統計解析ソフトRを用いたデータ解析
DAY.1

下川敏雄

和歌山県立医科大学 医学部／附属病院臨床研究センター

シェーマ

日程	内容
1日目(1)	データの要約の方法(1):連続尺度の場合
1日目(2)	データの要約の方法(2):離散尺度(カテゴリカル・データ)の場合
1日目(3)	プログラミング(その1):データの要約の方法
2日目(1)	統計的推測(1):連続尺度の場合
2日目(2)	統計的推測(2):離散尺度(カテゴリカル・データ)の場合
2日目(3)	プログラミング(その2):統計的推測

DAY.1：記述統計学とグラフの書き方

Section.1：連続尺度の要約とグラフ表示

CSVファイルの読み込み

ここでは，Cドライブの`Fukuoka_Seminor`というフォルダにある`data1.csv`というCSVファイルを読み込む．これは，2019年度の福岡県の市区町村別での交通事故発生件数のデータである．

Input

```
>   dat <- read.csv("C:/Fukuoka_Seminor/data1.csv",fileEncoding = "cp932")
>   head(dat)
```

- 関数`read.csv()`はcsvファイルを読み込むために用いられる．また，このファイルでは市区町村名が日本語で書かれている．
- 「`<-`」は代入を意味しているが，「`=`」でもかまわない．
- Rにおいて日本語を読み込ませるためには，Shift-JS形式での読み込みを引数「`fileEncoding = "cp932"`」で指定しなければならない．
- `head()`は，`dat`という変数名で入力されたデータの最初の部分を表示させるための関数である．
- なお，データは，データフレーム型という変数形式で代入される(データフレーム型とはExcelのSheetのようなもの)．

Output

```
          City  Accident
1       うきは市         7
2       みやま市         8
3  鞍手郡鞍手町         5
4  鞍手郡小竹町         3
5  遠賀郡芦屋町         3
6  遠賀郡遠賀町         5
```

- ここで，`City`，`Accident`は変数名，左端の`1`から`6`までの数字は個体のID番号(行名)を表す．

データフレームの操作

Input

```
>   dat$City
```

Output

```
 [1] "うきは市"          "みやま市"          "鞍手郡鞍手町"      "鞍手郡小竹町"      "遠賀郡芦屋町"      "遠賀郡遠賀町"
 [7] "遠賀郡岡垣町"      "遠賀郡水巻町"      "嘉穂郡桂川町"      "嘉麻市"            "久留米市"          "宮若市"
[13] "京都郡みやこ町"    "京都郡苅田町"      "古賀市"            "行橋市"            "三井郡大刀洗町"    "三潴郡大木町"
（以下省略）
```

Inputで指定されているのは，データフレームdatのなかの変数Cityの中身を表示させることを表している．
なお，dat[,1]と入力しても同じ意味である(データフレームdatの1列目を指定している)．

Input

```
>   dat[12,]
```

Output

```
    City Acceident
12 宮若市         17
```

Inputで指定されているのは，データフレームdatのなかの12番目の個体を指定している．

少し高度な利用方法：交通事故件数が50件以上の市区町村を表示する．

Input

```
>   dat[dat$Accident >= 50, ]
```

Output

```
             City  Accident
11         久留米市       115
53           飯塚市        60
55       福岡市西区        93
56      福岡市早良区        54
57      福岡市中央区        98
（以下省略）
```

データフレームの操作

Input
```
> colnames(dat)
```

Output
```
[1] "City"     "Accident"
```

Input
```
> rownames(dat)
```

Output
```
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14"
[15] "15" "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28"
[29] "29" "30" "31" "32" "33" "34" "35" "36" "37" "38" "39" "40" "41" "42"
[43] "43" "44" "45" "46" "47" "48" "49" "50" "51" "52" "53" "54" "55" "56"
[57] "57" "58" "59" "60" "61" "62" "63" "64" "65" "66" "67" "68" "69" "70"
```

関数colnames()は変数名を抽出するのに用いられ，関数rownames()は個体名を抽出するのに用いられる．

Input
```
> ncol(dat)
```

Output
```
[1] 2
```

Input
```
> nrow(dat)
```

Output
```
[1] 70
```

関数ncol()は変数の数(列数)を抽出するのに用いられ，関数nrow()は個体数(行数)を抽出するのに用いられる．

要約統計量の計算

要約統計量を計算する前に，データフレームdataのなかの変数Accidentの内容をAccに代入する．

Input

```
>   Acc <- dat$Accident
>   Acc
```

Output

```
 [1]   7   8   5   3   3   5  12  14   4   9 115  17   7  29  21  26   4   6  41  37  44  21
[23]  16   5  24  10  17  12  44  38  14  45  49   1   3   7  22  49  16  13   2  18  32   6
[45]   6   6   3   7  33  21   8  33  60  32  93  54  98 112  89 149  28  10  27  27 107 117
[67] 127  26  35  30
```

このように，1個の変数のみで構成されているデータ形式をベクトル型という．また，数値のベクトル型をnumeric(数値)型という．

なお，ここでは，ベクトル型で要約統計量を計算するが，データフレーム型のままでも計算できる．

先ほどのデータフレームとの違いを以下に示す．

データフレームdat

	City	Accident
1	うきは市	7
2	みやま市	8
3	鞍手郡鞍手町	5
4	鞍手郡小竹町	3
⋮	⋮	⋮

ベクトルAcc

1	2	3	4	...
7	8	5	3	...

データフレームは，複数の変数で構成されるため，変数名が存在する．一方で，ベクトルは1変数しか扱えないため，変数名は存在しない．

```
> dat[3,2]
> Acc[3]
```

➡ それぞれ，5が表示される．

平均値に基づく要約の方法

平均値の計算

Input
```
> mean(Acc)
```

Output
```
[1] 31.7
```

不偏分散の計算

Input
```
> var(Acc)
```

Output
```
[1] 1204.445
```

不偏標準偏差の計算

Input
```
> sd(Acc)
```

Output
```
[1] 34.70511
```

少し高度な利用方法：小数点以下3位を四捨五入して不偏標準偏差を求める

Input
```
> round(sd(Acc), 2)
```

Output
```
[1] 34.71
```

中央値の計算

Input
```
>   median(Acc)
```

Output
```
[1] 21
```

四分位範囲の計算

Input
```
>   Q1 <- quantile(Acc, 0.25)
>   Q3 <- quantile(Acc, 0.75)
>   Q3-Q1
```

Output
```
  75%
30.75
```

関数quantile()の引数0.25は第1四分位点（昇順に並べ替えたときの25%に位置する値），0.75は第3四分位点（昇順に並べ替えたときの75%に位置する値）を表す．ちなみに，0.50にすると中央値を表す．

少し高度な利用方法：中央値，第1四分位点，第3四分位点を一度に表示させる

Input
```
>   quantile(Acc,c(0.5,0.25,0.75))
```

Output
```
 50%   25%   75%
21.00  7.00 37.75
```

余談：欠測がある場合の注意点

要約統計量の関数は，いずれも欠測がある場合には，欠測値になってしまう．そのため，オプションとしてna.rm=TRUEを指定して，欠測は除外する必要がある．

ここでは，変数x（ベクトル型）を作成してその内容を理解する （ちなみに，NAは欠測の意味）．

Input

```
>   x <- c(10, 32, NA, 43, 23, 12, 53, 34)
>   mean(x)
```

Output

```
[1] NA
```

Input

```
>   mean(x, na.rm=TRUE)
```

Output

```
[1] 29.57143
```

Excelでデータを創る場合のヒント

エクセルでデータを作ったうえで，CSVファイルで保存したものを R で読み込ませる場合，欠測は空白にするのではなく，その部分にNAと入力して作る必要がある．空白は，データが記述されていないものと判断され，エラーメッセージが返されるので注意が必要．

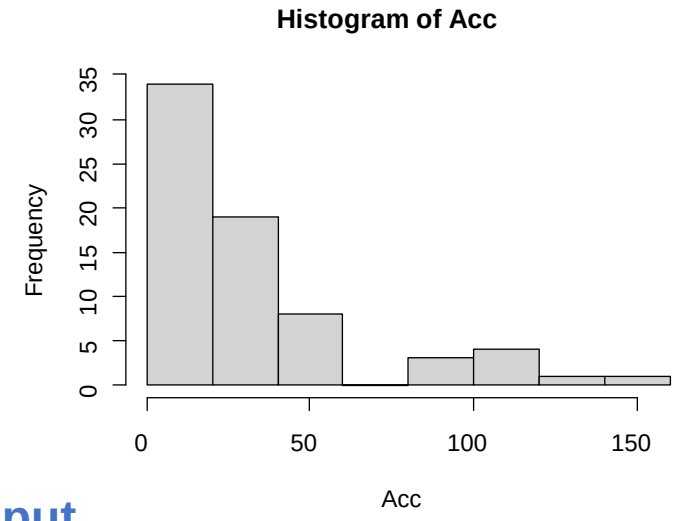
ヒストグラム

データの分布をグラフで表す方法がヒストグラムである．Rでは，hist()という関数を用いる．

hist()

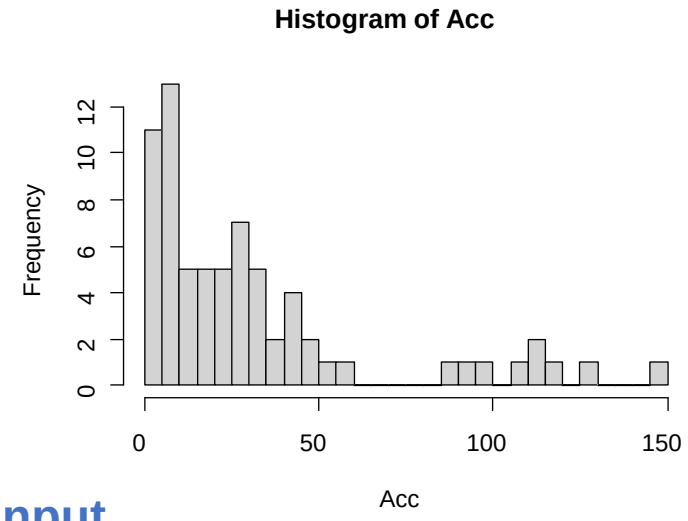
hist(x,breaks,freq,col,border,main,xlab,ylab)　その他にもオプションはあるが割愛

x	:データ（ベクトル形式）	breaks	ヒストグラムの棒の数の指定
freq	:TRUE（密度での記載），FALSE（度数での記載）	col	ヒストグラムを塗りつぶす色の指定
border	:ヒストグラムの線の色の指定	main	グラフのタイトル
xlab	:X軸のラベル	ylab	Y軸のラベル



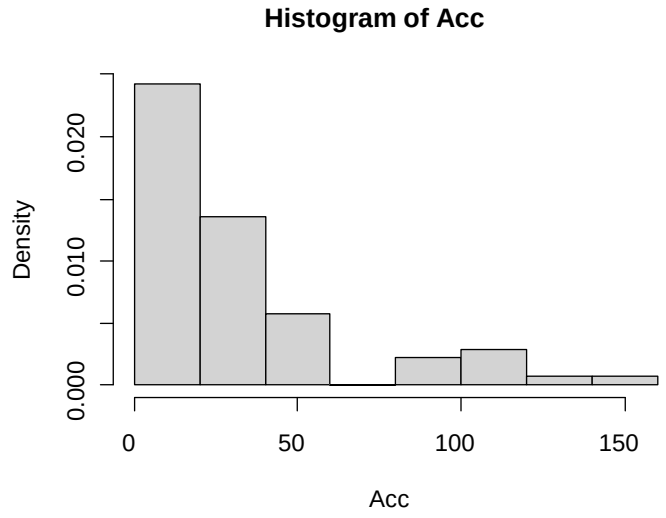
Input

```
> hist(Acc)
```



Input

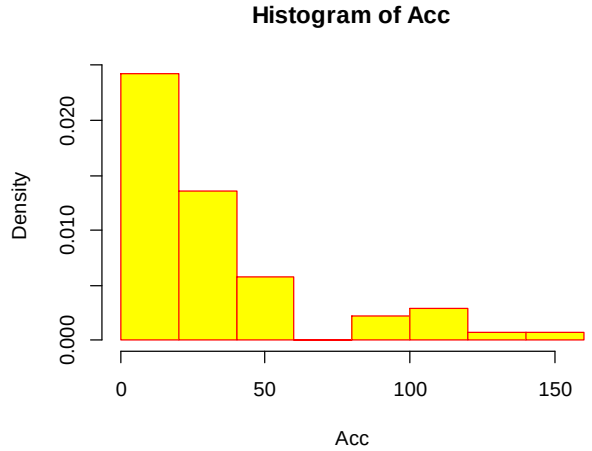
```
> hist(Acc,breaks=30)
```




```
> hist(Acc,breaks=30, freq=FALSE)
```

Rでは色の指定方法には4種類ある

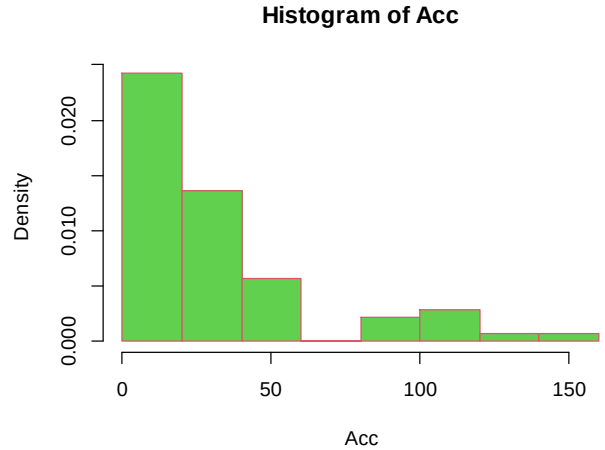
(1) 色の名前で指定する場合（色の名前は付録参照）



Input

```
> hist(Acc,freq=FALSE,col="yellow",border="red")
```

(2) 番号で指定する場合（色の名前は付録参照）

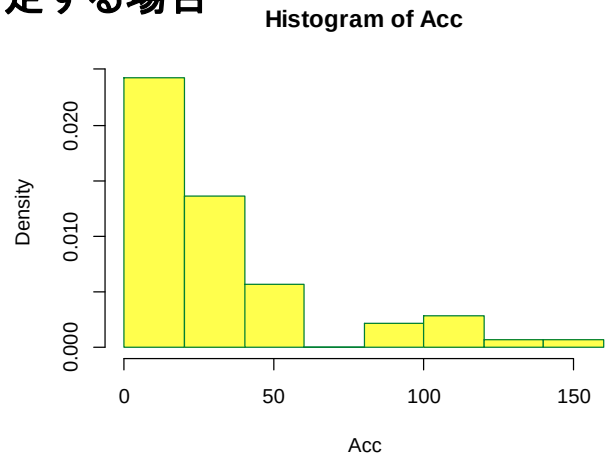


Input

```
> hist(Acc,freq=FALSE,col=3,border=2)
```

1:黒，2:赤，3:緑，4:青，5:シアン，6:マゼンダ，7:黄色(オレンジ)，8:グレイ

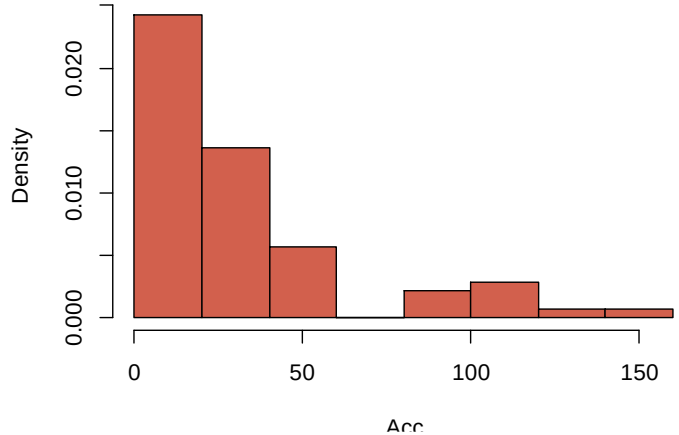
(3) RGBで指定する場合



Input

```
> hist(Acc,freq=FALSE,col=rgb(1,1,0.3),
+ border=rgb(0,0.5,0.2))
```

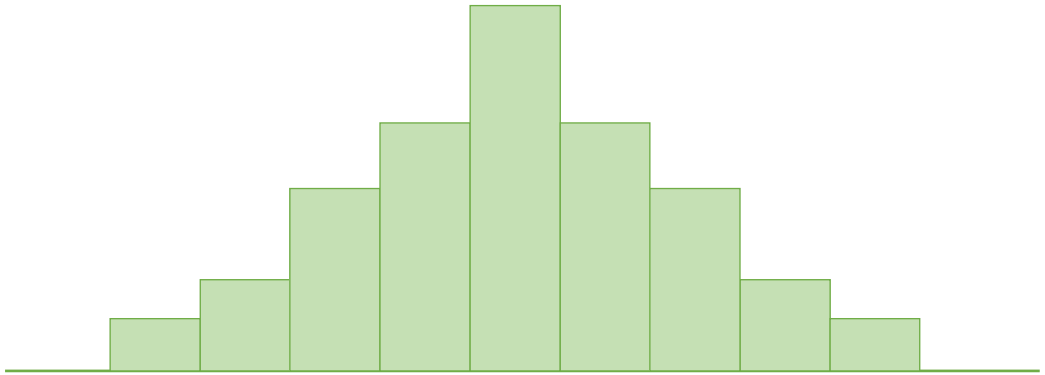
(4) 16進数で指定する場合（色の名前は付録参照）



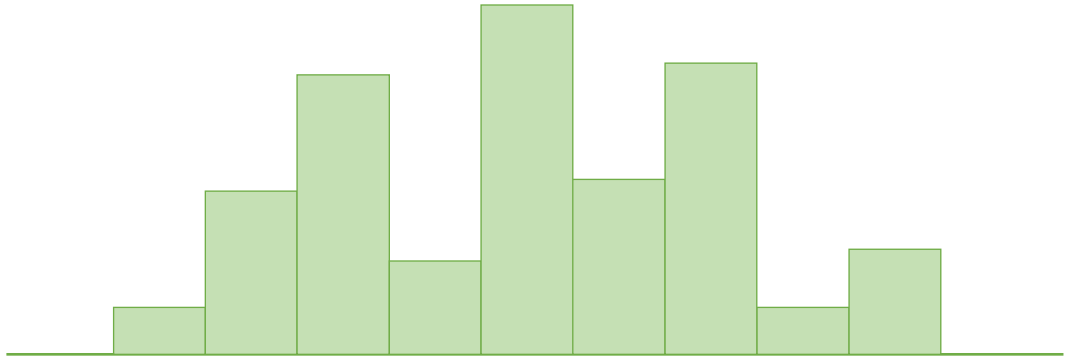
Input

```
> hist(Acc,freq=FALSE,col=3,border=2)
```

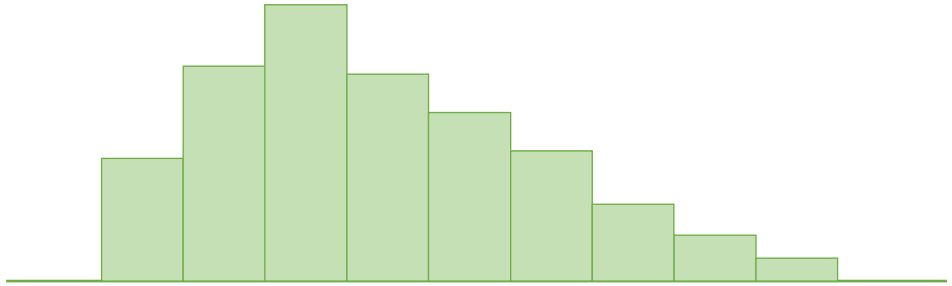
参考:ヒストグラムのパターン



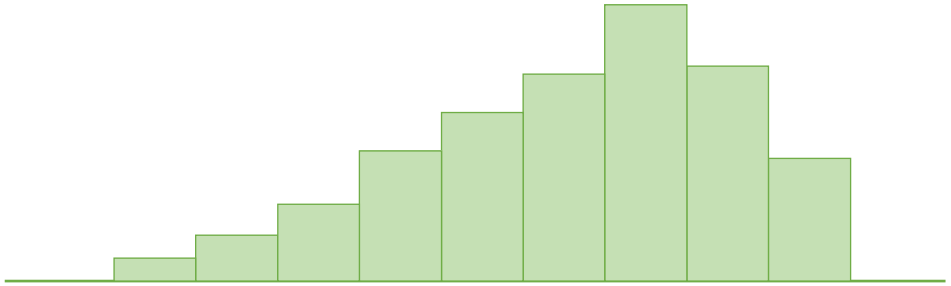
名前：対称分布.
形状：平均値，最頻値および中央値が重なりデータの存在範囲の中心である．平均値を中心に左右対称である.
説明：一般に現れる形.



名前：歯抜け分布.
形状：級の一つおきに度数が少なくなっている.
説明：標本サイズの割に級の数が多い，境界値のとり方が適当でないときに現れる.

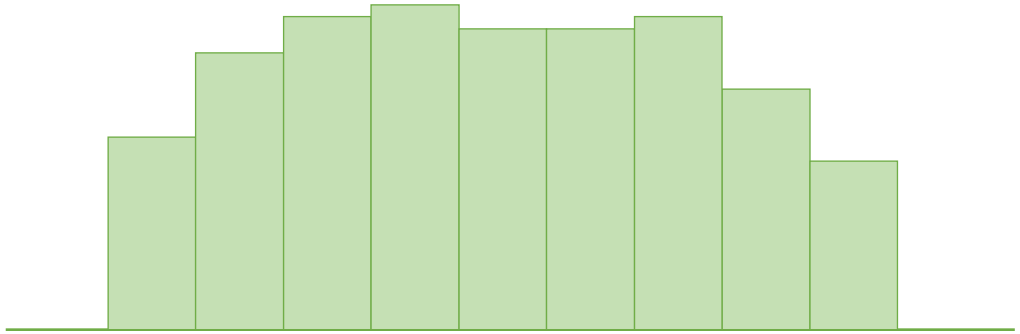


名前：左にひずんでいる形，右に裾が長い(L形状)．
形状：平均値．最頻値および中央値が一致しない場合である．すなわち平均値がデータの存在範囲より左にあり，左右対称ではない.
説明：理論的に下限がおさえられており，下限値以下のデータが存在しない場合に現れる.

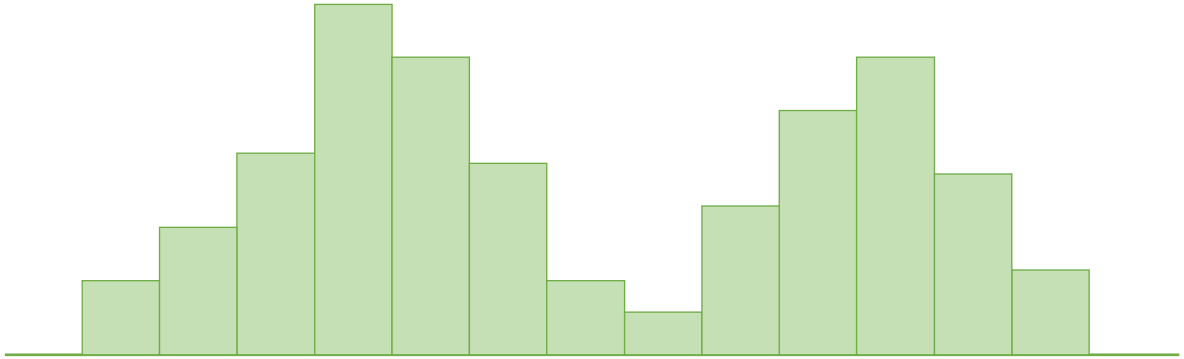


名前：右にひずんでいる形，左に裾が長い(J形状)．
形状：平均値．最頻値および中央値が一致しない場合である．すなわち平均値がデータの存在範囲より右にあり，左右対称ではない.
説明：理論値または規格値などによって，上限値以上のデータが存在しない場合に現れる.

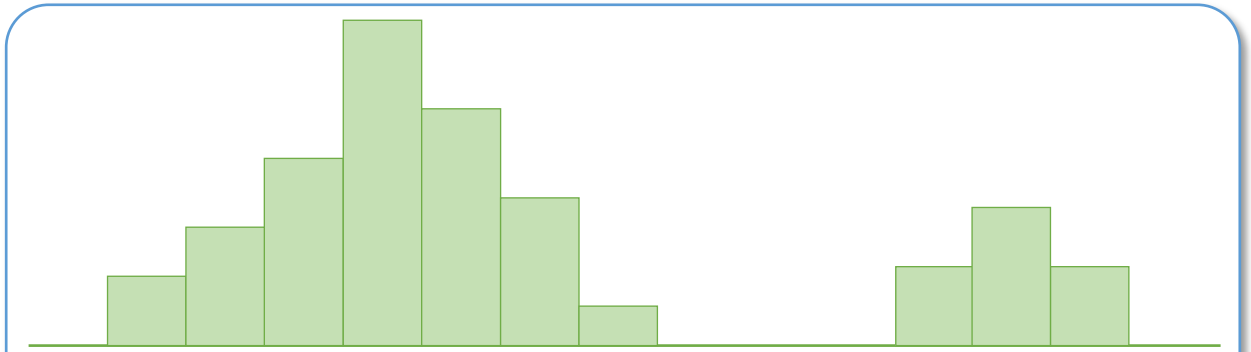
参考:ヒストグラムのパターン



名前：台形形(高原形)．
形状：各級に含まれる度数が両端を除いてほぼ等しい．
説明：平均値が異なるデータが混在するときに現れる（統計的評価は困難)．



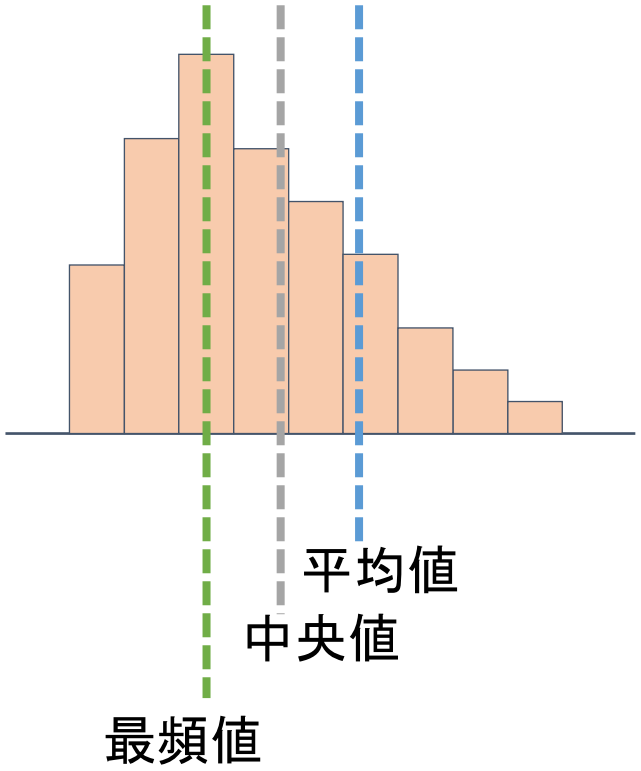
名前：二峰性形．
形状：データの存在する範囲の中心に度数が少なく，両端に山が２つある．
説明：平均値が極端に異なるデータが混在しているときに現れる．



名前：離れ島形．
形状：対称形のヒストグラムのほかに，それと離れたところに度数がある．
説明：データに外れ値が混入しているときに現れる．

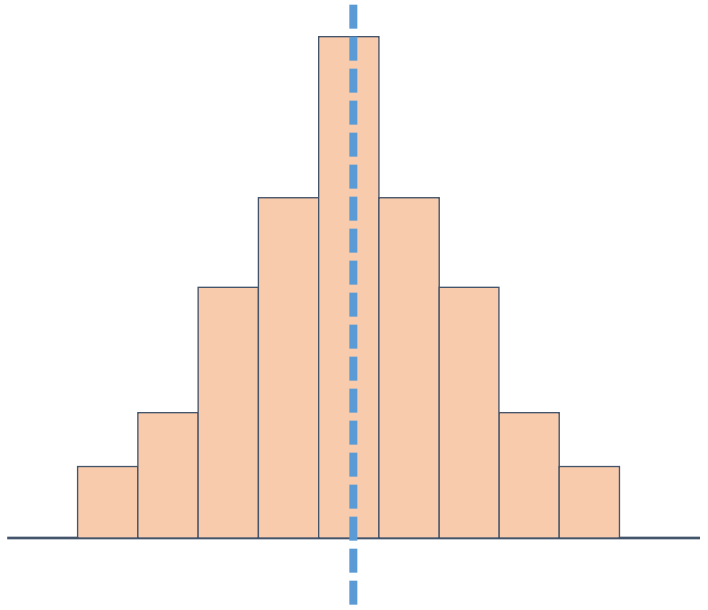
ヒストグラムと位置を表す測度の関係

左に歪んでいる分布
(非対称分布)



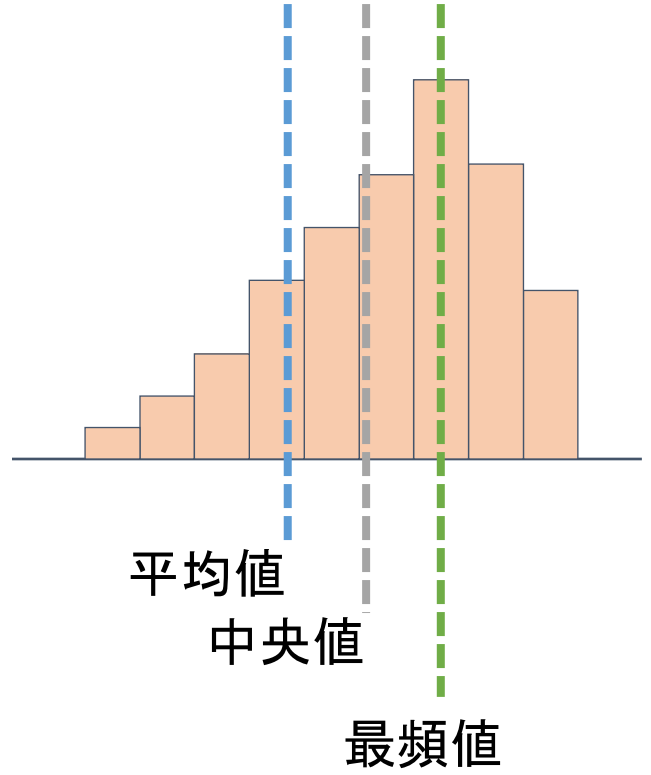
代表値：中央値
(最頻値)

対称分布



最も一般的な状況
代表値：平均値

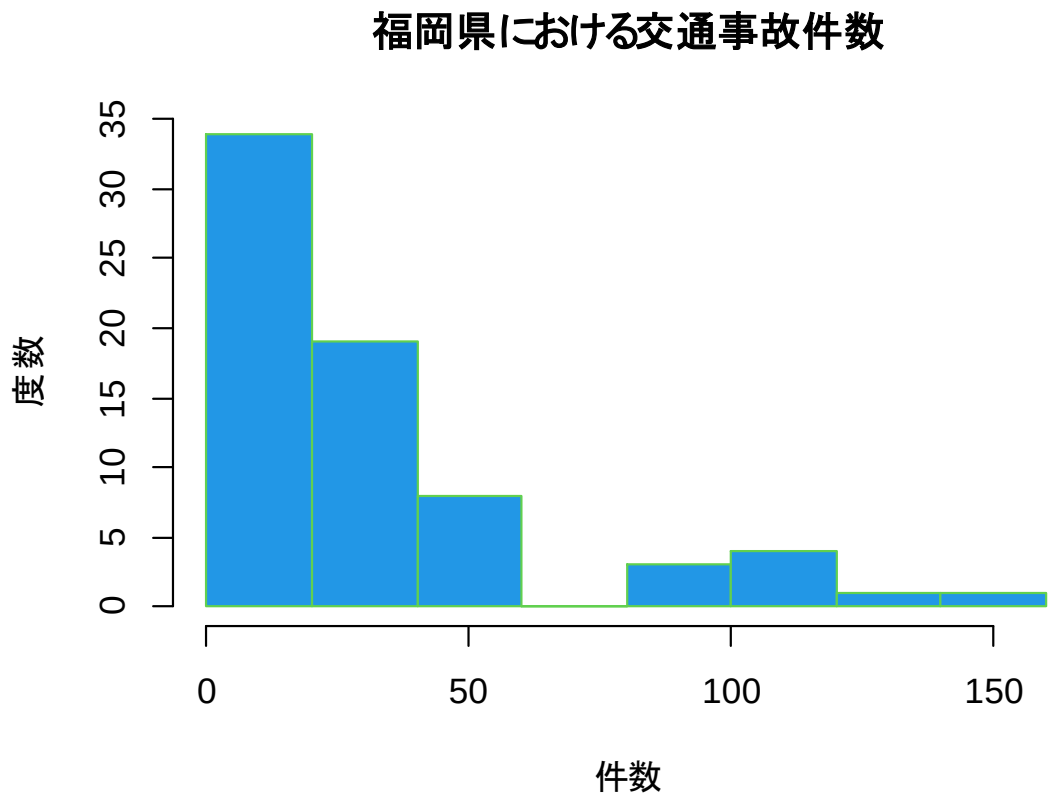
右に歪んでいる分布
(非対称分布)



代表値：中央値
(最頻値)

ヒストグラムを完成させる

```
> hist(Acc,col=4,border=3,main="福岡県における交通事故件数", xlab="件数", ylab="度数")
```



左に歪んだ分布形状を示している．すなわち，中央値で要約するほうが良い．

したがって，適切な要約の方法は，中央値 [第1四分位点，第3四分位点]で要約して

21.0 [7.0, 37.37]

と表すほうことが推奨される．

複数のデータでの要約

ここでは，Cドライブの`Fukuoka_Seminor`というフォルダにある`data2.csv`というCSVファイルを読み込む．これは，2017～2020年度の福岡県の市区町村別での交通事故発生件数のデータである．ここでは，Cityという変数に市区町村名が入っているので，それを列名にしたうえで計算する．

Input

```
>   dat <- read.csv("C:/Fukuoka_Seminor/data2.csv",fileEncoding = "cp932")
>   rownames(dat) <- dat$City
>   dat <- dat[,-1]
>   head(dat)
```

Output

```
          Y2017 Y2018 Y2019 Y2020
うきは市          9     9     7     3
みやま市         19    18     8    13
鞍手郡鞍手町      4    10     5     4
鞍手郡小竹町      1     6     3     0
遠賀郡芦屋町      5     6     3     3
遠賀郡遠賀町     18    12     5     6
```

2017年度～2020年度における年度別での平均値の計算

Input

```
>   Ms <- colMeans(dat)
>   Ms
```

Output

```
   Y2017    Y2018    Y2019    Y2020
41.78873 37.78873 31.61972 27.95775
```

Inputの1行目は，Msという変数(ベクトル型)に2017～2020年度の交通事故件数の平均値を代入していることを意味する．

2017年度～2020年度における年度別での中央値の計算

ここでは，2017～2020年度における年度別・市区町村別の中央値を計算するとともに，年度別および市区町村別で棒グラフを描く

Input

```
>   Meds <- apply(dat,2,"median")
>   Meds
```

Output

```
Y2017 Y2018 Y2019 Y2020
   29    25    21    17
```

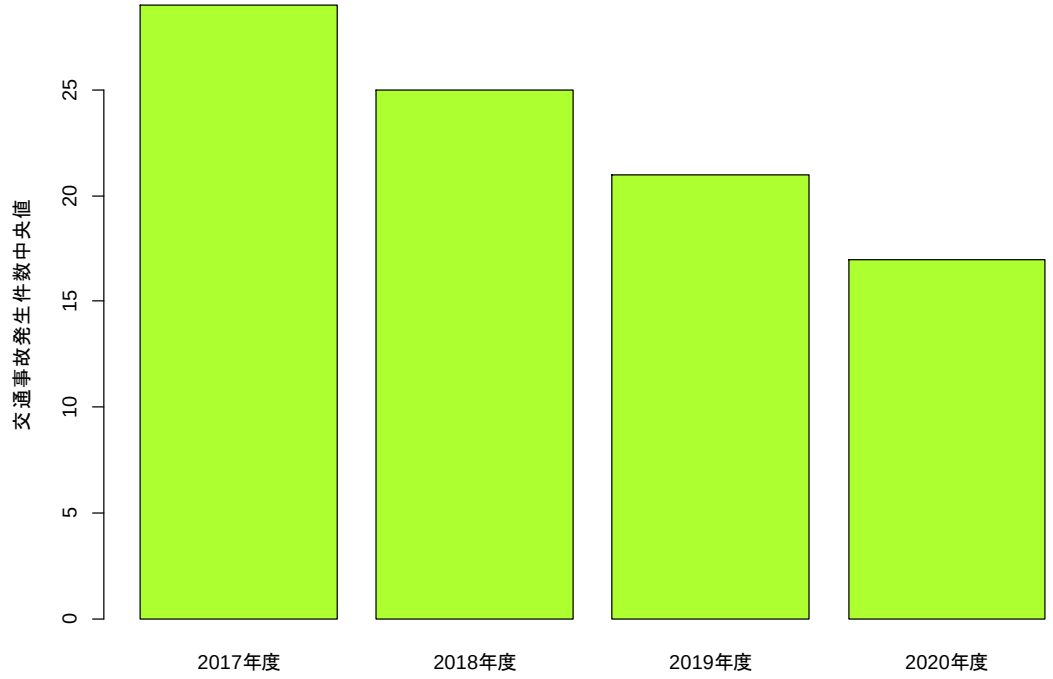
関数apply()とは，同じ処理(ここではmedian)を指定するデータフレームに一括して実施するための関数である．ここで，二つ目の引数の「2」は列毎に中央値を求めることを意味する．

結果を棒グラフで表す．

```
>   barplot(Meds, names=c("2017年度","2018年度","2019年度","2020年度")
+   , col="greenyellow", xlab="年度", ylab="交通事故発生件数中央値")
```

ここで引数namesは棒グラフのそれぞれの名前をベクトルで指定している．

交通事故発生件数が減少傾向を示していることがわかる．



Input

```
>   Mean.City <- apply(dat,1,"mean")
>   Mean.City
```

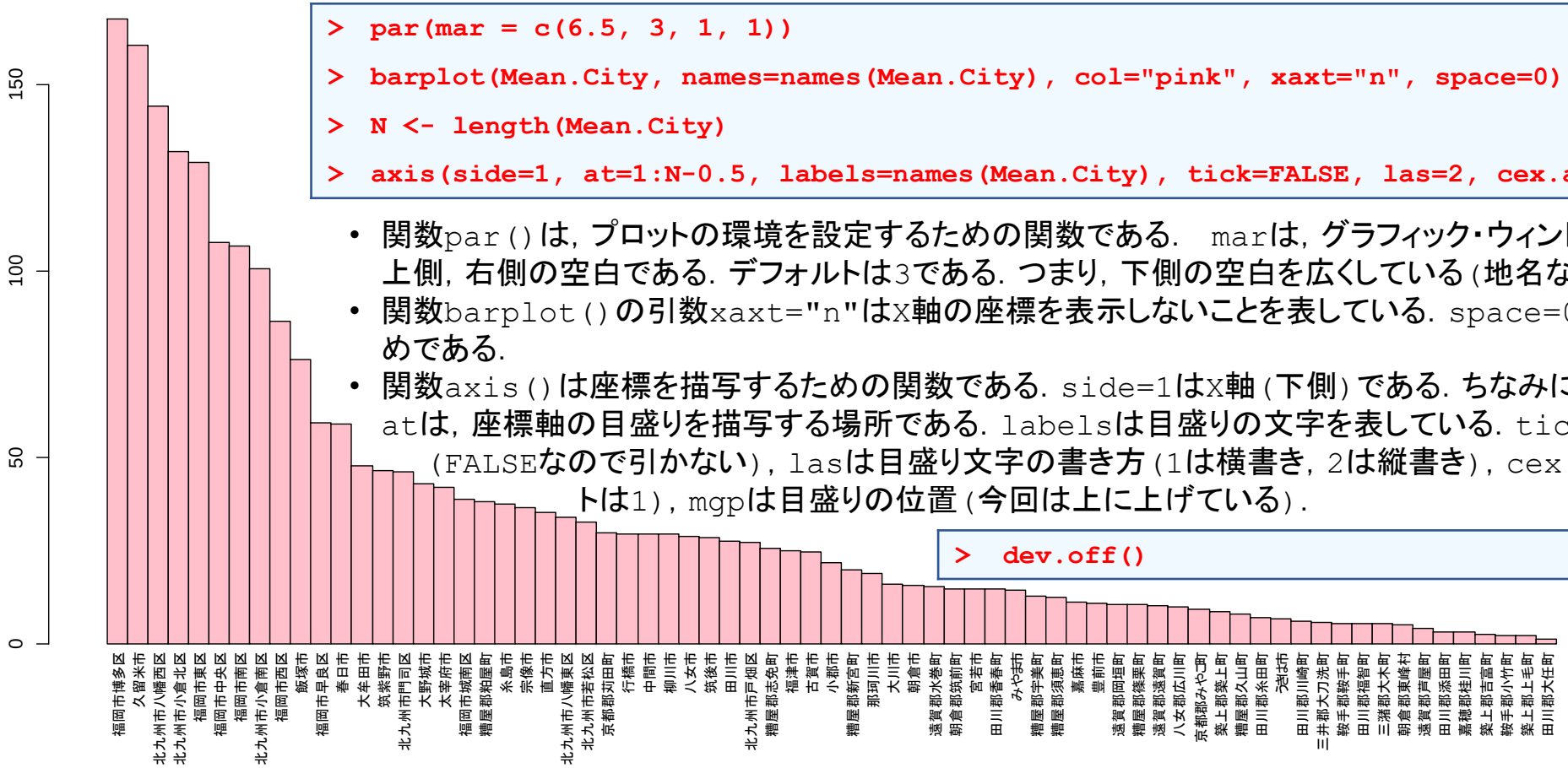
Output

```
        うきは市         みやま市      鞍手郡鞍手町      鞍手郡小竹町
            7.00           14.50            5.75            2.50
   遠賀郡芦屋町   遠賀郡遠賀町   遠賀郡岡垣町   遠賀郡水巻町
            4.25           10.25           10.75           15.50
（以下省略）
```

棒グラフで表す．

Input

```
>  par(mar = c(6.5, 3, 1, 1))
>  barplot(Mean.City, names=names(Mean.City), col="pink", xaxt="n", space=0)
>  N <- length(Mean.City)
>  axis(side=1, at=1:N-0.5, labels=names(Mean.City), tick=FALSE, las=2, cex.axis=0.7, mgp=c(1,0,1))
```

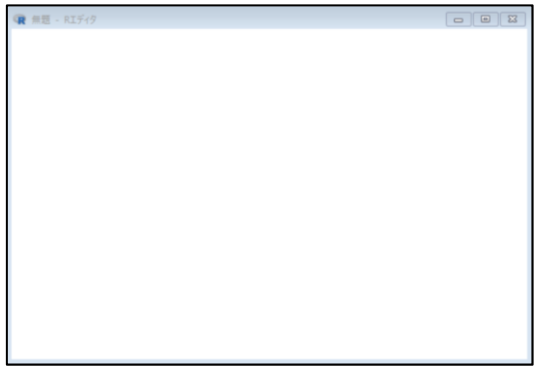


- 関数`par()`は，プロットの環境を設定するための関数である． `mar`は，グラフィック・ウィンドウの空白の表しており下側，左側，上側，右側の空白である．デフォルトは`3`である．つまり，下側の空白を広くしている（地名なので空白が広く必要なため）．
- 関数`barplot()`の引数`xaxt="n"`はX軸の座標を表示しないことを表している．`space=0`は棒グラフの間の空白を`0`にするためである．
- 関数`axis()`は座標を描写するための関数である．`side=1`はX軸（下側）である．ちなみに，`2`は左側，`3`は上側，`4`は右側である．`at`は，座標軸の目盛りを描写する場所である．`labels`は目盛りの文字を表している．`tick`は目盛りに線を引くかどうか（`FALSE`なので引かない），`las`は目盛り文字の書き方（`1`は横書き，`2`は縦書き），`cex.axis`は座標文字の大きさ（デフォルトは`1`），`mgp`は目盛りの位置（今回は上に上げている）．

```
>  dev.off()
```

関数を作る

先ほどの棒グラフは設定がたくさんあるため，複数のグラフを作る場合には，関数にしておくと便利.



「ファイル」→「新しいスクリプト」を選択 (左側のようなウィンドウ「Rエディッタ」が開く)

「Rエディッタ」にプログラムを記述する.

Rエディッターの内容

```
city.barplot <- function(dat){
        par(mar = c(6.5, 3, 1, 1))
        barplot(dat, col="pink", xaxt="n", space=0)
        N <- length(dat)
        axis(side=1, at=1:N-0.5, labels=names(dat), tick=FALSE, las=2, cex.axis=0.7, mgp=c(1,0,1))
}
```

入力したプログラム全体をドラックして，Ctrl+Rキーを押す(関数が R に読み込まれる).

Input

```
>   city.barplot(Mean.City)
```

先ほどと同じ棒グラフが表示される.

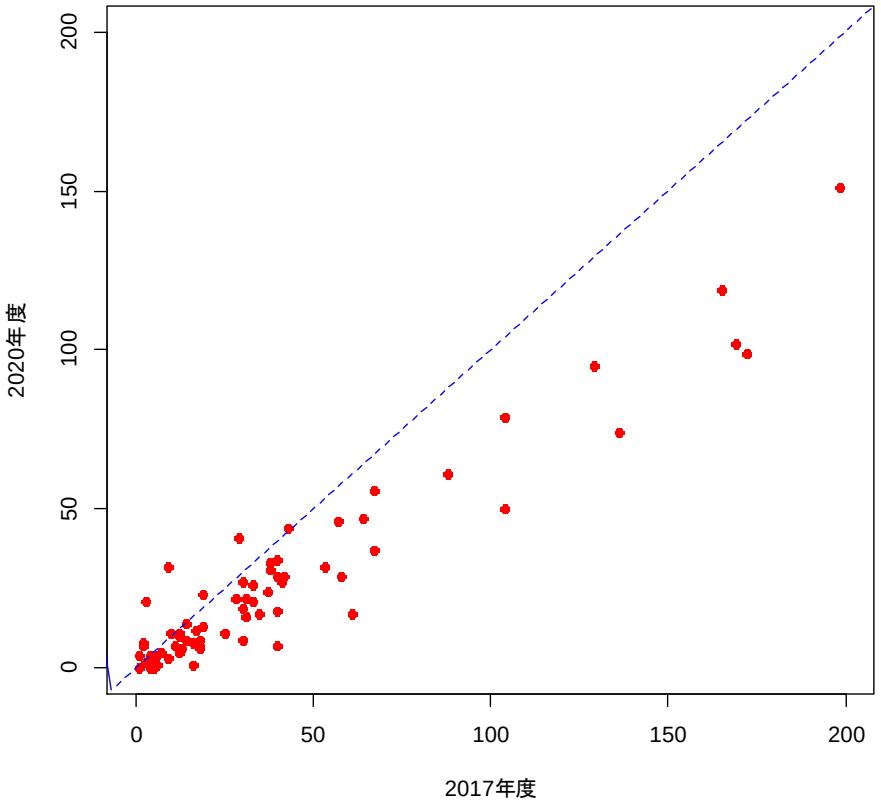
「ファイル」→「別名で保存」でRエディッタの内容が保存できる (このように関数を作ることで，業務が次第に効率化される).

散布図を作る

ここでは，データフレームdatに含まれている2017年度(Y2017)と2020年度(Y2020)のあいだで散布図を作る

Input

```
>  plot(dat$Y2017, dat$Y2020, xlab="2017年度", ylab="2020年度", pch=16, col="red", xlim=c(0,200), ylim=c(0,200))
>  abline(a=0, b=1, col="blue", lty=2)
```



関数plotの書式 `plot(x,y,options)`

関数plotのオプション

xlab:X軸の名前　　ylab:Y軸の名前

xlim:X軸の最小，最大(xlim=c(0,200)は0～200まで)

ylim:Y軸の最小，最大(ylim=c(0,200)は0～200まで)

pch:データ点の形状 (付録2を参照)

col:データ点の色 (付録1を参照)

- 関数ablineは，低水準グラフィクスと呼ばれ，プロット(高水準グラフィクス)に切片a，傾きbで直線を引くために用いられる．
- 関数ablineの引数ltyは直線の形状

6.'twodash'
5.'longdash'
4.'dotdash'
3.'dotted'
2.'dashed'
1.'solid'
0.'blank'

相関係数の計算

Rにおいて，相関係数を計算する関数はcor()である．

cor()

使い方1: `cor(x, y, method)` 変数xと変数yの相関係数を求める．

使い方2: `cor(M, method)` 行列のすべての組み合わせの相関係数を求める．

methodには，"pearson"（デフォルト），"spearman"， "kendall"の3種類がある．

- pearsonはいわゆる相関係数（Pearsonの積率相関係数）である．正規分布に従うことが想定される（分布が歪んでいない）．
- spearmanはSpearmanの順位相関係数，kendallはKendallの順位相関係数であり，ノンパラメトリック相関と呼ばれる．データが正規分布に従っていない（歪んでいる）場合に用いる．

いいかえれば，平均値で要約する場合はpearson，中央値で要約する場合にはspearmanかkendallを用いる．

2017年度と2020年度のPearsonの積率相関係数

Input
```
> cor(dat$Y2017,dat$Y2020)
```
Output
```
[1] 0.9594974
```

2017年度と2020年度のSpearmanの順位相関係数

Input
```
> cor(dat$Y2017,dat$Y2020,method="spearman")
```
Output
```
[1] 0.8699311
```

2017年度と2020年度のKendallの順位相関係数

Input
```
> cor(dat$Y2017,dat$Y2020,method="kendall")
```
Output
```
[1] 0.7262708
```

Note：欠測がある場合

欠測がある場合（NAがある場合）には，オプションとして，

```
na.rm=TRUE
```

を入れておかないと，NAで返されるので注意．

2017年度から2020年度までのPearsonの積率相関係数

Input

```
> cor(dat[,-1])
```

Output

```
          Y2017     Y2018     Y2019     Y2020
Y2017 1.0000000 0.9641707 0.9505330 0.9594974
Y2018 0.9641707 1.0000000 0.9756841 0.9764935
Y2019 0.9505330 0.9756841 1.0000000 0.9812387
Y2020 0.9594974 0.9764935 0.9812387 1.0000000
```

2017年度から2020年度までのSpearmanの順位相関係数

Input

```
> cor(dat[,-1], method="spearman")
```

Output

```
          Y2017     Y2018     Y2019     Y2020
Y2017 1.0000000 0.8458279 0.8730797 0.8699311
Y2018 0.8458279 1.0000000 0.9495602 0.9529501
Y2019 0.8730797 0.9495602 1.0000000 0.9766983
Y2020 0.8699311 0.9529501 0.9766983 1.0000000
```

2017年度から2020年度までのKendallの順位相関係数

Input

```
> cor(dat[,-1], method="kendall")
```

Output

```
          Y2017     Y2018     Y2019     Y2020
Y2017 1.0000000 0.6977526 0.7295368 0.7262708
Y2018 0.6977526 1.0000000 0.8340509 0.8389619
Y2019 0.7295368 0.8340509 1.0000000 0.8884804
Y2020 0.7262708 0.8389619 0.8884804 1.0000000
```

少し脱線：for文の利用方法

Rエディッタ

```
1 for (i in 1:5){
2           print(i)
3 }
```

Output

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

これは，1から5まで1個づつ変数iの値を増やしながら繰り返す繰り返し文といわれるものである．また，print(i)は変数iの内容を表示している．

Rエディッタ

```
1 for (i in 1:5){
2           print(Fukuoka[i,])
3 }
```

Output

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

これは，1から5まで1個づつ変数iの値を増やしながら，データフレームFukuokaのi行目を表示させている．

折れ線グラフを作る

ここでは，データフレームdatに含まれている福岡市の7区を抽出したうえで，折れ線グラフを用いて2017年から2020年案での交通事故発生件数の変化を描写する.

Rエディッターの内容 (CTRL+Rで実行できる)

```
1  idx <- grep("福岡市", rownames(dat))
2  Fukuoka <- dat[idx,]
3  N <- nrow(Fukuoka)
4  plot(1:4, Fukuoka[1,], type="n", ylim=c(1,200), xaxt="n",xlab="年度", ylab="交通事故発生件数")
5  axis(side=1, at=1:4, labels=c("2017年度","2018年度","2019年度","2020年度"))
6  for (i in 1:N){
7            points(1:4,Fukuoka[i,], pch=i+15, col=i+1)
8            lines(1:4,Fukuoka[i,], lty=i, col=i+1)
9  }
10 pchs <- 1:N + 15
11 cols <- 1:N + 1
12 ltys <- 1:N
13 legend("topright", legend = rownames(Fukuoka), col = cols, pch = pchs, lty = ltys, cex=0.7)
```

次ページから，各行の説明を行う

```
1 idx <- grep("福岡市", rownames(dat))
```

関数grep(a,b)のなかのaは，検索する文字列，bは検索されるリスト(ベクトル型)を表している．また，関数rownames()は，データフレームdatの列名を表す．なお，idxには，列名のなかで「福岡市」を含んでいる番号が返される．

```
2 Fukuoka <- dat[idx,]
```

関数grep()で検索した結果，「福岡市」を含むリストの番号idxの列を新しいデータフレームFukuokaとして代入

```
3 N <- nrow(Fukuoka)
```

データフレームFukuokaの列数(福岡市の区の交通事故発生件数)をNに代入．

```
4 plot(1:4, Fukuoka[1,], type="n", ylim=c(1,200), xaxt="n",xlab="年度", ylab="交通事故発生件数")
```

とりあえず，高水準グラフィクスのplotでアウトラインを作成している．ここでtype="n"とは，プロットを描写しないことを表している．

```
5 axis(side=1, at=1:4, labels=c("2017年度","2018年度","2019年度","2020年度"))
```

X軸の描写である．

```
6 for (i in 1:N){
7           points(1:4,Fukuoka[i,], pch=i+15, col=i+1)
8           lines(1:4,Fukuoka[i,], lty=i, col=i+1)
9 }
```

ここでは，for文というものを用いている(次ページで例を挙げて説明)．これは，1からNまで1個づつ値を増やしながらカッコ内を繰り返す繰り返し文といわれるものである．また，関数points，linesは低水準グラフィクスであり，それぞれ，点および線を引くための関数である(pch， lty， colは関数plotと同じ定義である)．

```
10 pchs <- 1:N + 15
11 cols <- 1:N + 1
12 ltys <- 1:N
13 legend("topright", legend = rownames(Fukuoka), col = cols, pch = pchs, lty = ltys, cex=0.7)
```

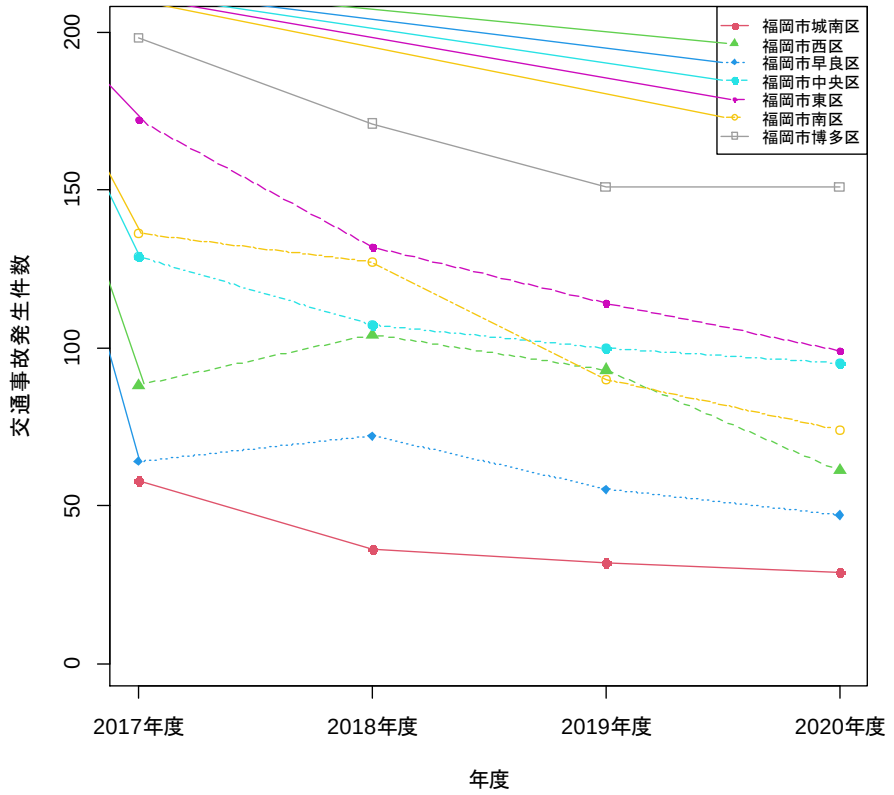
10行目の1:Nは，1からNまで1づつ上昇するベクトルを作成している．それに15を足し合わせている．N=7なので，pchのなかは下記のとおり．

```
[1] 16 17 18 19 20 21 22
```

11行目の1:Nは，1からNまで1づつ上昇するベクトルを作成している．それに1を足し合わせている．N=7なので，colsのなかは下記のとおり．

```
[1] 2 3 4 5 6 7 8
```

関数legend()は，凡例を表示させるための低水準グラフィクスである．最初の"topleft"は右上に表示させる関数．legendは凡例の名称，その他のオプションはplotと同じ．



殆どの区で減少傾向を示している．とくに，南区の減少は著しい．

箱ひげ図を作る

ここでは，Cドライブの`Fukuoka_Seminor`というフォルダにある`data3.csv`というCSVファイルを読み込む．これは，2020年度の福岡県の市区町村別での交通事故発生件数と2015年度の人口のデータである．ここでは，Cityという変数に市区町村名が入っているので，それを列名にしたうえで計算する．

Input

```
>   dat <- read.csv("C:/Fukuoka_Seminor/data3.csv",fileEncoding = "cp932")
>   rownames(dat) <- dat$City
>   dat <- dat[,-1]
>   head(dat)
```

Output

```
           Y2020 Population
うきは市       3      29509
みやま市      13      38139
鞍手郡鞍手町   4      16007
鞍手郡小竹町   0       7810
遠賀郡芦屋町   3      14208
遠賀郡遠賀町   6      18877
```

人口10,000人当たりでの交通事故件数を計算する．

Input

```
>   Ac10000 <- dat$Y2020/dat$Population*10000
>   Ac10000
```

Output

```
 [1] 1.016639 3.408584 2.498907 0.000000 2.111486 3.178471 2.216593 3.793496
 [9] 1.481921 2.323000 4.104389 2.845760 3.457985 6.292366 3.623251 3.825121
[17] 2.642357 3.527088 3.005960 3.419122 5.244428 3.966680 3.691302 7.294833
（以下省略）
```

人口の中央値を求めて，中央値以上を1，中央値未満を0とする新しい変数groupを作る．

Input

```
>   group <- numeric(nrow(dat))
>   group[dat$Population >= median(dat$Population)] <- 1
>   group
```

Output

```
 [1] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 1 1 1 1 0 0 1 0 0 0 1 1 0 1 1 0 0 0
[37] 1 1 0 0 0 1 1 0 0 0 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
```

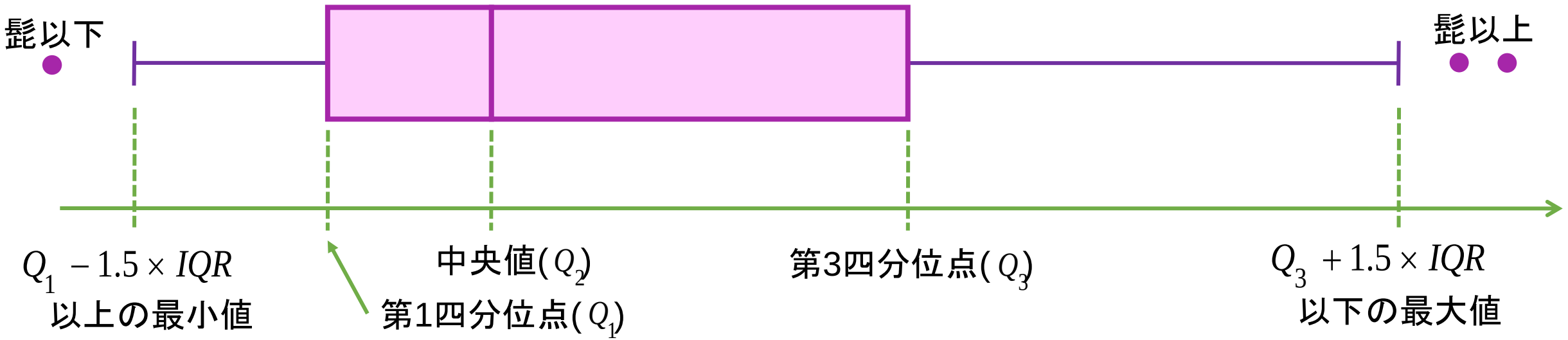
箱ひげ図を描く

boxplot()

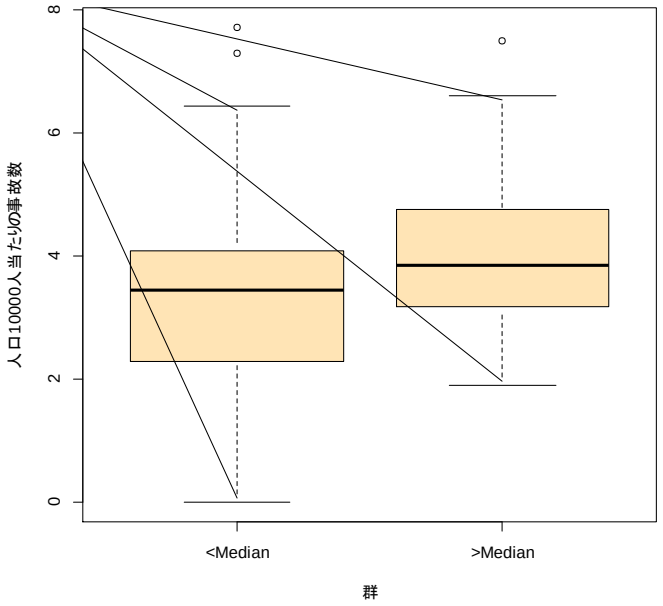
hist(formula,col,border,horizontal,main,xlab,ylab)　その他にもオプションはあるが割愛

formula	:データの形式を記載する．例えば，x~gorupとすれば，group毎にデータxのボックスプロットを描く		
col	:箱ひげ図を塗りつぶす色の指定	border	:箱ひげ図の輪郭を塗りつぶす色の指定
horizontal	:TRUEにすると水平の箱ひげ図が作成される	main	グラフのタイトル
xlab	:X軸のラベル	ylab	Y軸のラベル

箱ひげ図の概要

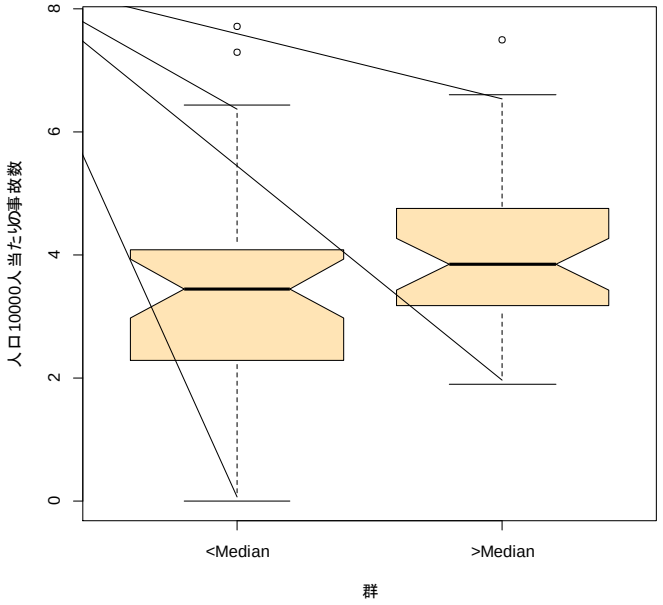


- 中央値，第1四分位点，第3四分位点に基づいて構成されるグラフ.
- 外れ値，分布形状，2標本の比較などに用いることができる.
- 単峰性が仮定されていることに注意



箱ひげ図(ボックスプロット)を描く

```
>   boxplot(Ac10000~group, names=c("<Median", ">Median"),
+   col="moccasin", xlab="群", ylab="人口10000人当たりの事故数")
```



ノッチ付き箱ひげ図(ノッチド・ボックスプロット)を描く

```
>   boxplot(Ac10000~group, names=c("<Median", ">Median"),notch=TRUE,
+   col="moccasin", xlab="群", ylab="人口10000人当たりの事故数")
```

ノッチとは，中央値(横線の部分)の信頼度を表しており，ノッチが重なっていない場合には中央値が異なる可能性が高いと判断される．

人口10,000万にあたりの交通事故件数ならびに，人口の中央値で分けたデータをデータフレームcompdatに保存するとともに，Cドライブの`Fukuoka_Seminor`にファイル名`compdata.csv`で保存する．

Input

```
>   compdat <- data.frame(accident=Ac10000, group=group)
>   write.csv(compdat, "C:/Fukuoka_Seminor/compdata.csv",row.names=FALSE)
```

	A	B
1	accident	group
2	1.016639	0
3	3.408584	0
4	2.498907	0
5	0	0
6	2.111486	0
7	3.178471	0
⋮	⋮	⋮

DAY.1：記述統計学とグラフの書き方

Section.2：離散尺度(カテゴリカルデータ)の要約とグラフ表示

度数の計算

ここでは，Cドライブの`Fukuoka_Seminor`というフォルダにある`data4.csv`というCSVファイルを読み込む．これは，福岡県内で発生した交通事故に関して，年齢層(高齢者/非高齢者)，時間帯(日中／夜)，天候に関するデータである．

Input

```
>   dat <- read.csv("C:/Fukuoka_Seminor/data4.csv",fileEncoding = "cp932")
>   head(dat)
```

Output

```
  Senior Day_Night Weather
1    高齢者         夜       曇
2 非高齢者         夜       晴
3 非高齢者         夜       晴
4 非高齢者         夜       晴
5 非高齢者         昼       晴
6    高齢者         昼       晴
```

天候別での度数を集計する(このとき，度数が高い順に並べ替える)．

Input

```
>   Weather.tbl <- sort(table(dat$Weather),decreasing=TRUE)
>   head(Weather.tbl)
```

Output

```
  晴    曇   雨  小雨
1164  738  179  137
```

円グラフの作成

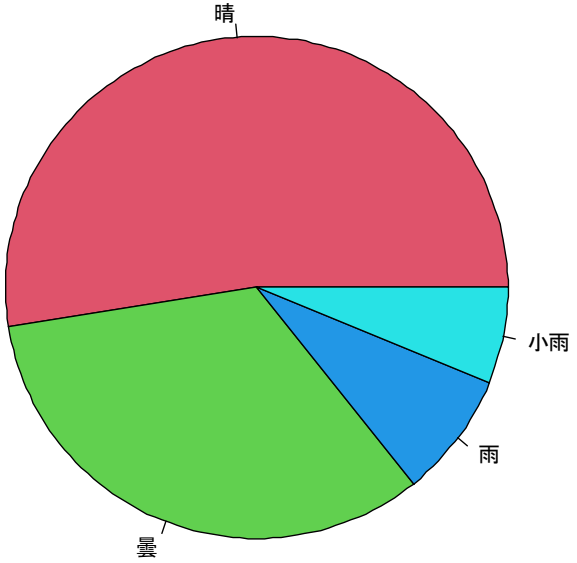
円グラフの描写について，Rではpie()という関数を用いる．

pie()

```
pie(x,labels,radius,clockwise,init.angle,density,col,angle)
```

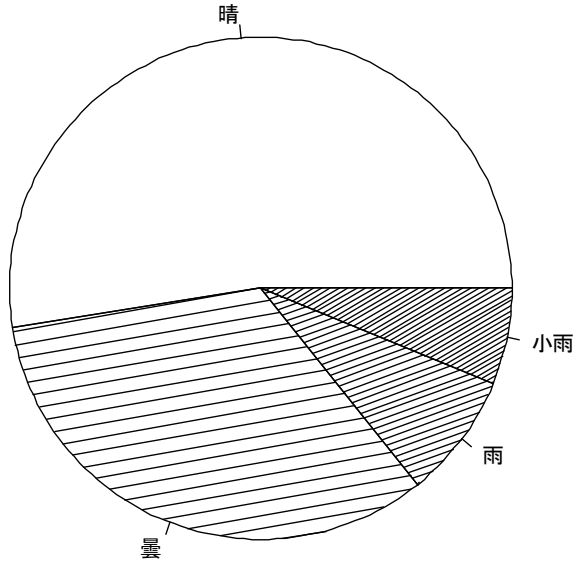
その他にもオプションはあるが割愛

x	:データ(ベクトル形式)	labels	:各扇状のラベル(文字列ベクトル)
radius	:円の大きさを(-1から1，デフォルトは0.8)	clockwise	:時計回り(TRUE)，反時計回り(FALSE)
init.angle	:始点を確度で指定(デフォルトは90)	deisity	:塗分ける際の線の本数
col	:塗りつぶしの色(ベクトル形式)	angle	:塗分ける際の線の確度



Input

```
>    cols <- 1:length(Weather.tbl)+1
>    pie(Weather.tbl, col=cols)
```



Input

```
>    dens <- (1:length(Weather.tbl)-1)*10
>    pie(Weather.tbl, density=dens, angle = dens)
```

関数sprintf()を用いて度数分布表のための関数を作成する.

ここでは，Freq.table()という新しい関数を作成する.

Rエディッターの内容

```
1  Freq.table <- function(dat){
2            tbl <- sort(table(dat),decreasing=TRUE)
3            N <- sum(tbl, na.rm=T)
4            pc <- tbl/N*100
5            res <- sprintf("%d (%3.1f)", tbl, pc)
6            return(res)
7  }
```

各カテゴリの度数を計算

データの個数を計算

各カテゴリの割合を計算

関数sprint()の説明は下記で説明

関数sprintf()は，書式付きの文字列を作る関数である．括弧()内の説明を以下に示す

- ""は文字列に含む書式を表している.
- %dの部分には整数が入ることを意味する.
- %3.1fの部分にはすべての桁数が3桁で，そのうち1桁が小数点以下の桁数であることを意味する(なお，マイナスも1文字になるので注意)
- ""のなかの括弧()は文字として認識される.
- ""の外側の変数においてtblは%dに入る数値，pcは%3.1fに入る数値.

Input

```
>   Freq.table(dat$Weather)
```

Output

```
          晴          曇          雨        小雨
"1164 (52.5)"  "738 (33.3)"  "179 (8.1)"  "137 (6.2)"
```

クロス集計表の作成

年齢層(Senior)と天候(Weather)のクロス集計表を作る．このとき，高齢者は「高齢者，非高齢者」の順番，天気は，「晴，曇，小雨，雨」の順番にする．

Input

```
>   Sn <- factor(dat$Senior)
>   levels(Sn)
```

Output

```
[1] "高齢者"    "非高齢者"
```

Input

```
>   We <- factor(dat$Weather)
>   levels(We)
```

Output

```
[1] "雨"    "小雨" "晴"    "曇"
```

Input

```
>   levels(We) <- c("晴","曇", "小雨", "雨")
>   levels(We)
```

Output

```
[1] "晴"    "曇"    "小雨" "雨"
```

Input

```
>   dt <- data.frame(Sn, We)
>   conj.tbl <- table(dt$Sn, dt$We)
>   conj.tbl
```

Output

```
           晴  曇 小雨  雨
高齢者      46  38  361 275
非高齢者 133  99  803 463
```

関数sprintf()を用いてクロス集計表のための関数を作成する.

ここでは，Conject.table()という新しい関数を作成する.

Rエディッターの内容

```
1  Conject.table <- function(X,Y,idx=NULL){
2            dat <- data.frame(X,Y)
3            tbl <- table(dat$X, dat$Y)
4            pct <- prop.table(tbl, idx)*100
5            N <- nrow(tbl)
6            M <- ncol(tbl)
7            res <- matrix(numeric(N*M), ncol=M)
8            for (i in 1:N){
9                     for (j in 1:M){
10                              res[i,j] <- sprintf("%d (%3.2f)", tbl[i,j], pct[i,j])
11                    }
12           }
13           colnames(res) <- colnames(tbl)
14           rownames(res) <- rownames(tbl)
15           return(res)
16 }
```

関数prop.table(x, margin)において，xはクロス集計表，marginはパーセントを求める方向を表しており，NULL(デフォルト)は全体パーセント，1は行パーセント(各行で合計すると100%になる)，2は列パーセント(各列で合計すると100%になる)を表している.

全体パーセントを加えたクロス集計表を作る

Input

```
>   Conject.table(Sn, We)
```

Output

```
          晴             曇             小雨            雨
高齢者    "46 (2.07)"   "38 (1.71)" "361 (16.28)" "275 (12.40)"
非高齢者 "133 (6.00)" "99 (4.46)" "803 (36.20)" "463 (20.87)"
```

行パーセントを加えたクロス集計表を作る

Input

```
>   Conject.table(Sn, We, 1)
```

Output

```
          晴             曇             小雨            雨
高齢者    "46 (6.39)"   "38 (5.28)" "361 (50.14)" "275 (38.19)"
非高齢者 "133 (8.88)" "99 (6.61)" "803 (53.60)" "463 (30.91)"
```

列パーセントを加えたクロス集計表を作る

Input

```
>   Conject.table(Sn, We, 2)
```

Output

```
          晴             曇             小雨            雨
高齢者    "46 (25.70)"  "38 (27.74)" "361 (31.01)" "275 (37.26)"
非高齢者 "133 (74.30)" "99 (72.26)" "803 (68.99)" "463 (62.74)"
```

帯グラフの作成

帯グラフは，クロス集計表におけるパーセントと関数`barplot()`を組み合わせることで描写できる．

conj.tbl

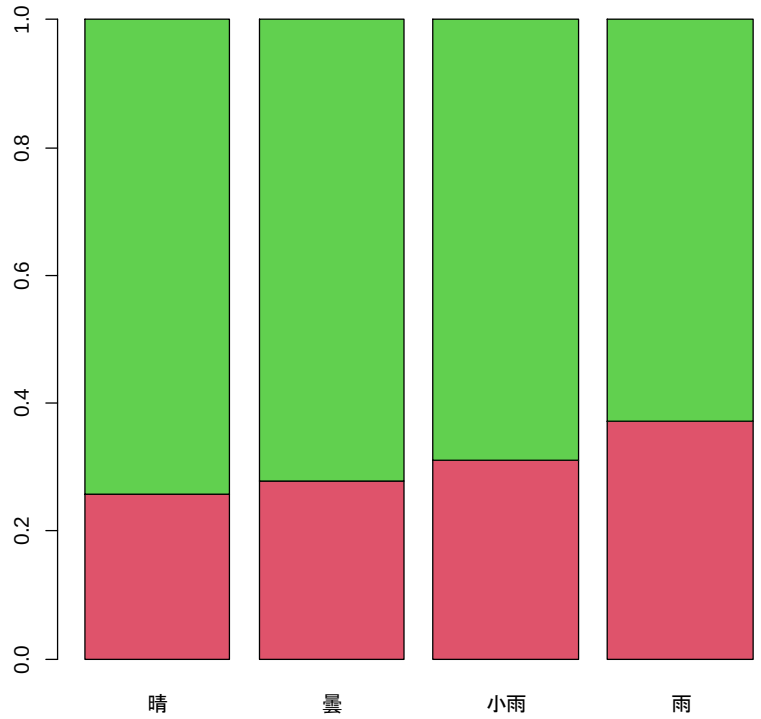
	晴	曇	小雨	雨
高齢者	46	38	361	275
非高齢者	133	99	803	463

t(conj.tbl)

	高齢者	非高齢者
晴	46	133
曇	38	99
小雨	361	803
雨	275	463

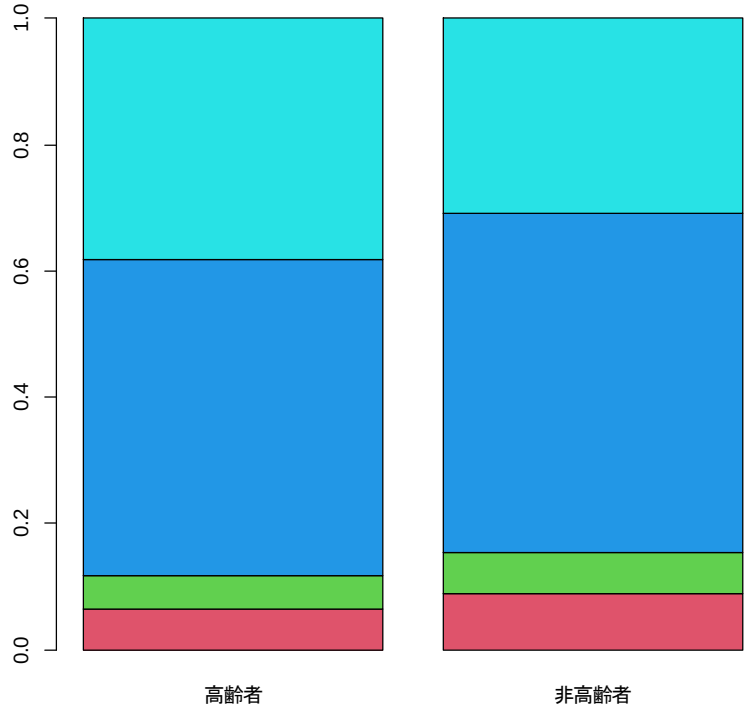
Input

```
> barplot(prop.table(conj.tbl,2), beside=F, col=c(2,3))
```



Input

```
> barplot(prop.table(t(conj.tbl),2), beside=F, col=c(2,3,4,5))
```



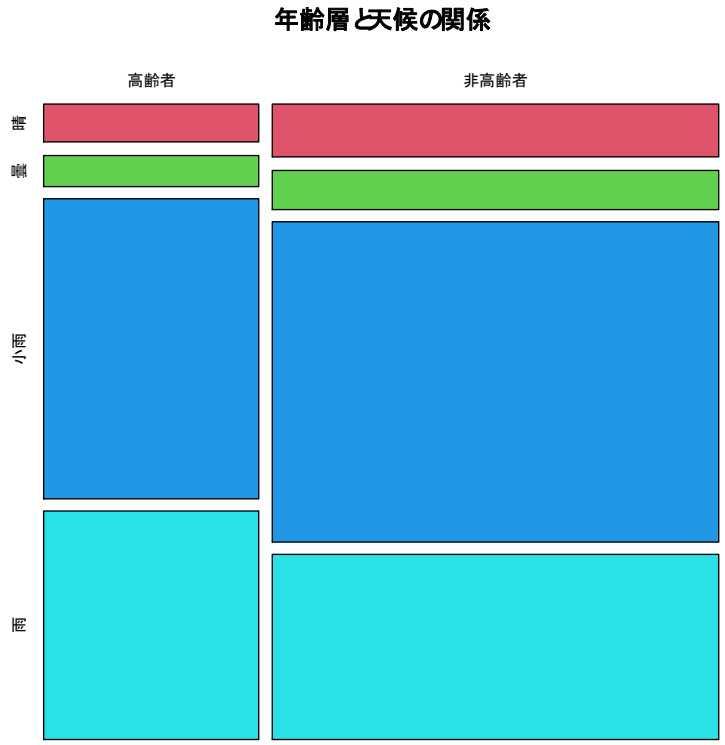
クロス集計表のグラフ表示

クロス集計表をグラフ表示する方法としてモザイク･プロットというものがある. では, mosaicplot() でびょうしゃできる.

mosaicplot()

`mosaicplot(x,main,xlab,ylab,col)` その他にもオプションはあるが割愛

x	:クロス集計表	main	:グラフのタイトル
xlab	:X軸の名前	ylab	:Y軸の名前
col	:塗りつぶしの色(ベクトル形式)		



Input

```
>  mosaicplot(conj.tbl, col=c(2,3,4,5),main="年齢層と天候の関係")
```

対応のあるクロス集計表

対応のないクロス集計表
(通常のクロス集計表)

いま，ゴミ処理施設を建設することを計画している．施設の周辺環境に対する環境への影響に関して，心配であるか否かを施設建設予定地付近の地域Aと，予定地から離れている地域Bでアンケート調査を行った．

	不安の有無	
	あり	なし
地域A	(a)	(b)
地域B	(c)	(d)

$$\text{地域Aの不安度} = \frac{(a)}{(a)+(b)}$$

比較

$$\text{地域Bの不安度} = \frac{(c)}{(c)+(d)}$$

対応のあるクロス集計表

いま，ゴミ処理施設を建設することを計画している．施設の周辺環境に対する環境への影響に関して，心配であるか否かを調査した後で，住民説明会を行った．その後，不安であるか否かを改めて調査した．

		説明後	
		不安あり	不安なし
説明前	不安あり	(a)	(b)
	不安なし	(c)	(d)

期待すること:

$$\text{不安あり} \rightarrow \text{不安なし} = \frac{(b)}{(a)+(b)+(c)+(d)}$$

あってはならないこと:

$$\text{不安なし} \rightarrow \text{不安あり} = \frac{(c)}{(a)+(b)+(c)+(d)}$$

対応のあるクロス集計表の例

ここでは，Cドライブの`Fukuoka_Seminor`というフォルダにある`data2.csv`というCSVファイルを読み込む．このデータを加工する．先ず，2018年度と2017年度を比較して上昇した市区町村にはup，そうでない市区町村にはdownとラベルとつけ，次いで，2019年度と2018年度を比較して，同様のラベルをつける．そのうえで，対応のあるクロス集計表をつくる．

Input

```
>   H2018 <- H2019 <- rep("Down", nrow(dat))
>   H2018[dat$Y2018 > dat$Y2017] <- "Up"
>   H2019[dat$Y2019 > dat$Y2018] <- "Up"
>   Conject.table(H2018,H2019)
```

関数`rep(a,N)`は`a`を`N`回繰り返すベクトルを作るための関数である．ここでは，`Down`という文字をデータフレーム`dat`の列数だけ作ることを意味する．

Output

```
             Down          Up
Down "32 (45.07)" "10 (14.08)"
Up   "28 (39.44)" "1 (1.41)"
```

行(縦方向)が2018年度，列(横方向)が2019年度である．両年度ともに前年度比で交通事故件数が減少している市区町村の割合は45.07%であった．また，2018年度は増加していたが2019年度には減少に転じた市区町村は39.44%であった．一方で，減少していたのに増加した市区町村は14.08%であり，各年度とも上昇傾向を示した市区町村は1件(1.41%)のみだった．

DAY.1：記述統計学とグラフの書き方

Section.3：自作したプログラムを動かしてみる

作成した関数を利用する方法

では，以前に作成した関数を使いまわすことで，業務を大幅に簡略化することができる．ここでは，3種類の関数を用意した．

平均値および標準偏差で要約：Summary.Means()

```
Summary.Means(dat,grp,dig)
```

dat　：データフレーム　　grp　：グループ変数（オプション）
dig　：表示する桁数（デフォルトは3）

中央値および第1・3四分位点で要約：Summary.Medians()

```
Summary.Medians(dat,grp,dig)
```

dat　：データフレーム　　grp　：グループ変数（オプション）
dig　：表示する桁数（デフォルトは3）

度数分布表を一括して計算：Summary.Freq()

```
Summary.Freq(dat)
```

dat　：データフレーム

自作した関数の読み込み

自作した関数は，関数source()を用いて読み込むことができる．ここでは，CドライブのFukuoka_Seminorに関数群が入っていることを想定する．

Input

```
> source("C:/Fukuoka_Seminor/Continuous_Summary.R")
```

Summary.Means()，Summary.Medians()が入っているソースを読み込み

Input

```
> source("C:/Fukuoka_Seminor/Categorical_Summary.R")
```

Summary.Freq() が入っているソースを読み込み

ここでは，CドライブのFukuoka_Seminorというフォルダにあるdata5.csvというCSVファイルを読み込む．これは，2017～2020年度の福岡県の市区町村別での交通事故発生件数のデータである．ここでは，Cityという変数に市区町村名が入っているので，それを列名にしたうえで計算する．

Input

```
> dat.Cont <- read.csv("C:/Fukuoka_Seminor/data5.csv",fileEncoding = "cp932")
> rownames(dat.Cont) <- dat.Cont [,1]
> dat.Cont <- dat.Cont[,-1]
> head(dat.Cont)
```

Output

	Y2017	Y2018	Y2019	Y2020	group
うきは市	9	9	7	3	0
みやま市	19	18	8	13	0
鞍手郡鞍手町	4	10	5	4	0
鞍手郡小竹町	1	6	3	0	0
遠賀郡芦屋町	5	6	3	3	0
遠賀郡遠賀町	18	12	5	6	0

ファイルdata2.csvにgroupを加えたものである．groupは人口が中央値以上を1，中央値未満を0とした変数である．

dat.Cont

	[,1]	[,2]	[,3]	[,4]	[,5]
	Y2017	Y2018	Y2019	Y2020	group
うきは市	9	9	7	3	0
みやま市	19	18	8	13	0
鞍手郡鞍手町	4	10	5	4	0
鞍手郡小竹町	1	6	3	0	0
遠賀郡芦屋町	5	6	3	3	0
遠賀郡遠賀町	18	12	5	6	0

各年度の平均および標準偏差を計算する.

Input

```
> Summary.Means(dat.Cont[,1:4])
```

Output

```
             [,1]
Y2017 "41.8(48.8)"
Y2018 "37.8(43.6)"
Y2019 "31.6(35.1)"
Y2020 "28(32.3)"
```

人口が中央値以上／未満で分けたうえで平均および標準偏差を計算する.

Input

```
> Summary.Means(dat.Cont[,1:4], dat.Cont[,5])
```

Output

```
                0            1
Y2017 "13.9(48.8)" "68.9(48.8)"
Y2018 "11.2(43.6)" "63.7(43.6)"
Y2019 "8.46(35.1)" "54.1(35.1)"
Y2020 "7.11(32.3)" "48.2(32.3)"
```

dat.Cont

	[,1] Y2017	[,2] Y2018	[,3] Y2019	[,4] Y2020	[,5] group
うきは市	9	9	7	3	0
みやま市	19	18	8	13	0
鞍手郡鞍手町	4	10	5	4	0
鞍手郡小竹町	1	6	3	0	0
遠賀郡芦屋町	5	6	3	3	0
遠賀郡遠賀町	18	12	5	6	0

各年度の中央値および第1・第3四分位点を計算する.

Input

```
>   Summary.Medians(dat.Cont[,1:4])
```

Output

```
      [,1]
Y2017 "29 [11.5, 42.5]"
Y2018 "25 [ 9.5, 40.5]"
Y2019 "21 [ 7.0, 38.0]"
Y2020 "17 [ 7.0, 32.5]"
```

人口が中央値以上／未満で分けたうえで中央値および第1・第3四分位点を計算する.

Input

```
>   Summary.Medians(dat.Cont[,1:4], dat.Cont[,5])
```

Output

```
                0                 1
Y2017 "12 [5.5, 17.5]" "41.5 [33.0, 92.0]"
Y2018 "10 [6.0, 14.0]" "40.5 [32.0, 92.8]"
Y2019 " 7 [4.5, 12.5]" "38.0 [28.0, 69.0]"
Y2020 " 7 [4.0,  9.5]" "32.5 [25.5, 57.2]"
```

平均値での結果を保存する

Input

```
>   result <- Summary.Means(dat.Cont[,1:4], dat.Cont[,5])
>   write.csv(result, "C:/Fukuoka_Seminor/Mean_Summary.csv",fileEncoding = "cp932")
```

	A	B	C
1		0	1
2	Y2017	13.9(48.8)	68.9(48.8)
3	Y2018	11.2(43.6)	63.7(43.6)
4	Y2019	8.46(35.1)	54.1(35.1)
5	Y2020	7.11(32.3)	48.2(32.3)

綺麗にExcelのファイルとして保存できる.

各市区町村の4年間の平均および標準偏差，中央値および第1・第3四分位点を計算する．

Input

```
> dat.Cont2 <- t(dat.Cont)[-5,]
> head(dat.Cont2)
```

dat.Cont

	[,1] Y2017	[,2] Y2018	[,3] Y2019	[,4] Y2020	[,5] group
うきは市	9	9	7	3	0
みやま市	19	18	8	13	0
鞍手郡鞍手町	4	10	5	4	0
鞍手郡小竹町	1	6	3	0	0
遠賀郡芦屋町	5	6	3	3	0
遠賀郡遠賀町	18	12	5	6	0

dat.Cont2

	[,1] うきは市	[,2] みやま市	[,3] 鞍手郡鞍手町	[,4] 鞍手郡小竹町	...
Y2017	9	19	4	1	...
Y2018	9	18	10	6	...
Y2019	7	8	5	3	...
Y2020	3	13	4	0	...

Input

```
> Summary.Means(dat.Cont2)
```

Output

```
                [,1]
うきは市        "7(2.83)"
みやま市        "14.5(5.07)"
鞍手郡鞍手町    "5.75(2.87)"
鞍手郡小竹町    "2.5(2.65)"
（以下省略）
```

Input

```
> Summary.Medians(dat.Cont2)
```

Output

```
                 [,1]
うきは市        "  8.0 [  6.00,   9.00]"
みやま市        " 15.5 [ 11.75,  18.25]"
鞍手郡鞍手町    "  4.5 [  4.00,   6.25]"
鞍手郡小竹町    "  2.0 [  0.75,   3.75]"
（以下省略）
```

ここでは，Cドライブの`Fukuoka_Seminor`というフォルダにある`data4.csv`というCSVファイルを読み込む．これは，福岡県内で発生した交通事故に関して，年齢層(高齢者/非高齢者)，時間帯(日中／夜)，天候に関するデータである．

Input

```
>   dat.Categ <- read.csv("C:/Fukuoka_Seminor/data4.csv",fileEncoding = "cp932")
>   head(dat.Categ)
```

Output

```
  Senior Day_Night Weather
1   高齢者         夜       曇
2 非高齢者         夜       晴
3 非高齢者         夜       晴
4 非高齢者         夜       晴
5 非高齢者         昼       晴
6   高齢者         昼       晴
```

各項目の度数分布表を計算する

Input

```
>   result <- Summary.Freq(dat.Categ)
>   result
```

Output

```
       vname     categ       result
1     Senior   高齢者   720 (32.5)
2            非高齢者 1498 (67.5)
3 Day_Night       昼 1283 (57.8)
4                 夜  935 (42.2)
5   Weather       雨   179 (8.1)
6                小雨   137 (6.2)
7                 晴 1164 (52.5)
8                 曇  738 (33.3)
```

結果を保存する

Input

```
>  write.csv(result, "C:/Fukuoka_Seminor/Freq_Summary.csv",fileEncoding = "cp932",row.names=FALSE)
```

	A	B	C
1	vname	categ	result
2	Senior	高齢者	720 (32.5)
3		非高齢者	1498 (67.5)
4	Day_Night	昼	1283 (57.8)
5		夜	935 (42.2)
6	Weather	雨	179 (8.1)
7		小雨	137 (6.2)
8		晴	1164 (52.5)
9		曇	738 (33.3)

綺麗にExcelのファイルとして保存できる.

Thank you for your kind attention

shimokaw@wakayama-med.ac.jp

toshibow2000@gmail.com

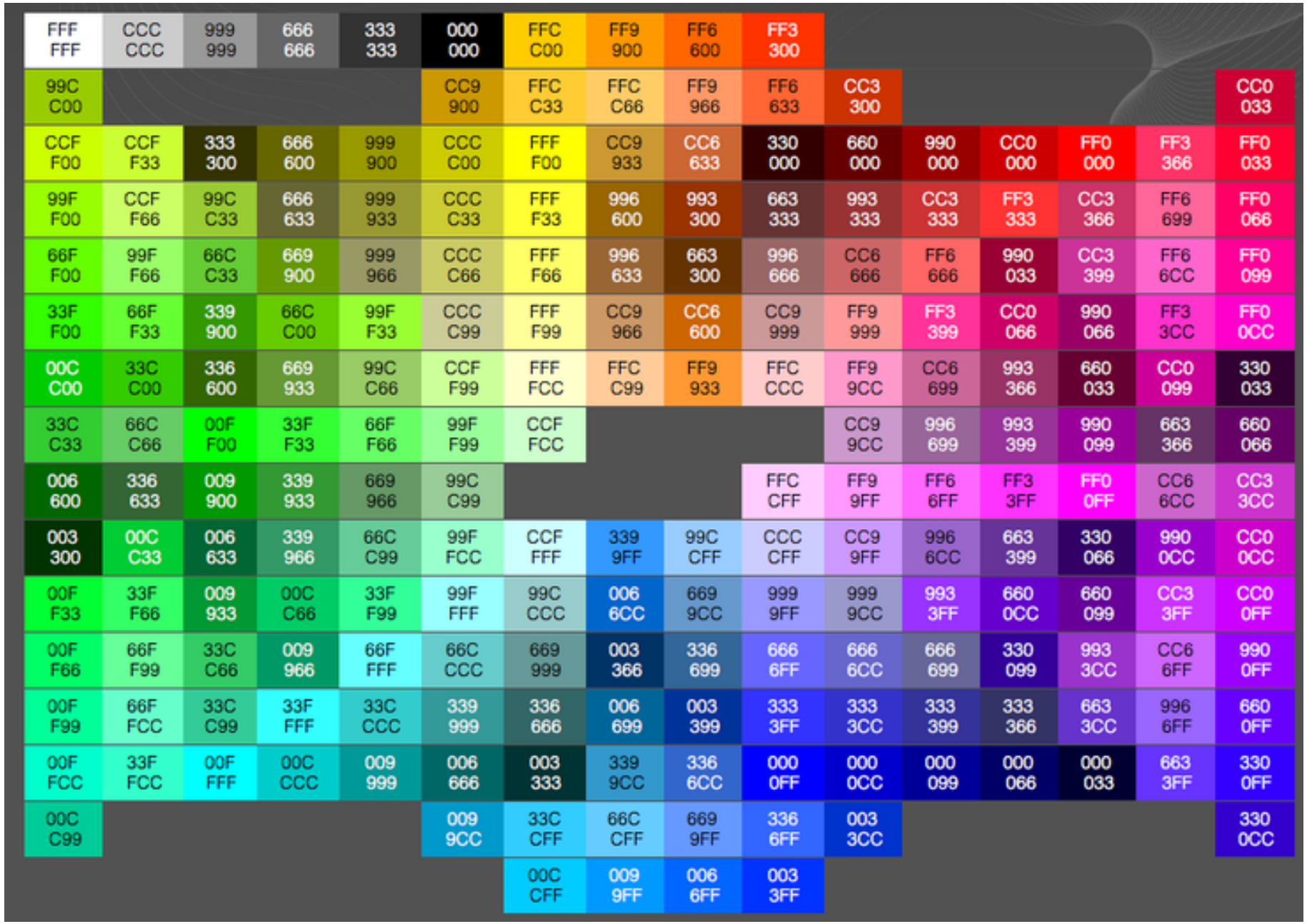
付録1:色の名前

brown4	darkorange4	gray	gray57	hotpink3	lightsalmon4	navajowhite1	plum3	slategray3	antiquewhite
brown3	darkorange3	goldenrod4	gray56	hotpink2	lightsalmon3	navajowhite	plum2	slategray2	aliceblue
brown2	darkorange2	goldenrod3	gray55	hotpink1	lightsalmon2	moccasin	plum1	slategray1	white
brown1	darkorange1	goldenrod2	gray54	hotpink	lightsalmon1	mistyrose4	plum	slategray	yellowgreen
brown	darkorange	goldenrod1	gray53	honeydew4	lightsalmon	mistyrose3	pink4	slateblue4	yellow4
blueviolet	darkolivegreen4	goldenrod	gray52	honeydew3	lightpink4	mistyrose2	pink3	slateblue3	yellow3
blue4	darkolivegreen3	gold4	gray51	honeydew2	lightpink3	mistyrose1	pink2	slateblue2	yellow2
blue3	darkolivegreen2	gold3	gray50	honeydew1	lightpink2	mistyrose	pink1	slateblue1	yellow1
blue2	darkolivegreen1	gold2	gray49	honeydew	lightpink1	mintcream	pink	slateblue	yellow
blue1	darkolivegreen	gold1	gray48	greenyellow	lightpink	midnightblue	peru	skyblue4	whitesmoke
blue	darkmagenta	gold	gray47	green4	lightgrey	mediumvioletred	peachpuff4	skyblue3	wheat4
blanchedalmond	darkkhaki	ghostwhite	gray46	green3	lightgreen	mediumturquoise	peachpuff3	skyblue2	wheat3
black	darkgrey	gainsboro	gray45	green2	lightgray	mediumspringgreen	peachpuff2	skyblue1	wheat2
bisque4	darkgreen	forestgreen	gray44	green1	lightgoldenrodyellow	mediumslateblue	peachpuff1	skyblue	wheat1
bisque3	darkgray	floralwhite	gray43	green	lightgoldenrod4	mediumseagreen	peachpuff	sienna4	wheat
bisque2	darkgoldenrod4	firebrick4	gray42	gray100	lightgoldenrod3	mediumpurple4	papayawhip	sienna3	violetred4
bisque1	darkgoldenrod3	firebrick3	gray41	gray99	lightgoldenrod2	mediumpurple3	palevioletred4	sienna2	violetred3
bisque	darkgoldenrod2	firebrick2	gray40	gray98	lightgoldenrod1	mediumpurple2	palevioletred3	sienna1	violetred2
beige	darkgoldenrod1	firebrick1	gray39	gray97	lightgoldenrod	mediumpurple1	palevioletred2	sienna	violetred1
azure4	darkgoldenrod	firebrick	gray38	gray96	lightcyan4	mediumpurple	palevioletred1	seashell4	violetred
azure3	darkcyan	dodgerblue4	gray37	gray95	lightcyan3	mediumorchid4	palevioletred	seashell3	violet
azure2	darkblue	dodgerblue3	gray36	gray94	lightcyan2	mediumorchid3	paleturquoise4	seashell2	turquoise4
azure1	cyan4	dodgerblue2	gray35	gray93	lightcyan1	mediumorchid2	paleturquoise3	seashell1	turquoise3
azure	cyan3	dodgerblue1	gray34	gray92	lightcyan	mediumorchid1	paleturquoise2	seashell	turquoise2
aquamarine4	cyan2	dodgerblue	gray33	gray91	lightcoral	mediumorchid	paleturquoise1	seagreen4	turquoise1
aquamarine3	cyan1	dimgrey	gray32	gray90	lightblue4	mediumblue	paleturquoise	seagreen3	turquoise
aquamarine2	cyan	dimgray	gray31	gray89	lightblue3	mediumaquamarine	palegreen4	seagreen2	tomato4
aquamarine1	cornsilk4	deepskyblue4	gray30	gray88	lightblue2	maroon4	palegreen3	seagreen1	tomato3
aquamarine	cornsilk3	deepskyblue3	gray29	gray87	lightblue1	maroon3	palegreen2	seagreen	tomato2
antiquewhite4	cornsilk2	deepskyblue2	gray28	gray86	lightblue	maroon2	palegreen1	sandybrown	tomato1
antiquewhite3	cornsilk1	deepskyblue1	gray27	gray85	lemonchiffon4	maroon1	palegreen	salmon4	tomato
antiquewhite2	cornsilk	deepskyblue	gray26	gray84	lemonchiffon3	maroon	palegoldenrod	salmon3	thistle4

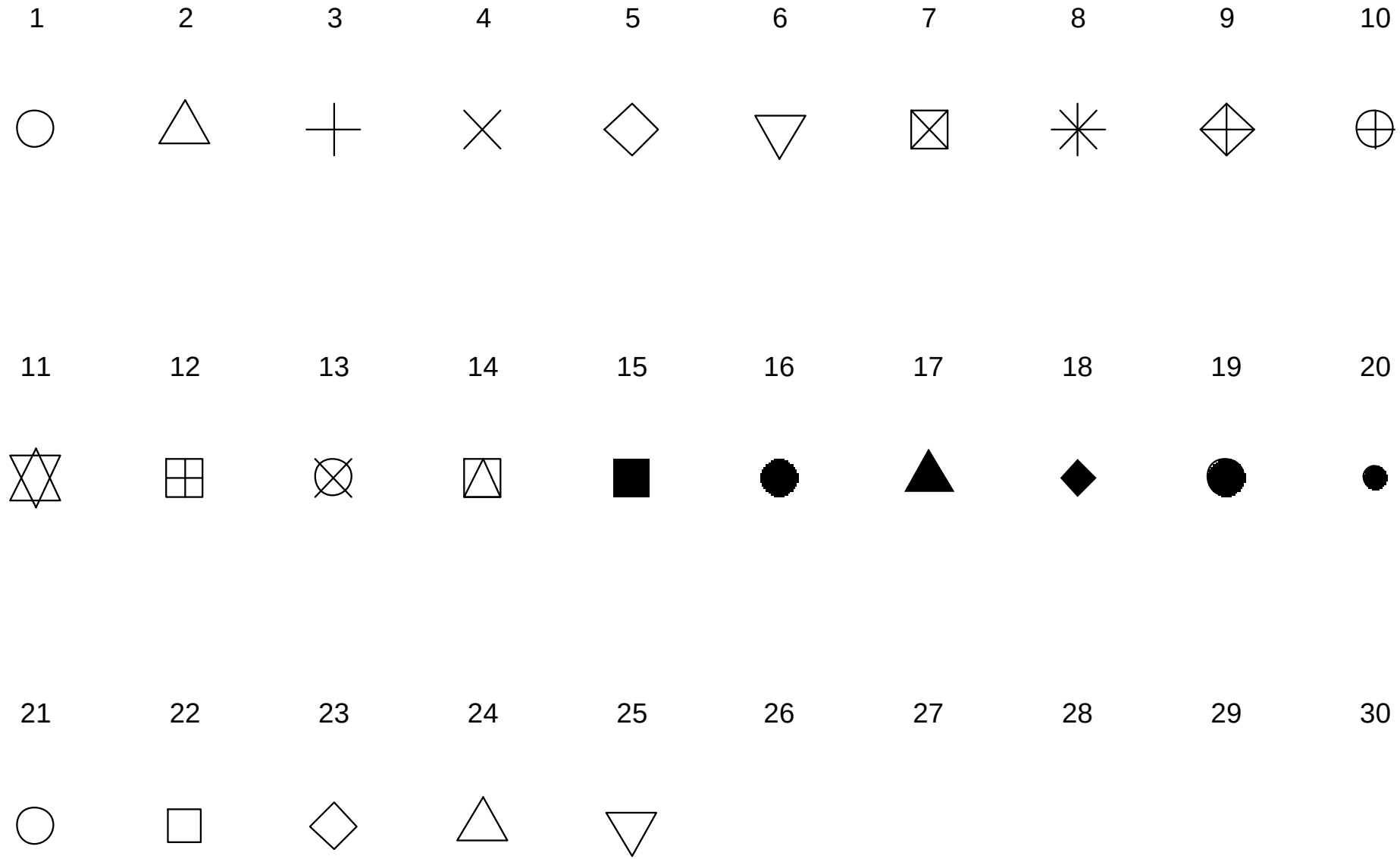
付録1:色の名前(続き)

antiquewhite1	cornflowerblue	deeppink4	gray25	gray83	lemonchiffon2	magenta4	orchid4	salmon2	thistle3
antiquewhite	coral4	deeppink3	gray24	gray82	lemonchiffon1	magenta3	orchid3	salmon1	thistle2
aliceblue	coral3	deeppink2	gray23	gray81	lemonchiffon	magenta2	orchid2	salmon	thistle1
white	coral2	deeppink1	gray22	gray80	lawngreen	magenta1	orchid1	saddlebrown	thistle
bisque3	coral1	deeppink	gray21	gray79	lavenderblush4	magenta	orchid	royalblue4	tan4
bisque2	coral	darkviolet	gray20	gray78	lavenderblush3	linen	orangered4	royalblue3	tan3
bisque1	chocolate4	darkturquoise	gray19	gray77	lavenderblush2	limegreen	orangered3	royalblue2	tan2
bisque	chocolate3	darkslategrey	gray18	gray76	lavenderblush1	lightyellow4	orangered2	royalblue1	tan1
beige	chocolate2	darkslategray4	gray17	gray75	lavenderblush	lightyellow3	orangered1	royalblue	tan
azure4	chocolate1	darkslategray3	gray16	gray74	lavender	lightyellow2	orangered	rosybrown4	steelblue4
azure3	chocolate	darkslategray2	gray15	gray73	khaki4	lightyellow1	orange4	rosybrown3	steelblue3
azure2	chartreuse4	darkslategray1	gray14	gray72	khaki3	lightyellow	orange3	rosybrown2	steelblue2
azure1	chartreuse3	darkslategray	gray13	gray71	khaki2	lightsteelblue4	orange2	rosybrown1	steelblue1
azure	chartreuse2	darkslateblue	gray12	gray70	khaki1	lightsteelblue3	orange1	rosybrown	steelblue
aquamarine4	chartreuse1	darkseagreen4	gray11	gray69	khaki	lightsteelblue2	orange	red4	springgreen4
aquamarine3	chartreuse	darkseagreen3	gray10	gray68	ivory4	lightsteelblue1	olivedrab4	red3	springgreen3
aquamarine2	cadetblue4	darkseagreen2	gray9	gray67	ivory3	lightsteelblue	olivedrab3	red2	springgreen2
aquamarine1	cadetblue3	darkseagreen1	gray8	gray66	ivory2	lightslategrey	olivedrab2	red1	springgreen1
aquamarine	cadetblue2	darkseagreen	gray7	gray65	ivory1	lightslategray	olivedrab1	red	springgreen
antiquewhite4	cadetblue1	darksalmon	gray6	gray64	ivory	lightslateblue	olivedrab	purple4	snow4
antiquewhite3	cadetblue	darkred	gray5	gray63	indianred4	lightskyblue4	oldlace	purple3	snow3
antiquewhite2	burlywood4	darkorchid4	gray4	gray62	indianred3	lightskyblue3	navyblue	purple2	snow2
antiquewhite1	burlywood3	darkorchid3	gray3	gray61	indianred2	lightskyblue2	navy	purple1	snow1
antiquewhite	burlywood2	darkorchid2	gray2	gray60	indianred1	lightskyblue1	navajowhite4	purple	snow
aliceblue	burlywood1	darkorchid1	gray1	gray59	indianred	lightskyblue	navajowhite3	powderblue	slategrey
white	burlywood	darkorchid	gray0	gray58	hotpink4	lightseagreen	navajowhite2	plum4	slategray4

付録1:色の名前 (Hex code)



付録2：pchの一覧


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

付録3：平均値と標準偏差を一括して計算するプログラムを作る

```
Summary.Means <- function(dat, grp=NULL, dig=3){
          P <- ncol(dat)
          if (is.null(grp)==TRUE){
                    Ms <- colMeans(dat, na.rm=T)
                    Sds <- apply(dat,2,sd,na.rm=T)
                    res <- matrix(numeric(P),ncol=1)
                    for (i in 1:P){
                              res[i,1] <- sprintf("%s(%s)", format(Ms[i], digit=dig), format(Sds[i], digit=dig))
                    }
                    rownames(res) <- colnames(dat)
          }
          if (is.null(grp)==FALSE){
                    gname = sort(unique(grp))
                    P.g <- length(gname)
                    res <- matrix(numeric(P*P.g), ncol=P.g)
                    for (j in 1:P.g){
                              dt <- dat[grp==gname[j],]
                              Ms <- colMeans(dt, na.rm=T)
                              Sds <- apply(dat,2,sd,na.rm=T)
                              for (i in 1:P){
                                        res[i,j] <- sprintf("%s(%s)", format(Ms[i], digit=dig), format(Sds[i], digit=dig))
                              }
                    }
                    rownames(res) <- colnames(dat)
                    colnames(res) <- gname
          }
          return(res)
}
```

付録4：中央値と第1,第3四分位点を一括して計算するプログラムを作る

```
Summary.Medians <- function(dat, grp=NULL, dig=3){
            P <- ncol(dat)
            if (is.null(grp)==TRUE){
                        Meds <- format(apply(dat,2,median, na.rm=T),digit=dig)
                        Q1s <- format(apply(dat,2,quantile,0.25,na.rm=T),digit=dig)
                        Q3s <- format(apply(dat,2,quantile,0.75,na.rm=T),digit=dig)
                        res <- matrix(numeric(P),ncol=1)
                        for (i in 1:P){
                                    res[i,1] <- sprintf("%s [%s, %s]", Meds[i], Q1s[i], Q3s[i])
                        }
                        rownames(res) <- colnames(dat)
            }
            if (is.null(grp)==FALSE){
                        gname = sort(unique(grp))
                        P.g <- length(gname)
                        res <- matrix(numeric(P*P.g), ncol=P.g)
                        for (j in 1:P.g){
                                    dt <- dat[grp==gname[j],]
                                    Meds <- format(apply(dt,2,median, na.rm=T),digit=dig)
                                    Q1s <- format(apply(dt,2,quantile,0.25,na.rm=T),digit=dig)
                                    Q3s <- format(apply(dt,2,quantile,0.75,na.rm=T),digit=dig)
                                    for (i in 1:P){
                                                res[i,j] <- sprintf("%s [%s, %s]", Meds[i], Q1s[i], Q3s[i])
                                    }
                        }
                        rownames(res) <- colnames(dat)
                        colnames(res) <- gname
            }
            return(res)
}
```

付録5:度数分布表を一括して計算するプログラムを作る

```
Summary.Freq <- function(dat){
            P <- ncol(dat)
            Re <- vn <- list()
            for (j in 1:P){
                        tb <- table(dat[,j])
                        pr <- prop.table(tb)*100
                        N <- length(tb)
                        st <- numeric(N)
                        vn[[j]] <- rep("",N)
                        vn[[j]][1] <- colnames(dat)[j]
                        names(st) <- names(tb)
                        for (i in 1:N){
                                    st[i] <- sprintf("%d (%3.1f)", tb[i], pr[i])
                        }
                        Re[[j]] <- st
            }
            unRe <- unlist(Re)
            result <- data.frame(vname=unlist(vn), categ=names(unRe), result=unRe)
            rownames(result) <- 1:nrow(result)
            return(result)
}
```