SYNTAX EXAMPLES FOR R STATISTICAL PROGRAM

by Matthew Keller, matthew.c.keller@gmail.com. 6/20/2006. Distribute freely.

R PROGRAMMING ENVIRONMENT

DEAL WITH OBJECTS:

```
ls()
length(ls())                    #how many objects in memory?
remove(list = ls())             #remove all objects
remove("dataset")               #remove a particular object
```

DEAL WITH ATTACHED FILES

```
search()
attach(myfile)                  #attach objects in myfile to 2nd position
attach(myfile, pos=5)           #attach them to 5th position
detach (5)                      #to detach objects in 5th position (if possible)
```

LOAD A SCRIPT

```
source("C:/Program Files/R/functions/mckeller/mckeller.functions.R")
```

CHANGE DIRECTORY

```
getwd()             #get the current working directory
setwd("c:/temp")
```

SAVE R IMAGE & HISTORY

```
save.image("PaperAnal4.RData")
savehistory(file = "MySession1.Rhistory")      #saves all commands you've inputed
history(max.show = 46, reverse = FALSE)        #displays last 46 commands
```

SAVE OUTPUT (DISALLOWS YOU TO SEE OUTPUT IN CONSOLE)

```
sink("MIsession12_9_05.Rlis")
```

SAVE BOTH COMMANDS AND OUTPUT, AT END OF SESSION:

```
<Under "File", Choose "Save to file…">
```

CHECK HOW LONG A FUNCTION TAKES

```
t1 <- Sys.time()
<SOME FUNCTION>
t2 <- Sys.time()
difftime(t1,t2)

#ALTERNATIVE
system.time(for (i in 1:runs){perm_results[i,] <- matrix(t(permstats[1:2,]),nrow=1)    })
```

EXAMPLES OF HOW FUNCTION IS USED (MEAN IN THIS CASE)

```
example(mean)
```

RENAME ALL OBJECTS IN ENVIRONMENT AND GIVE THEM PREFIX

```
vars <- ls()
for (i in vars){ cat(paste("MDD.",i," <- ",i,  sep=""),file="temp")
eval(parse(file="temp"))
cat(paste("remove(",i,")", sep=""),file="temp2")
eval(parse(file="temp2"))}
```

MAKE USEFUL LISTS

```
newlist <- list(
comment=matrix(c(
"I created this on 6-3-06.",
"There should be 6 columns and 50 rows." ,
```

```
"Make sure that if you want to use the data matrix, you index it with",
"two brackets rather than one, i.e., [[]] rather than [], since the latter",
"just provides another list."), nrow=6,ncol=1),
dimms=c("r","b","g","y","o","p"),
data=red[1:20,])
```

IMPORTING/EXPORTING

EXPORT DATA INTO A SPECIFIC FOLDER & FORMAT

```
write.table(dataset, file = "C:/temp/MyData", append = FALSE, quote = TRUE, sep = "&", eol = "\n", na
= "-999", dec = ".", row.names = FALSE, col.names = TRUE, qmethod ="double")
```

SAVE FILES FROM SPSS THAT ARE READABLE IN S-PLUS (IN SPSS LANGUAGE):

##Import directly from an excel file – OR -
##Always save as *.dbf, not text! Make sure to have S-Plus see it is a dbf file (filetype) before importing.
SAVE TRANSLATE OUTFILE='C:\Matts Folder\RESEARCH\S-Plus\attempt3.db4'/TYPE=TAB /FIELDNAMES
/KEEP=nmcode2 sex weeksbf k5 k1 k8 zcd_sad TO zcd_slp f.sad2 TO f.noeat2 /RENAME zcd_sad=zsad.

READ IN A DATA FRAME AND A HEADER FILE:

```
heads <- read.table("C:/temp/colnames.study1")
heads2 <- as.matrix(heads) #the data frame must be a matrix to change col & row #s
dim(heads2) <- c(1,50)           #originally, the header file was 10x5

##BUT THE ABOVE SCREWS IT UP… R READS DOWN COLUMN BY COLUMN, SO WE MUST TRANSPOSE 1ST:
heads <- read.table("C:/temp/colnames.study1")
heads2 <- as.matrix(t(heads))    #the data frame must be a matrix to change cols & rows
dim(heads2) <- c(1,50)
study1 <- read.table("C:/temp/study1.txt")
names(study1) <- heads2
```

READ & WRITE TABLES

```
data1 <- read.table("C:/temp/try2.txt", header = TRUE, sep = "&", dec = ".",as.is = TRUE,
      na.strings = c("NA", " ","999"),colClasses = NA, skip = 0, check.names = TRUE)

write.table(cor.table, file="C:/temp/Table2.txt")    #saves it to read it as a text file

save(cor.table,file="C:/temp/Table2.txt")            #saves it for reading back into R
```

EXPLORING DATA

MULTI-WAY IF/THEN STATEMENT:

```
x <- c(1,2,3,4,5)
if (x[1]==2) a <- 3 else if (x[1]==3) a <- 4 else if (x[1]==1) a <- 6 else a <- NA
```

RUN A SEQUENTIAL NUMBER OF STATISTICS ON MANY VARIABLES:

```
for (k in 152:171){
des.anova <- lm(scale(dataset[,k])  ~ var1*var2, na.action=na.exclude)
print("THIS IS FOR VARIABLE:")
print(names(dataset[k]))
print(summary(des.anova))
print(anova(des.anova))}
```

REMOVE CERTAIN VARIABLES FROM A REG EQUATION W/O RETYPING LOTS:

```
reg1 <- lm(var1 ~ var2+var3+var4+var5+var4*var5))
summary(update(reg1, .~. – var4 – var5-var4*var5))
```

LOOK AT A LOT OF MEANS AT ONCE

```
apply(dataset[,c(1,2,26,35,49,65,74)],2,mean)  !This gives you the means of each variable (COLUMN)
apply(dataset[,c(9,33,48,61,80)],1,mean)       !Gives you each subject's mean across vars (ROW)
```

APPLY A FUNCTION BROKEN UP BY SOME FACTOR (E.G., SEAS):
```
tapply(dataset[,5], dataset$seas, FUN=mean)
```

DO STATS ON SUBSETS OF DATA
```
summary(dataset[,c("var1","var2")])
summary(dataset[var1<40,]$var1)
summary(dataset[var1<40,]$var99)
summary(lm(var1~var2+var3, data=dataset[-150,]))
summary(lm(iwt5[iwt5$seas=='w',]$glob.loc ~ iwt7$tmp.sta + activ.tdy, na.action = na.omit))
summary(lm(var1~var2+var3, subset=(var22 == xor("death", "love")), data=dataset))
pairs(dataset[runif(446)<.10,214:226],labels=name.cols(dataset[,214:226]))
```

MEANS MATRIX
```
means.mat <- matrix(nrow=3, ncol=4)
ccc <- c("Belly","Poli","Admin")
for(col in 1:4){
   switch (col,
      "1" = for(i in 1:3) means.mat[i,col] <- mean(Read[grp==ccc[i]]),
      "2" = for(i in 1:3) means.mat[i,col] <- mean(Dance[grp==ccc[i]]),
      "3" = for(i in 1:3) means.mat[i,col] <- mean(TV[grp==ccc[i]]),
      "4" = for(i in 1:3) means.mat[i,col] <- mean(Ski[grp==ccc[i]])
      )}

means.mat <- aggregate(ximp,by=list(BWData_wn$eventnum),mean, na.rm=TRUE) #much easier
```

STATISTICS FOR COMBINED ROWS AND COLUMNS OF VARIABLES
```
meanvc <- vector(length=18)
sdvc <- vector(length=18)
for (i in 1:18){meanvc[i] <- mean(as.matrix(dataset[1:nrow(dataset),c(seq(41+i,131+i,by=18))]))
sdvc[i] <- sd(stack(dataset[1:nrow(dataset),c(seq(41+i,131+i,by=18))])$values)}
```

ROW MEANS
```
sympavg <- matrix(0,nrow(zsymptoms),18)
for (i in 1:18){sympavg[,i] <- rowMeans(dataset[1:nrow(dataset),c(seq(i,18*6,by=18))], na.rm =TRUE)}
```

LAGGED DIFFERENCE SCORES
```
diff(x, lag=1)
```

MANAGING DATA SETS

STANDARDIZE A VARIABLE CONDITIONALLY:
```
x <- vector(length=121)
   for(k in 1:121) {
   if(x.old[k,3]=="a") x[k] <- (x.old[k,4] - mean(x.old[var1=="a"]))/stdev(x.old[var1=="a"])
      else x[k] <- (x.old[k] - mean(x.old[var1=="b"]))/ stdev(x.old[var1=="b"]) }
```

REPLACE CERTAIN VALUES WITH 'NA':
```
dataset[5,2] <- NA         #replace a single cell with NA

for (k in 3:9){for (i in 1:121){if(dataset[i,k]==5) x[i,k] <- NA else dataset[i,k] <- dataset[i,k]}}

dataset[which(dataset[,3:9]==-999)] <- NA              #much easier & faster than looping

winred.ev <- as.matrix(winred[,c(33,64,94,123,152,184)]) #only works for matrices!
winred.ev[is.na(sumsymp)] <- NA        #for every place sumsymp is NA, winred.ev is now NA
```

RECODE DATA
```
new <- matrix(0,nrow(dataset),6)
new <- ((dataset[,22:27]==1)*1 + (dataset[,22:27]==2)*2 + (dataset[,22:27]==3)*8 +
(dataset[,22:27]==4)*3 + (dataset[,22:27]==5)*4 + (dataset[,22:27]==6)*7 +
```

```
(dataset[,22:27]==7)*5 + (dataset[,22:27]==8)*6 + (dataset[,22:27]==9)*9 +
(dataset[,22:27]==10)*7 + (dataset[,22:27]==11)*8 + (dataset[,22:27]==12)*8)
```

CHECK MISSING VALUES

```
which(is.na(dataset)==TRUE, arr.ind=TRUE))           #which subjects have missing data?
dim(which(is.na(dataset)==TRUE, arr.ind=TRUE) )[1]   #how many NAs in dataset?
```

SUBSET DATA:

```
dataset.subset <- dataset[seas=="spring" | seas=="summer",]    #only for spring & summer
dataset.subset2 <- dataset[var5 != NA,]                        #only non-missing on var5
dataset.subset2 <- dataset[dataset[,5] != NA,]                 #same thing
dataset3 <- dataset[,-c(5:20)]                                 #all columns except 5:20
dataset4 <- dataset[,c(1,4,23)]                                #only columns 1,4, &23
cbind.data.frame(categ2, daysbf, CONSV)[categ2!="other" & categ2 != "nosit",]
```

LOOK AT TWO COLUMNS OF DATA NEXT TO EACH OTHER:

```
dataset[,c(18,34)]                    #look at two columns side by side
cbind(var1, var3, var67)              #if dataset is attached
dataset2 <- edit(dataset)             #open in excel-like spreadsheet, saved in dataset2
file.show(dataset)                    #an alternative; opens in script editor windor
```

ORDER/SORT A DATASET

```
cbind(order(dataset$var1),dataset[order(dataset$var1),]) #1st column is original row#, 2nd is its value

a <- c(1,2,3,4,5,6,6,5,4,3,2)
b <- c(5,4,3,2,1,2,3,4,5,6,7)
c <- cbind(a,b)
c[order(a),]
```

RENAME 5th COLUMN IN DATA SET

```
names(dataset)[5] <- 'newname'
```

REARRANGE A SET OF COLUMNS

```
newdataset <- dataset[,c(1:13,seq(14,48,by=2),seq(15,49,by=2))]
```

USE CBIND.DATA.FRAME NOT CBIND IF WANT TO USE VARIABLE NAMES FOR SOMETHING

```
dataset <- cbind.data.frame(dataset[,],y[,])
```

CHANGE CLASS ATTRIBUTES IN A DATA SET

```
for (i in 1:dim(x)[2]){
   if (class(x[,i])=="factor") class(x[,i]) <- "character" else class(x[,i]) <- class(x[,i])}
```

CHECK IF THERE ARE ANY REPEAT #S IN A VECTOR

```
dim(as.matrix(vect))
dim(as.matrix(unique(vect)))
```

LOOK AT THE ROW OF DATA THAT FITS SOME CRITERIA:

```
dataset[dataset$FAMNO==52444,]
```

MAKE “DATA1, DATA2, DATA3” BY LOOPING

```
for (i in 1:5) {
cat(paste("newz",i, sep=""),"<- data[(i*5000-4999):(i*5000),]",file="temp")
eval(parse(file="temp"))}
```

RETURNS “TRUE” IF ALL VALUES ARE REAL, “FALSE” IF 1+ MISSING

```
all(is.na(x)==FALSE)
if (all(is.na(y)==FALSE)==TRUE) k <- 5 else k <- 5-1 # k=5 if no missing in y, k=4 ow
```

TAKE THE POSITION OF AN ELEMENT AND USE IT AS A VARIABLE

```
a <- dataset[1:nrow(dataset),28:33]
```

```
set2 <- matrix(0,nrow(dataset),6)
for (i in 1:dim(a)[1]){set2[i,1:length(which(a[i,]==1))] <- which(a[i,]==1)}
```

GIVE VARIABLES IN DATA FRAME ALTERED NAMES FROM ANOTHER DATA SET

```
winsymp <- as.data.frame(winsymp)
names(winsymp) <- paste("S",names(dataset[,42:149]),sep="")
```

RESHAPE DATA SET FROM "WIDE" TO "LONG" FORMAT

```
BWData <- reshape(Dataset, idvar=c("FAMNO","ID"),
varying=list(names(Dataset)[4:9],names(Dataset)[10:15],names(Dataset)[16:21],names(Dataset)[22:27],na
mes(Dataset)[28:33], names(Dataset)[h[1:6]], names(Dataset)[h[7:12]], names(Dataset)[h[13:18]],
names(Dataset)[h[19:24]], names(Dataset)[h[25:30]]), direction="long")
```

FUNCTIONS

SWITCH

```
for(i in c(-1:3,9,11))  print(switch(i, 1,2,3,4,5,6,7,8,99,10,"this is eleven"))
switch("a","a"=1,"cc"=2,"d"=3)
```

WRITE A FUNCTION WITH SET DEFAULTS

```
multi.hist <- function(x, file="C:/temp/Multihist.pdf"){
pdf(file, horizontal=FALSE, height=5, pointsize=10)
for(k in 1:dim(x)[[2]]) {
       hist(x[, k], main = paste("Histogram of" , names(x[k])), xlab = names(x[k]))} dev.off()  }
```

FACTOR ANALYSIS WITH BINARY DATA; (POLYCOR LIBRARY)

```
library(polycor)
#change numeric 0/1's to factor scores (binary data, to run polychoric corr)
factor.scores <- matrix(as.factor(score),nrow=5000,ncol=40,byrow=F)
mi.corr <- hetcor(binary.scores[,1:10],ML=FALSE)$correlations
factanal(covmat = mi.corr, factors=1,lower=.05, start=errcoef.mi, nstart=100)
```

VIEW & IMPUTE MISSING VALUES (NORM LIBRARY)

```
library(norm)              #use "norm" only if all vars normal; if not, use "mix" or "cat"
a <- prelim.norm(as.matrix(data_wn))
summary(a); a$nmis; a$r
cat("Total subject missing:"); cat(" ", sum(as.numeric(rownames(a$r)[2:length(rownames(a$r))])),"\n")
cat("Total values missing:"); cat(" ", sum(a$nmis),"\n")

thetahat <- em.norm(a)     #use expectation maximization (EM) algorithm
rngseed(599829)
ximp <- imp.norm(a,thetahat,data_wn)
```

GRAPHING

PLOT LEVERAGES WITH TEXT:

```
lm.model <- lm(var1~var2+var3)
x <- model.matrix(lm.model)
plot(hat(x), type="n", ylab="Leverages")
text(hat(x), labels=as.character(SUBNUM))
```

LINEPLOT OF MEANS MATRIX

```
matplot(1:4,t(means.mat),type="l",axes=F)
par(lwd=5)
axis(1,1:4,c("Read","Dance","TV","Ski"))
axis(2)
legend(3.5,9,c("Belly","Poli","Admin"),lty=1:3,ncol=1)
```

ADD NEW POINTS ONTO EXISTING GRAPH

```
matplot(x, y, pch=19,type="b",main="Change in Var",xlab="Degree",ylab="Var",ylim=c(-.1,.6))
```

```
matpoints(x, y2, pch=22,type="b",col=”red”)
```

INCLUDE TEXT AS POINTS ON A GRAPH:

```
plot(x,y, type="n", xlab="scaled after", ylab="scaled before")
text(x,y, as.character(subnum))

matplot(abs(mean1[1]-mean1[2:7]), type="c", lty=2, ylim=c(0,.05),col=1)
text(1:6,abs(mean1[1]-mean1[2:7]), "HELLO", font=4, col=1)  #”HELLO” at each point of graph
```

MAKE MATLINES HAVE RANDOM COLORS (ALL 657 OF THEM)

```
cool <- c(colors())
for (i in 2:10) {
cat("matlines(abs(",paste("mean",i,"[1]-", sep=""),paste("mean",i,"[8:13]", sep=""),"),
pch=19,type='b',col= sample(cool, 1, replace = FALSE))",file="temp")
eval(parse(file="temp"))}
```

SEE FUNCTION “PAR” FOR TONS OF GRAPHING PARAMETERS

```
?par
demo(plotmath)            #gives you how to get many symbols in R
example(points)           # gives lots of symbols
```

SAVE ALL GRAPHING OBJECTS YOU MAKE UNTIL USE “DEV.OFF” CMD

```
pdf("C:/temp/new.pdf", horizontal=FALSE, height=5, pointsize=10)
matplot(cor(data1[,c(1,8:13)])[2:7,1])
dev.off()

dev.print(file="C:/Matts Folder/NEWNEW.pdf", device=pdf) #or more simply, just this
?device            #see all the possible devices
```

CHANGE THE AXIS TICK MARK LABELS TO DEFAULT FOR Y AND A:F FOR X

```
matplot(1:6, x[c(1:6)], type="b", ylim=c(.75,.9), cex.main=.85,lwd=3, pch=19,xlim=c(.5,6),axes=FALSE)
axis(1,labels=c("a","b","c","d","e","f"))
axis(2)
```

MORE CONTROL OVER YOUR LABEL LOCATIONS

```
mtext("X", side=2, padj=-1,adj=.83)     #PADJ
```

FIND THE X & Y LIMITS ON YOUR GRAPH, FOR USE IN PLACING ITEMS IN GRAPH

```
lim <- par("usr")
text(lim[1]+(.46*(lim[2]-lim[1])),lim[3]+(.93*(lim[4]-lim[3])),"Death", adj=1)
text(lim[1]+(.5*(lim[2]-lim[1])),lim[3]+(.93*(lim[4]-lim[3])),"vs", adj=0.5)
text(lim[1]+(.54*(lim[2]-lim[1])),lim[3]+(.93*(lim[4]-lim[3])),"Failure", adj=0)
```

QQ PLOT

```
pdf("C:/temp/New.pdf", horizontal=FALSE, height=5, pointsize=10)
for (k in 1:10){qqplot(liab[,k],rnorm(10000))}
dev.off()
```

PLOT WITH GREEK CHARACTERS, HATS, ETC

```
?plotmath

#placed on same line
text(7,.3,substitute(paste(hat(eta)^2,"=",x,sep=""), list(x=effectsizes[1,1])),adj=0)

#ALTERNATIVE
text(-.5,-.5, expression(bold(M[1])*":"*hat(MPG)))
```

MOVE TO ONE PARTICULAR FIGURE IN MULTIPLE FIGURE ENVIRONMENT

```
par(mfg=c(2,1))     #goes to figure in row 2, col 1
text(5,.1,"hi")     #places “hi” there
```