R Coding

Necessity Really is a Mother

Paul E. Johnson[1][2]

[1]University of Kansas, Department of Political Science [2]Center for Research Methods and Data Analysis

2016

Outline

What is this Section?

- Tips from the school of hard knocks
- Criticisms of R code I've found or written

Functions replace "cut and paste" editing

If you find yourself using "copy and paste" to repeat stanzas with slight variations, **you are almost certainly doing the wrong thing**.

- Re-conceptualize, write a function that does the right thing
- Use the function over and over

Why is this important: AVOIDING mistakes due to editing mistakes

We Learn By Criticizing

I keep a folder of R code that troubles me.
This example is a more-or-less literal translation from SAS into R that does not use R's special features.

```
#---------------SPECIFICATIONS-------------------------#
iter=1000                              #how many iterations per
    condition
set.seed(7913025)                          # set random seed
#--------------END SPECIFICATIONS---------------------#
#n per cluster sample size
for (perclust in c(100)) {
#number of clusters - later for MLM application
  for (nclust in c(1) ){
#common correlation
    for (setcorr in c(1:8) ){
      if(setcorr==1){
        corr.10 <-1
        corr.20 <-0
        corr.30 <-0
        corr.40 <-0
```

We Learn By Criticizing ...

```
    corr.50 <-0
    corr.60 <-0
    corr.70 <-0
    corr.80 <-0
    }
  if(setcorr==2){
    corr.10 <-1
    corr.20 <-1
    corr.30 <-0
    corr.40 <-0
    corr.50 <-0
    corr.60 <-0
    corr.70 <-0
    corr.80 <-0
    }
  if(setcorr==3){
    corr.10 <-1
    corr.20 <-1
    corr.30 <-1
    corr.40 <-0
    corr.50 <-0
    corr.60 <-0
```

We Learn By Criticizing ...

```
    corr.70 <-0
    corr.80 <-0
    }
  if(setcorr==4){
    corr.10 <-1
    corr.20 <-1
    corr.30 <-1
    corr.40 <-1
    corr.50 <-0
    corr.60 <-0
    corr.70 <-0
    corr.80 <-0
    }
  if(setcorr==5){
    corr.10 <-1
    corr.20 <-1
    corr.30 <-1
    corr.40 <-1
    corr.50 <-1
    corr.60 <-0
    corr.70 <-0
    corr.80 <-0
```

We Learn By Criticizing ...

```
  }
if(setcorr==6){
  corr.10 <-1
  corr.20 <-1
  corr.30 <-1
  corr.40 <-1
  corr.50 <-1
  corr.60 <-1
  corr.70 <-0
  corr.80 <-0
  }
if(setcorr==7){
  corr.10 <-1
  corr.20 <-1
  corr.30 <-1
  corr.40 <-1
  corr.50 <-1
  corr.60 <-1
  corr.70 <-1
  corr.80 <-0
  }
if(setcorr==8){
```

We Learn By Criticizing ...

```
        corr.10 <-1
        corr.20 <-1
        corr.30 <-1
        corr.40 <-1
        corr.50 <-1
        corr.60 <-1
        corr.70 <-1
        corr.80 <-1
    }
```

That project defined several variables in that way, consuming 100s of lines.

Quick Exercise

How would you convert that into a vector in R, without writing 300 lines.
Hint. You want to end up with a vector `setcorr` with 0 elements, all either 0 or 1.
Requirement: You need some easy, flexible way to adjust the number of 0's and 1's

Another "noisy code" example

Note in the following

1. the author does not declare functions
   Rather, treats comments ahead of blocks of code as if they were function declarations.
2. repeated use of cat() with same file argument is an example of cut-and-paste coding.

```
##########################################
#        WRITE GENERATION CODE           #
##########################################

pathGEN <- paste(dirroot,perclust,nclust,setcorr,sep="\\")
gen <- paste(pathGEN,"generatecorrdata.inp",sep="\\")
testdata1 <- paste(pathGEN,"data1.dat",sep="\\")

cat('MONTECARLO: \n', file=gen)
cat('NAMES ARE x y q1-q8; \n', file=gen, append=T)
cat('NOBSERVATIONS = 1000 ; \n', file=gen, append=T)
```

Another "noisy code" example ...

```
cat(' NREPS = ',iter,'; \n', file=gen, append=T)
cat(' SEED = ',round(runif(1)*10000000), '; \n', file=gen,
    append=T)
cat(' REPSAVE=ALL; \n', file=gen, append=T)
cat(' SAVE=\n', pathGEN,'\\data*.dat; \n', file=gen, append=
    T, sep="")
cat('MODEL POPULATION: \n', file=gen, append=T)
cat('[q1-q8*0 x*0 y*0]; \n', file=gen, append=T)
cat('q1-q8*1; x*1; y*1;  \n', file=gen, append=T)
cat('q1-q8 with q1-q8*.50;  \n', file=gen, append=T)
cat('x with y*.50; \n', file=gen, append=T)
cat('x with q1-q8*.50; \n', file=gen, append=T)
cat('y with q1-q8*', .0+(corr.10*(.10+corr.20*.10+corr.30*
    .10+corr.40*+corr.40*.10+corr.50*.10+corr.60*.10+corr.70
    *.10+corr.80*.10)), '; \n', file=gen, append=T, sep="")

if(file.exists(testdata1)){
} else {
shell (paste("mplus.exe",gen, paste(pathGEN,"save1.out", sep
    ="\\"), sep=" "))
        }
```

Another "noisy code" example ...

```
#
    ------------------------------------------------------------------
    #

##########################################
#           GENERATE SAS CODE            #
##########################################
pathSAS <- paste(dirroot,perclust,nclust,setcorr,setpattern,
    percentmiss,aux,auxnumber,sep="\\")
pathSASdata <- paste(dirroot,perclust,nclust,setcorr,
    setpattern,percentmiss,sep="\\")
smcarmiss <- paste(pathSAS,"modifydata.sas",sep="\\")
testdata2 <- paste(pathSASdata,"data1.dat",sep="\\")

#-----------------------------------------------------#
#------------ import SIM data into SAS ---------------#
#-----------------------------------------------------#

if(file.exists(testdata2)){
} else {

cat('proc printto \n', file=smcarmiss, append=T)
```

Another "noisy code" example ...

```
cat('log="R:\\users\\username\\data\\simLOG2\\LOGLOG.log" \n
    ', file=smcarmiss, append=T)
cat('print="R:\\users\\username\\data\\simLOG2\\LSTLST.lst"
    \n', file=smcarmiss, append=T)
cat('new; \n', file=smcarmiss, append=T)
cat('run; \n', file=smcarmiss, append=T)
cat(' \n', file=smcarmiss, append=T)

cat('%macro importMPLUS; \n', file=smcarmiss, append=T)
cat('%do i=1 %to ', file=smcarmiss, append=T)
cat(paste(iter) , file=smcarmiss, append=T)
cat('; \n', file=smcarmiss, append=T)
cat('data work.data&i; \n', file=smcarmiss, append=T)
cat('infile ', file=smcarmiss, append=T)
cat('"', file=smcarmiss, append=T)
cat(paste(pathGEN,"data&i..dat",sep="\\") , file=smcarmiss,
    append=T)
cat('" ; \n', file=smcarmiss, append=T)
cat('INPUT x y q1 q2 q3 q4 q5 q6 q7 q8; /*<---insert
    variables here*/ \n', file=smcarmiss, append=T)
cat('RUN; \n', file=smcarmiss, append=T)
cat('%end; \n', file=smcarmiss, append=T)
```

Another "noisy code" example ...

```
cat('%mend; \n', file=smcarmiss, append=T)
cat('%importMPLUS \n', file=smcarmiss, append=T)
cat(' \n', file=smcarmiss, append=T)

##########################################
#      Draw random sample of size N      #
##########################################
cat('%macro samplesize; \n', file=smcarmiss, append=T)
cat('%do i=1 %to ', file=smcarmiss, append=T)
cat(paste(iter) , file=smcarmiss, append=T)
cat('; \n', file=smcarmiss, append=T)
cat('Proc surveyselect data=work.data&i out=work.sampledata&
    i method=SRS \n', file=smcarmiss, append=T)
if(perclust==50) {
cat('sampsize=50 \n', file=smcarmiss, append=T)
}
else if(perclust==75) {
cat('sampsize=75 \n', file=smcarmiss, append=T)
}
else if(perclust==100) {
cat('sampsize=100 \n', file=smcarmiss, append=T)
}
```

Another "noisy code" example ...

```
else if(perclust==200) {
cat('sampsize=200 \n', file=smcarmiss, append=T)
}
else if(perclust==400) {
cat('sampsize=400 \n', file=smcarmiss, append=T)
}
else if(perclust==800) {
cat('sampsize=800 \n', file=smcarmiss, append=T)
}
else if(perclust==1000) {
cat('sampsize=1000 \n', file=smcarmiss, append=T)
}
cat('SEED = ',round(runif(1)*10000000), '; \n', file=
    smcarmiss, append=T)
cat('RUN; \n', file=smcarmiss, append=T)
cat('%end; \n', file=smcarmiss, append=T)
cat('%mend; \n', file=smcarmiss, append=T)
cat('%samplesize \n', file=smcarmiss, append=T)
cat(' \n', file=smcarmiss, append=T)
```

This could be much better

- Weird indentation
- Use "/", not "backslash", even on Windows
- Use vectors
- Prolific copying and pasting of "cat" lines.
- Avoid `system(...)` except when R has not implemented some function. Never run `system(■mkdir whatever■)` when you ought instead run `dir.create(■whatever■)`. R has OS neutral functions to do many CLI checks, check directories, files, etc.

I found the prototype for that code in a previous project

```
gen <- paste(path,"generate.inp",sep="\\")

cat('MONTECARLO: \n', file=gen)
cat('NAMES ARE y1-y6; \n', file=gen, append=T)
cat(' NOBSERVATIONS = ',perclust*nclust, '; \n', file=gen,
    append=T)
if (perclust != 7.5) {
cat(' NCSIZES = 1; \n', file=gen, append=T)
cat(' CSIZES = ',nclust, '(', perclust, '); \n', file=gen,
    append=T)
}
if (perclust==7.5) {
cat(' NCSIZES = 2; \n', file=gen, append=T)
cat(' CSIZES = ',nclust/2, ' (7) ',nclust/2, ' (8); \n',
    file=gen, append=T)
}

#user-specified iterations
cat(' NREPS = ',iter,'; \n', file=gen, append=T)
cat(' SEED = 791305; \n', file=gen, append=T)
```

I found the prototype for that code in a previous project ...

```
cat(' REPSAVE=ALL; \n', file=gen, append=T)
cat(' SAVE=\n', path,'\\data*.dat; \n', file=gen, append=T,
    sep="")
cat(' ANALYSIS: TYPE = TWOLEVEL; \n', file=gen, append=T)

cat('MODEL POPULATION: \n', file=gen, append=T)

cat('%WITHIN% \n', file=gen, append=T)
# baseline loadings are all .3, add .4 if 'within' = 1, but
    change based on mod/strong
cat('FW BY y1-y2*', .3+(within*(.4+strong*.1)), ' \n', file=
    gen, append=T, sep="")
cat('y3-y4*', .3+(within*(.4-mod*.3)), ' \n', file=gen,
    append=T, sep="")
cat('y5-y6*', .3+(within*(.4-strong*.1)), '; \n', file=gen,
    append=T, sep="")
cat('FW@1;  \n', file=gen, append=T)
# residuals are just 1-loading^2
cat('y1-y2*', 1-(.3+(within*(.4+strong*.1)))^2, '; \n', file
    =gen, append=T, sep="")
cat('y3-y4*', 1-(.3+(within*(.4-mod*.3)))^2, '; \n', file=
    gen, append=T, sep="")
```

I found the prototype for that code in a previous project ...

```
cat('y5-y6*', 1-(.3+(within*(.4-strong*.1)))^2, '; \n', file
    =gen, append=T, sep="")
cat(' \n',file=gen, append=T, sep="")
cat('%BETWEEN% \n', file=gen, append=T)

# the ICC bit multiplies by .053 when ICC is low, by 1 when
    ICC is high
cat('FB BY y1-y2*', sqrt((.3+(between*(.4+strong*.1)))^2*
    (1+(icc*(.053-1)))), ' \n', file=gen, append=T, sep="")
cat('y3-y4*', sqrt((.3+(between*(.4-mod*.3)))^2 *(1+(icc*(
    .053-1)))), ' \n',file=gen, append=T, sep="")
cat('y5-y6*', sqrt((.3+(between*(.4-strong*.1)))^2 *(1+(icc*
    (.053-1)))), '; \n', file=gen, append=T, sep="")
cat('FB@1;  \n', file=gen, append=T)

#residuals are total variance (1 or .053, depending on  ICC)
     loading^2
cat('y1-y2*', 1+(icc*(.053-1))-sqrt((.3+(between*(.4+strong*
    .1)))^2*(1+(icc*(.053-1))))^2, '; \n', file=gen, append=
    T, sep="")
```

I found the prototype for that code in a previous project ...

```
cat('y3-y4*', 1+(icc*(.053-1))-sqrt((.3+(between*(.4-mod*.3)
    ))^2 *(1+(icc*(.053-1))))^2, '; \n', file=gen, append=T,
     sep="")
cat('y5-y6*', 1+(icc*(.053-1))-sqrt((.3+(between*(.4-strong*
    .1)))^2 *(1+(icc*(.053-1))))^2, '; \n', file=gen, append
    =T, sep="")
cat(' \n',file=gen, append=T, sep="")

#run the above syntax using Mplus

#shell (paste("cd", mplus, sep=" "))
shell (paste("mplus.exe",gen, paste(path,"save.out", sep="\\
    "), sep=" "))
```

Here was my Suggestion

Create separate functions to do separate parts of the work.
Avoid so much "cut and paste" coding. The cat function can include MANY separate quoted strings or values, there is no reason to write separate cat statements for each line.
Here was my suggestion for part of the revision

```
##Create one MPlus Input file corresponding to following
    parameters.
createInpFile <- function(path="apath", gen="afilename.inp"
    , perclust=2, nclust=100, iter=1000, mod=1, strong=1,
    between=1, within=1){

    ## Here is one cat that writes out the top stanza
    cat("MONTECARLO:
      NAMES ARE y1-y6;
      NOBSERVATIONS = ", perclust*nclust, "; \n",
      ifelse(perclust != 7.5 ,
             paste("NCSIZES = 1; \n    CSIZES = ", nclust, "
                 (", perclust, ");\n"),
```

Here was my Suggestion ...

```
            paste("NCSIZES = 2; \n   CSIZES = ", nclust/2,
                  " (7) ", nclust/2 , " (8); \n" ) ), file=
                gen,   append=T,
    sep="")

  ##Writes out another section: user-specified iterations
  cat( "NREPS = ", iter, ";
     SEED = 791305;
     REPSAVE=ALL;
     SAVE=", path, "\\data*.dat;
     ANALYSIS: TYPE = TWOLEVEL;
     MODEL POPULATION:
     %WITHIN% \n", file=gen, append=T, sep="")

  ## baseline loadings are all .3, add .4 if "within" =
       1, but change based on mod/strong
  ## Writes out section specifying models
  cat("FW BY y1-y2*", .3+(within*(.4+strong*.1)),
       "y3-y4*", .3+(within*(.4-mod*.3)),
       "y5-y6*", .3+(within*(.4-strong*.1)),
       "FW@1; \n",
       "y1-y2*", 1-(.3+(within*(.4+strong*.1)))^2, "; \n",
```

Here was my Suggestion ...

```
            "y3-y4*", 1-(.3+(within*(.4-mod*.3)))^2, "; \n",
            "y5-y6*", 1-(.3+(within*(.4-strong*.1)))^2, "; \n",
            " %BETWEEN% ",
            "FB BY y1-y2*", sqrt((.3+(between*(.4+strong*.1)))^
                2*(1+(icc*(.053-1)))),
            "y3-y4*", sqrt((.3+(between*(.4-mod*.3)))^2 *(1+(
                icc*(.053-1)))),
            "y5-y6*", sqrt((.3+(between*(.4-strong*.1)))^2 *
                (1+(icc*(.053-1)))),"; FB@1; \n",
            "y1-y2*", 1+(icc*(.053-1))-sqrt((.3+(between*(.4+
                strong*.1)))^2*(1+(icc*(.053-1))))^2, ";",
            "y3-y4*", 1+(icc*(.053-1))-sqrt((.3+(between*(
                .4-mod*.3)))^2 *(1+(icc*(.053-1))))^2, ";",
          "y5-y6*", 1+(icc*(.053-1))-sqrt((.3+(between*(
              .4-strong*.1)))^2 *(1+(icc*(.053-1))))^2, ";",
          "\n", file=gen, append=T, sep="")
}
```

Critique that!

- Benefit of re-write is isolation of code writing into a separate function
- We need to work on cleaning up use of "cat" to write files.