

そして csv 形式で保存し (mat30.csv という名前にしておきました), R に読み込みます。今回は行列ともに名前が入っているので

```
x <- read.csv("mat30.csv", header=TRUE, row.names = 1,  
             fileEncoding="CP932")
```

こんな感じでしょう。fileEncoding="CP932"はMacの場合は必要ですね。

次は, MASS パッケージに入っている `mvrnorm` 関数を使ってデータを発生させます。しかし, `mu=`の後に平均値0を30個並べるのもまた面倒です。そこで `rep(a, b)` という関数を使って手間を省きます。これは `a` を `b` 回繰り返ささいということです。データ数は, とりあえず200個にしておきましょうか。そしてできたデータをデータフレーム形式に変換しておきます。

```
library(MASS)  
mux <- rep(0, 30)  
data <- mvrnorm(n= 200, mu= mux, Sigma= x, empirical= TRUE)  
d.data <- data.frame(data)
```

データができたなら, 平均値や標準偏差, そして `cor(d.data)` を使って指定した通りの相関になっているかどうかを確認してみましょう。

これでデータができたので, 後は項目数を指定して α を計算してみます。 α 係数の算出は `psych` パッケージですので, これを読み込んでおきます。項目数は, `alpha` の後のカッコ内で指定する列数でコントロールすると簡単でしょう。たとえば3項目の α なら以下のように (もちろん `d.data[10:12]` などでも同じになりますが…)。

```
library(psych)  
alpha(d.data[1:3])
```

結果を見ると, α 係数は .75 です。つまり, 3項目間の平均相関係数が .50 であれば, α 係数は .75 になるということですね。軽く .70 をクリアしています。…ということは, 5項目で .70 に届かないということは, どれくらいの項目間相関なのでしょうかね…

```
> alpha(d.data[1:3])

Reliability analysis
Call: alpha(x = d.data[1:3])

  raw_alpha std.alpha G6(smc) average_r   mean   sd
    0.75      0.75    0.67      0.5 1.2e-17 0.82

Reliability if an item is dropped:
  raw_alpha std.alpha G6(smc) average_r
q1      0.67      0.67    0.5      0.5
q2      0.67      0.67    0.5      0.5
q3      0.67      0.67    0.5      0.5

Item statistics
  n    r r.cor r.drop   mean sd
q1 200 0.82 0.67  0.58 -3.4e-17 1
q2 200 0.82 0.67  0.58  1.7e-17 1
q3 200 0.82 0.67  0.58  3.7e-17 1
>
```

もしかすると、先の「3項目間の平均相関係数が .50 であれば」、というところに引っかかった人がいるかもしれません。しかし計算式上では、今回のデータのようにすべての相関係数が .5 でも、以下のように平均が .5 でも同じ値になります。

	qq1	qq2	qq3
qq1	1	0.3	0.7
qq2	0.3	1	0.5
qq3	0.7	0.5	1

懐疑的な人は、上の表について以下のように確認してみてください。

```
x0 <- matrix(c(1, 0.3, 0.7, 0.3, 1, 0.5, 0.7, 0.5, 1), ncol=3)
data0 <- mvrnorm(n = 200, mu = c(0, 0, 0), Sigma = x0, empirical = TRUE)
d.data0 <- data.frame(data0)
alpha(d.data0[1:3])
```

こちらも α 係数は .75 になります。

では、使う項目数をいろいろと変えてみて、 α 係数がどのように変化するかを確認してみてください。

さて、こういう手作業をやっていると、やはり一気にやりたくなります。2項目から 30項目までの α をグラフにするとどうなるのだろう…なんてことが気になってきます。

関連の時のやり方をちょっと変えれば簡単にできます。ただし、`raw_alpha` という数値は、`alpha` が出してくるオブジェクトの `total` の中に入っているので、一度 `total` を何かに保存し、さらにそこから `raw_alpha` 引き出すという手順になります。

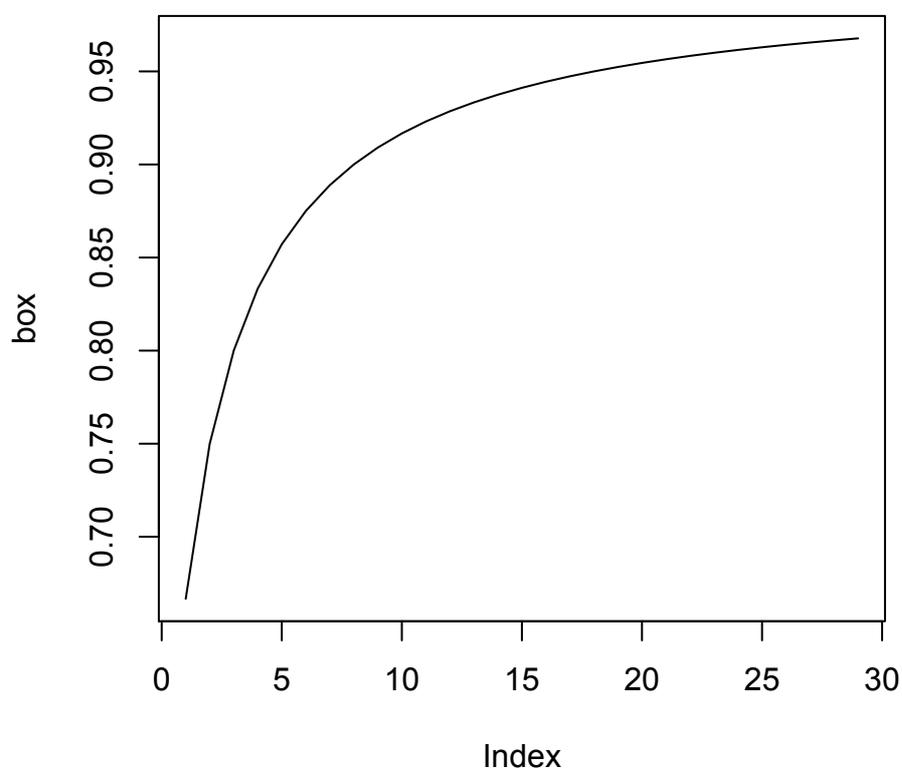
```
box0 <- rep(NA, 29)
box <- matrix(box0, ncol=1)
qq1 <- c(2:30)
for(n in 2:30) {
  al <- alpha(d.data[1:n])$total
  box [n-1,1] <- al$raw_alpha
}
plot(box, type="l")
```

なお、`raw_alpha` を一気に引き出すなら、以下のようにすればできます。

```
box0a <- rep(NA, 29)
boxa <- matrix(box0a, ncol=1)
qq1 <- c(2:30)
for(n in 2:30) {
  boxa [n-1,1] <- alpha(d.data[1:n])$total$raw_alpha
}
plot(boxa, type="l")
```

`box` に入れられた α 係数をグラフにしたら以下のようになります。

項目数が少ない場合は、1項目の増減が α 係数に大きく影響していますが、項目数が 10 あたりになってくると、それほど大きな影響は無いようです。



さて、こうなると .5以外の項目間相関の場合はどうなのかをやってみたくになりますね(?)。一度相関マトリックスを読み込んでおけば、その変更は簡単です。以下のような1行を加えれば、自由に変更できます。0.5を0.35に変えるなら…

```
x[x==0.5] <- 0.35
```

ただし、これをやると x が上書きされてしまうので注意してください。別のファイルを作ってやってもよいでしょう。

```
xx <- x  
xx[x==0.5] <- 0.35
```

実際にやってみてください。

もちろん α 係数は低くなりますが、グラフを描いてみると、上と似たようなカーブになります。

さて、R はグラフを重ね書きすることができます。今回も、項目間相関が .50 と .35 の場合のグラフをひとつにすることができます。

.50 の時の結果 (α 係数) は `box` という名前で存在しています。.35 の時の結果は、`box.x` という名前の入れ物に入れたとしましょう。これを使って、以下のようにすればひとつのグラフにしてくれます。

```
par(family="Osaka")          # Mac のみ
plot(box, ylim=c(0.5,1), xlab="", ylab="", type="l")
par(new = TRUE)
plot(box.x, ylim=c(0.5,1), xlab="項目数", ylab="α 係数", type="l",
col="red")
```

1 行目は Mac 用といってもよいでしょう。Win の場合は不要です。

2 行目が `box` を使ってグラフを描く命令、4 行目 (から 5 行目にかけて) が `box.x` を使ってグラフを描く命令ということはわかると思います。その間にある 3 行目が上書きをするためのものです。つまり `box` を使ったグラフの上に、そのまま `box.x` を使ってグラフを描くこととなります。

ホントにそのまま 2 重に描いてくれるので、軸の目盛などが違っているとグチャグチャになってしまいます。そこで、今回は X 軸は同じなので、Y 軸だけ目盛をそろえておきます。`ylim=c(0.5,1)` という部分がそれなのですが、カッコの中で目盛の最小と最大を指定するだけで OK です。

`xlab=""`, `ylab=""` の部分は、それぞれの軸に対する説明を書き込める部分です。これは何も指定しないとデフォルトの設定で表示されます。先の図で X 軸には **Index**, Y 軸には **box** と表示されていますが、これがデフォルトの結果です。これをコントロールする部分です。"" で囲んだ文字列を表示できるのですが、2 行目は文字列がありません。つまり、何も表示させないという設定です。上書きする時に、「項目数」と「 α 係数」を入れるようにしてあります。

最後の `col="red"`, そのままの意味で、赤で描きなさいということです。デフォルトは黒なので、黒の線が .50 の時の結果、赤の線が .35 の結果です。

これをやってみると、次のような図が出力されます。

やはりカーブの形は似ていますね。しかし、線の差は項目数が少ない時に大きく、項目数が増えると小さくなる傾向がわかります。

