７日目：項目のチェック（１）

データがきちんと読み込まれたことを確認したら，次にすることといえば，項目ごとのチェックでしょう。それには度数分布表をつくったり，平均値などの基礎統計量を計算したりすることが必要です。今日と明日，明後日は，このような基礎的な項目チェックをやってみましょう。

まずは，R を起動し，いつもの練習用のファイルを読み込んでください。はじめに，項目ごとの平均値や標準偏差，最小値，最大値などを算出してみたいと思います。

基本的なコマンドは以下のようです。

・最小値

```
min( )
```

・最大値

```
max( )
```

・平均値

```
mean( )
```

・標準偏差

```
sd( )
```

これらのカッコの中で，求めたい変数を指定することになります。

このような際，せっかちな私は，「ファイル名。つまり x をいれてみたらいいじゃない」などと発想してしまうわけです。

では x を入れて，min(x)などとして実行させるとどうなるか…

これだと，全然ダメ…というわけでもなく，何か計算はしてくれたり，興味深いアドバイスを返してくれたりします。「どういうこと？」と探ってみるのも，よい勉強になると思いますが…。時間があれば，ちょっとやってみてください。

順番に説明を加えていきましょう。

まず最小値と最大値については，min(x$年齢) とか，man(x[3])などと，ファイルと変数名，もしくは列番号で指定します。複数変数を一気にやろうとして min(x[1:3])などと指定してしまうと，おかしなことになります。これは，それぞれの列の最小値ではなく，１列目から３列目の全データの中の最小値を求めていることになるようです。ちなみに，昨日作った変数名をまとめる方法を使っても，うまくいきません。

また，min(x$b1)もうまくいかず，「NA」という結果を返してきます。これはb1に欠損値が含まれていることに起因します。欠損値が入っている列に対しては，「NA」と返してくるのです。そのため，「NA」を抜いて実行させるためには，max(x$b1, na.rm=TRUE)と，na.rm=TRUE(No AnswerをRe Moveと覚えればよいでしょう)を添えてやる必要があります。

次に平均値や標準偏差も，mean(x$年齢) とか，sd(x[3])などと記述すればOKです。ただし，これらも欠損値が含まれると「NA」という結果を返してきますので，mean(x$b1, na.rm=TRUE)と，na.rm=TRUEを添えてやる必要があります。

ところが最小値などと違い，こちらはmean(x[2:7], na.rm=TRUE) といった複数列指定ができます。

図のように警告は出てきますが，計算はできているようです。

```
> mean(x[2:7], na.rm=TRUE)
 性別  年齢    b1    b2    b3    b4
 1.70 19.35  2.95  3.85  4.45  4.00
 警告メッセージ:
mean(<data.frame>) is deprecated.
 Use colMeans() or sapply(*, mean) instead.
```

ちなみに，警告メッセージの内容は，「mean(<data.frame>)という形式はよくない。colMeans()か，sapply(*, mean) を使いなさい」ということですね。<data.frame>(データフレーム)とは，今回のデータのような，行ごとに1ケースのデータが，列に変数が配置されているようなデータ形式のことを指します。このような形式のデータをmean()でやるのは良くないよと教えて（？）くれているわけです。

では，代案通り，colMeans()でやってみましょう。カッコ内はx[2:7]で大丈夫です。ただし，こちらもna.rm=TRUEが必要なので，colMeans(x[2:7], na.rm=TRUE)となります。sapply(*, mean)は，「*」の部分にデータ名と変数名をいれます。こちらもna.rm=TRUEが必要なので，

sapply(x[2:7], mean, na.rm=TRUE)

となります。ちなみに，とても簡単に説明するとsapply(A, 関数B)は，Aに対して，関数Bを繰り返しなさいという命令です。

右図のように，確かに計算はうまくいっているようです。

```
> mean(x[2:7], na.rm=TRUE)
 性別  年齢    b1    b2    b3    b4
 1.70 19.35  2.95  3.85  4.45  4.00
 警告メッセージ:
mean(<data.frame>) is deprecated.
 Use colMeans() or sapply(*, mean) instead.
> colMeans(x[2:7], na.rm=TRUE)
 性別  年齢    b1    b2    b3    b4
 1.70 19.35  2.95  3.85  4.45  4.00
> sapply(x[2:7], mean, na.rm=TRUE)
 性別  年齢    b1    b2    b3    b4
 1.70 19.35  2.95  3.85  4.45  4.00
```

次に標準偏差の方ですが，こちらも sd(x[2:7], na.rm=TRUE) といった複数列指定ができるのですが，平均値と同じように警告が出ます。sapply を使えと。

ご指示に従い，sapply(x[2:7], sd, na.rm=TRUE)とするとうまくいきます。

```
> sd(x[2:7], na.rm=TRUE)
      性別       年齢         b1         b2         b3         b4
0.4701623 0.5871429 1.2763022 0.9333020 0.6863327 0.7254763
 警告メッセージ:
sd(<data.frame>) is deprecated.
 Use sapply(*, sd) instead.
> sapply(x[2:7], sd, na.rm=TRUE)
      性別       年齢         b1         b2         b3         b4
0.4701623 0.5871429 1.2763022 0.9333020 0.6863327 0.7254763
```

この sapply は，min や max にも使えます。となると，今回のデータの no をのぞく全変数の最小値，最大値，平均値，標準偏差を一気に求めるなら…

```
sapply(x[2:8], min, na.rm=TRUE)
sapply(x[2:8], max, na.rm=TRUE)
sapply(x[2:8], mean, na.rm=TRUE)
sapply(x[2:8], sd, na.rm=TRUE)
```

というのがお手軽（コピペで簡単に作れる）なのではないかと…

もちろん，以下のように変数名をまとめやっても OK です。

```
v1 <- c("性別","年齢","b1","b2","b3","b4","b5")
sapply(x[v1], min, na.rm=TRUE)
sapply(x[v1], max, na.rm=TRUE)
sapply(x[v1], mean, na.rm=TRUE)
sapply(x[v1], sd, na.rm=TRUE)
```

あとは，たとえば男女別に算出したいというような処理が必要な場合もあるでしょう。調べてみると，やり方はいくつもあるようです（このように，Rには複数のやり方がある場合が多くて…。SPSS から移る場合，ここは慣れが必要かもしれません。苦笑）。

美しくないやり方かもしれませんが，ここではたとえば男性のみ，女性のみのデータを抽出し，新しくデータ（データフレーム）を作る方法を紹介しておきます。

特定の行だけを抽出する方法もいくつかあるようですが，簡単なものとしては，subset(x, 性別==1)というようなコマンドがあります。x というデータから，性別が１のケースだけをとり出しなさいという命令です。なお R では，等しいという意味の表現するためには，「=」を２つ重ねて，「==」と表記します。これを使って，男女それぞれの新しいデータフレームを作ります。

```
x.m <- subset(x, 性別==1)
x.f <- subset(x, 性別==2)
```

こうやっておいて，再度本日の様々なコマンドをそれぞれのデータに対して実行してやれば，男女別に計算できます。

```
> x.m <- subset(x, 性別==1)
> x.f <- subset(x, 性別==2)
> x.m
     no 性別 年齢 b1 b2 b3 b4 b5
5  1005    1   20  3  5  5  5  5
6  1006    1   19  4  3  5  5  3
10 1010    1   19  4  4  5  3  2
11 1011    1   19  1  3  4  4  4
12 1012    1   19  4  4  5  3  2
19 1019    1   19  1  4  4  4  4
> x.f
     no 性別 年齢 b1 b2 b3 b4 b5
1  1001    2   20  2  5  5  4  2
2  1002    2   20  2  2  4  3  2
3  1003    2   19  3  4  4  4  3
4  1004    2   21  5  3  5  4  3
7  1007    2   20  5  5  5  4  4
8  1008    2   19  3  3  5  3  4
9  1009    2   19  2  4  3  4  3
13 1013    2   19  2  3  3  4  4
14 1014    2   19  2  3  4  4  3
15 1015    2   19  2  5  5  5  4
16 1016    2   19  2  5  4  5  2
17 1017    2   19  4  3  4  3  3
18 1018    2   19  3  4  5  4  3
20 1020    2   20  5  5  5  5  5
```

…ここで終わろうと思ったのですが，せっかく「やり方はいくつもある」と書いてあるので，おまけにもう一つ似たようなコマンドを紹介します（必須というわけでもないので，面倒だったら飛ばしてください）。by( )というコマンドです。具体的には，by(分析されるデータ，分けるための指標，関数)というふうに使います。

では，x の 3 列目から 8 列目を，性別によって分け，さらにそれぞれの平均を求めてみます。

```
by(x[3:8], x$性別, colMeans)
```

すると，右図のような結果が返ってきます。新しくデータを作る必要がなければ，この方法も使えると思います。

```
> by(x[3:8], x$性別, colMeans)
x$性別: 1
      年齢        b1        b2        b3        b4        b5
19.166667  2.833333  3.833333  4.666667  4.000000  3.333333
------------------------------------------------------------
x$性別: 2
      年齢        b1        b2        b3        b4        b5
19.428571  3.000000  3.857143  4.357143  4.000000  3.214286
```

これで７日目は終了です。本日は，いろいろと新しい情報が出てきました。面倒くさそう…という感じもすると思います。というか，面倒なやり方の方を紹介しました。R に慣れるには，簡単なものを先に覚えるより，試行錯誤した方がいいかなと（苦笑)。

明日は，これをもっと簡単にやる方法です。