Rのスクリプト作成時の注意点とエラーへの対処

スクリプト作成にあたっての一般的注意

スクリプトの冒頭に作業メモリを空にするコマンドを書く

Rでは，以前に扱ったデータを保存しておいて，次にまた利用できるという機能があります。Rを終了するときに，「作業スペースを保存しますか？」で「はい」と答えると「～.RData」というファイルが作成されます。この「～.RData」というファイルを開くと，前に使っていたときのように，データが作業メモリ上に復活します。

便利なのですが，スクリプトにエラーがある場合に問題を起こすことがあります。スクリプトにエラーがあると，メモリ上のデータは上書きされません。しかし，何らかのデータがメモリ上に残っていればそれが使われるため，エラーがあっても実行結果が表示されてしまいます。たとえば，新しいデータを読み込んで分析結果を表示させるはずだったのに，古いデータの分析結果が残っていてそれが表示されてしまうというようなことがおこります。ですので，Rを終了するときに「作業スペースを保存しますか？」は「いいえ」と答えることをお勧めします。

作業メモリ上のデータを消去するため，Rスクリプトの先頭に，次の一文をいれておくことを推奨します。

```
rm(list=ls())
```

これを実行すると，メモリ上のデータが消去されます。rmは remove，ls()はすべてのリスト（作業メモリ上にあるデータ）を意味します。つまり，上のスクリプトは，消去しなさい(rm)，対象リストは(list=)，すべてのリスト(ls())です，という命令になります。

バックスラッシュと円マーク

Rで作業ディレクトリを設定するとき，バックスラッシュ ＼ もしくは円マーク ￥ が出てきます（実際はいずれも半角)。同じ記号を**２つ入れます**（詳しくは，本スクリプト集の「**作業ディレクトリの設定**」の項を参照)。

コンピュータでは，バックスラッシュと円マークに同じコード番号を使っているため，その番号をどう翻訳するかで表示が変わってきます。

コード番号を記号に変換する翻訳体系として，UTF とか，Shift JIS などがあり，どの体系を使うかで，表示される記号が異なってきます。

コード番号は同じなので，スクリプト上の記号がどちらになっていても，問題はありません。

Rに限らず，コンピュータでソフトを使っているとき，コード翻訳体系を変えるとうまく動かないこともありますので，ソフトが指定してくるコード体系のままにしておいたほうが無難です。

つまり，バックスラッシュで表示されるならそのまま，円マークで表示されるならそのまま，コピーペーストして変わるなら変わったままにしておきます。

とりあえずは慣れて下さい。どうしても気になるときはコード体系を変えることが考えられますが，他に影響がでる可能性もあるので注意して下さい。

コメントを書くときは # を使う

スクリプト中でコメントを書きたいこともあります。そのような場合は，シャープ記号 # （半角）を用います。Rは，#で始まる文を読み飛ばします。例えば，

```
# 統計学レポート課題３
```

と冒頭に書いておくと，そのスクリプトが何のためのものか分かります。また，

```
xa <- abs(x)   # xの絶対値
```

のように，命令文の右にコメントを書くことも可能です。

半角ｶﾅ文字を使わない

Rは世界中で使われているソフトなので，日本語対応が完全でない場合があります。とくに半角ｶﾅ文字は特殊で，半角ｶﾅ文字を使用していることでエラーになることも多くあります。

それゆえ，データ，変数名，ファイル名，フォルダ名等のいずれにおいても，半角ｶﾅ文字は使用しないことを強く推奨します。どうしてもカナ文字を使う必要がある場合は，全角カナ文字を用いるようにします。

全角文字もなるべく使わない

半角ｶﾅ文字と同様，全角文字も使わないに越したことはありません。全角文字を使用していることでエラーになることがあります。Rを使うものについては，データ，変数名，ファイル名，フォルダ名等のいずれにおいても，なるべく全角文字は使わず，半角英数文字を使うことをお勧めします。

スクリプト画面においては，全角のスペースやカッコと，半角のスペースやカッコとの見分けがつきにくく，エラーのもとになります。ファイル名や変数名で全角文字を使用したときは，とくに注意が必要です。

なお，変数名については，ごく簡単な英単語を除き，ローマ字表記するのが分かりやすいです。

スクリプトの各行の先頭に「>」「+」「1:」などを書かない

スクリプトを実行すると，コンソール画面では，スクリプトの左端に「>」「+」「1:」などの記号をつけて表示します。「>」は各命令文の始まり，「+」や「1:」は命令文中の改行を意味します。コンソール画面の出力を参考にしてスクリプトを書くときは，これらの記号を外して（書かないで），スクリプトを書くようにしてください。

データファイルの変数名とRスクリプトの変数名を一致させる

データファイル（CSVファイル）の変数名と，Rスクリプトの変数名が一致していないと，エラーになります。注意しなければならないのは，CSVファイルの変数名におけるスペース（半角，全角）やハイフン（-）は，Rでは「.」（半角ピリオド）に変換されることです。

たとえば，CSVファイルでの「x 2」（半角x 半角スペース 半角2）という変数名は，R上では「x.2」（半角X 半角ピリオド 半角2）となります。Rで「x 2」という変数を使おうとしても，「x 2」という変数はないので（あるのは x.2），エラーとなります。この場合は，CSVファイルの変数名を x2 とするか，Rのスクリプトを x.2 とするかの，いずれかを行う必要があります。

このように混乱のもとになるので，データファイルの変数名にはスペースやハイフンを入れず，なるべく英数半角文字で書くことをお勧めします。ただし，先頭に数字を使うことはできません。

なお，例えば「情緒的サポート」の変数名を，support.emotion, supportEmotion, support_emothion などと書くように決めておくと，「道具的サポート」は，support.tangible, supportTangible, support_tangible のように統一的に表記することができ，何の変数であるかが分かりやすくなります。

Rの予約語をオブジェクト名に用いない

if, for, TRUE, NA など予約語と言われるものは用途が限定されているので，たとえば，if <- 1 のように値を代入しようとしてもエラーになります。Rの予約語は以下の通りです。

```
for     in      if      else    repeat  while   next    break
TRUE    FALSE   NULL    Inf     NaN     NA      function
NA_integer_     NA_real_        NA_complex_     NA_character_
```

pi, T, Fなどをオブジェクト名に用いない

また，pi，T，F はそれぞれ，円周率，TRUE，FALSE を意味する記号として初期設定されているので，これらの記号をオブジェクト名に使うのは避けるべきです。たとえば，pi はそのままでは円周率の値ですが，以下のように1という値を代入してしまうと，以降，円周率として使えなくなってしまいます。

```
> pi
[1] 3.141593
>
> pi <- 1
>
> pi
[1] 1
```

代入結果を表示するときは当該のスクリプト部分を () で囲う

Rでは，命令文を実行した結果（データ，値，分析結果など）を「<-」を用いて代入保存することができます。例えば，

```
x <- mean(c(1,2,3,4))
```

とすると，x には 1,2,3,4 という4つのデータの平均値 2.5 が代入されます。しかし，これを実行しても，コンソール画面上では，

```
>  x <- mean(c(1,2,3,4))
```

と表示されるのみで，xの値は表示されません。代入することしか命令されていないので，それを行ったらそれで終わりなのです。

結果を代入して，さらに，その値をコンソール画面上に表示させたいときは，

```
(x <- mean(c(1,2,3,4)))
```

のように，命令文全体を () で囲います。こうすると，

```
>  (x <- mean(c(1,2,3,4)))
[1] 2.5
```

のように結果が表示されます。

改行の位置に気をつける

スクリプトにおける改行のルールは基本的に以下の通りです。

・1つの行に複数の命令文を書くことは，基本的にはできない

例えば，head(d1)（d1の最初の数行を表示）と，nrow(d1)（d1の行数を表示）という2つの命令文を，

```
head(d1)  nrow(d1)
```

と書いてはいけないということです。

・1つの行に複数の命令文を書くときは，セミコロンなどで文を区切りを明示する

上記の場合，

```
head(d1); nrow(d1)
```

とのように，間を ; で明示すれば，ここが切れ目だとRが理解することができるので，1つの行に複数の命令文を書くことができます。

・ひとかたまりの単語（や項）の途中で改行しない
　例えば，10^rdgts（10のrdgts乗）というひとかたまりの項は，どこに改行をしてもいけないということです。これは，英単語を途中で改行してはいけないのと同じです。

・1つの命令文を複数の行に分けるときは，要素の切れ目で行変えする
　例えば，

```
d1$gr <- cut(d1$x1, breaks=c(-Inf, med1, Inf), right=F, labels=c("L","H"), ordered_result=TRUE)
```

というスクリプトであれば，

```
d1$gr <- cut(d1$x1, breaks=c(-Inf, med1, Inf), right=F,
             labels=c("L","H"), ordered_result=TRUE)
```

などとします。なお，改行したスクリプトを実行すると，コンソール画面では，

```
> d1$gr <- cut(d1$x1, breaks=c(-Inf, med1, Inf), right=F,
+              labels=c("L","H"), ordered_result=TRUE)
```

のように表示され，スクリプトの1行目に「>」，2行目以降に「+」がついて表示されます（「1:」のような番号がつくこともあります）。この「>」や「+」（および「1:」など）をスクリプトに書くとエラーになりますので，気をつけてください。

・計算式では，演算記号の後で改行する
　例えば，1+2+3+4+5を2行に分けるとき

```
1+2+3
+4+5
```

としてしまうと，1+2+3 と，4+5 という2つの計算式を入力したことになってしまい，実行すると 6 と 9 という2つの結果が出力されます。このような場合は，

```
1+2+3+
4+5
```

と1行目は「+」の後で改行し，まだ式は完結していないことを示した上で2行目を書きます。そうすると1つの計算式と認識され，15 という結果が得られます。

・for文，if文等における改行
　条件式は丸カッコ ()，命令文は波カッコ {} で括ります。命令文はセミコロン ; で区切って同じ行に書いても構いません。
　if-else文では，if文の最後の } と else { は同じ行に書きます。

```
for(条件式){
  命令文
  命令文
  …
  }

if (条件式){
  命令文
  命令文
  …
  } else {
  命令文
  命令文
  …
  }
```

欠測のあるデータの分析について

実際にデータを収集したときは，多くの場合で欠測データが生じます。欠測のあるデータをRで分析する場合，多くの関数は自動的に欠測データを除外して分析しますが，いくつかの関数では NA (Not Available) という結果を返したり，欠測の処理の仕方によって異なる結果を出力するので注意が必要です。詳しくは，本スクリプト集の「**欠測値の扱い**」の項を参照してください。

四捨五入について

Rで四捨五入を行う関数として round 関数があります。

round(数値, 小数点以下桁数)

と設定することにより，指定した桁数までに数値を四捨五入します。桁数を省略したときは整数になります。しかし，round関数について注意しなければならないのは，

```
> round(1.5)
[1] 2

> round(2.5)
[1] 2
```

のようにとなることです。通常の四捨五入なら 2, 3 という結果になるはずですが，Rのround関数では，～5 という数値があったとき，～が偶数なら切り捨て，～が奇数なら切り上げという計算をします。

通常の四捨五入を行うためには次のようにします。

```
sign(x)*(floor(abs(x)*10^p + 0.5)/10^p)
```

signはxの符号(+,-)を与える関数，floorは小数点以下を切り捨てる関数，absは絶対値を求める関数，pは小数点以下の桁数です。詳しくは，本スクリプト集の「**切り上げ，切り下げ，四捨五入**」の項を参照してください。

作成したスクリプトを少しずつ実行して確認する

スクリプトをすべて書いてから実行するよりも，少し書いてはその部分を実行することを繰り返したほうが，エラーを早く見つけることができます。確認にあたって，いつも上から全部実行する必要はなく，必要な箇所だけを選択して実行します。

エラー修正（プログラムのデバッグ）にあたっては，上から順に必要箇所を選択し，実行結果を確かめて行きます。1命令文ごとに実行し，結果を確認することをお勧めします。

for文は複数行にわたりますので，例えば i <- 1 のように初期値を設定しておいて，for文の{}中を1命令文ずつ実行していきます。

if文やifelse文では，条件式だけを実行して TRUE か FALSE の値が得られるか確認します。データに欠測値があると NA という結果になり，エラーの原因となります。

図を複数描く場合も，1つずつ実行しないと最後の図しか画面上に残らなくなります。たとえば，データを読み込んで，クロス表，円グラフ，帯グラフをそれぞれ作成したい場合は，まず，データやパッケージを取り込みます。これは共通部分で，全体を通して1回だけ行えば十分です。

```
rm(list=ls())
setwd("d:¥¥Rreport¥¥")
d1 <- read.table("度数分布_データ.csv", header=TRUE, sep=",")
library(descr)
```

そのあとクロス表を作成します。

```
ctd1 <- CrossTable(d1$sex, d1$grade)
```

円グラフは次のスクリプトを実行することにより作成できます。

```
t.grade <- table(d1$grade)
pie(t.grade, clockwise=TRUE)
```

帯グラフは以下を実行することにより作成されます。

```
t2 <- table(d1$grade, d1$sex)
p2 <- prop.table(t2, 2)
barplot(as.matrix(p2), horiz=TRUE, beside=FALSE, las=1, ylim=c(0,1), width=0.3, legend.text=T)
segments(0, 0, 0, 1)
```

クロス表，円グラフ，帯グラフの手順はそれぞれ独立していますので，それぞれ当該のスクリプトだけを範囲選択して実行します。

よくある質問・エラー

データが読み込めない（rm, setwd, read.table）

Rを使い始めの頃に良くあるのが，データの読み込みエラーです。rm, setwd, read.table等の関数でエラーが発生している可能性があります。それぞれのエラーの原因と対処法について説明します。

USBに「Report」というフォルダを作り，その中に「data1.csv」というcsvデータファイルがあるとします。USBドライブが「D」ドライブなら，データ読み込みのスクリプトは次のようになります。

```
rm(list=ls())
setwd("D:¥¥Report¥¥")
d1 <- read.table("data1.csv", header=TRUE, sep=",")
```

ここで，もし rm の行に間違いがあったら，
「... は名前あるいは文字列を含んでいなくてはなりません。」
のようなエラーが出ます。rm(list=ls()) と正しく書かれているか確認してください。

もし setwd の行に間違いがあったら，
「setwd("d:¥¥Rreport¥¥") でエラー： 作業ディレクトリを変更できません。」
のようなエラーがでます。いまの場合，フォルダ名を「Rreport」と書き間違えていますので，「Report」と修正します。

もし read.table の行のファイル名に間違いがあったら，
**「 file(file, "rt") でエラー： コネクションを開くことができません
追加情報： 警告メッセージ:file(file, "rt") で:ファイル 'data.csv' を開くことができません: No such file or directory」**
のようなエラーがでます。いまはファイル名を「data.csv」と間違えていますので，「data1.csv」とファイル名を正しく書き直します。

read.table の header や sep など，オプションの指定に間違いがあると，そのオプションは無視されるかエラーになります。エラーメッセージを表示してくれれば分かりますが，無視されたり，エラーメッセージを表示することなく実行できない場合は，オプションに誤りがないか，自分で検討をつけ対応する必要があります。古い結果しか出力されない，オプションを書き換えても結果が変わらない，エラーは出ないのに実行結果に問題があるなどの場合は，オプションが正しく書かれているか確認してください。

フォルダ名，ファイル名は，Windowsであればエクスプローラで確認することができます。気をつけなければならないのは，エクスプローラでは最初，拡張子というものを表示していないことです。

.csvなどは拡張子と言って，そのファイルがどのような形式か，どのソフトに関連しているかを表す記号です。エクスプローラの上のほうにある「表示」をクリックして，「ファイル名拡張子」にクリックを入れて下さい。そうすると，拡張子がついたファイル名が表示されます。

拡張子がついたファイル名を表示したとき，ファイル名が「data1.csv.csv」などとなっていたら，ファイル名をクリックして「data1.csv.csv」のように書き換えてください。

添え字が許される範囲外です と表示される

たとえば，以下のようなエラーが出たとします。

```
> describeBy(d1$tekiou, list(d1$future), mat=TRUE, digits=2)
 `[<-`(`*tmp*`, var, group + 1, value = dim.names[[group]][[groupi]]) でエラー:
  添え字が許される範囲外です
```

このような場合は，指定したデータや変数に何らかの問題があります。たいていの場合，そういう名前の変数が無いことにより，このエラーが生じています。データ名や変数名が正しいか，よく確認してください。

他の原因として，データがすべてNA，群分け変数（d1$future）にNAが含まれるなどの場合もあります。変数（d1$tekiou や d1$future）の中身を確認してください。

スクリプトにエラーはなくても，同じスクリプトを複数回実行するとエラーの原因をつくることもあります。スクリプトを先頭から実行しなおして下さい。

作成した覚えのない変数が存在する

データファイルで，想定範囲外の列のセルに何らかのデータをRで読み込んだときそのような列に対してRは自動で X, X.1, X.2, X.3 のような変数名をつけます。また，データファイルに同じ変数名が複数あるときも，同様の番号づけを行います。

X, X.1, X.2, X.3 のような作成した覚えのない変数がR上でできているときは，重複している変数名があればそれらを修正してください。また，想定される範囲外のセルにデータがあると考えられるときは，範囲外の行や列について，ある程度の領域を指定して削除を実行し，不要なデータを証拠してください。

incomplete final line found by readTableHeader と表示される

Rでread.table関数を実行した際，incomplete final line found by readTableHeader というエラーが出ることがあります。主にApple社製のパソコン（Mac）を使っている場合に出るメッセージです。

変数名やファイル名に全角文字があると，このメッセージが出ることがあります。変数名やファイル名を半角英数文字に変えてみてください。

それでもこのメッセージが出る場合は，「テキストエディット」などのエディタをまず起動し，そこから目的のCSVファイルを開いて，最終行の末尾に改行記号を入れて保存してください。詳しくは，本スクリプト集の「**Excelを使わずにCSVファイルを作成する方法**」の項を参照してください。

結果の数値に e-01 のような表記が付く

分析結果の出力が，例えば次のようになっているとします（重回帰分析の結果の一部です)。

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.497e-17  4.571e-02   0.000   1.0000
stress       4.509e-01  5.361e-02   8.411 3.61e-15 ***
kyoufu       1.210e-01  5.048e-02   2.397   0.0173 *
support     -3.508e-01  4.907e-02  -7.150 1.03e-11 ***
```

ここで，Estimate（推定値）の 4.497e-17, 4.509e-01 などの値はそれぞれ，
4.497e-17 = 4.497 × 10の-17乗 = 0.00000000000000004497 ≒ 0
4.509e-01 = 4.509 × 10の-1乗 = 0.4509 ≒0.45
を意味します。

e は指数（exponential）のことで，そのうしろにくる数値が指数であることを表しています。もし，eのうしろが +03 となっていたら，10の3乗（×1000）ということになります。

4.497e-17 は実質的には0ですが，数学的には0ではありません。コンピュータは数値計算で解を求めるので，数学的に厳密に正しい値ではなく，このような値を出力することがあります。

パッケージが開けない

パッケージはインストールできたが，library関数でそのパッケージを利用可能にしようとしても，開けないというトラブルがたまに発生します。いくつか原因が考えられますが，その１つとして，OSのアップデート（システムの更新）が未完了であることがあります。

Windowsであれば，スタートメニュー → 設定 → 更新とセキュリティ → 更新プログラムのチェック を行って，システムを最新の状態にしてください。おそらくパソコンの再起動が必要になります。OS（オペレーティングシステム）を最新の状態にして，Rを使うようにしてください。

パッケージの関数が使えない

パッケージに含まれる関数を使おうとすると反応しない（そのような関数はみつからないと言われる）ことがあります。

まずは，当該のパッケージをきちんとインストールしているか，library関数でそのパッケージを使える状態にしているか，スクリプトに誤記はないかなどを確認してください。

パッケージの更新を行って下さい。Rでパッケージ → パッケージの更新 → パッケージを選択してOK を実行してください。

これで解決することがあります。

これらを行っていてもパッケージの関数が使えないことがあります。そのようなときの原因の1つとして，ユーザー名の問題が考えられます。全角文字のユーザー名だと，このようなエラーが生じることがあります。この場合は，半角英数文字（先頭はアルファベット）のユーザーを新たに作成してください。

コントロールパネル → ユーザーアカウント → 別のアカウントの管理 → PC設定で新しいユーザーを追加 → その他のユーザーをこのPCに追加，と進んで，半角英数ユーザー名で新しいアカウントを作成してください。管理者権限をもたせることをお勧めします。

アカウントができたら，新しいユーザー名でサインインしてください。それで，パッケージも含めてRが正常に動くなら，今後Rを使うときはそのアカウントを使ってください。

新しいアカウントでもRがうまく動かないときは，いったんRをアンインストールしてください。

コントロールパネル → プログラム → プログラムのアンインストール ，と進んで，Rを選んでクリックします。アンインストールするか聞いて来たら，指示に従ってアンインストールしてください。

そして，新しいアカウントを使って，Rおよびパッケージをインストールしてください。パッケージの更新，もやってください。それでパッケージも含めてRが正常に動くなら，今後Rを使うときはそのアカウントを使ってください。

エラーが発生したときの対処

スクリプトを実行してエラーが発生するときは，ほとんどの場合，スクリプトに何らかの誤りがあります。また，Rが全然動かない（エラーすら出ない）ということも時々起こります。このような状態について，よくある誤りと対処法を説明します。なお，ある程度修正を試みてもうまくいかないときは，**修正を繰り返すよりも，スクリプトをすべて新しく書き直したほうが早い**です。

半角ｶﾅ文字を使っていませんか？

半角ｶﾅ文字は特殊な文字なので，Rのように世界中で使われるソフトには不向きです。データ，変数名，ファイル名，フォルダ名，ユーザ名等のすべていおいて，使わないで下さい。

お使いのPCのOSは何ですか？

本スクリプト集は主としてWindowsマシンを前提に書かれています。MacのパソコンでＲを使用する場合には，作業ディレクトリを設定する方法や，read.table関数でCSVファイルを読み込む方法などが異なります。具体的な方法は，本スクリプト集の「**Mac OSでＲを使用するときの注意点**」の項を参照してください。

エクセルでデータを保存したCSVデータと，Rでの読み込み形式は合っていますか？

コンピュータが数値，文字，記号を認識するためには，文字コードというものが必要になります。文字コードには，Shift JIS や UTF-8など，いくつか種類があります。

エクセルで「CSV（コンマ区切り)(*.csv)」で保存した場合は，文字コードは Shift JIS になります。このデータをRで読み込むには，
```
d1 <- read.table("dta1.csv",  header=TRUE, sep=",")
```
とします。

エクセルで「CSV **UTF-8**（コンマ区切り)(*.csv)」で保存した場合は，文字コードは BOM付きUTF-8 になります。BOMは「バイトオーダーマーク」というもので，データの先頭に，文字コードがUTF-8であることを意味する記号が付加されます。このデータをRで読み込むには，
```
d1 <- read.table("dta1.csv",  header=TRUE, sep=",", fileEncoding="UTF-8-BOM")
```
とします。

なお，BOMが付かないCSVデータの場合は，
```
d1 <- read.table("dta1.csv",  header=TRUE, sep=",", encoding="UTF-8")
```
とします。本スクリプト集の「**データの読み込み**」の項も参照してください。

作業メモリ上に余計なデータが保存されていませんか？

Rでは，計算や分析の結果を作業メモリ上に「オブジェクト」として置いておき，再利用することができます。スクリプトにエラーがあるのに，同名の古いオブジェクトが残っていると，過去の結果が使われて，ミスに気づかないことがあります。また，新しく分析結果を保存したいのに，古い分析結果の後ろに新しい分析結果が追加されてしまうこもあります。

これらの事態を避けるために，分析を始めるときは，いったんメモリ上のオブジェクトを消去しておきます。メモリを消去するには，以下のスクリプトを実行します。
```
rm(list=ls())
```
rmはメモリ消去をする関数，ls()はメモリ上にあるすべてのオブジェクト名を返す関数です。

実行するスクリプトの範囲選択は正しいですか？

スクリプト全体を実行する場合は，スクリプト画面上でマウスを右クリックし，「全て選択」を選択→「カーソル行または選択中のRコードを実行」を選択，で実行できます。

選択した範囲のスクリプトだけを実行する場合は，マウスの左ボタンを押しながら範囲を選択して左ボタンを放し，スクリプト画面上でマウスを右クリックして「カーソル行または選択中のRコードを実行」を選択，

で実行できます。

ここで，実行範囲を適切に選択しないと，エラーが生じます。行の途中から範囲指定する，最後のカッコを入れてないなどです。また，最後の改行を含んでいないため，最後の行が実行されないということもよくあります。範囲指定するときは，最後の行の次の行（空白行）まで範囲指定するようにします。
例)
```
setwd("D:¥¥Report¥¥") を選択すべきところを setwd("D:¥¥Report¥¥"
# 末尾の ) と 改行 が含まれていない
```

データのあるフォルダを正しく設定していますか？

データを読み込めないときによくあるエラーとして,setwd()の指定がうまくできていないことがあります。データの入っているドライブ名（c,d,eなど）と，フォルダ名（Reportなど）を正しく指定して下さい。ドライブ名の後ろは：(コロン)，フォルダ名の後ろは ¥¥（円マークを２つ）を入れるのを忘れないでください。カッコ内全体をダブルクオーテーションで囲うのも忘れないでください。なお，フォルダ名は，Windowsであればエクスプローラで確認することができます。
例)
データファイルが，USBメモリのReportというフォルダに入っていて，USBがDドライブの場合
```
setwd("D:¥¥Report¥¥")     #「D」は「d」でも構いません。
```

データファイル名は正しいですか？

データを読み込めないときよくある別のエラーとして，read.table()でファイル名が正しく書かれていないことがあります。ファイル名がdata1.csvのときに，データを読み込むスクリプトは次のようになります。
```
d1 <- read.table("data1.csv", header=TRUE, sep=",")
```

なお，ファイル名については，「**良くある質問・エラー**」の『**データが読み込めない**』の項も参照してください。

変数名を間違えていませんか？

データファイル（CSVファイル）で指定した変数名と，Rのスクリプトで書いている変数名が一致しないとエラーになります。変数名を確認し，正しい変数名を書いてください。

Rにおいて，read.table()など**関数**と呼ばれるものは綴りが決まっていますが，変数名やデータ名は自分で綴りを指定します。これらはなるべく英数半角文字で書くことをお勧めします。ただし，先頭に数字を使うことはできません。

データファイルの変数名にスペースが入っていませんか？

データファイルの変数名に半角スペース，全角スペースがあるとき，Rでは「.」(半角ピリオド）に変換されます。あまり好ましくないので，変数名にスペースを入れるべきではありません。
また，データファイルで変数名が無い場合（予想外のセルにデータが入っている場合など)，Rは自動的にX, X.1, X.2, X.3 のような変数名をつけます。
例)
X 1 が X.1

全角記号が混ざっていませんか？

Rは全角文字と半角文字を区別します。しかし，スクリプト画面では，とくに英数字やスペースについて，全角文字と半角文字の見分けがつきにくくなっています。それゆえ，半角で書かなければならないところが全角になっているのに気がつかず，エラーとなることがしばしばあります。
例)
```
d1 <- を d1  <-        # d1の後ろが全角スペースになっている
<- を＜－              # 代入記号（半角<-) が全角になっている
x1 を ｘ１             # 半角x1 を全角ｘ１にしている。
d1[d1$class=="A",] を d1[d1$class=="A",」       # 最後の ] が」になっている
```

スペルミスをしていませんか？

スペルミスがあれば，エラーになるか，正しく動きません。
例)
```
x1 を xl              # 1 (いち) と l (エル) を間違えている
colnames を colname   # sをつけ忘れている
```

スクリプトを書くときに「>」「+」「1:」などを左端に書いていませんか？

スクリプトを実行すると，コンソール画面には，実行されたスクリプトが表示されます。その際，左端に「>」「+」「1:」などの記号が付加されます。コンソール画面の出力を参考にしてスクリプトを書く場合，左端のこれらの記号は，スクリプトには書かないようにする必要があります。

大文字と小文字を間違えていませんか？

変数名や関数名で大文字と小文字を間違えても，エラーになります。
(ドライブ名，フォルダ名，ファイル名は，大文字小文字の区別はありません。)
例)
```
rowSums を rowsums
TRUE を True
```

記号を間違えていませんか？

$マーク，論理式，数式，などの記号を間違えると，エラーや計算間違いとなります。
例)
```
d1[d1$class=="A",] を d1[d1%class=="A",]    # $ を % としている
d1[d1$sex=="f",] を d1[d1$sex="f",]         # == を = としている
<- を <　または　<=                  # 代入記号「<-」を，大小関係を表す「<」や「<=」としている
```

カンマを忘れていませんか？

配列の行と列の区切り，変数や関数のオプションを複数指定するときなどは，要素間をカンマで区切りらなければなりません。
例)
```
d1[d1$class=="A",] を d1[d1$class=="A"]
table(d1$x1, d1$x2) を table(d1$x1 d1$x2)
c(x1, x2, x3) を c(x1 x2 x3)
```

カンマを多く入れていませんか？

余計なカンマが入っていても，エラーになります。
例)
```
table(d1$class, d1$grade) を table(d1$class,, d1$grade)
```

カンマとピリオドを間違えていませんか？

カンマを打つべきところをピリオドにすると，エラーになります。逆も同様です。
例)
```
ylim=c(1.5, 2.5) を ylim=c(1.5. 2.5)     # 1.5 の後ろのカンマをピリオドにしている
ylim=c(1.5, 2.5) を ylim=c(1,5, 2.5)     # 1.5 と書くべきところを 1 カンマ 5 としている
```

文字列をダブルクォーテーションで囲うのを忘れていませんか？

変数名，ファイル名，記号などの文字列は ″ ″ で囲います。片方または両方を付け忘れると，エラーになります。
例)
```
("D:¥¥Report¥¥") を ("D:¥¥Rreport¥¥)
″data1.csv″ を data1.csv
==″A″ を ==A                          # 「″A″」はAという文字，「A」はAという変数の意味になります。
```

カッコをつけ忘れていませんか？

開くカッコに対して，必ず閉じるカッコがなければなりません。
例)
```
round(data.frame(n.d1, mean.d1),2) を round(data.frame(n.d1, mean.d1, 2)
# 開くカッコが2つあるのに，閉じるカッコが1つしかない
```

カッコの種類を間違えていませんか？

Rはカッコの種類を区別します。開くカッコと閉じるカッコは，同じ種類でなければなりません。
例)
```
d1[,items] を d1[,items)
```

カッコで括らなければならないのを忘れていませんか？

if文の条件式はカッコで括らなければなりません。また，計算結果を代入値として使いたいときに，計算部分をカッコで括らなければならないことがあります。

開くカッコと閉じるカッコが同じ行にないとエラーになる場合があります。その場合は，長い文になっても，途中で改行せず１行のままにしておく必要があります。
例)
```
if (x==5) を if x==5
xlim=c((xmin - .5), (xmax + .5)) を xlim=c(xmin - .5, xmax + .5)
for(i in 1:(n+3)) を for(i in 1:n+3)
```

カッコをつける位置を間違えていませんか？

開くカッコと閉じるカッコの数があっていても，つける位置が間違っていると，エラーになったり，意図とは異なる結果になったりします。
例)
```
floor(round(24.5)+0.5) を floor(round(24.5+)0.5)       # エラーになる
floor(round(24.5)+0.5) を floor(round(24.5+0.5))       # 異なる結果になる
```

オプションは正しく書かれていますか？

header=TRUEなどオプションと言われるものに誤りがある場合，Rは関数によって，エラーを出すか，間違いのあるオプションを無視します。

エラーが表示される場合はそこで止まりますので，間違いを修正します。
無視される場合は，何も言わずに以前の分析結果を表示することがありますので，オプションを変えても結果が変わらない場合は，オプション指定に誤りがあることに自分で気がつく必要があります。

Rのヘルプを使って，関数の書式やオプションの指定法などを調べるのも有効な方法です。ヘルプの使い方については，本スクリプト集の「**ヘルプの使い方**」の項を参照してください。なお，パッケージに含まれて

いる関数は，そのパッケージをlibrary()で呼び出してからでないと，ヘルプをみることができません。

if - else文を書くとき，if 文の終わりと else 文の始まりの間で改行してませんか？

if - else文 では，else文はif文の終わりと同じ行に書かなくてはなりません。

例)

```
if (x==5){y <- 1} else y <- 0 を
if (x==5){y <- 1}
   else y <- 0
```

１行に複数の命令文を書いていませんか？

プログラムを書くとき，１つの行に複数の命令文を続けて書くことはできません。コンピュータには，どこが文の切れ目か分からないからです。どうしても１行に複数の命令文を書きたいときは，セミコロン「;」で区切りるようにします。

例)

```
head(d1)
nrow(d1)
を
head(d1) nrow(d1)            # head(d1);  nrow(d1)  ならOK
```

１つの数式を複数行に分けて書くとき，次の行の先頭が+ - * / などの記号になっていませんか？

数式をの途中で改行するとき，改行した次の行の先頭が + - * / などになっているとエラーになることがあります。

パッケージは正しくインストールされていますか？

library()でパッケージを呼び出す際，パッケージがインストールされていないと呼び出すことはできません。パッケージを利用する際は，あらかじめインストールしておく必要があります。本スクリプト集の「**パッケージのインストール**」の項などを参照して，パッケージをインストールしてください。

パッケージのインストールは，Rをインストールした後に1回行えば，あとは何度でも利用できます。ただし，異なるバージョンのRをインストールした場合は，パッケージも改めてインストールする必要があります。

なお，パッケージをインストールしたはずなのに，library()でパッケージを開けないことがあります。これは，パッケージをインストールするとき，Rが自動的に指定するフォルダの不具合で，通常とは異なるディレクトリにパッケージがインストールされるためです。この場合は，Rのインストールからやり直すと解決することが多いです。

R Studioを使っていませんか？

R Studioを使っている場合，まれに原因不明のエラーが生じます。他にエラーの原因が思い当たらない場合は，R Studioは使わずに，R本体を起動してRを実行してください。

Rが複数個，起動していませんか？

Rが複数個起動していて，いま操作していない別のRが何かの待ち状態になっていると，いま操作しているRも動かないことがあります。

Rが複数個起動していることは，画面のタスクバーに複数個のRのタブがあることなどで確認できます。これが原因でRが動かないときは，いま操作していない別のRの待ち状態を解消するか終了して下さい。

コンピュータが再起動待ちの状態になっていませんか？

コンピュータを使っている間に，システムの自動アップデートなどが行われて，コンピュータの再起動待ち状態になっていると，Rが動かないことがあります。

この場合は，Rを含めすべてのプログラムを終了して，コンピュータを再起動してシステムのアップデートを行ってから，Rを再度実行してください。システムの再起動後，システムの更新版のチェックを行って，最新の状態にしてからRを実行することをお勧めします。

全角文字を使っていることが原因のこともあります

全角文字に対応していない関数を使う場合などは，半角ｶﾅだけでなく，全角文字もエラーの原因となります。スペルミスなど，他に思い当たる原因がない場合は，これが原因になっている可能性があります。データ，変数名等を半角英数文字にしてください。

パソコンのユーザー名が全角文字になっていることが原因のこともあります

パソコンのユーザー名が全角文字だと，エラーが出る，または，動かない関数があります。この場合は，管理者権限のある半角英数文字のユーザーアカウントを作成し，そのアカウントでパソコンにサインインして，Rを使ってください。

Rやコンピュータを再起動することも有効です

再起動待ちでもなく，原因は分からないけどとにかくRが動かなくなったという場合も，Rを終了して再起動，コンピュータを再起動などすることにより，動くことがあります。

Rを再インストールすることも有効です

何をやってもエラーが出るという場合，Rを再インストールしたら解決したという事例も，パッケージを利用する場合などに，まれにあります。