高校生のためのR入門

授業の前に

タッチタイピングに慣れよう。

　タッチタイピングとは、パソコンにキーボード入力を行う際に、キーボード面を注目せずに、指先の感覚をもとにキーを叩くタイピング技法で、プログラミングなどでは欠かせないもの。

自習用に自分のパソコンにRをインストールしよう。

　ネット上で検索して、Rのダウンロードサイトからサービスを受けることができる。Rはフリーソフトである。

入力方法

R console のみ利用の場合

R console とR Editorの利用の場合（推奨）

・「ファイル」→「新しいスクリプト」を選択。
・「R Editor」が起動し「R Console」と2画面になる。
「R Editor」に命令を打つごとに「**Ctrl+R**」を打つと実行結果がその都度,
「R Console」反映される。

Rの基本演算

まず，＞というプロンプトにより〝4＋3〝と入力しEnterキーを押す。

```
> 4+3                                  # キー入力
[1] 7                                  # 出力結果（和）
```

上と同様に，差を求めるには次のように入力する。

```
> 12－3                                #　引き算
[1] 9
```

次は商（割り算）を求める。

```
> 24/6                                 #　割り算
[1] 4
```

3の2乗は，次のように入力する。

```
> 3^2                                  #　累乗
[1] 9
```

保存の方法（適当なフォルダに保存）

○ **テキストファイルで保存：**「ファイル」→「ファイルを保存」で「***.txt」と保存

○ **Rファイルで保存：**「R Editor」のファイルはRファイル「***.R」と保存（推奨）

読込と実行の方法

○ **テキストファイル**：「R Console」に貼り付けて実行。

○「**Rファイル**」：「ファイル」→「スクリプトを開く」で読み込む。
一行ごとに実行できるが，「編集」→「全て実行」で一括して実行できる。（推奨）

Rの基本演算（組み込み関数の例：平方根、合計）

Rにはたくさんの組み込み関数が用意されている。

正の平方根は,

```
> sqrt(16)                                    #　平方根
［1］4
```

次はデータ（リスト形式）の合計を返す。

```
> sum(1,2,3,4,5)                              #　合計
[1］15
```

Rでの統計処理

プログラミング言語としての**R**の概念

Rはオブジェクト型言語.
オブジェクトは「関数オブジェクト」, 「データオブジェクト」から成り立つ.

関数オブジェクト：

関数（引数）　　#　functin(argument)

sqrt(16)

データオブジェクト（関数c を用いた場合）：

c (データ)　　# 結合：concatenate

c (1, 2, 3)

関数オブジェクトやデータオブジェクトに名前をつける：

オブジェクト名　<― 関数（引数）　　# 代入：assignment

data<-c(1, 2, 3)

簡単な統計処理

【実習３】（データの作成）

```
> c (1, 2, 3, 4)                    # データの作成
 [1] [1,2,3,4]

> data<-c(1, 2, 3, 4, 5)            # データをオブジェクトdataに代入

> sum(data)                         # データdata合計
 [1] 15

> mean(data)                        #  データdataの平均値
 [1] 3

> data^2                            # 各データdataの２乗
[1] 1, 4, 9, 16, 25
```

簡単な統計処理

【実習４】（data1を定義してその代表値を求める）

```
> data1<-c(7, 6, 4, 8, 9, 6, 5, 7, 8, 5, 8, 7, 10, 6, 7, 7, 8, 9, 4)      # data1の定義

> sum(data1)                                                              # data1の合計
 [1] 131

> mean(data1)                                                             # data1の平均値
 [1] 6.894737

> median(data1)                                                           # data1の中央値
 [1] 7
```

簡単な統計処理

【実習４】（data1を定義してその代表値を求める）

Rには最頻値の組み込み関数はないので，tableコマンドで頻度を求める。

```
> table(data1)
 data1
  4  5  6  7  8  9 10                    # data1
  2  2  3  5  4  2  1                    # data1の頻度

> max(data1)                             # data1の最大値
 [1] 10

> min(data1)                             # data1の最小値
 [1] 4

> length(data1)                          # data1のデータの個数
[1] 19
```

簡単な統計処理

【実習５】（分散、標準偏差）

＞ var(data1)　　#　data1の不偏分散

［1］2.766082

この結果は不偏分散{÷（n-1)}、標本分散は｛÷ｎ｝）。

標本分散（1学期に定義した分散）を求めるには定義もしくは定義から導いた式を用いて計算することで標本分散を求める。標本分散の計算（1学期に学習済）に従って入力する。

＞ mean(data1^2)-mean(data1)^2　　#　data1の標本分散（2乗の値の平均値ー平均値の2乗）

［1］2.620499

別の方法として次の計算でも求まる。

＞ var(data1)*18/19　　#　data1の標本分散

［1］2.620499

標準偏差は分散の正の平方根である。

>sqrt(mean(data1^2)-mean(data1)^2)　　# data1の標準偏差

[1]　1.618795

簡単な統計処理

【実習６、７】（ユーザー定義関数・要約）

```
> varp<-function(x){var(x)*(length(x)-1)/length(x)}          #ユーザー定義
```
上のように定義をすると，**varp**コマンドが有効になる。

```
> varp(data1)
[1] 2.620499                          # data1の標本分散（ユーザー定義関数を用いた）
```

```
> summary(data1)                                              #　要約
   Min.   1st Qu.  Median    Mean    3rd Qu.    Max.
   4.000   6.000   7.000     6.895     8.000    10.000
```
ここで，1st Qu. は第1四分位数，3rd Qu. は第3四分位数である。

簡単な統計処理

【実習８】（相関係数）

```
>x<-c(7, 6, 4, 8, 9, 6, 5, 7, 8, 5, 8, 7, 10, 6, 7, 7, 8, 9, 4)
>y<-c(3, 5, 1, 8, 3, 5, 2, 9, 3, 9, 1, 5, 9, 4, 6, 2, 9, 5, 3)
>cor(x, y)                                          # x と y の相関係数
 [1] 0.3070567
```

と計算できる。

Rでのデータ形式

Rのデータ形式

「データの型」
数値型(**numeric**)，文字型(**character**)など
これらを組み合わせて複雑なデータを定義できる。

「データ構造」（データの集合）
ベクトル(**vector**)，行列(**matrix**)，配列(**array**)，リスト(**list**),
データフレーム(**dataframe**)

【実習9】（ベクトル）

```
> vec1<-c(1,2,3,4)                    # オブジェクト名<-関数（引数）
```

同様に，vec2として，

```
> vec2<-c(5,6,7,8)                    # オブジェクト名<-関数（引数）
> vec1+vec2                           #  ベクトルの和
  [1] 6  8 10 12
> vec1－vec2                          #  ベクトルの差
  [1] -4 -4 -4 -4
> vec1*vec2                           # 成分ごとの積のデータ
  [1] 5 12 21 32
> vec1%*%vec2                         # ベクトルの内積
  [1,] 70
> vec1[2];  vec1[1:3]                 # vec1の第2要素；vec1の第1~第3要素の取出
[1] 2          [1] 1 2 3
```

行列と配列（matrix　array）

```
> m<-matrix(c(1,2,3,4,5,6),2,3)        #　行列(2行3列）のコマンド（定義）
> m
     [,1]  [,2]  [,3]
[1,]    1    3    5
[2,]    2    4    6

> a<-array(1：4,dim=c(4))                    #　1次元配列（1行4列）
> a<-array(1：6, dim=c(2,3))                 #　2次元配列（2行3列）
> a<-array(1:6,dim=c(2,3,2))                 #　3次元配列
```

（各自で実行しよう）

【実習10】（データフレーム）

```
>fr<-data.frame(name=c("青木","大木","加藤","山崎"),mark=c(20,24,12,32))
> fr                                                    # データフレーム作成
  name mark
1 青木   20
2 大木   24
3 加藤   12
4 山崎   32

> mean(fr$mark)                  # 要素の操作：＄でmark変数にアクセス（平均値）
[1] 22
```

【実習11】（リスト list）

```
> list1<-list(“2年”,“早稲田太郎”,c(10,20,32))          # 異なる型にも対応
> list1
    [[1]]                                              # １番目のリスト
    [1] "2年"
    [[2]]                                              # 2番目のリスト
    [1] "早稲田太郎"
    [[3]]                                              # 3番目のリスト
    [1] 10 20 32
> list1[[1]]                                  #１番目を直接取り出すコマンド
  [1] "2年"
```

Rでのデータの操作

データの読み込み（インポート）

【実習12】データセットの読み込み

```
>WorldPhones
>class(WorldPhones)                                    # データの型の確認
[1] "matrix“
>wp <- data.frame(WorldPhones)                         # データフレーム形式に変換
```

(2) データの読み込み（※ディレクトリについて）

例えば、Jドライブの Rフォルダにある”sheet1.csv”を読み込むときは次のようにする。

```
 data1 <- read.csv("J:/R/sheet1.csv")
```

同じ階層の場合（デスクトップ上であれば）

```
 data1 <- read.csv("sheet1.csv")
```

データの読み込み（インポート）

【実習13】データの読み込み

data1 <- read.csv("J:/R/sheet1.csv", header=TRUE)

header=TRUE (最初の行にラベル有), FALSE（ラベル無）

	sex	region	name	Japanese	Math	English
1	1	tokyo	aoki	23	56	65
2	2	tokyo	ehara	43	45	44
3	2	saitama	kato	11	97	76
4	1	chiba	kayama	56	32	23
5	2	chiba	kinoshita	89	41	54

> class(data1) # データの型の確認

[1] "data.frame"

データの加工

【実習14】データの加工

①データ構造の確認

>str(data1) # データの構造

> dim(data1) # データの配列

[1] 30 7

②抽出

```
> data1[1,2]                                          #  1行2列目の抽出
 [1]  Tokyo
> data1[1,]          data1[1,1:6]と同じ                #  1行目の抽出
 sex  region name Japanese Math English
  1  tokyo   aoki       23     56      65
> data1[1:3,]          data1[1:3,1:6]と同じ            # 1~3行目の抽出
 sex  region  name Japanese Math English
  1   tokyo   aoki       23   56      65
  2   tokyo   ehara      43   45      44
  2   saitama  kato      11   97      76
> data1[,5]              data1[1:30,5]と同じ           #  5列目の抽出
 [1] 56 45 97 32 41 78 45 55  (略) 86 76 54 67 76 91 55 69 48 83 81 79 89
> data1$Math                                          # $Math変数へアクセス
 [1] 56 45 97 32 41 78 45 55  (略) 86 76 54 67 76 91 55 69 48 83 81 79 89
> data1[,4]+data1[,5]                                 #  演算（4列目＋5列目）
```

③並べ替え

```
> sort(data1$Math)                                    #　Math列の並べ替え(昇順)
[1] 12 31 32 34 41 44 45 45 45 48 54 （略）78 78 79 79 81 83 86 89 91 97

> sort(data1$Math,decreasing=TRUE)      #　Math列の並べ替え(降順)
[1] 97 91 89 86 83 81 79 79 78 78 76 （略） 48 45 45 45 44 41 34 32 31 12
```

④ apply関数：apply, tapply, sapply, lapply：（部分和）

```
apply(data,1,sum)     # apply（data,1：行/2：列,関数名）
```
行列・データフレーム対応

```
> apply(data1[,4:6],1,sum)                    #  各行の合計（個人の三教科合計）
 [1] 144 132 184 111 184 215 125 131 176 108 169 181 253 154 204 187 180
 165 182 178 145 188 265 167 188 164 189 203 223 193

> apply(data1[,4:6],2,sum)                    #  各列の合計（各教科の合計）
 Japanese     Math  English
     1681     1861     1746

> sapply(data1[,4:6],sum)                     #  各列の合計（各教科の合計）
 Japanese     Math  English
     1681     1861     1746
> lapply(data1[,4:6],sum)                     #  各列の合計（各教科の合計をリスト形式で出力）
```

apply

sapply（ベクトル・データフレームの列に対応）

⑤表の再構成（**data2**をつくる）

```
> mark <- apply(data1[,4:6],1,sum)        #合計をmarkオブジェクト
> data2 <- data.frame(data1,mark)         # data1とmarkを結合
> data2
```

	sex	region	name	Japanese	Math	English	mark
1	1	tokyo	aoki	23	56	65	144
2	2	tokyo	ehara	43	45	44	132
3	2	saitama	kato	11	97	76	184
4	1	chiba	kayama	56	32	23	111

データの一部にアクセス（抽出機能）

【実習15】
(1) 男子のみの抽出
(2) 東京都出身を抽出
(3) 英語が60点以上を抽出
(4) 女子で国語が60点以上を抽出

```
> data1[data1$sex==1,]
> data1[data1$region=="tokyo",]
> data1[data1$English>=60,]
> data1[data1$Japanese>=60 & data1$sex==2,]
```

データの一部にアクセス（抽出機能）

【実習16】
(1) 東京出身の国語の合計点
(2) 埼玉出身の英語の平均点
(3) 数学が50点以上の人数
(4) 神奈川県出身で英語が80点未満の人数
(5) 三教科の合計が180点以上の人数

```
> attach(data1)                          # data1の各変数への直接アクセス可能
> tapply(Japanes,region,sum)
                          # tapply(data1$Japanese,data1$region,sum)と同じ
> tapply(English,region,mean)
                          # tapply(data1$English,data1$region,sum)と同じ
> dim(data1[data1$Math>=50,])
> dim(data1[data1$English<80 & data1$region=="kanagawa",])
> dim(data2[data2$mark>=180,])
```

データの書き込み

新たに三教科合計の欄を追加した表（data2）をsheet2として保存する。

【実習１７】”data2”をsheet2.csvで保存（Hドライブに保存）

```
> write.csv(data2,"H:sheet2.csv",row.names=FALSE)
```

row.name=FALSEとすると最左列のヘッダを付けない。

データに欠損値がある場合【発展】

```
> data3<-read.csv("J:/R/sheet3.csv",header=TRUE)
> data3
```

	sex	region	name	Japanese	Math	English
1	1	tokyo	aoki	23	56	65
2	2	tokyo	ehara	43	45	44
3	2	saitama	kato	11	97	76
4	1	chiba	kayama	56	32	23
5	2	chiba	kinoshita	89	41	54
6	1	kanagawa	kojima	58	78	79
7	2	saitama	saito	54	45	26
8	1	saitama	sakamoto	33	55	43
9	2	USA	suzuki	NA	NA	NA
10	2	kanagawa	sugiyama	33	31	44

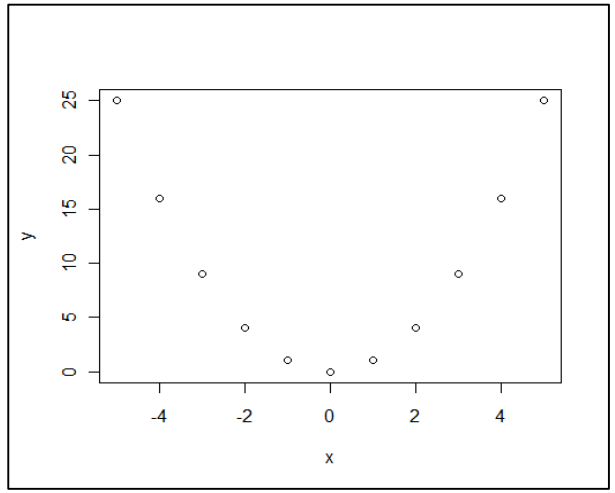
データに欠損値がある場合【発展】

```
> apply(data3[4:6],1,sum,na.rm=TRUE)          # 欠損値を無視した列の部分和

 [1] 144 132 184 111 184 215 125 131   0 108 169 181 253 154 204 187 180
168 182 178 145 188 265 113 188 164 189 203 223 193

> apply(data3[4:6],2,sum,na.rm=TRUE)          # 欠損値を無視した行の部分和
 Japanese     Math  English
     1551     1827     1683

> na.omit(data3)                                          # 欠損値データの削除
```

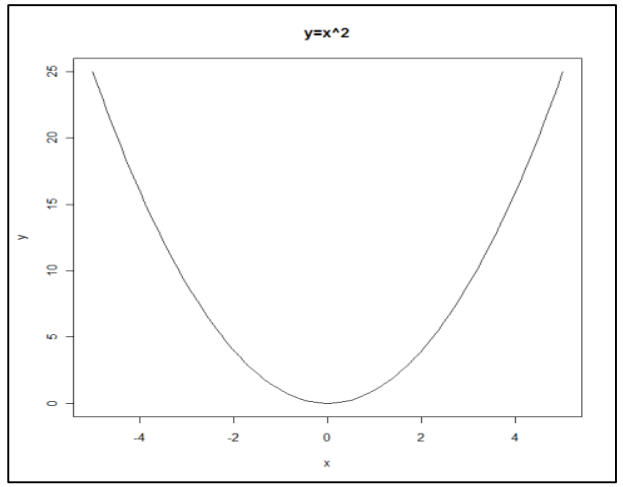
Rでのグラフィックス

散布図(plot)

```
> x <- c(-5:5)                                  # x軸
> y <- c(25, 16, 9, 4, 1, 0, 1, 4, 9, 16, 25)   # y軸

> plot(x, y, type=‘p’, xlab='x', ylab='y')
```

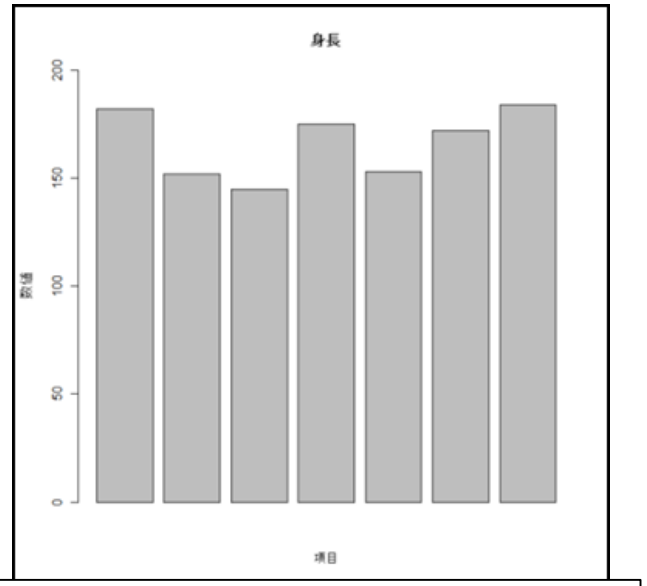


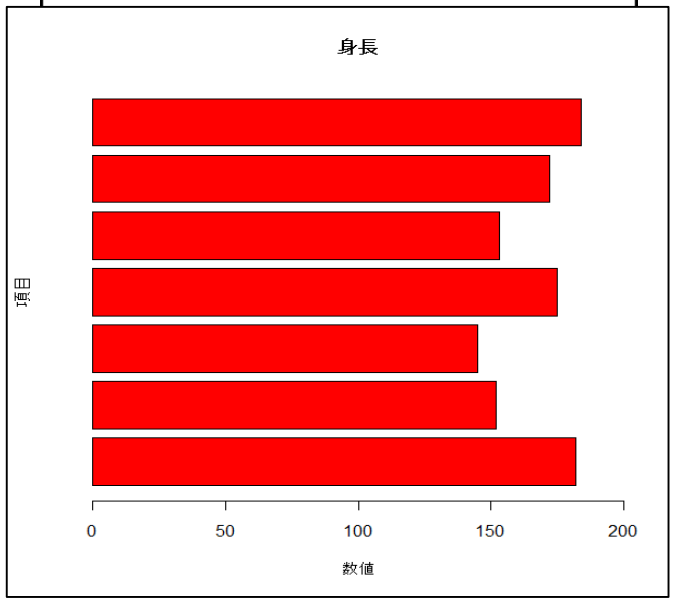

```
> x <- seq(-5, 5, length=100)
                    # xを-5から5まで100の長さの数列を生成
> y <- x^2                                      # yを関数で指定

> plot(x, y, type='l', xlab='x', ylab='y', main='y=x^2')
```



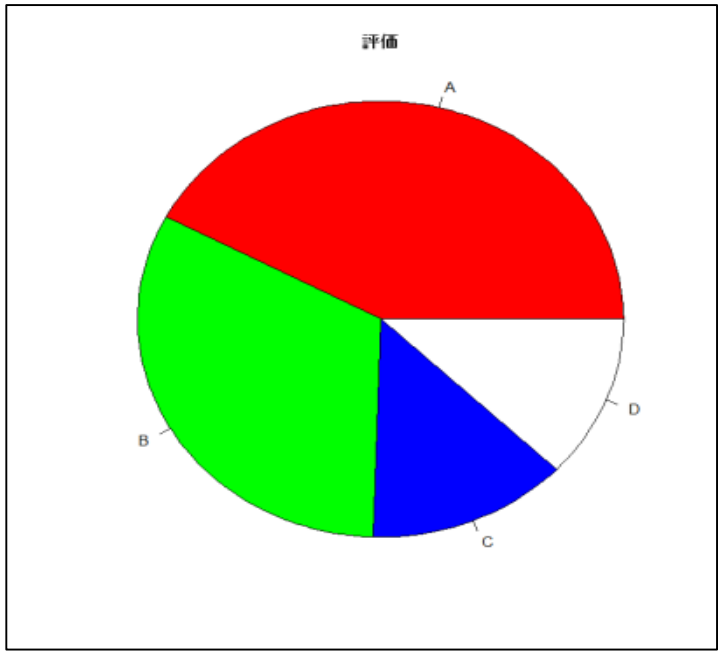
棒グラフ(barplot)

```
> y <- c(182,152,145,175,153,172,184)
                                          #ベクトル型データ
> barplot(y,ylim=c(0,200),
    xlab='項目',
    ylab='数値',
    main='身長`
  )

                                          #  縦グラフ
> barplot(y,xlim=c(0,200),
     xlab='数値',
     ylab='項目',
     main='身長',
     beside=TRUE,horiz=TRUE,col='red`
  )
                                          #   横グラフ
```





円グラフ(pie)




```
> vec1 <- c(42.2, 32.3, 13.4, 12.1)
                                    #非負ベクトル型データ

> names(vec1) <- c('A','B','C','D')
                                    # ヘッダの追加：ラベル名

> vec1.col <-c('red','green','blue','white')
                                    # 色データ

> pie(vec1,radius=1,col=vec1.col,main='評価')
```

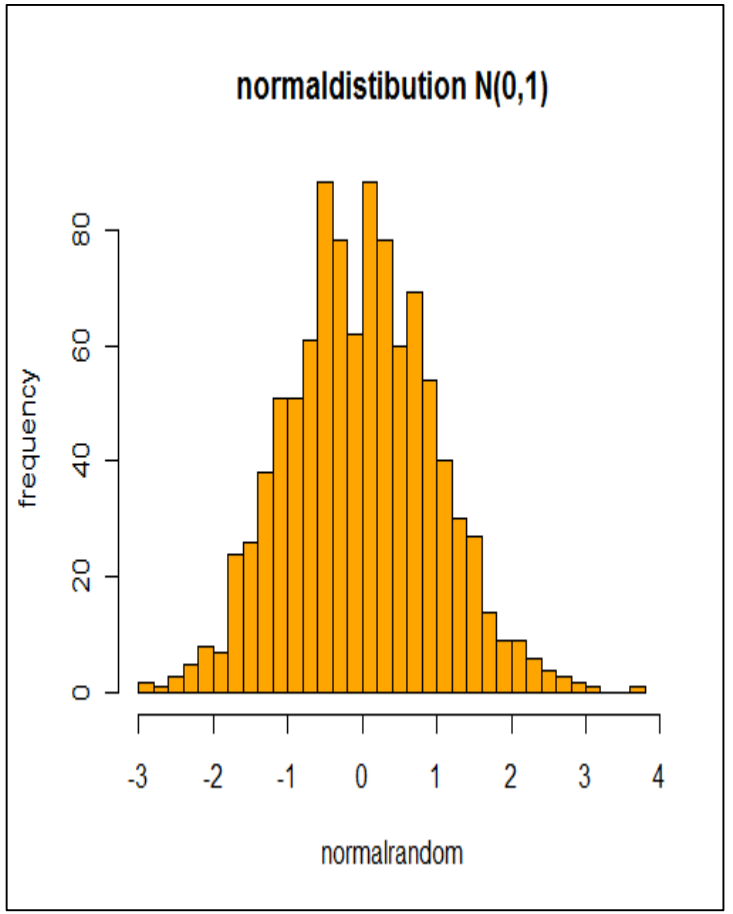
ヒストグラム(hist)

```
> nrd = rnorm(1000,0,1)
```

ベクトル型データ

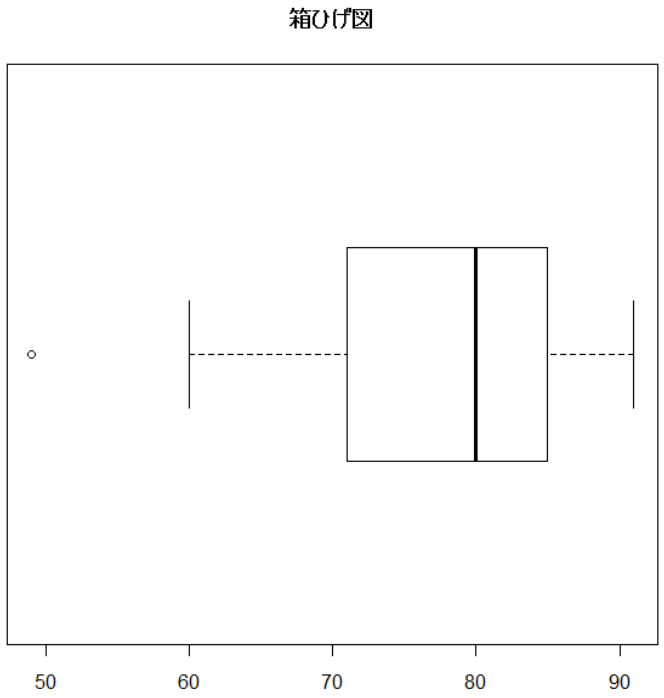
平均0,標準偏差1である正規分布をなす乱数1000個を生成

```
>hist(nrd,breaks=25,
 xlab="normalrandom",
 ylab="frequency",
 main="normaldistibutionN(0,1)",
 col="orange"
 )
```



箱ひげ図(boxplot)

```
> vec2 <- c(79,90,80,81,83,60,76,66,91,49,87)
                                          #  ベクトル型データ
> range(vec2)                                    #  範囲
[1] 49 91
> quantile(vec2)                               #  四分位数
  0%  25%  50%  75% 100%
  49   71   80   85   91
> IQR(vec2)                                  #  四分位範囲
 [1] 14
> boxplot(vec2,horizontal=TRUE main='箱ひげ図')
                                          #  横型指定
```



グラフィックスの保存

保存の方法は主に2通りある。

一つは，作成したグラフを右クリックして，コピー・保存メニューを選択する（下左図）。

もう一つは，「**R Graphics**ウインドウ」がアクティブ（前面）のとき，
「ファイル」→「別名で保存」でいくつかのファイル形式で保存する（下右図）。