R

コード資料

lab.kenrikodaka (2024)

R
第１講

アトミックベクトル

```
#（2024年版）

################################################
################################################
####### [BASIC] 基礎（主にベクトル） ##########
################################################
################################################


#-----------------------------------
# [BASIC.1] 変数と代入
#-----------------------------------

# コメントアウトは#を行頭につける

# 変数aに1を代入
# イコールも矢印もどちらも使えます。
a<-1
a=1
# Rでは矢印が一般的ですが、この授業ではイコール記法を採用します。

# 実行する行を選択し、「⌘RETURN」でコードを実行
# コンソールに以下のように実行結果が表示されます
# >[1] 1
a
#[1] 1

# ピリオドを変数の表記に使うことも可（先頭は不可）
A.a = 0.4; A.b = 0.8
A.a; A.b #セミコロンを使うと一行に複数の文を並べることができます。
#[1] 0.4
#[1] 0.8

# 大文字と小文字は区別します
name = -1; Name = -2
name+1;Name+1
#[1] 2
#[1] 3


#-----------------------------------
# [BASIC.2] 型
#-----------------------------------


#https://qiita.com/maruman029/items/365a2abcdaaf99b720be

a = 1 #整数はデフォルトではdouble型として扱われます
b = 0.2 #double型
c = TRUE; C = T #logical型（T、F表記も可能）
d = 1L #整数型（integer型）として扱いたい時はLをつける
e = "Hello" #character型

# データ型はtypeofで調べる
typeof(a);typeof(b);typeof(c);typeof(d);typeof(e)
#[1] "double"
#[1] "double"
#[1] "logical"
#[1] "integer"
#[1] "character"

# 型変換にはas.xxxを使う
a = 1; typeof(a)
#[1] "double"
```

```
b = as.integer(a) #double型からinteger型に変換
b;typeof(b);
#[1] 1
#[1] "integer"
c = as.character(a) #double型からcharacter型に変換（1は"1"に変換されます）
c;typeof(c)
#[1] "1"
#[1] "character"

# numericはintegerとdoubleを合わせた型（mode関数で確認可能）
mode(a);mode(b)
#[1] "numeric"
#[1] "numeric"



#-----------------------------------
# [BASIC.3] 一次元ベクトルの生成
#-----------------------------------

#------
##(3.1) 一次元ベクトルを生成する方法
#------

# c(n1,n2,n3)
# cはcombineに由来

a=c(1,2,3)
#[1] 1 2 3
a=c(T,T,F)
#[1] T T F
a=c("T","T","F") #文字列の配列
#[1] "T" "T" "F"

# c(a:b)で一次元配列ベクトル（a,a+1,,b）を作成
a=c(101:105)
#[1] 101 102 103 104 105
a=c(9:1); # b<aのとき（a,a-1, .. b）
#[1] 9 8 7 6 5 4 3 2 1
a = c(-3:9999333)
#[1]  -3  -2  -1   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
16  17  18  19  20  21
#[26]  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
41  42  43  44  45  46
#（以下略）
length(a) #ベクトルの長さはlength関数で取得可能
#[1] 9999337

# c(a,b)の引数が非整数の場合がある。
# 初項a、公差1で、b以下のものを集めたベクトルを返す
a=c(2.3:4)
#[1] 2.3 3.3

# 「:」記法の場合, cは省略可能
a=101:105
#[1] 101 102 103 104 105
a=9:1
#[1] 9 8 7 6 5 4 3 2 1
a=2.3:4
#[1] 2.3 3.3

a = seq(1,10,length=5) #1~10を5分割
#[1] 1.00  3.25  5.50  7.75 10.00
a= seq(1,10,by=2) #1から2ずつ増やして10を越える手前まで
#[1] 1 3 5 7 9
```

```
a = rep(1:3,times=3) #(1,2,3)を3回
#[1] 1 2 3 1 2 3 1 2 3
a = rep(1:3,length=5) #(1,2,3)を要素数5となるまで繰り返す
#[1] 1 2 3 1 2


# 無作為サンプリング関数（sample）を使う
a = sample(1:6,1) #サイコロを1回振る
#[1] 5
#[1] 3
a = sample(1:6,6) #1~6から無作為サンプルを6回（重複なし）
#[1] 4 6 5 1 3 2

a = sample(1:6,7) #エラー（母数よりサンプル数が多い）
a = sample(1:6,7,replace = TRUE) #replaceをTRUEにセットすると重複ありモードへ
# [1] 6 1 2 6 5 6 5

# 空のベクトルを作成
a=vector("integer") #integer型の空のベクトルを生成（中身は空）
#integer(0)
a=vector("logical") #logical型の空のベクトルを生成（中身は空）
#logical(0)
a=vector("integer",10) #引数2はベクトルの長さ（初期値は0）
#[1] 0 0 0 0 0 0 0 0 0 0
a=vector("logical",10) #logical型の場合、初期値はFALSE
#[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE


#------
##(3.2)　アトミックベクトルの性質
#------

# Rでは、全てのメンバが同じ型を持つタイプのベクトルを
# アトミックベクトルと呼ぶ。

# 後で示すように、
# アトミックベクトルは一次元ベクトルとは限りません。

# アトミックベクトルかどうかを確認する
a1=2:5; is.atomic(a1)
#[1] TRUE

# リストはアトミックベクトルでは無い（次週学びます）
# （参考）リストは複数の型を扱います
a2=list(2:5,"R"); is.atomic(a2)
#[1] FALSE

# データフレームはアトミックベクトルでは無い（次週学びます）
a3=data.frame(c(1,2),c("KODAKA","KENRI")); is.atomic(a3)
#[1] FALSE

# （参考）is.vectorは、名前以外の属性を持つかを調べます。
is.vector(a1);is.vector(a2);is.vector(a3)
#[1] TRUE
#[1] TRUE
#[1] FALSE

# アトミックベクトルの型はtypeofで調べる。
a=c(T,T,F); b=c(101:105)
typeof(a); typeof(b)
#[1] "logical"
#[1] "integer"
```

```
#  異なる型を入れた場合、適当な型に型変換されます。
#  以下で、Tと102は文字列型（character）に型変換されます。
a=c(T,102,"hello"); typeof(a)
#[1] "character"

a[1];a[2];a[3]
#[1] "TRUE"
#[1] "102"
#[1] "hello"

# 数値も要素1のベクトルです。
a=2;is.atomic(a)
#[1] TRUE
a[1]
#[1] 2


#------
##(3.3) ベクトルの結合
#------

a1 = c(101:105)
a2 = c(a1,201,202) #a1に2つの要素を追加
#[1] 101 102 103 104 105 201 202

# append関数を使って結合する
a = c(101:105)
b = c(301:305)
append(a,b) #aとbを結合
#[1] 101 102 103 104 105 301 302 303 304 305
append(a,b,after=2) #3番目の要素から挿入
#[1] 101 102 301 302 303 304 305 103 104 105

a1 = c(101:105)
a2 = c(201,a1,202) #a2の2番目の要素にa1を追加
#[1] 201 101 102 103 104 105 202

# この場合a1の要素数は7となり、
# 例えばa1の3番目要素はa2の4番目の要素となります。
a2[4]
#[1] 103

# 多次元配列の様な構造を持つことはできません。
# よって以下の様な書き方をしても、103とはなりません。
a2[2][3]
#[1] NA

# 要するに一次元ベクトルの要素に
# 別の一次元ベクトルを配置すると、
# 新しいサイズの一次元ベクトルが作られます。

a1 = c(101:105)
a2 = c(201,a1,202)
a3 = c(301,a2,302);
#[1] 301 201 101 102 103 104 105 202 302


#------
##(3.4) ベクトルの要素抽出・置換
#------

# n番目の要素を取り出す
a=101:105
a[3]
#[1] 103
```

```
# n番目の要素を書き換える
a=101:105; a[3] = 3;  a;
#[1] 101 102 3 104 105

a=101:105
a[c(2,4)] #2番目と4番目の要素を取り出してベクトルにする
#[1] 102 104
a[c(-2,-4)] #2番目と4番目の要素を取り除く
#[1] 101 103 105
a[4:8] #範囲外はNAを要素とする
#[1] 104 105  NA  NA  NA

# 特定の条件を満たす要素のみを取り出す時は
# which()関数で、引数に条件を記述する
a=101:120
which(a-10==105) #10を引いて105となる要素の添字を取り出す
#[1] 15

# 添字が存在しない場合、空の整数ベクトルが返される
which(a-10==120) #10を引いて120となる要素の添字を取り出す
#integer(0)

a[which(a-10==105)] #aの要素のうち、10を引いて105となる要素を取り出す
#[1] 115
a[which(a-10==120)] #aの要素のうち、10を引いて120となる要素を取り出す
#integer(0) #存在しない

# 整数商は「%/%」、割った余りは「%%」
# 例えば, 17わる3は、5余り2
17 %/% 3; 17 %% 3
#[1] 5
#[1] 2
# これを使うと、、
a=101:125
a[which(a%%2==0)] #偶数の要素のみを取り出す
#[1] 102 104 106 108 110 112 114 116 118 120 122 124
a[which(a%/%103<1)] #103で割った余りが1より小さい
#[1] 101 102



#-------------------------------------
# [BASIC.4] ベクトルの演算
#-------------------------------------

dice=1:6
#[1] 1 2 3 4 5 6
dice-1
#[1] 0 1 2 3 4 5
dice/2
#[1] 0.5 1.0 1.5 2.0 2.5 3.0
dice * dice
#[1]  1  4  9 16 25 36
dice %% 3 #3で割った余り
#[1] 1 2 0 1 2 0

#  （リサイクル規則）
# ベクトルのサイズが異なる場合の演算は、
# 短い方のベクトルの要素が繰り返し使われます。
x = c(2,2,3,3,4,4)
y = c(0.1,0.2)
z = x+y
```

```
#[1] 2.1 2.2 3.1 3.2 4.1 4.2
z = x*y
#[1] 0.2 0.4 0.3 0.6 0.4 0.8



#-----------------------------------
# [BASIC.5] ベクトルに適用される関数
#-----------------------------------

vec = 1:6
sum(vec) #総和
#[1] 21
mean(vec) #平均
#[1] 3.5
round(mean(vec))#四捨五入
#[1] 4


# ランダムサンプリング（既出）
sample(1:4, 2)
#[1] 2 4
#[1] 4 1

# 引数を明示して関数を実行する
#  （xとsizeは引数の名前として指定）
sample(x=1:4,size=2)
# 名前が無いもの（1:4）は、xとして扱う
# sampleの場合、xが引数のデフォルト値
sample(size=2,1:4)

# 関数の引数の名前（および引数のデフォルト値 + 引数の順序）を調べる
args(sample)
#function (x, size, replace = FALSE, prob = NULL)

# replaceは同一の値のサンプリングを許可するか
sample(1:4, replace=TRUE)
#[1] 1 2 2 3
#[1] 4 3 3 2

# 名前がなければ、2番目の引数はsizeが期待されるためエラー
sample(1:4, TRUE)
#Error in sample.int(length(x), size, replace, prob) :
#  invalid 'size' argument

# 誤った引数の名前はエラーとなる
sample(1:4,saizu=2)
#Error in sample(1:4, saizu = 2) : unused argument (saizu = 2)



#-----------------------------------
# [BASIC.6] 配列
#-----------------------------------

# アトミックベクトルを2行3列の配列に変換
a = array(1:6,dim=c(2,3))
#[,1] [,2] [,3]
#[1,]    1    3    5
#[2,]    2    4    6

is.array(a) #aは配列
#[1] TRUE
is.atomic(a) #aは変わらずアトミックベクトル
#[1] TRUE
```

```
length(a) #長さは6
#[1] 6

a[2,1] #2行1列目の要素
#[1] 2
a[2,3] #2行3列目の要素
#[1] 6

a[4] #a[2,2]に同じ
#[1] 4
a[5] #a[1,3]に同じ
#[1] 5

a[2,] #2行の要素群
#[1] 2 4 6
a[,3] #3列の要素群
#[1] 5 6

a[,c(1,3)] #1列目と3列目を切り出す（配列）
#[,1] [,2]
#[1,]    1    5
#[2,]    2    6


# コマンドの複製（replicate）

a = replicate(3, 1+1) #1+1の結果を3回繰り返しベクトルへ
#[1] 2 2 2
a = replicate(10,sample(1:6,1)) #サイコロの無作為抽出を10回繰り返す
#[1] 4 3 4 6 3 3 5 2 2 3
is.array(a) #aは配列ではなく、単なる（一次元）ベクトル
#[1] FALSE

# 要素数n（>1）のベクトルを返す関数をm回replicateすると、
# 結果はn x mの配列となる
a = replicate(10,sample(1:6,size=2))
#[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#[1,]    2    5    4    2    2    1    6    3    1     6
#[2,]    1    1    6    6    1    2    2    1    2     2
is.array(a) #aは配列
#[1] TRUE
is.atomic(a) #aは変わらずアトミックベクトル
#[1] TRUE
length(a) #長さは20（2x10）
#[1] 20


#-----------------------------------
# [BASIC.7] 繰り返し文・条件文
#-----------------------------------

# 1:6に含まれる個々の要素をaとし,
# aを出力する（print関数）
for(a in 1:6){
  print(a)
}
#[1] 1
#[1] 2
#[1] 3
#[1] 4
#[1] 5
#[1] 6
```

```
# for(a in vec)のaは必ずしも使う必要はない
# 以下はサイコロの無作為抽出を5回繰り返したものの加算を計算したもの
sum = 0
for(a in 1:5){
  sum = sum + sample(1:6,1) #+=の記法はエラー（なぜ？）
  print(sum)
}
#[1] 2
#[1] 7
#[1] 10
#[1] 15
#[1] 18

# サイコロの総和が50以上となるまで加算
sum=0
while(sum<50){
  sum = sum + sample(1:6,1)
  print(sum)
}
#[1] 3
#[1] 5
#[1] 7
#中略
#[1] 35
#[1] 41
#[1] 45
#[1] 50

# repeat関数はwhile(true)に相当
# breakが実行されるまで永遠に実行する

sum = 0
repeat{
  sum = sum + sample(1:6,1)
  if(sum>50){
    break
  }
  print(sum)
}

#[1] 4
#[1] 10
#[1] 13
#中略
#[1] 41
#[1] 46
#[1] 49


#-----------------------------------
# [BASIC.8] 関数定義（独自の関数をつくる）
#-----------------------------------

# 2個のサイコロを振って、総和を得るプログラム
dice = 1:6
dice2 = sample(dice, size=2, replace = TRUE)
sum(dice2)

# これをroll()として関数化する
roll = function(){
  dice = 1:6
  dice2 = sample(dice, size=2, replace = TRUE)
  sum(dice2)
}

# 関数定義の最後の行が返り値となる（returnは使わないことに注意！！）
for(i in 1:3){
```

```
  ret = roll()
  print(ret)
}
#[1] 9
#[1] 10
#[1] 6


# n個のサイコロを振って、総和を得るプログラム
# 引数ありの関数をつくります。
roll.n = function(n){
  dice = 1:6
  dice2 = sample(dice, size=n, replace = TRUE)
  sum(dice2)
}

rolls=vector(mode="integer",length=1000)


result = vector("integer",10) #サイズ10の全ての要素がゼロのベクトル
result = vector(mode="integer",length=10) #同じです
for(i in 1:10){
  result[i] = roll.n(i)
}
result
#[1]  6  8 14 13 19 16 22 30 39 38 #（EXAMPLE）



#roll関数を10000回繰り返し、一次元ベクトルに格納
rolls = replicate(1000, roll())
#[1]  6  7  9  6  9  4  3  9  8  4  7 10  8  8  7  9  8 10  7  8  8  6
#[23]  4 12  5  5 12  7 10  6  9  5  5  5 11  4 12  7  8 10  7 11  7  6
#[45]  5 12  8 11 10  7  5 11  9  5  9  7  5  4  8  6 10  9  5  6  9  9
#...
#[991]  6  6  7  7 10 10  5  9  9 12

#for文で書き直しています（同じことです）。
rolls=vector("integer",1000) #integer型の空のベクトルを生成
for(i in 1:1000){
  rolls[i] = roll()
}


#ヒストグラムの計算（roll関数を使います）
imax = 10000
rolls = replicate(imax, roll())

hi=vector("integer",12) #中身が0の長さ12のベクトルをつくる
for(i in 1:imax){
  hi[rolls[i]] = hi[rolls[i]]+1
}
#[1] 0  28  53  85  93 140 180 146 112  89  54  20

# 棒グラフで可視化（barplot）
barplot(hi, main="Double dice",names.arg=1:12, ylab="Frequency")
# mainはタイトル
# names.argはx軸の値
# ylabはy軸のラベル

# hist関数を使うと、サンプルベクトルから自動でヒストグラムを生成できる
hist(rolls,breaks=0:13,xlim=c(0,12),ylim=c(0,2000))
# breaksは区切りのベクトル
# xlim/ylimはx/yの範囲
```

R
第２講

リスト・データフレーム

```
#（2024年版）

################################
################################
####### [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


# a[1]は、11:13（a[[1]]）を要素に持つサイズ1のリスト（復習）
a1 = a[1]
#[[1]]
#[1] 11 12 13

typeof(a1);length(a1)
#[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,breaks=1:20)

# 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]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
hi[[1]] #リストなので二重括弧を使うこともできます。
#[1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

#個々のサンプル数（例です）
hi$counts #hi[[2]]でも同じです。
#[1]   0   4  10  26  54  73 111 109 124 129 106 101  60  43  32  16
2   0   0



#---------------------------------
#（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))

#breaksは区切りのライン
#2.5~3.5, 3.5~4.5, 4.5~5.5, ...17.5~18.5をサンプリング
hi = hist(sample,breaks=2.5:18.5)

# hi$XXXはベクトルのため、同じ探索方法が適用できます。

# hi$midsは、各ビンの中央のサンプル値
hi$mids
#[1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
hi$counts
#[1]   7  10  35  53  51 105 114 106 138 119  97  66  60
#[14]  23  15   1

# 総和10のサンプル数は？
#  （後述の論理値サーチを参照）
hi$counts[hi$mids==10]
# [1] 106

# 総和10以上の各サンプル数は？
hi$counts[hi$mids>=10]
#[1] 106 138 119  97  66  60  23  15   1
```

```
# 総和10以上のサンプル数の総和は、
sum(hi$counts[hi$mids>=10]) #欠損値があるとNAになります
#[1] 625

# 総和10以上の確率密度は？
sum(hi$counts[hi$mids>=10])/sum(hi$counts)
#[1] 0.625



####################################
####################################
####### [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")] #名前を入れ替えると順番も変わる
#  pop  name
#1 716 AICHI
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

# 二重括弧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、メジャーリーグの各年の成績データが入手できます。
# https://baseballsavant.mlb.com/leaderboard

# 以下は、上記から2021〜2022年の成績をCSVとしてDLしたものです。
# バッターの成績：2021 - 2022
# http://lab.kenrikodaka.com/_download/class/2023_AppliedMedia/
mlb.b.2021-22.csv
# ピッチャーの成績：2021 - 2022
# http://lab.kenrikodaka.com/_download/class/2023_AppliedMedia/
mlb.p.2021-22.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。
# 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。
# last_name,...,year,player_age,...home_run,...
# Frazier,..,2021,29,...,5,...
# Altuve,...,2021,31,...,31,...
# Gallo,....,2021,38,...,38,...
# Sano,.....,2021,30,...,30,...
# ...

# last_name（ラストネーム）,first_name（ファーストネーム）,
# player_id（id）,year（成績年）,player_age（年齢）,
# ab（打席数）,hit（安打数）,single（単打）,double（二塁打）,triple（三塁
打）,
# home_run（本塁打）,strikeout（三振）,walk（四球）,
# k_percent（三振率）,bb_parcent（四球率）,
# batting_avg（打率）,on_base_percent（出塁率）,on_base_plus_slg（OPS）
# wOBA（打撃貢献指標）, X（?）

## CSVファイルのインクルード
###（1）ローカルファイルから

# 作業ディレクトリを指定します。（以下は例です。）
setwd("/Users/kenrikodaka/Dropbox/DocClass/_2023/B3_情報処理応用")
# メニューからも指定できます（Session → ChooseDirectory）
```

```
# setwdをしておくと、データを読み込むときに楽になります。
# 以下のようにカレントディレクトリからの相対パスでデータを指定できます。
# read.csvを用いると、csvデータをデータフレームとして読み込むことができます。
datb = read.csv("_dat/mlb.b.2021-22.csv") #打撃成績
datp = read.csv("_dat/mlb.p.2021-22.csv") #投手成績


###（2）インターネットから直接

# バッターの成績：2021 - 2022
url_b = "http://lab.kenrikodaka.com/_download/class/
2023_AppliedMedia/mlb.b.2021-22.csv"
# ピッチャーの成績：2021 - 2022
url_p = "http://lab.kenrikodaka.com/_download/class/
2023_AppliedMedia/mlb.p.2021-22.csv"

datb = read.csv(url_b) #打撃成績
datp = read.csv(url_p) #投手成績


#-------------------------------------------
# [DF.6] 練習（MLBのデータから）
#-------------------------------------------


# 打撃成績データの最初の6行をちょっと出し
head(datb)

#last_name first_name player_id year player_age  ab  pa hit single
#1      Frazier       Adam    624428 2021         29 577 639 176
130
#2       Altuve       Jose    514888 2021         31 601 678 167
103
#3        Gallo       Joey    608336 2021         27 498 616  99
47
#4         Sano     Miguel    593934 2021         28 470 532 105
51
#5 Kiner-Falefa      Isiah    643396 2021         26 635 677 172
136
#6        Edman      Tommy    669242 2021         26 641 691 168
113
#double triple home_run strikeout walk k_percent bb_percent
batting_avg
#1     36      5        5        69   48      10.8        7.5
0.305
#2     32      1       31        91   66      13.4        9.7
0.278
#3     13      1       38       213  111      34.6       18.0
0.199
#4     24      0       30       183   59      34.4       11.1
0.223
```

```
#5     25      3        8        90   28      13.3        4.1
0.271
#6     41      3       11        95   38      13.7        5.5
0.262
#slg_percent on_base_percent on_base_plus_slg  woba  X
#1       0.411           0.368            0.779 0.341 NA
#2       0.489           0.350            0.839 0.357 NA
#3       0.458           0.351            0.809 0.348 NA
#4       0.466           0.312            0.778 0.332 NA
#5       0.357           0.312            0.669 0.293 NA
#6       0.387           0.308            0.695 0.301 NA

# 打率が3割を超えている選手を列挙
datb$last_name[datb$batting_avg>0.3]
#[1] "Frazier"      "Harper"       "Marte"        "Anderson"
"Soto"         "Castellanos"  "Brantley Jr."
#[8] "Riley"        "Reynolds"     "Gurriel"      "Turner"
"Guerrero Jr." "Freeman"      "Alvarez"
#[15] "Benintendi"   "Judge"        "Goldschmidt"  "Bogaerts"
"Lowe"         "McNeil"       "Abreu"
#[22] "Arraez"


# ホームランを40本以上売っている選手の名前を列挙
datb$last_name[datb$home_run>=40]
#[1] "Tatis Jr."    "Ohtani"       "Perez"        "Guerrero Jr."
"Semien"       "Schwarber"    "Judge"
#[8] "Alonso"

# ホームランを40本以上売っている選手の成績を抽出
# （データフレームとして出力されます）
datb[datb$home_run>=40,]
#last_name first_name player_id year player_age
#15     Tatis Jr.   Fernando    665487 2021         22
#27        Ohtani     Shohei    660271 2021         26
#110        Perez   Salvador    521692 2021         31
#125 Guerrero Jr.   Vladimir    665489 2021         22
#131       Semien     Marcus    543760 2021         30
#170    Schwarber       Kyle    656941 2022         29
#189        Judge      Aaron    592450 2022         30
#252       Alonso       Pete    624413 2022         27

# ホームランを40本以上の記録を列挙
# （アトミックベクトルでの出力）
datb$home_run[datb$home_run>=40]
#[1] 42 46 48 48 45 46 62 40

# ホームランを40本以上の選手の名前・成績年・本塁打数を
# データフレームで抽出
datb[datb$home_run>=40,c("last_name","year","home_run")]
#last_name year home_run
#15     Tatis Jr. 2021       42
```

```
#27        Ohtani 2021       46
#110        Perez 2021       48
#125 Guerrero Jr. 2021       48
#131       Semien 2021       45
#170    Schwarber 2022       46
#189        Judge 2022       62
#252       Alonso 2022       40


# 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）


# 再び、打率が3割を超えている選手を列挙
datb$last_name[datb$batting_avg>0.3]
#[1] "Frazier"      "Harper"       "Marte"        "Anderson"
"Soto"         "Castellanos"  "Brantley Jr."
#[8] "Riley"        "Reynolds"     "Gurriel"      "Turner"
"Guerrero Jr." "Freeman"      "Alvarez"
#[15] "Benintendi"   "Judge"        "Goldschmidt"  "Bogaerts"
"Lowe"         "McNeil"       "Abreu"
#[22] "Arraez"

# 2021年に限定（&は論理和）
bat2021 = datb$last_name[datb$year==2021 & datb$batting_avg>0.3]
#[1] "Frazier"      "Harper"       "Marte"        "Anderson"
"Soto"         "Castellanos"  "Brantley Jr."
#[8] "Riley"        "Reynolds"     "Gurriel"      "Turner"
"Guerrero Jr."

# 2022年に限定
bat2022 = datb$last_name[datb$year==2022 & datb$batting_avg>0.3]
#[1] "Freeman"     "Alvarez"     "Benintendi"  "Judge"
"Goldschmidt" "Bogaerts"    "Lowe"
#[8] "McNeil"      "Abreu"       "Arraez"



# 2年連続3割を超えた打者を探します。
# まずはintersectの使い方を覚えます。

# intersectは2つのベクトルの中の共通の要素を取り出します。
intersect(1:50,30:80)
#[1] 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
# 2の倍数と3の倍数の共通要素（0～50）は6の倍数
intersect(seq(0,50,by=2),seq(0,50,by=3))
```

```
#[1]  0  6 12 18 24 30 36 42 48

# 2年連続の3割打者はいない
intersect(bat2021,bat2022)
# character(0)

# 2年連続、2割8部超えは8人（ジャッジ、ボガーツ、ゴールドシュミット他）
tmp2021 = datb$last_name[datb$year==2021 & datb$batting_avg>0.28]
tmp2022 = datb$last_name[datb$year==2022 & datb$batting_avg>0.28]
intersect(tmp2021,tmp2022)
#[1] "Marte"       "Judge"       "Bogaerts"    "Rosario"
"Goldschmidt" "Turner"      "Freeman"
#[8] "Bichette"


# 同様に、2年連続40本以上の選手は？
tmp2021 = datb$last_name[datb$year==2021 & datb$home_run>=40]
tmp2022 = datb$last_name[datb$year==2022 & datb$home_run>=40]
intersect(tmp2021,tmp2022)
# character(0) #（いない）

# 2年連続34本以上の選手は？
tmp2021 = datb$last_name[datb$year==2021 & datb$home_run>=34]
tmp2022 = datb$last_name[datb$year==2022 & datb$home_run>=34]
intersect(tmp2021,tmp2022)
# [1] "Ohtani" "Judge"  "Alonso" "Olson" #（4人いた）

# 関数化します。（BASICの復習）
getHomerunBatter = function(n){
  tmp2021 = datb$last_name[datb$year==2021 & datb$home_run>=n]
  tmp2022 = datb$last_name[datb$year==2022 & datb$home_run>=n]
  intersect(tmp2021,tmp2022)
}

# こんな感じで同じことができますね。
getHomerunBatter(40)
#character(0)
getHomerunBatter(39)
#[1] "Judge"
getHomerunBatter(38)
#[1] "Judge"
getHomerunBatter(37)
#[1] "Judge"  "Alonso"


# 打撃成績にも投手成績にもどちらにも顔を出している選手のIDは？
#  （規定打席と規定投球回数のいずれも満たしている選手です）
intersect(datb$player_id,datp$player_id)
#[1] 660271

#誰だ？
```

```
#名前、成績年、打席数
datb[datb$player_id==660271,c("last_name","year","ab")]
#last_name year ab
#27     Ohtani 2021 537
#216    Ohtani 2022 586

#名前、成績年、投球回数
datp[datp$player_id==660271,c("last_name","year","p_formatted_ip")]
#last_name year p_formatted_ip
#84     Ohtani 2021          130.1
#179    Ohtani 2022          166.0
#(大谷は2021年は規定投球回数満たしてないよね、、)
```

R
第３講

グラフ（ggplot2）の基本

```
#(2024年度版)
#--------------------------------
# [GRAPH.1] いろいろなグラフ
#--------------------------------

#--------------------------------
# (A) 基本的な流れ
#--------------------------------

# ggpplot2ライブラリを使えるようにします
install.packages("ggplot2")
library(ggplot2)

# gcookbookライブラリを使えるようにします
install.packages("gcookbook")
library(gcookbook)

# 例えば以下のようなデータフレームが使えます。
head(heightweight)
#. sex ageYear ageMonth heightIn weightLb
#1   f   11.92      143     56.3     85.0
#2   f   12.92      155     62.3    105.0
#3   f   12.75      153     63.3    108.0
#4   f   13.42      161     59.0     92.0
#5   f   15.92      191     62.5    112.5
#6   f   14.25      171     62.5    112.0

# それではグラフを作っていきましょう。


#（1） エステティックマッピング
## x軸に年齢、y軸に身長、fill属性に性別を割り振る
## この時点では何のグラフも出力されません
gp = ggplot(heightweight,aes(x=ageYear,y=heightIn,fill=sex)); gp

#（2）散布図としてグラフ化する
gp = gp + geom_point(shape=21,size=6); gp

#（3）軸の設定
gp = gp + scale_x_continuous(name="Age",
limits=c(10,20),breaks=c(10,15,20)); gp
gp = gp + scale_y_continuous(name="Height
(inch)",limits=c(0,100),breaks=c(0,50,100)); gp

#（4A）凡例の設定
gp1 = gp + scale_fill_discrete(name = "Gender",
labels=c("Female","Male")); gp1


#（4B）凡例と色の設定

gp1 = gp + scale_fill_grey(name = "Gender", labels=c("Female","Male"),
                           start=0.2, end=0.8); gp1
gp1 = gp + scale_fill_brewer(name = "Gender", labels=c("Female","Male"),
                             palette="Set2"); gp1


#（5）全体の体裁
gp2 = gp1 + theme(
```

```
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-A1



#--------------------------------
# (B) 棒グラフ１：geom_col
#--------------------------------

#---
#デフォルトの出力
#---

install.packages("gcookbook")
# gcookbookのライブラリを使います
library(gcookbook)

# 使用するデータフレーム
BOD
#Time demand
#1    1    8.3
#2    2   10.3
#3    3   19.0
#4    4   16.0
#5    5   15.6
#6    7   19.8

# 基本レイヤを作成
## この時点では何も出力されません
gp = ggplot(BOD,aes(x=Time,y=demand)); gp

# デフォルトの棒グラフ（塗りは黒、幅は0.9）
gp1 = gp + geom_col(); gp1

# 棒グラフの幅を最大幅の半分とする（デフォルトは0.9）
gp1 = gp + geom_col(width=0.5); gp1

# 棒グラフの塗りつぶしを白、枠線を黒とし、枠線のサイズを0.2とする。
gp1 = gp + geom_col(fill="white",colour="black", size=0.2)
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-B1


#---
#属性の要素ごとに分ける方法
#---

library(gcookbook)
cabbage_exp
#. Cultivar Date Weight        sd  n         se
```

```
#1      c39  d16   3.18 0.9566144 10 0.30250803
#2      c39  d20   2.80 0.2788867 10 0.08819171
#3      c39  d21   2.74 0.9834181 10 0.31098410
#4      c52  d16   2.26 0.4452215 10 0.14079141
#5      c52  d20   3.11 0.7908505 10 0.25008887
#6      c52  d21   1.47 0.2110819 10 0.06674995

# CultivarとDateは「Factor」（ラベル付きのinteger）
# 数字でラベル付けされた「文字列」と考えて問題ない
typeof(cabbage_exp$Cultivar)
#[1] "integer"
class(cabbage_exp$Cultivar)
#[1] "factor"

# 順序は以下で確認（Levelsの並びに注目）
cabbage_exp$Cultivar
#[1] c39 c39 c39 c52 c52 c52
#Levels: c39 c52

cabbage_exp$Date
#[1] d16 d20 d21 d16 d20 d21
#Levels: d16 d20 d21


# 基本レイヤを作成
## x軸をData、y軸をWeight、塗りをCultivarにマッピング
gp = ggplot(cabbage_exp,aes(x=Date,y=Weight,fill=Cultivar))

# デフォルトはfillを積み上げる（棒グラフ）
gp1 = gp + geom_col(); gp1
# 引数で「積み上げ」を明示的に指定（factor順序：上から下へ...）
gp1 = gp + geom_col(position = position_stack()); gp1
## 積み上げ順序の反転（factor順序：下から上へ）
gp1 = gp + geom_col(position = position_stack(reverse = TRUE)) ; gp1

# 100%積み上げ棒グラフとする
gp1 = gp + geom_col(position = "fill")
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-B2



# 水平方向に並べる（棒グラフ）
## デフォルトはfill属性同士はぴたりと張り付く
gp1 = gp + geom_col(position = position_dodge()); gp1
# 幅を0.7とし、隙間を0.1だけ空ける。
## この場合、「幅+隙間」をposition_dodgeの引数に与える
gp1 = gp + geom_col(position = position_dodge(0.8),width=0.7); gp1

#隙間を狭める
gp1 = gp + geom_col(position = position_dodge(0.72),width=0.7)
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
```

```
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-B3



# fillではなくinteractionを使っても同じことができます
gp = ggplot(cabbage_exp,aes(x=interaction(Date,Cultivar),
                            y=Weight,fill=Cultivar))
# デフォルトの棒グラフにする
gp1 = gp + geom_col()
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-B4


#---
#エラーバーの作成
#（すでに標準誤差が計算されDFMに書き出されている場合）
#---

# cabbage_expより、Cultivarがc39のものだけ取り出す
library(gcookbook)
cabbage_exp0 = cabbage_exp[cabbage_exp$Cultivar=="c39",]
#  Cultivar Date Weight        sd  n         se
#1      c39  d16   3.18 0.9566144 10 0.30250803
#2      c39  d20   2.80 0.2788867 10 0.08819171
#3      c39  d21   2.74 0.9834181 10 0.31098410

gp = ggplot(cabbage_exp0,aes(x=Date,y=Weight))

# 塗りを白に、枠線を黒に
gp1 = gp + geom_col(fill="white",colour="black") ; gp1
# エラーバーの作成（widthはエラーバーの横幅）
gp1 = gp1 + geom_errorbar(aes(ymin=Weight-se,ymax=Weight+se),width=.2);
gp1

# 横ならびタイプ（dodge）でエラーバーを追加する時
## 棒グラフのデフォルトのずらし幅（0.9）に合わせる。
gp1 = gp + geom_col(fill="white", colour="black", position =
position_dodge()) +
  geom_errorbar(aes(ymin=Weight-se,ymax=Weight+se),
                position=position_dodge(0.9), width=.2)
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-B5
```

```
#--------------------------------
# (C) 棒グラフ２：geom_bar
#--------------------------------

#---
#同じ値のものを数え上げる
#---

library(gcookbook)
# mtcarsの最初の6行を表示
head(mtcars)
#                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
#Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
#Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
#Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
#Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
#Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
#Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1

# 列名cylのデータを確認（cylはシリンダーの数らしい）
mtcars$cyl
#[1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4

# 各要素の数を数える
table(mtcars$cyl)
# 4  6  8
#11  7 14

# 列名cylをカテゴリカル変数に変更し、
# 各要素の個数を数え上げて棒グラフとして表示
ggplot(mtcars,aes(x=factor(cyl))) + geom_bar()
# factorにしない場合、X軸のラベルに5と7も表示されることに注意
ggplot(mtcars,aes(x=cyl)) + geom_bar()

# 数え上げた個数を棒グラフ上に表示
gp1 = ggplot(mtcars,aes(x=factor(cyl))) + geom_bar() +
  geom_text(aes(label=..count..),stat="count",
            vjust=1.5,colour="white", size=10)
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-C1



#---
#属性の要素ごとに分ける方法
#---

# 列名gearのデータを確認（ギアの数）
mtcars$gear
#[1] 4 4 4 3 3 3 3 4 4 4 4 3 3 3 3 3 3 4 4 4 3 3 3 3 3 4 5 5 5 5 5 4

# $cylと$gearの組み合わせ
```

```
table(mtcars[,c("cyl","gear")])
#     gear
#cyl  3  4  5
#  4  1  8  2
#  6  2  4  1
#  8 12  0  2

# 同じ
table(mtcars$cyl,mtcars$gear)
#   3  4  5
#4  1  8  2
#6  2  4  1
#8 12  0  2

# 変数cylと変数gearの組み合わせについて、gearで色分け
ggplot(mtcars,aes(x=interaction(cyl,gear),fill=factor(gear))) + geom_bar()
# 変数cylで色分け
gp1 = ggplot(mtcars,aes(x=interaction(cyl,gear),fill=factor(cyl))) +
geom_bar()
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-C2

# fillにマッピングされた変数（数値）をファクタにしない場合、
# 連続値として扱われ、カラーバーで色付けされることに注意
ggplot(mtcars,aes(x=interaction(cyl,gear),fill=gear)) + geom_bar()
ggplot(mtcars,aes(x=interaction(cyl,gear),fill=cyl)) + geom_bar()

# interactionを使わない方法の例
## デフォルトは積み上げ
ggplot(mtcars,aes(x=cyl,fill=factor(gear))) + geom_bar()
ggplot(mtcars,aes(x=gear,fill=factor(cyl))) + geom_bar()
## 引数にposition=position_dodge()で横並びにする
ggplot(mtcars,aes(x=cyl,fill=factor(gear))) +
geom_bar(position=position_dodge())

gp1 = ggplot(mtcars,aes(x=gear,fill=factor(cyl))) +
geom_bar(position=position_dodge())
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-C3

#---------------------------------
# (D) 折れ線グラフ：geom_line
#---------------------------------

#---
#基本
#---

library(gcookbook)
```

```
BOD
#Time demand
#1    1    8.3
#2    2   10.3
#3    3   19.0
#4    4   16.0
#5    5   15.6
#6    7   19.8

# 基本レイヤの出力
gp = ggplot(BOD,aes(x=Time,y=demand))

# デフォルトのグラフ
gp + geom_line()

# 線の色を深緑にして、線の太さを1.5とする。
gp + geom_line(colour = "dark green",size=1.5)

# 線を点線とする。
gp + geom_line(linetype="dashed")

# xをファクタにすると、カテゴリカルな系列と解釈され、
# x軸の6が取り除かれます。
## aes(group=1)は、Timeが同じグループに属していることを
## 明示しています。（無いとエラーとなります）
gp = ggplot(BOD,aes(x=factor(Time),y=demand,group=1))
gp1 = gp + geom_line()
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-D1

#---
#属性の要素ごとに分ける方法
#---

library(gcookbook)
tg
#. supp dose length
#1   OJ  0.5  13.23
#2   OJ  1.0  22.70
#3   OJ  2.0  26.06
#4   VC  0.5   7.98
#5   VC  1.0  16.77
#6   VC  2.0  26.14

# 基本レイヤの作成
gp = ggplot(tg,aes(x=dose,y=length,colour=supp))

# suppの要素（OJ、VC）ごとに、色分けして出力
gp + geom_line()

# 線の描画をグループ間で左右に0.2だけずらす
gp + geom_line(position = position_dodge(0.2))

# 同じ位置に点を重ねる
```

```
gp + geom_line(position = position_dodge(0.2)) +
  geom_point(position=position_dodge(0.2),shape=21,size=5,fill="white")

# suppをlinetypeでマッピング
gp1 = ggplot(tg,aes(x=dose,y=length,linetype=supp)) + geom_line()
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-D2


#---
#エラーバーの描画
#---

library(gcookbook)
cabbage_exp
#. Cultivar Date Weight        sd  n         se
#1      c39  d16   3.18 0.9566144 10 0.30250803
#2      c39  d20   2.80 0.2788867 10 0.08819171
#3      c39  d21   2.74 0.9834181 10 0.31098410
#4      c52  d16   2.26 0.4452215 10 0.14079141
#5      c52  d20   3.11 0.7908505 10 0.25008887
#6      c52  d21   1.47 0.2110819 10 0.06674995

# ドッジの設定を変数に保存
pd = position_dodge(0.3)

# Cultivarの要素に基づき、折れ線グラフを分けて描画
# 折れ線に関して、Cutivarでグループ分けすることを
# 明示的に伝える必要がある（colourをerrorbarでも使っているため）
gp = ggplot(cabbage_exp,
            aes(x=Date,y=Weight,colour=Cultivar,group=Cultivar))
# エラーバーの描画
gp1 = gp + geom_errorbar(
  aes(ymin=Weight-se,ymax=Weight+se),
  width=.2,size=0.25,colour="black",position=pd) +
  geom_line(position=pd) +
  geom_point(position=pd,size=5)
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-D3



#--------------------------------
#  (E) 散布図：geom_point
#--------------------------------

#---
#基本
```

```
#---

library(gcookbook)

# heightweightの最初の6行
head(heightweight)
#. sex ageYear ageMonth heightIn weightLb
#1   f   11.92      143     56.3     85.0
#2   f   12.92      155     62.3    105.0
#3   f   12.75      153     63.3    108.0
#4   f   13.42      161     59.0     92.0
#5   f   15.92      191     62.5    112.5
#6   f   14.25      171     62.5    112.0

# 基本レイヤ
gp = ggplot(heightweight,aes(x=ageYear,y=heightIn))

# デフォルトの散布図を出力
gp + geom_point()

# 正方形の点（22）、サイズを3、塗りつぶしの色を白に
gp + geom_point(shape=22, size = 3, fill = "white")

# 点の枠線をdarkredにする（他は上を参照）。
gp1 = gp + geom_point(shape=22, size = 4, colour="darkred", fill =
"white")
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-E1


#---
#グループ分け
#---

library(gcookbook)

# heightweightの最初の6行
head(heightweight)
#. sex ageYear ageMonth heightIn weightLb
#1   f   11.92      143     56.3     85.0
#2   f   12.92      155     62.3    105.0
#3   f   12.75      153     63.3    108.0
#4   f   13.42      161     59.0     92.0
#5   f   15.92      191     62.5    112.5
#6   f   14.25      171     62.5    112.0

# 性別（sex）で色分け
gp_col = ggplot(heightweight,aes(x=ageYear,y=heightIn,colour=sex))
gp_col + geom_point()

# 性別（sex）で形状を変える
gp_shape = ggplot(heightweight,aes(x=ageYear,y=heightIn,shape=sex))
gp_shape + geom_point()

# 性別（sex）で色も形も変える
```

```
gp_colshape =
  ggplot(heightweight,aes(x=ageYear,y=heightIn,colour=sex,shape=sex))
gp1 = gp_colshape + geom_point(size=5)
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-E2



#--------------------------------
# （F）ヒストグラム：geom_histogram
#--------------------------------

#---
#基本
#---

library(gcookbook)

# faithfulの最初の6行
head(faithful)
#  eruptions waiting
#1     3.600      79
#2     1.800      54
#3     3.333      74
#4     2.283      62
#5     4.533      85
#6     2.883      55

# 基本レイヤ
gp = ggplot(faithful,aes(x=waiting))

# デフォルトのヒストグラム
gp + geom_histogram()

# ヒストグラムの塗りつぶしの色を白に、枠線の色を黒とする。
gp + geom_histogram(fill="white",colour="black")

# ビン幅を8に、ビンの開始を31とする。
# [31~39), [39~47), [47~55), ...（最大はデータによる）
gp + geom_histogram(binwidth=8, boundary = 31,
                    fill="white",colour="black")

# ビン幅を[31~39), [39~47), [47~55), ...[87~95)とする
gp1 = gp + geom_histogram(breaks = seq(31,95,by=8),
                    fill="white",colour="black")
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-F1
```

```
#---
#グループ分け
#---

library(dplyr)
library(MASS)

# faithful（出生時体重）の最初の6行
head(birthwt)
#   low age lwt race smoke ptl ht ui ftv  bwt
#85   0  19 182    2     0   0  0  1   0 2523
#86   0  33 155    3     0   0  0  0   3 2551
#87   0  20 105    1     1   0  0  0   1 2557
#88   0  21 108    1     1   0  0  1   2 2594
#89   0  18 107    1     1   0  0  1   0 2600
#91   0  21 124    3     0   0  0  0   0 2622

# smoke=0：喫煙なし
# smoke=1：喫煙あり
# bwt：出生児体重

# 喫煙の有無ごとにヒストグラムを分けて描画
gp = ggplot(birthwt,aes(x=bwt,fill=factor(smoke)))
## デフォルトでは積み上げで表示
gp + geom_histogram()
## 横並びではよくわからない
gp + geom_histogram(position=position_dodge())

## 重ねて表示（半透明とする）
gp1 = gp + geom_histogram(position=position_identity(),alpha=0.6)
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size=20)
);gp2 #--------------- Graph1-F2


# ファセット（facet_grid）を使うと、
# グループごとに異なるグラフで描画をしてくれる
## facet_grid関数の引数に、グループ分けしたい変数を指定
## 書式：facet_grid(XXX ~ .)

# 喫煙の有無（smoke）でグループ分け
gp = ggplot(birthwt,aes(x=bwt))
gp + geom_histogram(fill="white",colour="black") +
  facet_grid(smoke ~ .)

# 人種（race）でグループ分け
gp1 = gp + geom_histogram(fill="white",colour="black") +
  facet_grid(race ~ .)
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
```

```
  legend.text = element_text(size = 20),
  strip.text = element_text(size = 20) #ファセットタイトルのサイズ
);gp2 #--------------- Graph1-F3




#--------------------------------
#  (G) ボックスプロット（箱ひげ図）：geom_boxplot
#--------------------------------

#---
#基本
#---

library(dplyr)
library(MASS)

# faithful（出生時体重）の最初の6行
head(birthwt)
#   low age lwt race smoke ptl ht ui ftv  bwt
#85   0  19 182    2     0   0  0  1   0 2523
#86   0  33 155    3     0   0  0  0   3 2551
#87   0  20 105    1     1   0  0  0   1 2557
#88   0  21 108    1     1   0  0  1   2 2594
#89   0  18 107    1     1   0  0  1   0 2600
#91   0  21 124    3     0   0  0  0   0 2622

# race（人種）は1,2,3のいずれか
table(birthwt$race)
# 1  2  3
#96 26 67

# 基本レイヤの作成
## race（人種） をファクタに変換
gp = ggplot(birthwt,aes(x=factor(race),y=bwt))

# デフォルトの箱ひげ図
gp + geom_boxplot()

# 外れ値の範囲設定（IQR x 1.5）および、描画の形
gp + geom_boxplot(outlier.size=1.5, outlier.shape = 21)

# 外れ値を表示しない
## dotplotを重ね書きするときには表示しないのがよい
gp + geom_boxplot(outlier.shape = NA)

# birthwtの全ての行を対象に箱ひげ図を作成
## xに任意の値を与える（下の場合１）
ggplot(birthwt,aes(x=1,y=bwt)) + geom_boxplot()

## x軸の目盛とラベルを除く
gp1 = ggplot(birthwt,aes(x=1,y=bwt)) + geom_boxplot() +
  scale_x_continuous(breaks=NULL) +
  theme(axis.title.x = element_blank())
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
```

```
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20)
);gp2 #--------------- Graph1-G1

#---
# n x mの箱ひげ
#---

library(dplyr)
library(MASS)

# faithful（出生時体重）の最初の6行
head(birthwt)
#   low age lwt race smoke ptl ht ui ftv  bwt
#85   0  19 182    2     0   0  0  1   0 2523
#86   0  33 155    3     0   0  0  0   3 2551
#87   0  20 105    1     1   0  0  0   1 2557
#88   0  21 108    1     1   0  0  1   2 2594
#89   0  18 107    1     1   0  0  1   0 2600
#91   0  21 124    3     0   0  0  0   0 2622

# smoke（喫煙の有無: 0,1）
# race（人種：1,2,3）
# smoke x raceは合計6種類
table(birthwt[,c("race","smoke")])
#     smoke
#race  0  1
#   1 44 52
#   2 16 10
#   3 55 12

# 上記の6つのグループそれぞれで箱ひげ図を出力

# smokeで色分けする場合（いずれもファクタに変換する）
ggplot(birthwt,aes(x=factor(race),y=bwt,fill=factor(smoke))) +
  geom_boxplot()
## 凡例ラベルの表記を変更
ggplot(birthwt,aes(x=factor(race),y=bwt,fill=factor(smoke))) +
  geom_boxplot() +
  scale_fill_discrete(labels=c("NO SMOKE","SMOKE"))

# raceで色分けする場合
gp1 = ggplot(birthwt,aes(x=factor(smoke),y=bwt,fill=factor(race))) +
  geom_boxplot()
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20)
);gp2 #--------------- Graph1-G2


#---
# dotplotを重ねて描画
#---

library(dplyr)
```

```
library(MASS)

# faithful（出生時体重）の最初の6行
head(birthwt)
#   low age lwt race smoke ptl ht ui ftv  bwt
#85   0  19 182    2     0   0  0  1   0 2523
#86   0  33 155    3     0   0  0  0   3 2551
#87   0  20 105    1     1   0  0  0   1 2557
#88   0  21 108    1     1   0  0  1   2 2594
#89   0  18 107    1     1   0  0  1   0 2600
#91   0  21 124    3     0   0  0  0   0 2622


# 基本レイヤの作成
gp = ggplot(birthwt,aes(x=factor(race),y=bwt))

# ビンをY軸上に区切って（ビン幅：50）ドットをX軸方向に積み上げる
# デフォルトは棒グラフの中心から右に配列される
gp + geom_boxplot() +
  geom_dotplot(binaxis="y", binwidth=50)

# 中央寄せとして、色を中抜きとする。
gp + geom_boxplot() +
  geom_dotplot(binaxis="y", binwidth=50,
               stackdir="center", fill=NA)

# 平均値の追加（stat_summaryを使用する）
gp1 = gp + geom_boxplot() +
  geom_dotplot(binaxis="y", binwidth=50,
               stackdir="center", fill=NA) +
  stat_summary(fun.y="mean",geom="point",
               shape=23,size=5,fill="red")
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20)
);gp2 #--------------- Graph1-G3


# N x Mの箱ひげ図でdotplotを重ねる
## smoke=0を白に、smoke=1を灰色にマッピング
##  （boxplotとdotplotとstat_summaryのfillを個別に設定できればよいが
##   やり方わからず、わかる人いたらご教示ください）
gp1 = ggplot(birthwt,aes(x=factor(race),y=bwt,fill=factor(smoke))) +
  geom_boxplot() +
  scale_fill_manual(values=c("white","gray"),
                    labels=c("NO SMOKE","SMOKE")) +
  geom_dotplot(binaxis="y", binwidth=35,
               stackdir="center",
               position = position_dodge(0.75)) +
  stat_summary(fun.y="mean",geom="point",
               shape=23,size=5,
               position = position_dodge(0.75))
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
```

```
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph1-G4









#--------------------------------
# [GRAPH.2] 軸の設定
#--------------------------------

#---
# 使用するデータフレーム
#---

library(gcookbook)

head(heightweight)
#. sex ageYear ageMonth heightIn weightLb
#1   f   11.92      143     56.3     85.0
#2   f   12.92      155     62.3    105.0
#3   f   12.75      153     63.3    108.0
#4   f   13.42      161     59.0     92.0
#5   f   15.92      191     62.5    112.5
#6   f   14.25      171     62.5    112.0

str(heightweight)
#'data.frame':    236 obs. of  7 variables:
#$ sex      : Factor w/ 2 levels "f","m": 1 1 1 1 1 1 1 1 1 1 ...
#$ ageYear  : num  11.9 12.9 12.8 13.4 15.9 ...
#$ ageMonth : int  143 155 153 161 191 171 185 142 160 140 ...
#$ heightIn : num  56.3 62.3 63.3 59 62.5 62.5 59 56.5 62 53.8 ...
#$ weightLb : num  85 105 108 92 112 ...
#$ ageYearf : Factor w/ 7 levels "11","12","13",..: 1 2 2 3 5 4 5 1 3
1 ...
#$ ageYear.f: Factor w/ 7 levels "11","12","13",..: 1 2 2 3 5 4 5 1 3
1 ...

#---
# 軸のタイトル
#---

library(gcookbook)
# 散布図を描画（xを年齢、yを身長）
gp = ggplot(heightweight,aes(x=ageYear,y=heightIn))
gp = gp + geom_point()

# x軸・y軸のタイトルを設定しなおす
## デフォルトではデータフレームの列名が使われる。
gp + xlab("Age Year") + ylab("Height")
## 別の方法（1）
gp + labs(x="Age Year", y = "Height")
## 別の方法（2）
## scale_xy_continuousは連続値変数を扱う軸の設定
gp + scale_x_continuous(name = "Age Year") +
  scale_y_continuous(name = "Height")
```

```
## x軸のタイトルを非表示にする
gp + xlab(NULL)
gp + scale_x_continuous(name=NULL)
# x軸の目盛とラベルを除く
gp1 = gp + scale_x_continuous(breaks = NULL)
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph2-1


#---
# 目盛と範囲の設定（連続値変数）
#---

library(gcookbook)
# 散布図を描画（xを年齢、yを身長）
gp = ggplot(heightweight,aes(x=ageYear,y=heightIn))
gp = gp + geom_point()

# x軸の目盛を1刻みとする（範囲では無い）
gp + scale_x_continuous(breaks = 10:20)
# x軸の範囲を0〜20とする。
gp + scale_x_continuous(limits=c(10,20))
# x軸の目盛と範囲を同時に変える。
gp + scale_x_continuous(breaks=seq(10,20,by=2),limits=c(10,20))
# y軸の目盛と範囲を同時に変える。
gp + scale_y_continuous(breaks=seq(0,100,by=10),limits=c(0,100))
# xim、ylimを使っても同じ
gp + xlim(10,20) + ylim(40,80)
gp + xlim(10,20) + ylim(80,40) #y軸を反転


## y軸の反転（および範囲の設定）
gp + scale_y_reverse(limits=c(80,40))

# データの内容から軸の範囲を決定
gp + xlim(min(heightweight$ageYear)-2,
          max(heightweight$ageYear)+2) +
  ylim(min(heightweight$heightIn)-5,
       max(heightweight$heightIn)+5)
# 横軸が「y=0」を含む様に拡大
gp + expand_limits(x=0)


# 任意に設定した目盛の位置に位置にラベルを打つ
gp1 = gp + scale_y_continuous(breaks=c(50,56,60,72),
                          labels=c("Tiny","Short","Medium","Tallish"))
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph2-2
```

```
#---
# 離散値変数の軸設定（scale_xy_discrete）
#---

library(gcookbook)

# 年齢の整数部分だけを取り出し、離散値とし、さらにファクタとする
heightweight$ageYear.f = factor(floor(heightweight$ageYear))
## 各年齢の行数
table(heightweight$ageYear.f)
#11 12 13 14 15 16 17
#30 63 46 44 39 10  4

# 単に年齢（11〜17）の個数を数え上げたグラフ
## 横軸のxをファクタとして扱う。
gp = ggplot(heightweight,aes(x=ageYear.f))
gp = gp + geom_bar()

# 現在の状態はx軸は、11から17までの7つ
gp
# 順番を変える
gp + scale_x_discrete(limits=c("12","13","14","15","11","16","17"))
# 順番を反転させる
gp + scale_x_discrete(limits=rev(levels(heightweight$ageYear.f)))
# 一部のみを表示する
gp + scale_x_discrete(limits=c("12","13","14"))

# 軸のラベルに新しい名前を対応させる
gp1 = gp + scale_x_discrete(limits=c("12","14","16"),
                      breaks=c("12","14","16"),
                      labels=c("Twelve","Fourteen","Sixteen"))
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph2-3


#---
# スケールの変更
#---

library(gcookbook)
# 散布図を描画（xを年齢、yを身長）
gp = ggplot(heightweight,aes(x=ageYear,y=heightIn))
gp = gp + geom_point()

# x軸とy軸を反転
gp1 = gp + coord_flip()
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
```

```
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph2-4



# x軸とy軸のスケールを同じにする。
gp + coord_fixed()
## y軸のスケールをx軸の2倍とする。
gp1 = gp + coord_fixed(ratio = 1/2)
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph2-5



#---------------------------------
# [GRAPH.3] 凡例の設定
#---------------------------------

#---
# 使用するデータフレーム
#---

library(gcookbook)

# 離散値用のデータフレーム
head(PlantGrowth)
#. weight group
#1   4.17  ctrl
#2   5.58  ctrl
#3   5.18  ctrl
#4   6.11  ctrl
#5   4.50  ctrl
#6   4.61  ctrl

# 行数は30
str(PlantGrowth)
#'data.frame':    30 obs. of  2 variables:
#$ weight: num  4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
#$ group : Factor w/ 3 levels "ctrl","trt1",..: 1 1 1 1 1 1 1 1 1 1 ...

# grouopは要素数3のファクタ
PlantGrowth$group
#[1] ctrl ctrl ctrl ctrl ctrl ctrl ctrl ctrl ctrl ctrl
#[11] trt1 trt1 trt1 trt1 trt1 trt1 trt1 trt1 trt1 trt1
#[21] trt2 trt2 trt2 trt2 trt2 trt2 trt2 trt2 trt2 trt2
#Levels: ctrl trt1 trt2


# 連続値用のデータフレーム
head(heightweight)
#  sex ageYear ageMonth heightIn weightLb
#1   f   11.92      143     56.3     85.0
#2   f   12.92      155     62.3    105.0
#3   f   12.75      153     63.3    108.0
#4   f   13.42      161     59.0     92.0
```

```
#5   f   15.92      191     62.5    112.5
#6   f   14.25      171     62.5    112.0

str(heightweight)
#'data.frame':    236 obs. of  7 variables:
#$ sex      : Factor w/ 2 levels "f","m": 1 1 1 1 1 1 1 1 1 1 ...
#$ ageYear  : num  11.9 12.9 12.8 13.4 15.9 ...
#$ ageMonth : int  143 155 153 161 191 171 185 142 160 140 ...
#$ heightIn : num  56.3 62.3 63.3 59 62.5 62.5 59 56.5 62 53.8 ...
#$ weightLb : num  85 105 108 92 112 ...



#---
# 凡例の作成
#---

library(gcookbook)

# ボックスプロットの作成（fillは離散値groupにマッピング）
## 凡例は離散表現となる
gp_dc = ggplot(PlantGrowth,aes(x=group,y=weight,fill=group))
gp_dc = gp_dc + geom_boxplot()
gp2 = gp_dc + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph3-1

# 散布図の作成（fillは連続値ageYearにマッピング）
## fillをプロットに反映させるには、塗りのあるshapeを選ぶ必要あり
## 凡例はカラーバーとなる
gp_con = ggplot(heightweight,aes(x=heightIn,y=weightLb,fill=ageYear))
gp_con = gp_con + geom_point(size=3,shape=21)
gp2 = gp_con + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph3-2

#---
# 凡例タイトルの変更
#---

# 離散値の凡例のタイトルを変更（デフォルトは軸名）
gp_dc + scale_fill_discrete(name="Condition") #同じ
gp_dc + labs(fill="Condition") #省略記法

# 連続値の凡例タイトルの変更
gp_con + scale_fill_continuous(name="Condition") #同じ
gp_con + labs(fill="Condition") #省略記法



#---
```

```
# 凡例ラベルの変更
#---

# 凡例のラベル（デフォルトは要素名）を更新
gp1 = gp_dc + scale_fill_discrete(
  labels=c("Control","Treatment 1","Treatment 2"))
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph3-3


## 要素数が不足しているところはNAとなる
gp_dc + scale_fill_discrete(
  labels=c("Control","Treatment 1"))


# 連続値凡例のカラーバーの目盛と範囲を設定
gp_con + scale_fill_continuous(
  limits=c(10,20), breaks=seq(10,20,by=5)
  )

# 目盛に対応するラベルを更新
gp_con + scale_fill_continuous(
  limits=c(10,20), breaks=seq(10,20,by=5),
  labels=c("Ten","Fifteen","Twenty"))


# 離散値凡例の色をグレースケールで0.5～1.0の範囲とする。
gp_dc + scale_fill_grey(
  start=.5,end = 1,limits=c("ctrl","trt1","trt2"))

## カラーバーの最小値を黒、最大値を白とする
gp_con + scale_fill_gradient(
  low="black", high="white",
  limits=c(10,20),breaks=seq(10,20,by=5))

## カラーバーの代わりに離散的な凡例を用いる。
gp1 = gp_con + scale_fill_gradient(
  low="white", high="black",
  limits=c(12,18),breaks=seq(12,18,by=3),
  guide = guide_legend())
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph3-4




#---
# 凡例の非表示
#---
```

```
# fill凡例を非表示
gp_dc + scale_fill_discrete(guide=FALSE)
gp_con + scale_fill_continuous(guide=FALSE)

# 全てのエステてヒックマッピングを一気に非表示
gp_dc + theme(legend.position="none")
gp_con + theme(legend.position="none")

#---
# 凡例の位置を変える
#---

# 上部に表示
gp_dc + theme(legend.position="top")

# 座標で指定：左下が(0,0), 右上が(1,1)
gp1 = gp_dc + theme(legend.position=c(0.8,0.2))
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph3-5

# 凡例を4隅に合わせる
gp_dc + theme(legend.position=c(0,1),
              legend.justification=c(0,1))
gp_dc + theme(legend.position=c(0,0),
              legend.justification=c(0,0))

#---
# その他
#---

# 凡例の枠に境界線をつける

gp_dc + theme(legend.background =
                element_rect(fill="white",colour="black"))


# 凡例ラベルの体裁を一気に変更
gp_dc  + theme(legend.text=element_text(
  face="italic",family="Times",colour="red",size=14))

gp1 = gp_con  + theme(legend.text=element_text(
  face="italic",family="Times",colour="brown",size=20))
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph3-6
```

```
#--------------------------------
# [GRAPH.4] 色の設定
#--------------------------------

# 連続値用のデータフレーム
head(heightweight)
#  sex ageYear ageMonth heightIn weightLb
#1   f   11.92      143     56.3     85.0
#2   f   12.92      155     62.3    105.0
#3   f   12.75      153     63.3    108.0
#4   f   13.42      161     59.0     92.0
#5   f   15.92      191     62.5    112.5
#6   f   14.25      171     62.5    112.0

str(heightweight)
#'data.frame':    236 obs. of  7 variables:
#$ sex      : Factor w/ 2 levels "f","m": 1 1 1 1 1 1 1 1 1 1 ...
#$ ageYear  : num  11.9 12.9 12.8 13.4 15.9 ...
#$ ageMonth : int  143 155 153 161 191 171 185 142 160 140 ...
#$ heightIn : num  56.3 62.3 63.3 59 62.5 62.5 59 56.5 62 53.8 ...
#$ weightLb : num  85 105 108 92 112 ...

# ageYearの少数部分を切り捨て
heightweight$ageYear.i = floor(heightweight$ageYear)
#[1] 11 12 12 13 15 14 15 11 13 ...
# 年齢（ageYear.i）をファクタとする
heightweight$ageYear.f = factor(heightweight$ageYear.i)
#[1] 11 12 12 13 15 14 15 11 13 ...

# デフォルトのboxplot
## fillを指定していない場合、白色が塗られる。
gp = ggplot(heightweight,aes(x=ageYear.f,y=heightIn))
gp + geom_boxplot()

# 一律に色を塗る
gp = ggplot(heightweight,aes(x=ageYear.f,y=heightIn))
gp + geom_boxplot(colour="black",fill="purple")

# エステティックマッピングでfillに軸名を対応させると
# ファクトの要素ごとに異なる色の配色となる
gp = ggplot(heightweight,aes(x=ageYear.f,y=heightIn,fill=ageYear.f))
gp = gp + geom_boxplot()

# デフォルトの配色パレット
## 色相で等距離にあるもの
gp + scale_fill_discrete()
gp1 = gp + scale_fill_hue() #上に同じ
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph4-1

# グレースケール
gp + scale_fill_grey()
```

```
## 始まりと終わりのグレースケールの値を指定
gp1 = gp + scale_fill_grey(start=1.0,end=0.5)
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph4-2

# viridisパレット
# library(viridis)
gp1 = gp + scale_fill_viridis_d()
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph4-3


# ColorBrewerパレット（デフォルト）
gp1 = gp + scale_fill_brewer()
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph4-4

## ColorBrewerで使用できるパレットを調べる
library(RColorBrewer)
display.brewer.all()

## 引数でパレットを指定
gp + scale_fill_brewer(palette = "Oranges")
gp + scale_fill_brewer(palette = "PiYG")
gp + scale_fill_brewer(palette = "Paired")
gp1 = gp + scale_fill_brewer(palette = "Set2")
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph4-5



# 手動で色指定（valuesで要素の数だけ指定します）
gp1 = gp + scale_fill_manual(
  values=c("purple","pink","purple","pink","purple","pink","purple"))
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
```

```
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph4-6


gp1 = gp + scale_x_discrete(limits=c("12","16"),
                      labels=c("Twelve","Sixteen")) +
  scale_fill_manual(values=c("#CC6666","#7777DD"))
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph4-7


# そのほかによく使われているものとして、
# Nature Publishing Groupのパレットががあります。

# ggsciのライブラリを使用します。
install.packages("ggsci")
library(ggsci)

gp1 = gp + scale_fill_npg()
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph4-8



#--------------------------------
# [GRAPH.5] その他
#--------------------------------

#---
# ファセットの利用
#---

# 使用するデータフレーム
head(heightweight)
#  sex ageYear ageMonth heightIn weightLb
#1   f   11.92      143     56.3     85.0
#2   f   12.92      155     62.3    105.0
#3   f   12.75      153     63.3    108.0
#4   f   13.42      161     59.0     92.0
#5   f   15.92      191     62.5    112.5
#6   f   14.25      171     62.5    112.0

str(heightweight)
#'data.frame':    236 obs. of  7 variables:
#$ sex      : Factor w/ 2 levels "f","m": 1 1 1 1 1 1 1 1 1 1 ...
#$ ageYear  : num  11.9 12.9 12.8 13.4 15.9 ...
#$ ageMonth : int  143 155 153 161 191 171 185 142 160 140 ...
#$ heightIn : num  56.3 62.3 63.3 59 62.5 62.5 59 56.5 62 53.8 ...
```

```
#$ weightLb : num  85 105 108 92 112 ...

# ageYearの少数部分を切り捨て
heightweight$ageYear.i = floor(heightweight$ageYear)
#[1] 11 12 12 13 15 14 15 11 13 ...
# 年齢（ageYear.i）をファクタとする
heightweight$ageYear.f = factor(heightweight$ageYear.i)
#[1] 11 12 12 13 15 14 15 11 13 ...

# 身長（x=heightIn）と体調（y=weightLb）の相関図をプロット
gp = ggplot(heightweight,aes(x=heightIn,y=weightLb))
gp + geom_point(size=3, shape=21)

# 年齢で色分けする
gp = ggplot(heightweight,aes(x=heightIn,y=weightLb,fill=ageYear.f))
gp = gp + geom_point(size=3, shape=21)
gp1 = gp + scale_fill_discrete()
gp2 = gp1 + theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
);gp2 #--------------- Graph5-1

# 性別で色分けする
gp = ggplot(heightweight,aes(x=heightIn,y=weightLb,fill=sex))
gp = gp + geom_point(size=3, shape=21)
gp + scale_fill_discrete()

# 要素ごとに異なるグラフを作成する（ファセットの利用）

## 基本プロットの作成
gp = ggplot(heightweight,aes(x=heightIn,y=weightLb))
gp = gp + geom_point(size=3, shape=21, fill="black")

## 垂直方向に分割
gp_a = gp + facet_grid(ageYear.f ~ .)
gp_b = gp + facet_grid(sex ~ .)

## 水平方向に分割
gp_c = gp + facet_grid(. ~ ageYear.f)
gp_d = gp + facet_grid(. ~ sex)

## フォントサイズを整えます。
gp2 = gp_a+ theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 15),
  legend.text = element_text(size = 20),
  strip.text = element_text(size = 20)
);gp2 #--------------- Graph5-2

gp2 = gp_d+ theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
```

```
  axis.text.y = element_text(size = 20),
  legend.text = element_text(size = 20),
  strip.text = element_text(size = 50)
);gp2 #--------------- Graph5-3



## なるべく同じ行数列数で配置
gp + facet_wrap( ~ ageYear.f)
## 行数を指定
gp + facet_wrap( ~ ageYear.f, nrow=4)
## 列数を指定
gp1 = gp + facet_wrap( ~ ageYear.f, ncol=4)
gp2 = gp1+ theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 15),
  legend.text = element_text(size = 20),
  strip.text = element_text(size = 20)
);gp2 #--------------- Graph5-4



## xとyをフリースケールにする
##（軸の範囲が可変となります）
gp + facet_wrap( ~ ageYear.f, scales = "free")
## yのみをフリースケールにする
gp + facet_wrap( ~ ageYear.f, scales = "free_y")


# ファセットラベルのテキストを変更する

## 凡例ラベルと異なり、
## ファセットラベルを変更するには、
## 大元のデータフレームのデータを変更しなければな

# install.packages(dplyr)
library(dplyr)
# dplyrライブラリのrecode関数を使うと、
# 文字列を異なる文字列に一括に変換してくれます。
heightweight$gender =
  recode(factor(heightweight$sex), "f"="Female","m"="Male")

## 次のように変わりました.
###[BEFORE]
heightweight$sex
#[100] f f f f f f f f f f f m m m

###[AFTER]
heightweight$gender
#[109] Female Female Female Male   Male   Male
#...

## これでファセットラベルが「Female」「Male」となりました
gp = ggplot(heightweight,aes(x=heightIn,y=weightLb,fill=ageYear.f))
gp = gp + geom_point(size=3, shape=21)
```

```
gp + facet_grid(. ~ gender)

# ファセットラベルのサイズを変える
gp + facet_grid(. ~ gender) +
  theme(strip.text = element_text(size=40))

# ファセットラベルの背景も変える
gp1 = gp + facet_grid(. ~ gender) +
  theme(strip.text = element_text(face="bold",size=rel(2)),
        strip.background =
          element_rect(fill="lightblue",colour="black",size=1))
gp2 = gp1+ theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 15),
  legend.text = element_text(size = 20)
);gp2 #--------------- Graph5-5



# ファセットごとに異なる色を与える.
## fillを使いますが、判例とファセットで表記が重なるため,
## 凡例を消去します.
gp = ggplot(heightweight,aes(x=heightIn,y=weightLb,fill=gender))
gp = gp + geom_point(size=3, shape=21)
gp = gp + scale_fill_discrete(guide=FALSE)
gp1 = gp + facet_grid(. ~ gender) +
  theme(strip.text = element_text(size=40))
gp2 = gp1+ theme(
  legend.title = element_text(size = 30),
  axis.title.x = element_text(size = 30),
  axis.title.y = element_text(size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 15),
  legend.text = element_text(size = 20)
);gp2 #--------------- Graph5-6
```

R
第４講

グラフの実践１：カテゴリー変数の集計
（奇数と偶数と誕生日）

```
########################################
########################################
####### グラフ２（実践１） #############
####### geom_bar / geom_col ############
########################################
########################################


library(ggplot2)


#----------------------------------------
# [準備1] データフレームをインポート
#----------------------------------------


# 以下から、奇数偶数の好み実験結果の架空（N=2865）のデータをインポートしてください。
url = "https://lab.kenrikodaka.com/_download/csv/oddeven_2865.csv"
source = read.csv(url)

source

#誕生日と奇数偶数の好みに関するアンケートの架空のデータ（2865人分）です。
#month（誕生月）・day（誕生日）・preference（奇数が好き：ODD, 偶数が好き：EVEN）
#gender（男性：MALE, 女性：FEMALE）・$domhand（利き手が左：LEFT, 利き手が右：
RIGHT）
#age（年齢）

#最初の6行をちょっと出し
source[1:6,]
#  month day preference gender age domhand
#1    10  26        ODD   MALE  19   RIGHT
#2     4   6        ODD   MALE  NA    <NA>
#3     9  14       EVEN FEMALE  20   RIGHT
#4     1  21        ODD FEMALE  17   RIGHT
#5     4   7       EVEN FEMALE  20   RIGHT
#6     7  27       EVEN FEMALE  18   RIGHT

##NAは未定です。

# 翻訳すると以下の様になります。
# 10月26日、奇数好き、男性、19歳、右利き
# 04月06日、奇数好き、男性、不明、不明
# 09月14日、偶数好き、女性、18歳、右利き
# 01月21日、奇数好き、女性、18歳、右利き
# 04月07日、偶数好き、女性、18歳、右利き
# 07月27日、偶数好き、女性、19歳、右利き



#----------------------------------------
# [準備2] mutateの使い方
#----------------------------------------

#mutateを使うと
#データフレームの属性を簡単に追加／更新できます。
```

```
#mutateを使うには以下のライブラリをインクルードします。
library(dplyr)

# mutateの構文は以下の通り
#----------------------------------------
## 新しいデータフレーム =
## mutate(古いデータフレーム, 新しい属性 = 既存の属性同士の計算)
#----------------------------------------


# sourceをdatにコピーします。
dat = source
dat[1:3,]
#   month day preference gender age domhand
#1      10  26        ODD   MALE  19   RIGHT
#2       4   6        ODD   MALE  NA    <NA>
#3       9  14       EVEN FEMALE  20   RIGHT

# datに月日の奇数／偶数のラベル（mtype）を新たに追加します。
dat2 = mutate(dat, mtype=month%%2)
dat2[1:4,]
#   month day preference gender age domhand mtype
#1      10  26        ODD   MALE  19   RIGHT     0
#2       4   6        ODD   MALE  NA    <NA>     0
#3       9  14       EVEN FEMALE  20   RIGHT     1
#4       1  21        ODD FEMALE  17   RIGHT     1

# if_else（条件, 条件が真のとき, それ以外）という構文となります。
dat2 = mutate(dat2, mtype = if_else(mtype==0,"EVEN","ODD"))
dat2[1:4,]

#    month day preference gender age domhand mtype
#1      10  26        ODD   MALE  19   RIGHT  EVEN
#2       4   6        ODD   MALE  NA    <NA>  EVEN
#3       9  14       EVEN FEMALE  20   RIGHT   ODD
#4       1  21        ODD FEMALE  17   RIGHT   ODD

# 同様にdtypeも追加
dat2 = mutate(dat2, dtype = if_else(day%%2==0,"EVEN","ODD"))
dat2[1:4,]
#     month day preference gender age domhand mtype dtype
# 1      10  26        ODD   MALE  19   RIGHT  EVEN  EVEN
# 2       4   6        ODD   MALE  NA    <NA>  EVEN  EVEN
# 3       9  14       EVEN FEMALE  20   RIGHT   ODD  EVEN
# 4       1  21        ODD FEMALE  17   RIGHT   ODD   ODD


# 3つ以上の分岐はcase_whenを使いいます。
dat2 = mutate(dat2, combo =
                case_when(mtype=="EVEN"&dtype=="EVEN"~"EE",
                          mtype=="ODD"&dtype=="ODD"~"OO",
                          mtype=="EVEN"&dtype=="ODD"~"EO",
                          TRUE~"OE"));
dat2[1:4,]

#    month day preference gender age domhand mtype dtype combo
#1      10  26        ODD   MALE  19   RIGHT  EVEN  EVEN    EE
#2       4   6        ODD   MALE  NA    <NA>  EVEN  EVEN    EE
#3       9  14       EVEN FEMALE  20   RIGHT   ODD  EVEN    OE
#4       1  21        ODD FEMALE  17   RIGHT   ODD   ODD    OO
```

```
# グラフを作成します
## 4つのcomboごとに、偶数好きと奇数好きの人数を数え上げます。
g = ggplot(dat2, aes(x=combo, fill=preference))
g = g + geom_bar(position = position_dodge()) #fillを横に並べる
g = g + labs(title = "Graph0_Mutate") #グラフのタイトル

g0 = g
g0 #out Graph0_Mutate



#----------------------------------------
# [Graph1_Gender]
# 男女別の奇数偶数好き
#----------------------------------------

# sourceをそのままdatに移します。
dat = source

# [ggplot]
## x軸をgenderに、枠線と塗りつぶしをpreferenceに設定
g = ggplot(dat,aes(x=gender,color=preference, fill=preference))

# [geom_bar] genderごとにpreferenceを数え上げ
## position_dodge：横並び, width：グラフの幅, alpha：透明度
g = g + geom_bar(position=position_dodge(width=0.9),alpha=0.9)

# X軸は離散値、Y軸は連続値
g = g + scale_x_discrete(limits = c("FEMALE", "MALE"))
g = g + scale_y_continuous(limits = c(-100,
1100),breaks=c(0,250,500,750,1000))

# 塗りつぶしのパレットをnejmに変更
g = g + scale_fill_nejm()
# 90度回転
g = g + coord_flip()

# [geom_text]
## geom_barの数え上げの量を表示
## just=0（左揃え）, 0.5（中央揃え）, 1（右揃え）
g = g + geom_text(stat = "count",
                  aes(label = paste("(",..count..,")",sep=""),
                      y=..count..*0.95, hjust=1),
                  position = position_dodge(width=0.9),
                  color="white", size=10)
# [labs]
## グラフのタイトル、XY軸のタイトル
g = g + labs(title = "Graph1_Gender", x = "Gender", y = "Population")

# [theme]
g = g + theme(
  # 縦横比を3:4
  aspect.ratio=3/4,
  # ベースを太字、サイズ36に
  text = element_text(face = "bold", size = 36),
  # 軸の目盛りのサイズ
  axis.text.x = element_text(size = 30),
```

```
  axis.text.y = element_text(size = 25),
  # 凡例関係の設定
  legend.text = element_text(size=20),
  legend.title = element_text(size=25),
  legend.position=c(0.95,0.95),
  legend.justification=c(1,1),
  legend.background =
    element_rect(fill = "white", colour = "black"))

g1 = g
g1 #out Graph1_Gender



#----------------------------------------
# [Graph2_FemaleMonth]
# 女性の月の数字の影響
#----------------------------------------

# 女性だけを集めたデータフレームをdat_とする
dat_ = source[source$gender=="FEMALE",]

# 女性の数を確認
nrow(dat_)
# [1] 1603

# 奇素数（2を除く素数）を判定する関数の作成
#------------------------------------------------------------
oddprimeSingle = function(n){
  result = FALSE
  if(n==3||n==5||n==7||n==11||n==13||n==17||n==19||n==23||n==29||n==31){
    result = T
  }
  result
}

oddprime = function(v){
  result = vector("logical", length(v))
  for(i in 1:length(v)){
    result[i] = oddprimeSingle(v[i])
  }
  result
}
#------------------------------------------------------------


# 月が偶数、奇数、素数の好みをベクトルで取り出す
pref_even = dat_$preference[dat_$month%%2==0]
pref_odd = dat_$preference[dat_$month%%2==1]
pref_prime = dat_$preference[oddprime(dat_$month)]

# それぞれの偶数好きの割合の重み（50%を0とする）
r1 = 100*(sum(pref_even=="EVEN")/length(pref_even)-0.5)
r2 = 100*(sum(pref_odd=="EVEN")/length(pref_odd)-0.5)
r3 = 100*(sum(pref_prime=="EVEN")/length(pref_prime)-0.5)

# 偶数好きの重みをデータフレームとする
x = data.frame(
  ntype = c("EVEN", "ODD", "PRIME"),
  weight = c(r1, r2,r3)
)
```

```
# [ggplot]
g = ggplot(x, aes(x = ntype, y = weight, fill = ntype, colour = ntype))

# [geom_bar]
## 個数（stat="count"）でなく数値（stat="identity）を描画
g = g + geom_bar(stat = "identity",alpha=4/5)

# yは連続値
g = g + scale_y_continuous(limits=c(-10,30))

# 縦線（hline）と横線（vline）の描画
g = g + geom_hline(yintercept = 0, linetype = "solid",
size=1,colour="black")
g = g + geom_vline(xintercept = 1.5, linetype = "dotted",
size=1,colour="black")
g = g + geom_vline(xintercept = 2.5, linetype = "dotted",
size=1,colour="black")

# 色の設定
col1 = rgb(174/255,69/255,50/255) #for even
col2 = rgb(58/255,128/255,77/255) #for odd
col3 = rgb(48/255,112/255,176/255) #for prime

g = g + scale_fill_manual(values = c(col1,col2,col3), name="number type")
g = g + scale_colour_manual(values = c(col1,col2,col3), name = "number
type")

# 90度回転
g = g + coord_flip()

# [labs]
## グラフのタイトル、XY軸のタイトル
g = g + labs(title = "Graph2_FemaleMonth",
             x = "Number type", y = "Even number preference (%)")

# [theme]
g = g + theme(
  # 縦横比を3:4
  aspect.ratio=3/4,
  # ベースを太字、サイズ36に
  text = element_text(face = "bold", size = 36),
  # 軸の目盛りのサイズ
  axis.text.x = element_text(size = 30),
  axis.text.y = element_text(size = 25),
  # 凡例関係の設定
  legend.text = element_text(size=20),
  legend.title = element_text(size=25),
  legend.position=c(0.95,0.95),
  legend.justification=c(1,1),
  legend.background =
    element_rect(fill = "white", colour = "black"))

g2 = g
g2 #out Graph2_FemaleMonth


#---------------------------------------
# [Graph3_FemaleMonthDay]
```

```
# 女性の月x日の数字の影響
#----------------------------------------


# 月と日にちが偶数、奇数、素数の好みをベクトルで取り出す
pref_even = dat_$preference[dat_$month%%2==0 & dat_$day%%2==0]
pref_odd = dat_$preference[dat_$month%%2==1 & dat_$day%%2==1]
pref_prime = dat_$preference[oddprime(dat_$month) & oddprime(dat_$day)]

#以下はGraph2とほぼ同様

# それぞれの偶数好きの割合の重み（50%を0とする）
r1 = 100*(sum(pref_even=="EVEN")/length(pref_even)-0.5)
r2 = 100*(sum(pref_odd=="EVEN")/length(pref_odd)-0.5)
r3 = 100*(sum(pref_prime=="EVEN")/length(pref_prime)-0.5)

x = data.frame(
  ntype = c("EVEN", "ODD", "PRIME"),
  weight = c(r1, r2,r3)
)

g = ggplot(x, aes(x = ntype, y = weight, fill = ntype, colour = ntype))
g = g + geom_bar(stat = "identity",alpha=4/5)
g = g + scale_y_continuous(limits=c(-10,30))

# x軸（離散値）のラベルを変更
g = g + scale_x_discrete(limits=c("EVEN","ODD","PRIME"),
                         label=c("EVENxEVEN","ODDxODD","PRIMExPRIME"))

g = g + geom_hline(yintercept = 0, linetype = "solid",
size=1,colour="black")
g = g + geom_vline(xintercept = 1.5, linetype = "dotted",
size=1,colour="black")
g = g + geom_vline(xintercept = 2.5, linetype = "dotted",
size=1,colour="black")
g = g + scale_fill_manual(values = c(col1,col2,col3), name="number type")
g = g + scale_colour_manual(values = c(col1,col2,col3), name = "number
type")
g = g + coord_flip()
g = g + labs(title = "Graph3_FemaleMonthDay",
             x = "Number type", y = "Even number preference (%)")
g = g + theme(
  aspect.ratio=4/4.5,
  text = element_text(face = "bold", size = 36),
  axis.text.x = element_text(size = 30),
  axis.text.y = element_text(size = 25),
  legend.text = element_text(size=20),
  legend.title = element_text(size=25),
  legend.position=c(0.95,0.95),
  legend.justification=c(1,1),
  legend.background =
    element_rect(fill = "white", colour = "black"))

g3 = g
g3 #Graph3_FemaleMonthDay



#----------------------------------------
# [Graph4_DayRank]
# 誕生日の日にちランキング（男女分けない）
#----------------------------------------
```

```
dat_ = source;


#----------------------------------------
# グラフ用のデータフレームの作成
#----------------------------------------

#day毎のpreferenceの数を集計
daypref = table(dat_$day, dat_$preference)

#     EVEN ODD
# 1    62  54
# 2    64  27
# 3    55  33

# tableをデータフレームに変換する
df = as.data.frame(daypref)
#   Var1 Var2 Freq
#1     1 EVEN   62
#2     2 EVEN   64
#3     3 EVEN   55

df.e= df[df$Var2=="EVEN",]
df.o= df[df$Var2=="ODD",]

# 列として接続
df = cbind(df.e,df.o)
#   Var1 Var2 Freq Var1 Var2 Freq
#1     1 EVEN   62    1  ODD   54
#2     2 EVEN   64    2  ODD   27
#3     3 EVEN   55    3  ODD   33

# 1,3,6列目だけを取り出す
df = df[,c(1,3,6)]
colnames(df) = c("number","even","odd")
#   number even odd
#1       1   62  54
#2       2   64  27
#3       3   55  33

# 新しい属性erate（偶数を好む確率）をdfに追加
df = mutate(df,erate = even/(even+odd))
#   number even odd     erate
#1       1   62  54 0.5344828
#2       2   64  27 0.7032967
#3       3   55  33 0.6250000

# number属性をfactorからintegerに変更します。
class(df$number)
#[1] "factor"
df$number = as.integer(df$number)
class(df$number)
#[1] "integer"


#mutateを使った分岐
df = mutate(df, type = case_when(number%%2==0~"EVEN",
                          oddprime(number)~"PRIME",
                          TRUE~"ODD"))
#if_elseを使った場合（上と同じ）
```

```
df = mutate(df, type = if_else(number%%2==0,"EVEN",
                                  if_else(oddprime(number),"PRIME","ODD")))

#   number even odd     erate  type
#1       1   62  54 0.5344828   ODD
#2       2   64  27 0.7032967  EVEN
#3       3   55  33 0.6250000 PRIME
#4       4   51  33 0.6071429  EVEN
#5       5   43  43 0.5000000 PRIME
#...

# erateの大きい順に並べる
df = arrange(df,desc(erate)); df
#   number even odd     erate  type
#1      22   57  23 0.7125000  EVEN
#2       2   64  27 0.7032967  EVEN
#3      21   72  39 0.6486486   ODD

# 順位の属性を加える（）
df = cbind(df, data.frame(order=1:31)); df
#   number even odd     erate  type order
#1      22   57  23 0.7125000  EVEN     1
#2       2   64  27 0.7032967  EVEN     2
#3      21   72  39 0.6486486   ODD     3


dat_order = df

#-----------------------------------------
# グラフの作成
#-----------------------------------------

g = ggplot(dat_order,aes(x=order,y=100*erate,fill=type)); g
g = g + geom_bar(stat = "identity", colour="black"); g

#geom_bar(stat="identity")とgeom_colは同じなので、こちらでも同じ結果です。
#g = g + geom_col(colour ="black")

g = g + scale_x_continuous(limits=c(0.5,31.5),breaks=c(1,10,20,30)); g
g = g + scale_y_continuous(limits=c(0,87),breaks=c(0,25,50,75)); g

# 50%のところに線を引く
g = g + geom_hline(yintercept = 50, linetype = "dotted",
size=1,colour="black");
# 各順位に対応する数字をバーの上端より1.5文字分下に描画する
g = g + geom_text(aes(label = number), colour = "white",  size = 5, vjust
= 1.5)

# 凡例の色をマニュアルで指定する
g = g + scale_fill_manual(values = c(col1,col2,col3),name="Number type")
g = g + scale_colour_manual(values = c(col1,col2,col3),name="Number type")

g = g + labs(title="Graph4_DayRank",x="ORDER",y="EVEN-PREF RATE")
g = g + theme(
  aspect.ratio=1/4,
  text = element_text(face = "bold", size = 36),
  axis.text.x = element_text(size = 30),
  axis.text.y = element_text(size = 25),
  legend.text = element_text(size=20),
  legend.title = element_text(size=25),
  legend.position=c(1.0,1.03),
  legend.justification=c(1,0),
```

```
  legend.background =
    element_rect(fill = "white", colour = "black"))

g4 = g
g4 #Graph4_FemaleMonthDay


#----------------------------------------
# [確認課題]
# ApMedia04_Work
# 男女別の月の偶数好きの数字ランキングのグラフを作成してください。
# 締切は11月中とします.
# ファイル名は「2250xx_work4.R」としてください。
#----------------------------------------


# グラフを結合する場合、以下を参照してください。

## グラフの結合の方法
grid.arrange(g2,g3,nrow=1) #1行で並べる
grid.arrange(g2,g3,ncol=1) #1列で並べる

## 3つを配置する例
g23 = grid.arrange(g2,g3,nrow=1)
grid.arrange(g4,g23,ncol=1)
```

R
第５講

グラフの実践２：時系列データを扱う
（集団フリーシャッター課題）

```
########################################
########################################
####### グラフ３（実践２） #############
####### フリーシャッター課題 ###########
########################################
########################################

# 主要なライブラリを読み込みます

#install.packages(ggplot2)
library(ggplot2)

#install.packages(dplyr)
library(dplyr)


# まずは以下のCSVを読み込んでください。

url = "https://lab.kenrikodaka.com/_download/csv/freeshutter_230601.csv"
dat_230601 = read.csv(url)
head(dat_230601)

#  seat  a1  b1   b2   b3 seat_ix seat_iy
#1  192 2.2 1.5  2.7  5.6      12      16
#2  168 0.8 4.4 10.0 14.7      12      14
#3   59 0.6 0.6  0.8  0.9      11       5
#4   60 2.9 3.4  5.3  9.9      12       5
#5   89 6.2 0.3 10.2 18.1       5       8
#6  180 0.1 0.1  0.5  0.8      12      15

# 集団フリーシャッター実験
# 実験A（20秒間の間に１度だけシャッターを押す）
# 実験B（20秒間の間に最大で３回シャッターを押す）

#seat（座席番号）
#a1（実験Aのシャッター時間）
#b1（実験Bのシャッター時間：1回目）
#b2（実験Bのシャッター時間：2回目）
#b3（実験Bのシャッター時間：3回目）
#seat_ix（座席の左右位置：1–12）
#seat_iy（座席の奥行き位置：1–22）

# データ数（119人のデータ）
n = nrow(dat_230601)
n
# [1] 119


# dat0を、以下のような形式に整列し直します。
## – 行数を4倍とし、新たに変数「exp」と「order」を用いて、
## – 個々の時間の、実験属性（A or B）と順序（1、2、3回目）を紐づけます。


seat0 = rep(dat_230601$seat,4) #座席番号（参加者ID）
seatx = rep(dat_230601$seat_ix,4) #座席位置（Y）
seaty = rep(dat_230601$seat_iy,4) #座席位置（Y）
time0 = c(dat_230601$a1,dat_230601$b1,dat_230601$b2,dat_230601$b3) #シャッター時間
exp0 = factor(c(rep("Single",n),rep("Triple",3*n))) #実験の種類（A or B）
order0 = factor(c(rep(1,n),rep(1,n),rep(2,n),rep(3,n))) #何回目のシャッターか

#データフレーム化します。
dat = data.frame(seat=seat0,seatx=seatx,seaty=seaty,
                 time=time0,exp=exp0,order=order0)

dat
# 以下のような構造となっています。
str(dat)
#'data.frame':        476 obs. of  4 variables:
#$ seat : int  192 168 59 60 89 180 223 213 214 191 ...
#$ seatx: int  16 14 5 5 8 15 19 18 18 16 ...
```

```
#$ seaty: int  12 12 11 12 5 12 7 9 10 11 ...
#$ time : num  2.2 0.8 0.6 2.9 6.2 0.1 14.8 0.7 1.9 0.3 ...
#$ exp  : Factor w/ 2 levels "Single","Triple": 1 1 1 1 1 1 1 1 1 1 ...
#$ order: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
```

```
#-------------------------------------------------------------
#-------------------------------------------------------------
# [Graph1_ShutterHistogram]
# シャッターのヒストグラム
#-------------------------------------------------------------
#-------------------------------------------------------------
```

```
#----------------------------------------
# Graph1A（Single実験）
#----------------------------------------

# Single実験のデータフレームを取得
dat_si = dat[dat$exp=="Single",]

# [ggplot]
## x軸をtimeに設定
gp_a = ggplot(dat_si,aes(x=time))

# [geom_histogram] ヒストグラムの作成
## 一つのビンの幅を0.1sec、塗りつぶしを白、枠線を黒
gp_a = gp_a + geom_histogram(binwidth=0.1, fill="white",colour="black")

# Y軸（Count）の最大を10とする
gp_a = gp_a + scale_y_continuous(limits=c(0,10),breaks=seq(0,10,by=2))
# X軸の範囲と目盛の位置を決める
gp_a = gp_a + scale_x_continuous(limits=c(0,20),breaks=seq(0,20,by=5))

# [labs]
## グラフのタイトル、XY軸のタイトル
gp_a = gp_a + labs(title = "Graph1A_ShutterHistogram", x = "Time[s]", y = "Count")

# [theme]
gp_a = gp_a + theme(
  # 縦横比を1:3
  aspect.ratio=1/3,
  # ベースを太字、サイズ36に
  text = element_text(face = "bold", size = 36),
  # 軸の目盛りのサイズ
  axis.text.x = element_text(size = 30),
  axis.text.y = element_text(size = 25),
  # 凡例関係の設定
  legend.text = element_text(size=20),
  legend.title = element_text(size=25))

gp_a #out Graph1A_ShutterHistogram

#----------------------------------------
# Graph1B（Triple実験）
#----------------------------------------

# Triple実験のデータフレームを取得
dat_tri = dat[dat$exp=="Triple",]

# [ggplot]
## x軸をtimeに設定
gp_b = ggplot(dat_tri,aes(x=time))
gp_b = gp_b + geom_histogram(binwidth=0.1, fill="white",colour="black")
gp_b = gp_b + scale_y_continuous(limits=c(0,10),breaks=seq(0,10,by=2))
gp_b = gp_b + scale_x_continuous(limits=c(0,20),breaks=seq(0,20,by=5))
gp_b = gp_b + labs(title = "Graph1B_ShutterHistogram", x = "Time[s]", y = "Count")
```

```
# [theme]
gp_b = gp_b + theme(
  aspect.ratio=1/3,
  text = element_text(face = "bold", size = 36),
  axis.text.x = element_text(size = 30),
  axis.text.y = element_text(size = 25))
gp_b #out Graph1B_ShutterHistogram


#----------------------------------------
# Graph1C（Triple実験、順序別）
#----------------------------------------

# [ggplot]
## x軸をtimeに設定, 塗りつぶしをorderに設定
gp_c = ggplot(dat_tri,aes(x=time,fill=order))

# 積み上げ順序の反転（factor順序：下から上へ）
## 1->2->3の順に積み上げるようにする
gp_c = gp_c + geom_histogram(binwidth=0.1,colour="black",
                             position = position_stack(reverse = TRUE))

gp_c = gp_c + scale_y_continuous(limits=c(0,10),breaks=seq(0,10,by=2))
gp_c = gp_c + scale_x_continuous(limits=c(0,20),breaks=seq(0,20,by=5))
gp_c = gp_c + labs(title = "Graph1C_ShutterHistogram", x = "Time[s]", y = "Count")

# 塗りつぶしの対応を決める。
## orderはfactor（非数値）のため、limits=c(1,2,3)では不可
gp_c = gp_c + scale_fill_discrete(
  limits=c("1","2","3"),labels=c("1st","2nd","3rd"))

# [theme]
gp_c = gp_c + theme(
  aspect.ratio=1/3,
  text = element_text(face = "bold", size = 36),
  axis.text.x = element_text(size = 30),
  axis.text.y = element_text(size = 25))
gp_c #out Graph1C_ShutterHistogram

#----------------------------------------
# Graph1D（Triple実験、順序別）
#----------------------------------------

gp_d = ggplot(dat_tri,aes(x=time,fill=order))
gp_d = gp_d + geom_histogram(binwidth=0.1,colour="black")
gp_d = gp_d + scale_y_continuous(limits=c(0,10),breaks=seq(0,10,by=5))
gp_d = gp_d + scale_x_continuous(limits=c(0,20),breaks=seq(0,20,by=5))
gp_d = gp_d + scale_fill_discrete(
  limits=c("1","2","3"),labels=c("1st","2nd","3rd"))
gp_d = gp_d + labs(title = "Graph1D_ShutterHistogram", x = "Time[s]", y = "Count")
gp_d = gp_d + theme(
  aspect.ratio=1/3,
  text = element_text(face = "bold", size = 36),
  axis.text.x = element_text(size = 30),
  axis.text.y = element_text(size = 25))

# facetをorderで分割、factの背景なども整える
gp_d = gp_d + facet_grid(order ~ .)
gp_d = gp_d + theme(
  strip.text = element_text(size = 25),
  strip.background =
    element_rect(fill="gray",colour="black",size=1))

gp_d  #out Graph1D_ShutterHistogram









#------------------------------------------------------------------
```

```
#-----------------------------------------------------------------
# [Graph2_ShutterBox]
# シャッター時間のボックスプロット
#-----------------------------------------------------------------
#-----------------------------------------------------------------

# [ggplot]
## x軸をexp、y軸にtime, 塗りつぶしをorderに設定
gp = ggplot(dat,aes(x=exp,y=time,fill=order))

# [geom_boxplot]ボックスプロットによる描画
gp = gp + geom_boxplot()

gp = gp + scale_x_discrete(name="Experiment")
gp = gp + scale_y_continuous(name="Time [sec]")

gp = gp + scale_fill_discrete(
  limits=c("1","2","3"),labels=c("1st","2nd","3rd"))

gp = gp + labs(title = "Graph2_ShutterBox",
               x = "Experiment", y = "Time[s]")

gp = gp + theme(
  aspect.ratio=3/2,
  text = element_text(face = "bold", size = 30),
  axis.text.x = element_text(size = 30),
  axis.text.y = element_text(size = 25),
  #凡例の位置の調整、背景を加える
  legend.position=c(1.05,1.00),
  legend.justification=c(0,1),
  legend.background =
    element_rect(fill = "white", colour = "black"))

gp
```



```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
# [Graph3_ShutterComma]
# コンマ何秒でシャッターを押していたか
#-----------------------------------------------------------------
#-----------------------------------------------------------------

# オリジナルのデータは476行
nrow(dat)
#[1] 476

# datには欠損値が42ある。
sum(is.na(dat$time))

# 欠損値を除いたdat_を作成（行数434）
dat_ = dat[is.na(dat$time)==FALSE,]
nrow(dat_)
# [1] 434

#コンマN秒に相当するtime2属性を追加したデータフレームdat2を作成
## roundは四捨五入、digitsは桁の指定
dat2 = mutate(dat_,time2 = round(10*(time - floor(time)),digits=1))

## 正しく変換できていることを確認
dat2[,c("time","time2")]
#     time time2
# 1    2.2     2
# 2    0.8     8
# 3    0.6     6
# 4    2.9     9
# 5    6.2     2
```

```
# 6    0.1     1
# 7   14.8     8
# 8    0.7     7
# 9    1.9     9
# ...

# expごとのtime2の集計
table(dat2$time2,dat2$exp)
#   Single Triple
# 0     17     42
# 1      5     25
# 2     16     32
# 3     13     32
# 4     10     35
# 5     11     28
# 6     11     20
# 7     12     31
# 8     14     33
# 9     10     37

# このテーブルをdata.frame化する
dat_comma = as.data.frame(table(dat2$time2,dat2$exp))
colnames(dat_comma) = c("n","exp","count");dat_comma
#    n    exp count
# 1  0 Single    17
# 2  1 Single     5
# 3  2 Single    16
# 4  3 Single    13
# 5  4 Single    10
# 6  5 Single    11
# 7  6 Single    11
# 8  7 Single    12
# 9  8 Single    14
# 10 9 Single    10
# 11 0 Triple    42
# 12 1 Triple    25
# 13 2 Triple    32
# 14 3 Triple    32
# 15 4 Triple    35
# 16 5 Triple    28
# 17 6 Triple    20
# 18 7 Triple    31
# 19 8 Triple    33
# 20 9 Triple    37

#----------------------------------------
# Graph3A（X軸を0:9の順で）
#----------------------------------------

# number属性をfactorからintegerに変更します。
class(dat_comma$n)
#[1] "factor"
## - 文字列にしてから整数へ
## - !!）直接as.integerをするとファクタのIDが対象となります
dat_comma$n = as.integer(as.character(dat_comma$n))
class(dat_comma$n)
#[1] "integer"


gp_a = ggplot(dat_comma,aes(x=n,y=count,colour=exp))
gp_a = gp_a + geom_line(size=1.5)
gp_a = gp_a + geom_point(size=10)
gp_a = gp_a + scale_x_continuous(limits=c(-0.5,9.5),breaks=0:9)
gp_a = gp_a + scale_y_continuous(limits=c(0,50),breaks=seq(0,50,by=10))
gp_a = gp_a + labs(title="Graph3A_ShutterComma",x="Comma.N",y="Count")
#g3 = g3 + scale_colour_discrete()
gp_a = gp_a + theme(
  aspect.ratio=3/4,
  text = element_text(face = "bold", size = 36),
  axis.text.x = element_text(size = 30),
  axis.text.y = element_text(size = 25)
)
gp_a #out Graph3A_ShutterComma


#----------------------------------------
```

```
# Graph3B（X軸を6,7,8,9,0,1,2,3,4,5の順で）
#----------------------------------------

# n = c(0,1,2,3,4,5,6,7,8,9)を
# n2 = c(4,5,6,7,8,9,0,1,2,3)にマッピング
## n2の0:9が、nの6:9,0:5に対応していることに注意

dat_comma$n
dat_comma = mutate(dat_comma,
                   n2 = if_else(n>=6,n-6,n+4))


gp_b = ggplot(dat_comma,aes(x=n2,y=count,colour=exp))
gp_b = gp_b + geom_vline(xintercept = 4, linetype = "dotted",
                         size=1.5,colour="black")

gp_b = gp_b + geom_line(size=1.5)
gp_b = gp_b + geom_point(size=10)

# ラベルをずらしてつけるのがポイント
gp_b = gp_b + scale_x_continuous(
  limits=c(-0.5,9.5),breaks=0:9,
  labels=c(6:9,0:5));
gp_b = gp_b + scale_y_continuous(limits=c(0,50),breaks=seq(0,50,by=10))

gp_b = gp_b + labs(title="Graph3B_ShutterComma",x="Comma.N",y="Count");
gp_b = gp_b + theme(
  aspect.ratio=3/4,
  text = element_text(face = "bold", size = 36),
  axis.text.x = element_text(size = 30),
  axis.text.y = element_text(size = 25)
)
gp_b #out Graph3B_ShutterComma









#----------------------------------------------------------------
#----------------------------------------------------------------
# [Graph4_Correlation]
# シャッターとシャッターの時間差の相関（Triple実験）
#----------------------------------------------------------------
#----------------------------------------------------------------

# Triple実験のデータフレームを取得
dat_tri = dat[dat$exp=="Triple",]

# 以下、dat_として参照します。
dat_ = dat_tri;

#席順に並べる（昇順に）
dat_ = arrange(dat_,seat)
#arrange(dat_,desc(seat)) #降順
dat_ = dat_[,c("seat","seatx","seaty","time","order")];

dat_[1:6,]
# seat seatx seaty time    exp order
# 1    4     1     4  3.2 Triple     1
# 2    4     1     4  6.9 Triple     2
# 3    4     1     4 10.1 Triple     3
# 4    5     1     5   NA Triple     1
# 5    5     1     5   NA Triple     2
# 6    5     1     5   NA Triple     3

# orderごとのデータを切り出す
dat_t1 = dat_[dat_$order==1,]; dat_t1
dat_t2 = dat_[dat_$order==2,]; dat_t2
dat_t3 = dat_[dat_$order==3,]; dat_t3
```

```
# これらを列でバインド
dat_ =
  cbind(dat_t1[,c("seat","seatx","seaty","time")],
        dat_t2[,c("time")],
        dat_t3[,c("time")])

# 列名を変更
colnames(dat_) = c("seat","seatx","seaty","time1","time2","time3")
dat_[1:6,]
#    seat seatx seaty time1 time2 time3
# 1     4     1     4   3.2   6.9  10.1
# 4     5     1     5    NA    NA    NA
# 7     9     1     9   0.5   3.6  19.5
# 10   13     2     1   3.4  10.0  13.0
# 13   28     3     4   5.0   9.0  10.2
# 16   36     3    12   2.7   6.1   9.0

# シャッター1からシャッタ2の時間差をtime21
# シャッター2からシャッタ3の時間差をtime32として属名を追加
dat_ = mutate(dat_,time21 = time2-time1)
dat_ = mutate(dat_,time32 = time3-time2)
dat_[1:6,]
#    seat seatx seaty time1 time2 time3 time21 time32
# 1     4     1     4   3.2   6.9  10.1    3.7    3.2
# 4     5     1     5    NA    NA    NA     NA     NA
# 7     9     1     9   0.5   3.6  19.5    3.1   15.9
# 10   13     2     1   3.4  10.0  13.0    6.6    3.0
# 13   28     3     4   5.0   9.0  10.2    4.0    1.2
# 16   36     3    12   2.7   6.1   9.0    3.4    2.9


#-----------------------------------------
# Graph4A（time1 と time21の相関図）
#-----------------------------------------

gp_a = ggplot(dat_, aes(x = time1, y = time21))

# [geom_point] 散布図（同一行のxとyの関係）
gp_a = gp_a + geom_point(size=8,shape=21)

# [geom_smooth] 線形回帰直線
gp_a = gp_a + geom_smooth(method="lm",fomula='y~x')

gp_a = gp_a + scale_x_continuous(limits=c(0,10),breaks=c(0,5,10))
gp_a = gp_a + scale_y_continuous(limits=c(0,10),breaks=c(0,5,10))

# [geom_segment] 任意の直線
# (x,y) to (xend,yend)の座標間に線を引く
gp_a = gp_a + geom_segment(x=0,y=0,xend=10,yend=10,
                           size=0.5,linetype="dotted")

gp_a = gp_a + labs(title = "Graph4A_Correlation1",
                   x = "Time1", y = "Time1 to Time2")
gp_a = gp_a + theme(
  aspect.ratio=1/1,
  text = element_text(face = "bold", size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20))

gp_a #out Graph4A_Correlation1

#-----------------------------------------
# Graph4B（time21 と time32の相関図）
#-----------------------------------------

gp_b = ggplot(dat_, aes(x = time21, y = time32))
gp_b = gp_b + geom_point(size=8,shape=21)
gp_b = gp_b + geom_smooth(method="lm",fomula='y~x')
gp_b = gp_b + scale_x_continuous(limits=c(0,10),breaks=c(0,5,10))
gp_b = gp_b + scale_y_continuous(limits=c(0,10),breaks=c(0,5,10))
gp_b = gp_b + geom_segment(x=0,y=0,xend=10,yend=10,
                           size=0.5,linetype="dotted")
gp_b = gp_b + labs(title = "Graph4B_Correlation2",
                   x = "Time1 to Time2", y = "Time2 to Time3")
gp_b = gp_b + theme(
```

```
  aspect.ratio=1/1,
  text = element_text(face = "bold", size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20))

gp_b #out Graph4B_Correlation2


#install.packages(gridExtra)
library(gridExtra)
grid.arrange(gp_a,gp_b,nrow=1) #1行で並べる







#-----------------------------------------------------------------
#-----------------------------------------------------------------
# [Graph5_DirectionEffect]
# 隣接空間（方向）の影響
# 前の人に影響されるか？後ろの人に影響されるか？
#-----------------------------------------------------------------
#-----------------------------------------------------------------


# Single実験のデータを取り出す。
dat_si = dat[dat$exp=="Single",]
dat_ = dat_si

# 注目するのは、座席の位置と時間のみなので、それだけを切り出します。
dat_ = dat_[,c("seatx","seaty","time")]
dat_[1:6,]

# seatx seaty time
# 1    16    12  2.2
# 2    14    12  0.8
# 3     5    11  0.6
# 4     5    12  2.9
# 5     8     5  6.2
# 6    15    12  0.1



# まずは隣接関係にあるものたちのシャッター時間を算出する関数を作成します。
## 引数は、自分の座席のxy座標です。
## timeForward：前方（最大）3人の平均シャッター時間の算出
## timeNeighbor：両隣（最大）2人の平均シャッター時間の算出
## timeBackward：後方（最大）3人の平均シャッター時間の算出

timeForward = function(sx,sy){
  forward = dat_[dat_$seatx>=sx-1 & dat_$seatx<=sx+1 & dat_$seaty==sy-1,]
  mean(forward$time) #平均値（引数が空の場合NA）
}
timeNeighbor = function(sx,sy){
  neighbor = dat_[(dat_$seatx==sx-1 | dat_$seatx==sx+1) & dat_$seaty==sy,]
  mean(neighbor$time) #平均値（引数が空の場合NA）
}
timeBackward = function(sx,sy){
  backward = dat_[dat_$seatx>=sx-1 & dat_$seatx<=sx+1 & dat_$seaty==sy+1,]
  mean(backward$time) #平均値（引数が空の場合NA）
}


# dat_の各行に、
# 3方向区間のシャッター時間（time.f, time.n, time.b）を付加します。
#-------------------------------------------------------------------------

## mutateはなぜかうまくいなかい、、。（なぜだ？）
# mutate(dat_,time.f = timeforward(seatx,seaty))
# mutate(dat_,time.n = timeneighbor(seatx,seaty))
# mutate(dat_,time.b = timebackward(seatx,seaty))
```

```
## dat_と同じ行数のベクトルを3つ作成
time.f = vector("double",nrow(dat_))
time.n = vector("double",nrow(dat_))
time.b = vector("double",nrow(dat_))

## それぞれに、3方向区間の撮影時間を代入
for(i in 1:nrow(dat_)){
  seatx = dat_[i,c("seatx")]
  seaty = dat_[i,c("seaty")]
  time.f[i]  = timeForward(seatx,seaty)
  time.n[i]  = timeNeighbor(seatx,seaty)
  time.b[i]  = timeBackward(seatx,seaty)
}

## データフレームの属性を新たに追加
dat_$time.f = time.f
dat_$time.n = time.n
dat_$time.b = time.b

## 最初の6行を確認します。
## NaNは対象区間に誰もいない場合と対応します。
dat_[1:6,]
#   seatx seaty time    time.f time.n time.b
# 1    16    12  2.2  0.300000   1.15    NaN
# 2    14    12  0.8 20.000000   9.60    NaN
# 3     5    11  0.6       NaN    NaN   2.45
# 4     5    12  2.9  0.600000   2.00    NaN
# 5     8     5  6.2  8.566667  13.50  15.80
# 6    15    12  0.1 10.150000   1.50    NaN

#-------------------------------------------------------------------------


# 3区間（前方・隣接・後方）の撮影時間と比べて
# 早くシャッターを押したか（EARLY）か遅く押したか（LATE）
# mutate関数で、新たに属性を追加し並べる
dat_ = mutate(dat_, order.f = if_else(time - time.f>=0,"LATE","EARLY"))
dat_ = mutate(dat_, order.n = if_else(time - time.n>=0,"LATE","EARLY"))
dat_ = mutate(dat_, order.b = if_else(time - time.b>=0,"LATE","EARLY"))

# 最初の20行を, 1-3,7-9列目のみを確認
dat_[1:20,c(1:3,7:9)]
#      seatx seaty time order.f order.n order.b
#   1     16    12  2.2    LATE    LATE    <NA>
#   2     14    12  0.8   EARLY   EARLY    <NA>
#   3      5    11  0.6    <NA>    <NA>   EARLY
#   4      5    12  2.9    LATE    LATE    <NA>
#   5      8     5  6.2   EARLY   EARLY   EARLY
#   6     15    12  0.1   EARLY   EARLY    <NA>
#   7     19     7 14.8    LATE    LATE    LATE
#   8     18     9  0.7    <NA>   EARLY   EARLY
#   9     18    10  1.9   EARLY    <NA>   EARLY
#   10    16    11  0.3   EARLY    <NA>   EARLY
#   11    17     9  7.4    LATE    LATE    LATE
#   12    16     9  9.8    LATE    LATE   EARLY
#   13     6     7  2.5    <NA>   EARLY   EARLY
#   14     6     8  6.1   EARLY    LATE    LATE
#   15     7    10  3.0    LATE    LATE   EARLY
#   16     7     9  3.7    LATE    LATE    LATE
#   17    13    12 19.1    LATE    LATE    <NA>
#   18    17    12  2.2    LATE    LATE    <NA>
#   19    11     9  9.7    LATE    LATE   EARLY
#   20    11    10 16.8    LATE    LATE    LATE

##（例）
### - 席(8,5)は前方・両隣・後方よりも早くシャッターを押している。
#   5      8     5  6.2   EARLY   EARLY   EARLY
### - 席(7,10)は前方・両隣よりも遅くシャッターを押し、
### - 後方よりも早くシャッターを押している。
#   15     7    10  3.0    LATE    LATE   EARLY


# 前方との比較：ほぼ変わらない
table(dat_$order.f)
```

```
# EARLY  LATE
# 43    44

# 両隣との比較：ほぼ変わらない
table(dat_$order.n)
# EARLY  LATE
# 43    44

# 後方との比較：後方よりも早く押す傾向
table(dat_$order.b)
# EARLY  LATE
# 52    36

# グラフにするために、新たなデータフレームを作成

dat_dir = rbind(
  data.frame(direction = "FORWARD", dorder = dat_$order.f),
  data.frame(direction = "NEIGHBOR", dorder = dat_$order.n),
  data.frame(direction = "BACKWARD", dorder = dat_$order.b))

# NAはとる
dat_dir = dat_dir[is.na(dat_dir$dorder)==FALSE,]
dat_dir

#     direction dorder
# 1   FORWARD   LATE
# 2   FORWARD  EARLY
# 4   FORWARD   LATE
# ...
# 120  NEIGHBOR   LATE
# 121  NEIGHBOR  EARLY
# 123  NEIGHBOR   LATE
# ...
# 241  BACKWARD  EARLY
# 243  BACKWARD  EARLY
# 245  BACKWARD   LATE

## x軸をgenderに、枠線と塗りつぶしをpreferenceに設定
gp = ggplot(dat_dir,aes(x=direction, fill=dorder))
gp = gp + geom_bar(position=position_dodge(width=0.9), colour="black")
gp = gp + scale_x_discrete(limits=c("FORWARD","NEIGHBOR","BACKWARD"))
gp = gp + scale_y_continuous(limits=c(0,60))
gp = gp + scale_fill_discrete(limits=c("EARLY","LATE"),name="ORDER")
gp = gp + labs(title = "Graph5_DirectionEffect",
                x = "DIRECTION", y = "COUNT")
gp = gp + theme(
  aspect.ratio=2/3,
  text = element_text(face = "bold", size = 30),
  axis.text.x = element_text(size = 20),
  axis.text.y = element_text(size = 20))

gp #out Graph5_DirectionEffect

#（発展）カイ二乗分布
## p>0.05で、方向による影響は統計的には認められない
chisq.test(dat_dir$direction, dat_dir$dorder, correct=F)
# data:  df_dir$direction and df_dir$dorder
# X-squared = 2.1902, df = 2, p-value = 0.3345




#----------------------------------------
# [確認課題]
#----------------------------------------

#----------------------------------------
# [Work5_CommaDistribution]
# 20秒間で10回押すフリーシャッター実験の結果を用いて、
# 「コンマN秒」の分布を作成していください。
# この際、前半1~5回目のシャッターを「1ST」
# 後半6~10回目のシャッターを「2ND」として、
```

```
# 前後半別で分布を重ねて示してください。

# 締切は11月中とします.
# ファイル名は「2250xx_work5.R」としてください。
#-----------------------------------------

# [10times フリーシャッター実験]
# 20秒の間にシャッターを10回押してください実験の結果（24年7月11日）

url = "https://lab.kenrikodaka.com/_download/csv/freeshutter_240711.csv"
dat_240711 = read.csv(url)

# 参加者は109人
nrow(dat_240711)
#[1] 109

# 最初と最後の6行
dat_240711[c(1:6,104:109),]
#     seatx seaty  t1  t2  t3  t4   t5   t6   t7   t8   t9  t10
# 1       9     2 0.3 1.1 1.4 1.6  1.7  2.1  2.6  2.7  3.6  5.1
# 2      12     9 0.4 1.2 1.6 2.3  3.0 13.3 14.0 15.0 16.2 19.0
# 3       4     4 1.2 2.9 4.9 6.8  9.0 10.7 12.5 15.0 17.2 19.2
# 4      10     2 6.0 6.5 6.6 6.9  7.2  7.5  7.7  7.9  8.2  8.4
# 5       6    14 1.3 2.7 3.2 8.2  8.9 11.2 13.3 17.3 17.7 18.0
# 6       9     6 1.1 1.5 2.1 2.4  3.5  4.0  5.4  7.7  9.8 11.1
# 104     9     1 1.1 2.4 3.8 5.5  6.9  8.3  9.2 10.8 12.1   NA
# 105    12     4 1.2 2.4 4.2 5.2  6.2  7.4  8.2  9.4 10.5 11.5
# 106    12    15 2.3 3.4 4.5 6.7 10.3 11.8 16.9 18.9 20.1 21.0
# 107    10    15 0.9 1.6 2.2 3.5  4.6  5.2  5.9  6.5  8.1  9.9
# 108    12    15 2.0 3.6 4.6 5.7 10.2 11.5 16.7 17.9 18.7 19.9
# 109    11     2 2.9 3.1 3.6 4.0  5.2 14.2 15.7 16.0 16.7 18.8


## seatx：座席の水平方向（生徒視点で左から1,2,.....）
## seaty：座席の奥行き方向（教卓に近い方から1,2,...）
## t1, t2, t3, ...：1回目のシャッター時間, 2回目のシャッター時間, ...
## NAは欠損データ（例えば、104行目のt10）


#-----------------------------------------------------------------
# 途中までヒント（必ずしも、以下の方法に従う必要はありません）
#-----------------------------------------------------------------


# 以下、dat_として扱います。
dat_ = dat_240711

# t1~t10について、少数第一位の数字を求めます。

comma1 = round(10*(dat_$t1-floor(dat_$t1)),digits=1)
comma2 = round(10*(dat_$t2-floor(dat_$t2)),digits=1)
comma3 = round(10*(dat_$t3-floor(dat_$t3)),digits=1)
comma4 = round(10*(dat_$t4-floor(dat_$t4)),digits=1)
comma5 = round(10*(dat_$t5-floor(dat_$t5)),digits=1)
comma6 = round(10*(dat_$t6-floor(dat_$t6)),digits=1)
comma7 = round(10*(dat_$t7-floor(dat_$t7)),digits=1)
comma8 = round(10*(dat_$t8-floor(dat_$t8)),digits=1)
comma9 = round(10*(dat_$t9-floor(dat_$t9)),digits=1)
comma10 = round(10*(dat_$t10-floor(dat_$t10)),digits=1)


# これ以降は、Graph3Aを参考にすすめてください。
```

R
第６講

グラフの実践３：連続計測値の整形
（ブッダの耳錯覚実験）

```
########################################
########################################
####### グラフ４（実践３） #############
####### ブッダの耳錯覚実験 #############
########################################
########################################

# 主要なライブラリを読み込みます

#install.packages(ggplot2)
library(ggplot2)

#install.packages(dplyr)
library(dplyr)


# まずは以下のCSVを読み込んでください。

url = "https://lab.kenrikodaka.com/_download/csv/buddhaexp2024.csv"
source = read.csv(url)

str(source)

# 'data.frame':       16 obs. of  29 variables:
# $ SBJ   : int  1 2 3 4 5 6 7 8 9 10 ...
# $ Q1T0B0: num  1 1.5 0 1 2 0 5 1 1.5 0.5 ...
# $ Q1T0B1: num  3.5 4 5 3 3 1 5.5 2 4 3 ...
# $ Q1T1B0: num  2.5 1.5 0 3 4 0 4.5 1 2.5 0 ...
# $ Q1T1B1: num  5.5 4.5 6 3.5 5.5 1.5 5.5 3.5 5.5 3 ...
# $ Q2T0B0: num  0 2.5 0 1.5 2 0 5 1 1.5 1.5 ...
# $ Q2T0B1: num  2.5 3 3.5 0 4 0 5.5 1 3 4 ...
# $ Q2T1B0: num  1.5 1.5 0 1.5 4 0 5 1.5 2 2.5 ...
# $ Q2T1B1: num  3 4.5 4.5 3 5.5 0 5 1 4 6 ...
# $ Q3T0B0: num  0 0.5 0 0.5 0 0 2 0 0 0 ...
# $ Q3T0B1: num  2.5 1.5 0 1.5 0 0 1.5 1 1.5 0 ...
# $ Q3T1B0: num  1.5 0 0 3 0.5 0 1 0 0 0 ...
# $ Q3T1B1: num  3 0 0 3 0 0 1 0 1 0.5 ...
# $ Q4T0B0: num  0 0 0 2.5 0 0 0 1.5 0 0 ...
# $ Q4T0B1: num  0 0 0 2 0.5 0 0 0.5 1.5 0 ...
# $ Q4T1B0: num  0 0 0 2.5 1.5 0 0 0.5 0 0 ...
# $ Q4T1B1: num  0 0 0 2 0.5 0 0 0 1 0 ...
# $ E0T0B0: num  4.25 13.15 -0.25 3.6 -2.1 ...
# $ E0T0B1: num  15.45 14.15 3.05 8.2 -1.3 ...
# $ E0T1B0: num  20.1 11.1 1.1 -1.5 1.4 4.15 5.2 -0.4 5.5 1.25 ...
# $ E0T1B1: num  1.7 11.1 0.2 -0.2 3.3 0.75 6.4 -0.2 5 2.95 ...
# $ E1T0B0: num  36.3 16.55 4.55 5.6 -1.95 ...
# $ E1T0B1: num  49 55 55.5 9.1 40.9 ...
# $ E1T1B0: num  32.8 41.4 4.8 5.2 30 ...
# $ E1T1B1: num  50.2 46.4 77.3 6.8 75 ...
# $ ORDER : int  0 1 0 0 1 0 1 1 0 1 ...


# SBJ 被験者のID（1-16）

# 変数名[XX][TaBb]の説明

# [XX] Q1-Q4はアンケートの設問に対する主観評価値（0-6の7段階）
## （主観評価実験）
## Q1:「自分の耳たぶが通常よりも伸びている感じがした。」
## Q2:「何も無い空間に「見えない」皮膚が存在しているように感じた。」
## Q3:「自分の耳全体が下に移動しているような感じがした。」
## Q4:「自分の耳全体の面積が小さくなる感じがした。」
## （行動実験）
## E0:耳上端の主観位置の錯覚前後の降下量（cm）
## E1:耳下端の主観位置の錯覚前後の降下量（cm）

# [TaBb] Taは体験者側、Bbは実験者側の条件
## T0B0:（体験者が）耳上端に触れない x （実験者が）耳下端をつまむ（だけ）
## T0B1:（体験者が）耳上端に触れない x （実験者が）耳下端の下を引っ張る
## T1B0:（体験者が）耳上端の上をつまむ x （実験者が）耳下端をつまむ（だけ）
## T0B1:（体験者が）耳上端の上をつまむ x （実験者が）耳下端の下を引っ張る
```

```
# ORDER 行動実験の順序（今回の演習では、ORDERは無視します）
## 0：T1B1→T0B1（錯覚条件に関してT1が先、その後にT0）
## 1：T0B1→T1B1（錯覚条件に関してT0が先、その後にT1）



# データ数（16人のデータ）
n = nrow(source)
# [1] 16


#------------------------------------------------------------
# グラフ作成用に２種類のデータフレームを整形する

# dat_Q：アンケート実験に関するデータフレーム
## rate（評価値）、stmt（質問項目）、etop（耳上端条件）、ebot（耳下端条件）

# dat_B：行動実験に関するデータフレーム
## drift（位置変化量）、epos（変化部位）、etop（耳上端条件）、ebot（耳下端条件）
#------------------------------------------------------------


# sourceをdat_にコピーします。
dat_ = source

# 実験参加者ID（長さ：16x16）を引き延ばす
v_sbj = rep(dat_[,1],16)

# アンケートの全ての評価値を格納したvRate（長さ：16x16）を作成
# 2列目（Q1T0B0）から17列目（Q4T1B1）をつなげる
v_rate = dat_[,2]
for(i in 3:17){
  v_rate = c(v_rate,dat_[,i])
}

v_stmt = c(rep("Q1",16*4),rep("Q2",16*4),rep("Q3",16*4),rep("Q4",16*4))
v_etop = rep(c(rep("T0",16*2),rep("T1",16*2)),4)
v_ebot = rep(rep(c(rep("B0",16),rep("B1",16)),2),4)

# アンケートに対するデータフレームを作成
dat_Q = data.frame(sbj = v_sbj, rate = v_rate, stmt = v_stmt, etop = v_etop, ebot = v_ebot)
dat_Q[1:5,]
#   sbj rate stmt etop ebot
# 1   1  1.0   Q1   T0   B0
# 2   2  1.5   Q1   T0   B0
# 3   3  0.0   Q1   T0   B0
# 4   4  1.0   Q1   T0   B0
# 5   5  2.0   Q1   T0   B0


# 実験参加者ID（長さ：16x8）を引き延ばす
v_sbj = rep(dat_[,1],8)

# 行動実験の全ての位置変化量を格納したvDrift（長さ：16x8）を作成
# 18列目（Q1T0B0）から25列目（Q4T1B1）をつなげる
v_drift = dat_[,18]
for(i in 19:25){
  v_drift = c(v_drift,dat_[,i])
}

v_epos = c(rep("TOP",16*4),rep("BOT",16*4))
v_etop = rep(c(rep("T0",16*2),rep("T1",16*2)),2)
v_ebot = rep(rep(c(rep("B0",16),rep("B1",16)),2),2)

# 行動実験（ドリフト）に対するデータフレームを作成
dat_D = data.frame(sbj = v_sbj, drift = v_drift, epos = v_epos,
                   etop = v_etop, ebot = v_ebot)
dat_D[1:5,]
#   sbj drift epos etop ebot
# 1   1  4.25  TOP   T0   B0
# 2   2 13.15  TOP   T0   B0
# 3   3 -0.25  TOP   T0   B0
# 4   4  3.60  TOP   T0   B0
```

```
# 5   5 -2.10  TOP   T0   B0
```

```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
# [Graph1_QuestionnaireBoxplot]
# アンケート結果をボックスプロットでまとめる
#-----------------------------------------------------------------
#-----------------------------------------------------------------



# [ggplot]
## interactionは、複数の変数の組み合わせをグラフの属性にマップできる
## interaction(etop,ebot)は「T0.B0」「T1.B0」「T0.B1」「T1.B1」の4つの要素を作る
gp = ggplot(dat_Q,aes(x=interaction(etop,ebot),
                      y=rate,fill=interaction(etop,ebot)))

# [geom_boxplot]ボックスプロットによる描画
gp = gp + geom_boxplot()

gp = gp + scale_x_discrete(
  labels=c("T0B0","T1B0","T0B1","T1B1"))
gp = gp + scale_y_continuous(limits=c(0,6),breaks=0:6)
gp = gp + scale_fill_discrete(
  name = "CONDITION",
  labels=c("(T0B0) TOP:none, BOT:pinched",
           "(T1B0) TOP:self-pinched, BOT:pinched",
           "(T0B1) TOP:none, BOT:pulled",
           "(T1B1) TOP:self-pinched, BOT:pulled"))

gp = gp + labs(title = "Graph1_QuestionnaireBoxplot",
               x = "CONDITION", y = "RATE")

# gp = gp + facet_grid(stmt ~ .)
gp = gp + facet_grid(. ~ stmt)


gp = gp + theme(
  aspect.ratio=3/2,
  text = element_text(face = "bold", size = 30),
  axis.text.x = element_text(size = 25),
  axis.text.y = element_text(size = 25),
  #凡例の位置の調整、背景を加える
  legend.position=c(1,1.08),
  legend.justification=c(1,0),
  legend.background =
    element_rect(fill = "white", colour = "black"),
  legend.text = element_text(size = 30))

gp
```

```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
# [Graph2_QuestionnaireBar]
# アンケート結果をバープロットでまとめる
#-----------------------------------------------------------------
#-----------------------------------------------------------------

#個々の条件の平均値・標準誤差を算出する

# まずはdat_Qの構造を復習
## stmt（4）・etop（2）・ebot（2）の4x2x2=16通り
dat_Q[1:3,]
#   sbj rate stmt etop ebot
# 1   1  1.0   Q1   T0   B0
# 2   2  1.5   Q1   T0   B0
# 3   3  0.0   Q1   T0   B0

# 例えばQ1、T0、B0の平均値と標準誤差は以下の手順で算出

## (1)Q1,T0,B0のrateを全て取り出す(16人分)
rates = dat_Q[dat_Q$stmt=="Q1" & dat_Q$etop=="T0" & dat_Q$ebot=="B0",c("rate")];rates
#[1] 1.0 1.5 0.0 1.0 2.0 0.0 5.0 1.0 1.5 0.5 3.5 0.0 1.0 0.0 2.0 0.5

## (2)平均値の算出：meanを使う
mean(rates) #[1] 1.28125

## (3)標準誤差：標準偏差（sd）をサンプルサイズの平方根で割る
sd(rates) / sqrt(length(rates)) #[1] 0.3414697
### 以下も同じ（varは分散、sdは分散の平方根）
sqrt(var(rates)) / sqrt(length(rates)) #[1] 0.3414697


# 平均値・標準誤差の組を16通りの条件で一気に吐き出すために関数を定義

getMeanSE = function(Qn,Tn,Bn){

  rates = dat_Q[dat_Q$stmt==Qn & dat_Q$etop==Tn & dat_Q$ebot==Bn,c("rate")]
  mean = mean(rates)
  SE = sd(rates) / sqrt(length(rates))

  c(mean,SE)

}

# 列名だけ存在する空のデータフレームを作成
dat_stat = data.frame(mean=as.numeric(),se=as.numeric(),
                      stmt=as.character(),etop=as.character(),ebot=as.character())
dat_stat
# [1] mean se   stmt etop ebot
# <0 rows> (or 0-length row.names)

for(Qn in c("Q1","Q2","Q3","Q4")){
  for(Tn in c("T0","T1")){
    for(Bn in c("B0","B1")){
      mean.se = getMeanSE(Qn,Tn,Bn)
        add_data = data.frame(mean=mean.se[1],se=mean.se[2],stmt=Qn,etop=Tn,ebot=Bn)
        dat_stat = rbind(dat_stat,add_data)
    }
  }
}

# これで準備完了!
# >dat_stat
#       mean        se stmt etop ebot
# 1  1.28125 0.3414697   Q1   T0   B0
# 2  3.28125 0.3817961   Q1   T0   B1
# 3  1.68750 0.4327119   Q1   T1   B0
# 4  4.59375 0.3169475   Q1   T1   B1
# 5  1.25000 0.3505947   Q2   T0   B0
# 6  2.46875 0.4017378   Q2   T0   B1
# 7  1.53125 0.3776814   Q2   T1   B0
# 8  3.53125 0.4017378   Q2   T1   B1
# 9  0.34375 0.1562500   Q3   T0   B0
# 10 1.09375 0.2858421   Q3   T0   B1
# 11 0.46875 0.2163559   Q3   T1   B0
```

```
# 12 0.84375 0.3119787   Q3   T1   B1
# 13 0.25000 0.1767767   Q4   T0   B0
# 14 0.28125 0.1511673   Q4   T0   B1
# 15 0.28125 0.1765925   Q4   T1   B0
# 16 0.21875 0.1366927   Q4   T1   B1



#-----------------------------------------------------------------
# [Graph2A_QuestionnaireBar]
# エラーバーなし
#-----------------------------------------------------------------

gp = ggplot(dat_stat,aes(x=interaction(etop,ebot), y=mean,
                         fill=interaction(etop,ebot)))
# geom_barは数え上げ、geom_colは数値をそのまま表示することに注意
## position_dodgeは横に並べるの意味
gp = gp + geom_col(position=position_dodge(width=0.9))

gp = gp + scale_x_discrete(labels=c("T0B0","T1B0","T0B1","T1B1"))
gp = gp + scale_y_continuous(limits=c(0,6),breaks=0:6)
gp = gp + scale_fill_discrete(
  name = "CONDITION",
  labels=c("(T0B0) Top:none, Bot:pinched",
           "(T1B0) Top:self-pinched, Bot:pinched",
           "(T0B1) Top:none, Bot:pulled",
           "(T1B1) Top:self-pinched, Bot:pulled"))

gp = gp + facet_grid(. ~ stmt)

gp = gp + labs(title="Graph2A_QuestionnaireBar",x="CONDITION",y="RATE")

gp = gp + theme(
  aspect.ratio=3/2,
  text = element_text(face = "bold", size = 30),
  axis.text.x = element_text(size = 25),
  axis.text.y = element_text(size = 25),
  #凡例の位置の調整、背景を加える
  legend.position=c(1,1.08),
  legend.justification=c(1,0),
  legend.background =
    element_rect(fill = "white", colour = "black"),
  legend.text = element_text(size = 30))

gp


#-----------------------------------------------------------------
# [Graph2B_QuestionnaireBar]
# エラーバーあり
#-----------------------------------------------------------------

gp = ggplot(dat_stat,aes(x=interaction(etop,ebot), y=mean,
                         fill=interaction(etop,ebot)))

gp = gp + geom_col(position=position_dodge(width=0.9))

#[geom_errobar]
## aes内のyminはエラーバーの下限ymaxは上限に対応
## yminは「平均値−標準誤差」、ymaxは「平均値+標準誤差」に対応
## widthは横幅（1が棒グラフの横幅いっぱい）
gp = gp + geom_errorbar(aes(ymin = mean-se, ymax = mean+se), width=0.3)

gp = gp + scale_x_discrete(labels=c("T0B0","T1B0","T0B1","T1B1"))
gp = gp + scale_y_continuous(limits=c(0,6),breaks=0:6)
gp = gp + scale_fill_discrete(
  name = "CONDITION",
  labels=c("(T0B0) Top:none, Bot:pinched",
           "(T1B0) Top:self-pinched, Bot:pinched",
           "(T0B1) Top:none, Bot:pulled",
           "(T1B1) Top:self-pinched, Bot:pulled"))

gp = gp + facet_grid(. ~ stmt)

gp = gp + labs(title="Graph2B_QuestionnaireBar",x="CONDITION",y="RATE")
```

```
gp = gp + theme(
  aspect.ratio=3/2,
  text = element_text(face = "bold", size = 30),
  axis.text.x = element_text(size = 25),
  axis.text.y = element_text(size = 25),
  #凡例の位置の調整、背景を加える
  legend.position=c(1,1.08),
  legend.justification=c(1,0),
  legend.background =
    element_rect(fill = "white", colour = "black"),
  legend.text = element_text(size = 30))

gp
```

```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
# [Graph3_EarPositionDrift]
# 錯覚前後の耳（上端・下端）の主観位置の変化
#-----------------------------------------------------------------
#-----------------------------------------------------------------

# まずはdat_Dの構造を復習
## drift：錯覚前後の位置の変化
## epos：耳が上端（TOP）か、下端（BOT）か
## etop：耳の上を被験者がつままない（T0）or つまむ（T1）
## ebot：耳の下を実験者がつまむ（B0） or 引っ張る（B1）
## order：（0）T0→T1、（1）T1→T0
str(dat_D)
# 'data.frame':       128 obs. of  6 variables:
#   $ sbj  : int  1 2 3 4 5 6 7 8 9 10 ...
# $ drift: num  4.25 13.15 -0.25 3.6 -2.1 ...
# $ epos : chr  "TOP" "TOP" "TOP" "TOP" ...
# $ etop : chr  "T0" "T0" "T0" "T0" ...
# $ ebot : chr  "B0" "B0" "B0" "B0" ...
# $ order: int  0 1 0 0 1 0 1 1 0 1 ...


#-- 前準備 --
# fillにマップされるeposの並び順を左からTOP->BOTにする
dat_D$epos = factor(dat_D$epos,levels=c("TOP","BOT"))
#------------



# [ggplot]
gp = ggplot(dat_D,aes(x=interaction(etop,ebot),
                      y=drift,fill=epos))

# [geom_hline] y=0とy=50に横線を引く。solidは実線、dottedは点線
gp = gp + geom_hline(yintercept = 0, linetype="solid",linewidth=1)
gp = gp + geom_hline(yintercept = 50, linetype="dotted",linewidth=1)

# [geom_boxplot]ボックスプロットによる描画
gp = gp + geom_boxplot()
gp = gp + scale_x_discrete(
  limits=c("T0.B0","T1.B0","T0.B1","T1.B1"),
  labels=c("T0B0","T1B0","T0B1","T1B1"))

gp = gp + scale_x_discrete(
  limits=c("T0.B0","T1.B0","T0.B1","T1.B1"),
  labels=c("T0B0","T1B0","T0B1","T1B1"))

gp = gp + scale_y_continuous(limits=c(80,-5),breaks=seq(80,-10,by=-10),trans="reverse")

# 今回はbrewerパレットを使います（scale_fill_discreteと色の割り当てが異なる）。
# RColorBrewer::display.brewer.all() #色のパレットを見て、pairedを選ぶ。
gp = gp + scale_fill_brewer(
    name = "EAR'S POSITION", labels=c("TOP","BOTTOM"),palette="Paired")
# gp = gp + scale_fill_discrete(
#   name = "EAR'S POSITION", labels=c("TOP","BOTTOM"))

gp = gp + labs(title = "Graph3_EarPositionDrift",
               x = "CONDITION", y = "SUBJECTIVE EAR-LOCATION'S DRIFT(cm)")

gp = gp + theme(
  aspect.ratio=3/2,
  text = element_text(face = "bold", size = 30),
  axis.text.x = element_text(size = 25),
  axis.text.y = element_text(size = 25),
  #凡例の位置の調整、背景を加える
  legend.position=c(0.05,0.05),
  legend.justification=c(0,0),
  legend.background =
    element_rect(fill = "white", colour = "black"),
  legend.text = element_text(size = 30))

gp
```

```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
# [Graph4_EarSizeDeformation]
# 錯覚前後の耳のサイズの変化
#-----------------------------------------------------------------
#-----------------------------------------------------------------

# 最初に確認（eposはGraph3でFactor化されていることに注意）
str(dat_D)
# 'data.frame':       128 obs. of  6 variables:
# $ sbj  : int  1 2 3 4 5 6 7 8 9 10 ...
# $ drift: num  4.25 13.15 -0.25 3.6 -2.1 ...
# $ epos : Factor w/ 2 levels "TOP","BOT": 1 1 1 1 1 1 1 1 1 1 ...
# $ etop : chr  "T0" "T0" "T0" "T0" ...
# $ ebot : chr  "B0" "B0" "B0" "B0" ...
# $ order: int  0 1 0 0 1 0 1 1 0 1 ...


# 耳のサイズの（下方向）変化（dsize）はebotからetopを引いたもの
# dsize = ebot - etopとして、新たな属性dsizeを追加しdat_D2とする


# 特定の参加者（id）の、TnBn条件におけるサイズ変化を計算する
getSizeDeformation = function(id,Tn,Bn){

  drift_eartop = dat_D[dat_D$sbj==id & dat_D$etop==Tn
                       & dat_D$ebot==Bn & dat_D$epos=="TOP",c("drift")]
  drift_earbot = dat_D[dat_D$sbj==id & dat_D$etop==Tn
                       & dat_D$ebot==Bn & dat_D$epos=="BOT",c("drift")]

  dsize = drift_earbot - drift_eartop
  dsize

}

# [使い方]
## 参加a者1のT1B1におけるサイズ変化は48.5cm
getSizeDeformation(1,"T1","B1") #[1] 48.5
## 参加者13のT1B0におけるサイズ変化は3.35cm
getSizeDeformation(13,"T1","B0") #[1] 3.35


#全ての被験者、条件を一気に計算し、データフレームに流し込む

## (1) まずは空のデータフレームを作成
dat_dsize = data.frame(sbj=as.numeric(),dsize=as.numeric(),
                       etop=as.character(),ebot=as.character())
# > dat_esize
# [1] sbj   dsize etop  ebot
# <0 rows> (or 0-length row.names)




## (2) 繰り返し文で一気に計算
## - rbind（追加元DF、追加するDF）による行の結合を行います
for(id in 1:16){
  for(Tn in c("T0","T1")){
    for(Bn in c("B0","B1")){

      dsize = getSizeDeformation(id,Tn,Bn)
      print(dsize)
      dat_dsize = rbind(dat_dsize,
                        data.frame(sbj=id,dsize=dsize,etop=Tn,ebot=Bn))

    }
  }
}

# 準備完了!
dat_dsize

str(dat_dsize)
# 'data.frame':       64 obs. of  4 variables:
```

```
# $ sbj  : int  1 1 1 1 2 2 2 2 3 3 ...
# $ dsize: num  32.1 33.6 12.7 48.5 3.4 ...
# $ etop : chr  "T0" "T0" "T1" "T1" ...
# $ ebot : chr  "B0" "B1" "B0" "B1" ...


# [ggplot]
gp = ggplot(dat_dsize,aes(x=interaction(etop,ebot),
                      y=dsize,fill=interaction(etop,ebot)))

# [geom_boxplot]ボックスプロットによる描画
gp = gp + geom_boxplot()

gp = gp + scale_x_discrete(
  labels=c("T0B0","T1B0","T0B1","T1B1"))
gp = gp + scale_y_continuous(limits=c(0,100),breaks=seq(0,100,by=10))
gp = gp + scale_fill_discrete(
  name = "CONDITION",
  labels=c("(T0B0) top:none, bot:pinched",
           "(T1B0) top:self-pinched, bot:pinched",
           "(T0B1) top:none, bot:pulled",
           "(T1B1) top:self-pinched, bot:pulled"))

gp = gp + labs(title = "Graph4_EarSizeDeformation",
               x = "CONDITION", y = "DEFORMATION SIZE(cm)")

gp = gp + theme(
  aspect.ratio=3/2,
  text = element_text(face = "bold", size = 30),
  axis.text.x = element_text(size = 25),
  axis.text.y = element_text(size = 25),
  #凡例の位置の調整、背景を加える
  legend.position=c(0.02,0.98),
  legend.justification=c(0,1),
  legend.background =
    element_rect(fill = "white", colour = "black"),
  legend.text = element_text(size = 25))

gp
```

```
#-----------------------------------------------------------------
#-----------------------------------------------------------------
# [確認課題]
#-----------------------------------------------------------------
#-----------------------------------------------------------------

#----------------------------------------
# [Work6_EarSizeDeformationBar]

# Graph4_EarSizeDeformationを
# エラーバー付きの棒グラフにしてください。
# 締切は11月中とします.
# ファイル名は「2250xx_work6.R」としてください。
#----------------------------------------



#-----------------------------------------------------------------
# 途中までヒント（必ずしも、以下の方法に従う必要はありません）
#-----------------------------------------------------------------

# 以下、dat_として扱います。
dat_ = dat_dsize
dat_dsize[1:3,]

# 平均値・標準誤差の組を4通りの条件で一気に吐き出すために関数を定義（上書き）
getMeanSE = function(Tn,Bn){
  ## 16人分の変形量をベクトルで取り出す
  dsizes = dat_[dat_$etop==Tn & dat_$ebot==Bn,c("dsize")]
  mean = mean(dsizes)
  SE = sd(dsizes) / sqrt(length(dsizes))

  c(mean,SE)

}

# 以降は[Graph2_QuestionnaireBar] を参考にすすめてください。
```

```
#-------------------------------------------------------------
#-------------------------------------------------------------
# （補講：パイプ演算子、統計量の要約）

# [%>%]
# [group_by(data_frame, 列属性1, 列属性2, ...)]
# [summarize（arg1 = mean(計測値属性), arg2 = sd(計測値属性), ...）]

# パイプ演算子を使って、
# 属性毎の統計値をデータフレームとして一気に出力する
#-------------------------------------------------------------
#-------------------------------------------------------------

# group_by関数を使うためにはdplyrライブラリが必要です。

#install.package(dplyr)
library(dplyr)


#-----------------------------------------
# パイプ演算子 %>%
#-----------------------------------------

# アンケート実験の結果
dat_Q
#     sbj rate stmt etop ebot
# 1     1  1.0   Q1   T0   B0
# 2     2  1.5   Q1   T0   B0
# 3     3  0.0   Q1   T0   B0
# ...
# 198   6  0.0   Q4   T0   B0
# 199   7  0.0   Q4   T0   B0
# 200   8  1.5   Q4   T0   B0

# （復習）新しい列属性（sbj2）を追加する方法
mutate(dat_Q,sbj2 = sbj-1)
#     sbj rate stmt etop ebot sbj2
# 1     1  1.0   Q1   T0   B0    0
# 2     2  1.5   Q1   T0   B0    1
# 3     3  0.0   Q1   T0   B0    2
#...

# 同じことを、異なる記法（パイプ演算子）でも表現できます。

dat_Q %>%
  mutate(sbj2 = sbj-1)
#     sbj rate stmt etop ebot sbj2
# 1     1  1.0   Q1   T0   B0    0
# 2     2  1.5   Q1   T0   B0    1
# 3     3  0.0   Q1   T0   B0    2
#...


# (1)と(2)は同じ意味です。

# (1)
# dat %>%
#   func(arg1,arg2,...)

# (2)
# func(dat,arg1,arg2,...)


# 階層はいくらでも深くできます。
# （この場合、明らかに(1)の方が視認性が高くなります）

# (1)
# dat %>%
#   func1(arg1,arg2,...) %>%
#   func2(arg1,arg2,...)

# (2)
# fun2(func1(dat,arg1,arg2,...),arg1,arg2,...)
```

```
#-----------------------------------------
# group_by, summarize
#-----------------------------------------

dat_Q
#     sbj rate stmt etop ebot
# 1     1  1.0   Q1   T0   B0
# 2     2  1.5   Q1   T0   B0
# 3     3  0.0   Q1   T0   B0
# ...

# dat_Qの列属性の構造は以下の通り
# sbj : 1,2,3,..,16（16人）
# stmt: Q1,Q2,Q3,Q4（4種類：質問の種類）
# etop: T0,T1（2種類：耳上端条件）
# ebot: B0,B1（2種類：耳下端条件）

# stmtごとの平均値の出力（4条件を無視）

## group_byで、グループ属性（stmt）を指定し、
## その結果をsummarizeにパイプし、
## その中で、計算したい統計値（mean:平均値）を指定する

dat_Q %>%
  group_by(stmt) %>%
  summarize(
    mean = mean(rate)
  )

# A tibble: 4 × 2
# stmt   mean
# <chr> <dbl>
# 1 Q1    2.71
# 2 Q2    2.20
# 3 Q3    0.688
# 4 Q4    0.258

# stmt x etop x ebotごとの平均値の出力（全16グループ）

dat_Q %>%
  group_by(stmt,etop,ebot) %>%
  summarize(
    mean = mean(rate)
  )

# # A tibble: 16 × 4
# # Groups:   stmt, etop [8]
# stmt  etop  ebot   mean
# <chr> <chr> <chr> <dbl>
#  1 Q1    T0    B0    1.28
#  2 Q1    T0    B1    3.28
#  3 Q1    T1    B0    1.69
#  4 Q1    T1    B1    4.59
#  5 Q2    T0    B0    1.25
#  6 Q2    T0    B1    2.47
#  7 Q2    T1    B0    1.53
#  8 Q2    T1    B1    3.53
#  9 Q3    T0    B0    0.344
# 10 Q3    T0    B1    1.09
# 11 Q3    T1    B0    0.469
# 12 Q3    T1    B1    0.844
# 13 Q4    T0    B0    0.25
# 14 Q4    T0    B1    0.281
# 15 Q4    T1    B0    0.281
# 16 Q4    T1    B1    0.219


## 標準誤差も合わせて求める

### sd(rate)はrateの標準偏差、n()はサンプル数

dat_Q %>%
  group_by(stmt,etop,ebot) %>%
```

```
  summarize(
    mean = mean(rate),
    sd = sd(rate),
    n = n(),
    se = sd / sqrt(n))

#  stmt  etop  ebot   mean    sd     n    se
#    <chr> <chr> <chr> <dbl> <dbl> <int> <dbl>
#  1 Q1    T0    B0    1.28  1.37     16 0.341
#  2 Q1    T0    B1    3.28  1.53     16 0.382
#  3 Q1    T1    B0    1.69  1.73     16 0.433
#  4 Q1    T1    B1    4.59  1.27     16 0.317
#  5 Q2    T0    B0    1.25  1.40     16 0.351
#  6 Q2    T0    B1    2.47  1.61     16 0.402
#  7 Q2    T1    B0    1.53  1.51     16 0.378
#  8 Q2    T1    B1    3.53  1.61     16 0.402
#  9 Q3    T0    B0    0.344 0.625    16 0.156
# 10 Q3    T0    B1    1.09  1.14     16 0.286
# 11 Q3    T1    B0    0.469 0.865    16 0.216
# 12 Q3    T1    B1    0.844 1.25     16 0.312
# 13 Q4    T0    B0    0.25  0.707    16 0.177
# 14 Q4    T0    B1    0.281 0.605    16 0.151
# 15 Q4    T1    B0    0.281 0.706    16 0.177
# 16 Q4    T1    B1    0.219 0.547    16 0.137

# 出力結果をデータフレームに変換するには
# 以下のようにすればよい。

tmp = dat_Q %>%
  group_by(stmt,etop,ebot) %>%
  summarize(
    mean = mean(rate),
    sd = sd(rate),
    n = n(),
    se = sd / sqrt(n))

dat_stat = as.data.frame(tmp)
# > dat_stat
#     stmt etop ebot    mean        sd  n        se
# 1    Q1   T0   B0 1.28125 1.3658788 16 0.3414697
# 2    Q1   T0   B1 3.28125 1.5271842 16 0.3817961
# 3    Q1   T1   B0 1.68750 1.7308476 16 0.4327119
#...



# 同様にdat_D関連の統計値を要約する

dat_D
#     sbj  drift epos etop ebot
# 1     1  4.250  TOP   T0   B0
# 2     2 13.150  TOP   T0   B0
# 3     3 -0.250  TOP   T0   B0
# ...
# 126  14 41.100  BOT   T1   B1
# 127  15 42.600  BOT   T1   B1
# 128  16 39.000  BOT   T1   B1

## drift量の統計量

dat_D %>%
  group_by(epos,etop,ebot) %>%
  summarize(
    mean = mean(drift),
    sd = sd(drift),
    n = n(),
    se = sd / sqrt(n)
  ) %>%
  as.data.frame()

#   epos etop ebot      mean        sd  n        se
# 1  BOT   T0   B0  8.240937 10.012798 16 2.5031994
# 2  BOT   T0   B1 32.172187 19.936722 16 4.9841805
# 3  BOT   T1   B0 14.928125 13.671990 16 3.4179976
# 4  BOT   T1   B1 36.621875 25.877306 16 6.4693266
# 5  TOP   T0   B0  1.678125  3.902904 16 0.9757259
```

```
# 6  TOP   T0   B1  4.028125  4.982870 16 1.2457174
# 7  TOP   T1   B0  4.421875  6.185762 16 1.5464406
# 8  TOP   T1   B1  3.265625  3.781478 16 0.9453694
```