Automated bootstrap selection of matrix survey block design

Ryan Scott and Tyler Scott

March 21, 2014

In order to devise a block structure that optimized the eventual imputation power of the data set, we develop a method to use bootstrap replicates of a full pilot data set to test multiple possibilities for the matrix design based on random selection of questions out of a progressively constrained list of possible questions. Selection of the optimal matrix design is selected based on the matrix that provides the lowest sum of the residual between the pilot and imputed correlation.

In order to select the optimal design, we start with a pilot survey with complete responses for all individuals. This data set thus becomes the basis for establishing the "true" correlation structure as well as the basis for the bootstrap expansion of the data set based on the block design.

The concept for our methodology is based on the concept of a survey design as a matrix of questions, with each column representing a question and each row representing a block. Thus, a survey with 12 blocks and 5 questions per block can be represented as a 12*5 matrix. Within a survey design, one likely will use screener questions, and such questions are allocated to every block, leaving a subset of each block unfilled. Because of the ease of the method, we utilize a constrained vector of possible questions and randomly select a question for each unfilled location in the design matrix. The vector is continually contained such that no question can appear within a block twice, and such that no question can occur more than a set number of times.

Once the design matrix is complete, the blocks are each completed with the questions selected through the randomization routine through the creation of bootstrap replicates of observations from the pilot data set, where only questions asked within the blocks are complete for each observation, and all others are re coded as missing. The number of observations that are drawn is the choice of the user. This method results in a k blocks of n observations, where q questions are responded to within each block, and all others are missing.

We then use the MI package in R to impute a the missing values based on the entire data frame, using noise control methods to limit the impact of imputing values with values that were themselves imputed. Once the multiple imputation is completed, we extract the correlation matrix corresponding to the imputed data set. This correlation matrix is compared back to the original correlation matrix that resulted from the original pilot study. By taking the residuals of the two matrices, we are able to select the design matrix that most effectively minimizes any residual between the original and imputed correlation structure. With a perfectly imputed data set, any resulting correlations would be identical, and thus, the "best" matrix design would result in a residual score of 0. Therefor, from the randomly designed matrices, the program selects the matrix that minimizes the resulting residuals as the optimal design.

The procedure can be replicated as many times as necessary, but being that for each replicate an entire survey is imputed, the method is computationally intensive and increasing the number of trials greatly increases the burden. However, a data set with fewer imputations needed or a stronger design matrix will take significantly less time to analyze than a trial where a completely useless matrix design is used. Thus, setting a time limit for each trial imputation may be an effective way of minimizing the time necessary to complete the procedure.

```
#setwd('/hwb_project')
rm(list=ls())

sd<-read.csv("bakerclass_pilot.csv")
sd<-sd[-26,]
require(mi)

summary(out$Q12)
Question12 #and i dont have time to do it as much as I like and I dont have access.
```

```
levels(sd[,2])<-c("16-21","22-29","30-39","40-49","50-59")
levels(sd[,3])<-c("tac","pierce","balance","far")

#remove extra columns
sd<-sd[,4:18]

#convert letters to numbers
for(i in 1:ncol(sd)){
  sd[,i]<-as.numeric(sd[,i])
}

#code numbers as strings
for(i in 1:ncol(sd)){
    sd[,i]<-as.character(sd[,i])
}

#code levels for variables

#Q3: How long have you lived in PS
levels(sd[,1])<-c("1","1-3","4-10",">10")
#Q4: I am attached to Puget Sound region
levels(sd[,2])<-c(2,1,0,-1,-2)
#Q5: I identify with Puget Sound region
levels(sd[,3])<-c(2,1,0,-1,-2)
#Q6 How satisfied with life as whole
levels(sd[,4])<-c(2,1,0,-1,-2)
#Q7 How frequently feel inspired in nature
levels(sd[,5])<-c(1,.75,.5,.25,0)
#Q8 How frequently nature reduces stress
levels(sd[,6])<-c(1,.75,.5,.25,0)
#Q9: How often outdoor in Winter
levels(sd[,7])<-c(5,2,1,.25,0)
#Q10: How often outdoor in summer
levels(sd[,8])<-c(5,2,1,.25,0)
#Q11: How often gather resources
levels(sd[,9])<-c(0,1,2,3,4)
#Q12: able to harvest enough resources
levels(sd[,10])<-c(4,3,2,1)
#Q13: participate in cultural activties
levels(sd[,11])<-c(0,1,3,12,52)
#Q14: participate in environmental stewardship
levels(sd[,12])<-c(0,1,3,12,52)
#Q15: work with others in community
levels(sd[,13])<-c(0,1,3,12,52)
#Q16: trust local policy makers
levels(sd[,14])<-c(1,.66,.5,.33,0)
#Q17: trust experts
levels(sd[,15])<-c(1,.66,.5,.33,0)

out<-sd
for(i in 1:ncol(out)){
    out[,i]<-as.double(out[,i])
}
```

```
out<-sd
for(i in 1:ncol(out)){
out[,i]<-ordered(out[,i])
}



#SIMULATE A BLOCK DESIGN
makedesign <- function(nblocks=12,qsperblock=5,screeners=c(1,4),numqs=15,maxoccurence=8)
{design<-matrix(0,ncol=qsperblock,nrow=nblocks)
  for(p in 1:length(screeners)){design[,p]<-screeners[p]}
#generate sample list
for(i in 1:nblocks){
  for(q in (length(screeners)+1):ncol(design)){
    possibleqs<-seq(1,numqs,1)[tabulate(design,nbins=numqs)<maxoccurence]
    s1<-possibleqs[possibleqs %in% design[i,]==FALSE]
    s<-sample(s1,1)
    design[i,q]<-s}}
  return(design)}


#GENERATE MULTIPLE BLOCK DESIGNS
multdesigns <- function(ndesigns=2,nblocks=12,qsperblock=5,screeners=c(1,4),numqs=15,maxoccurence=8)
{
  designs <- list()
  for (i in 1:ndesigns)
  {
    designs[[i]]<-as.data.frame(makedesign(nblocks,qsperblock,screeners,numqs,maxoccurence))
  }
  return(designs)
}

#Build Fake Data for 1 design
fakesample<-function(design,data=sd,nblocks=12,qsperblock=5,numinblock=100)
{
total<-numinblock*nblocks
allblocks<-data.frame(matrix(NA,nrow=total,ncol=ncol(data)))
for(i in 1:nblocks)
{
  beg<-1+(numinblock*(i-1))
  end<-numinblock+(numinblock*(i-1))
  #sample from observed complete individuals
  peopletodraw<-c(sample(1:nrow(data),numinblock,replace=TRUE))
  #which questions to draw
  mm<- c(as.numeric(design[i,]))
  #pull from observed data, questions in mm
  df<-data[peopletodraw,mm]
  allblocks[beg:end,mm]<-df
}
return(allblocks)
}

library(snow)
```

```
require(mi)
library(lme4)

#BUILD LIST OF FAKE DATA
#generate 16 designs
temp<-multdesigns(ndesigns=1500)
#generate 16 fake datasets based upon 16 designs
temp1<-lapply(1:length(temp), function(x) fakesample(design=temp[[x]]))
vec<-unlist(lapply(1:length(temp1), function(x) sum(colSums(apply(temp1[[x]],2,is.na))==nrow(temp1[[x]]
temp1<-temp1[vec]

require(foreach)
require(doParallel)
#run multiple imputation models
cl<-makeCluster(16)
registerDoParallel(cl)
multimputes<-foreach(i = 1:length(temp1), .packages=c('mi')) %dopar%
  mi(object=temp1[[i]],n.iter=30,n.imp=3, add.noise=noise.control(method="reshuffling", K=1))
stopCluster(cl)


library(dplyr)
library(plyr)
#run imputation on each fake dataset
da<-lapply(1:length(multimputes),function(x) mi.completed(multimputes[[x]]))

#make each imputed dataset a data frame, select one of three imputed sets randomly for each imputation
da1<-lapply(1:length(da),function(x) as.data.frame(da[[x]][1]))
                                                    #[sample(1:3,1)]))

#make values numeric
da2<-lapply(1:length(da1),function(x) apply(da1[[x]],2,as.numeric))

numdat<-apply(sd,2,as.numeric)


fakecor<-lapply(1:length(da2),function(x) cor(da2[[x]], use='pairwise.complete.obs',method='spearman'))

obscor<-cor(numdat,use='pairwise.complete.obs',method='spearman')

#compar faked to observed (toss out values where NA for one question)
cordiff.score<-lapply(1:length(fakecor),function(x) sum(abs(obscor-fakecor[[x]])))
save.image('searchfordesign.sim7.RData')
```