICHI","GIFU","MIE","SHIZUOKA"), #県の名前
                   pop = c(716,211,186,379), #人口
                   order = c(4,17,22,10), #人口順位
                   capital = c("nagoya","gifu","mie","shizuoka"), #県庁所在
地
                   shinkansen = c(T,T,F,T)) #新幹線通過の有無

# (LIST.5)で扱った論理値による探索は
# データフレームで大活躍します。


chubu[1,c(T,T,F,F,F)] #愛知県の名前と人口のみ
#   name pop
#1 AICHI 716
chubu[c(T,F,F,F),c(1,2)] #同じです。
#   name pop
#1 AICHI 716


#  人口順位のみをベクトルとして取り出す
chubu$order
#[1]  4 17 22 10

#  人口順位10番以内の県をサーチ（真偽値ベクトルが返ります）
chubu$order<=10
#[1]  TRUE FALSE FALSE  TRUE

#  条件を満たす行をデータフレームとして取り出す
chubu[chubu$order<=10,]
#      name pop order  capital shinkansen
#1    AICHI 716     4   nagoya       TRUE
#4 SHIZUOKA 379    10 shizuoka       TRUE

# 条件を満たす県をベクトルで取り出す
# 以下は、人口トップ10の県を取り出す例
chubu[chubu$order<=10,1]
#[1] "AICHI"    "SHIZUOKA"
chubu[[1]][chubu$order<=10]
#[1] "AICHI"    "SHIZUOKA"
chubu$name[chubu$order<=10]
#[1] "AICHI"    "SHIZUOKA"


# 新幹線の通らない県の人口順位
chubu$order[chubu$shinkansen==F]
```

```
#[1] 22

# 新幹線の通る県の人口順位
chubu$order[chubu$shinkansen==T]
#[1]  4 17 10




#-----------------------------------------
# [DF.5] CSVファイルからデータフレームを読み取り
#-----------------------------------------

# 以下から、2020年度入学学生の仮配属情報を入手してください。
# http://lab.kenrikodaka.com/_download/class/2022_AppliedMedia/imd2050.csv

# CSVファイルは以下のようなフォーマットになっています。
# （実際は、EXCELファイルから書き出したものです。）
# 一行目に名前属性、二行目以降に各データが並んでいます。

#id,name,sex,lab,department,class
#205001,ISHIKAWA,F,KAMINUMA,imd,F
#205002,ISONO,F,HANAWA,imd,T
#205003,IDA,F,KOBAYASHI,imd,F
#205004,IJICHI,F,KAMINUMA,imd,F
#...

# 右上のEnvironmentパネルから、Import Dataset（From Text (base)）を選びます。
# 以下の要領でcsvファイルをデータフレームとして読み込んでください。

#Name：imd2050（入力してください）
#Encoding:Automatic
#Heading:Yes
#Row names:Automatic
#Separator:Comma
#Decimal:Period
#Quote：Double
#Coomen:None
#na.stfings:NA
#Strings as factors:Checked（チェックを入れてください）


#2020年度にimd入学で仮配属が済んでいる学生のリストです。
#id（学籍番号）・name（苗字）・sex（性別）
#lab（配属先）・department（配属研究室）
#class（この授業を受講しているか否か）

#最初の6行をちょっと出し
head(imd2050)

#      id     name sex       lab department class
#1 205001 ISHIKAWA   F  KAMINUMA        imd FALSE
#2 205002    ISONO   F    HANAWA        imd  TRUE
#3 205003      IDA   F KOBAYASHI        imd FALSE
#4 205004   IJICHI   F  KAMINUMA        imd FALSE
#5 205005 ICHIHARA   F  KURIHARA        imd  TRUE
#6 205006      ITO   F    KODAKA        imd FALSE
```

```
imd2050$name[imd2050$class==T]
#[1] ISONO    ICHIHARA ITO      KAGAMI   KANO     YOSHIDA  YOSHIDA
#28 Levels: FUKUTOMI ICHIHARA IDA IJICHI ISHIKAWA ISONO ITO IWASAKI ...
YOSHIDA

# 配属研究室が産業の学生グループ（データフレーム）
imd2050[imd2050$department=="iid",]
#id   name sex       lab department class
#8  205008 IWASAKI   F TSUJIMURA        iid FALSE
#17 205019  TAJIMA   M TAKAHASHI        iid FALSE

# 配属研究室が産業の学生の名前
imd2050$name[imd2050$department=="iid"]
#[1] "IWASAKI" "TAJIMA"

# 配属研究室が産業の研究室のリスト
imd2050$lab[imd2050$department=="iid"]
#[1] TSUJIMURA TAKAHASHI
#Levels: HANAWA KAMINUMA KOBAYASHI KODAKA KURIHARA MIZUNO NAKAGAWA
TAKAHASHI TSUJIMURA


# ブール演算子
# A & B（AかつBが真のとき真）
# A | B（AまたはBが真のとき真）
# xor(A,B)（AとBのうち1つだけが真のとき真）
# !A（Aが偽のとき偽）
# any(A,B,C,...)（いずれかが真のとき真）
# all(A,B,C,...)（いずれも真のとき真）

# 配属研究室が小鷹研でこのクラスをとっている学生
imd2050$name[imd2050$lab=="KODAKA" & imd2050$class==T]
#[1] "KAGAMI"

# 男の吉田の学籍番号
imd2050$id[imd2050$name=="YOSHIDA" & imd2050$sex=="M"]
#[1] 205032

# このクラスをとっていない塙研の学生（そんな学生はいない）
imd2050$name[imd2050$lab=="HANAWA" & imd2050$class==F]
#character(0)


# ある条件を満たす総数を調べる方法
# 例えば、このクラスをとっている学生の総数を数えます。

# まず、このクラスをとっている学生を真偽値ベクトルをつくります。
imd2050$class==T
#[1] FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE
TRUE
#[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
FALSE
#[25] FALSE FALSE FALSE FALSE  TRUE  TRUE

# 真偽値ベクトルに含まれるTRUEの数はsumが使えます。
sum(imd2050$class==T)
#[1] 7
```

```
# このクラスをとっている学生の割合は、（例えば）以下のように算出できます。
sum((imd2050$class==T)) / (sum(imd2050$class==T) + sum(imd2050$class==F))
#[1] 0.2333333
```