ManLabs2 R Code Review

ManyLabs2 (Corresponding coder: Fred Hasselman)

3 February 2016

Contents

Instruction for reviewers

Analysis Strategy

The ManyLabs2 data analysis strategy attempts to regard three principles that maximize research transparency:

1. **Principle of Equality**: All data should be treated equally by a code. That is, the code should do its job generating results while at the same time being as naive as possible to the particular facts of the study being analysed. This will reduce any chances of bias with respect to the outcomes of a certain dataset or a particular study. If it is necessary to add study specific code, the second principle should be regarded.

2. **Principle of Transparency**: All operations that are crucial for obtaining an analysis result should be available for inspection by anyone who wishes to do so. This should be possible without the help of the auhtors that generated the code. The operations concern the application of data filtering rules, computation of variables derived from original measurements, running an analysis and constructing graphs, tables and figures. If full transparency is not possible, the third principle should be regarded.

3. **Principle of Reproducibility**: The most basic requirement for analysis results is that they should be reproducable given the original code and the original data set. However, any new implementation of the same analysis strategy in a different context, or application of the code to a different dataset, e.g. a replication study, should not be problematic. That is, outcomes may differ between data sets, but this should not be attributable to any details of the code or the analysis strategy.

R as a parser of online code.

The pre-registered Manylabs2 protocol describes a number of analyses per replication study that can be categorised as **Primary** (target replications per site), **Secondary** (additional analyses per site, e.g. on subgroups), and **Global** (analyses on the entire dataset).

These *promised analyses* have all been implemented in R in a transparent way and this implementation is now ready for an independent review.

Implementation

Functions avalaible in an R package on GitHub extract information and instructions about each promised analysis a table that is openly accessible, the masteRkey spreadsheet.

Each row in the table represents an analysis, the columns contain specific information about the analysis:

- Columns A through E are identifiers for study, analysis and slate.
- Column F and G contain R commands which will extract and label the columnms from the dataset needed for the analysis.
- Column H and I contain filter instructions for cases and subsamnples.
- Columns J through L contain information about the nature of the analysis (Global, Primary Secondary).
- Column M lists the name of a analysis specific *variable function* (varfun.) which in most cases just reorganises the variables specified in previous columns so they can be passed to the analysis code. In some cases these function perform specific calculations required by the original analyses.
- Columns N through S contain information about the statistiscal tests

What is in need of review by R experts?

The code runs and perfoms analyses on the data, that much we know :).

We would like to get your expert opinion on the following:

1. Does the R code in column O (stat.test) reflect the promised analyses in the protocol?
    - In order to evaluate this you'll need to look at the analysis plan for a specific study in the protocol and figure out whether the R code in rows of column O for that study represents all the tests that is described there.
    - In many cases this will be straightforward, without any need to actually run any code ooking at the way in which the variables are grouped and labelled and, filters are applied and which settings are used for the analysis, e.g. direction of the test (column P, stat.params).
    - In some cases you will need to inspect the contents of the varfun listed in column M. All varfuns are available in a sourceable file on Github. There is also a pdf of the manual pages

```
require(devtools)
source_url("https://raw.githubusercontent.com/FredHasselman/ManyLabRs/master/manylabRs/R/ML2_variable_functions.R")
```

2. Does the output correspond to what may be expected by the R code in column O?
    - Think about the test-statistic, the parameters (df) and N.
    - The results files can be found in the dropbox associated with a private project on OSF: /TestOutput/
        - RAW.DATA.PRIVATE - Contains the output of the data merging and filtering rules described in ML2_data_cleaning.html

---

Appendix

The analysis cor.test.fisherZ() defined as follows:

```
cor.test.fisherZ <- function(r1=NULL,r2=NULL,n1=NULL,n2=NULL,p=TRUE){
    if((dim(as.matrix(r1))[2]==2)&(dim(as.matrix(r2))[2]==2)){
        r1 <- cor(r1[,1],r1[,2],use="pairwise.complete.obs")
        r2 <- cor(r2[,1],r2[,2],use="pairwise.complete.obs")
    } else{
        if(all((dim(as.matrix(r1))!=1))&all(dim(as.matrix(r2))!=1)){
            disp(message = "r1 and r2 each need to be:", header = "cor.test.fisherZ", footer = FALSE)
            disp(message = "- Either a single numerical value representing a correlation,", header = FALSE, footer = FALSE)
            disp(message = "- Or a 2 column matrix from which a correlation r1 and r2 can be calculated", header = FALSE)
        }
    }
    z <- ((atanh(r1)-atanh(r2))/((1/(n1-3))+(1/(n2-3)))^0.5)
    if(p){p<-2*(1-pnorm(abs(z)))} else {p=NULL}
    stat.test <- structure(list(statistic = z,
                                method="Fisher r-to-Z transformed test for 2 independent correlations",
                                parameter=n1+n2,
                                p=p)
    )
    class(stat.test) <- "htest"
    return(stat.test)
}
```

The function any2r() is defined as follows:

```
#' any2r
#'
#' Converts most common test statistics into (signed) effect size \code{r}.
#'
#' @param s     Value(s) of a test statistic.
#' @param df1     Degrees of freedom
```

```
#' @param df2     NULL or degrees of freedom of the denominator for the f-distribution.
#' @param N     Number of data points used in calculation of test-statistic.
#' @param esType     Type of test statistic. One of: "t", "lm.t", "f", "lm.f", "r", "X2", "Z", "lm.Z"
#' @param CIcalc     If \code{TRUE} (default) the Confidence Interval for the test statistic in \code{x} will be calculated using the "Conf
#' @param CL     Confidence Limit (default: .95).
#' @param keepSign     Return effect size with sign of test statistic? (default = TRUE).
#'
#' @details The prefix "lm" is currently disregarded, but will be implemented in future versions to indicate the test statistic is in fact a fix
#'
#' @author
#' CHJ Hartgerink (original code)
#' F Hasselman (added vector input and CI calculation )
#'
#' @return The effect size \code{r} corresponding to the test statistic(s).
#' @export
#'
#' @examples
any2r <- function(st, df1 = NULL, df2 = NULL, N = NULL, esType = NA, CIcalc = TRUE, CL = .95, keepSign = TRUE){
    require(MBESS)
    esType <- gsub("lm.","",esType)
    if(CIcalc){
        sCI <- get.ncpCI(st, df1, df2, N, esType, CL)
    } else {
        sCI <- st
    }

    esComp <- list()
    cnt <- 0

    for(cnt in seq_along(sCI)){

        x <- sCI[cnt]

        esComp[[cnt]] <- ifelse(grepl("t",esType), sqrt((x^2*(1 / df1)) / (((x^2*1) / df1) + 1)),
                                ifelse(grepl("f",esType), sqrt((x*(df2 / df1)) / (((x*df2) / df1) + 1))*sqrt(1/df2),
                                       ifelse(grepl("r",esType), x,
                                              ifelse(grepl("X2",esType), sqrt(x/N),
                                                     ifelse(grepl("Z",esType), tanh(x * sqrt(1/(N-3))),
                                                            NA)
                                              )
                                       )
                                )
        )
    }

    ES <- cbind(sCI,rbind(unlist(esComp)))
    colnames(ES) <- c(c("ncp","ncp.lo","ncp.hi")[1:cnt],c("r","r.ciL","r.ciU")[1:cnt])
    if(keepSign){
```

```
        ES <- ES * c(c(1,1,1)[1:cnt], sign(sCl)[1:cnt])
    }
    return(ES)
}


get.ncpCI <- function(x, df1, df2, N, esType, CL=.95){
    require(MBESS)
    esType <- gsub("lm.","",esType)
    ncCI <- list()
    switch(esType,
           t = ncCI <- MBESS::conf.limits.nct(t.value=x, conf.level=CL, df=df1),
           f = ncCI <- MBESS::conf.limits.ncf(F.value=x, conf.level=CL, df.1=df1, df.2=df2),
           r = ncCI <- list(Lower.Limit = ci.R(R=x, conf.level=CL, N=(df1+2), K=1)$Lower.Conf.Limit.R,
                            Upper.Limit = ci.R(R=x, conf.level=CL, N=(df1+2), K=1)$Upper.Conf.Limit.R),
           X2 = ncCI <- MBESS::conf.limits.nc.chisq(Chi.Square=x, df=df1,
                                                    alpha.lower=0, alpha.upper=1-CL, conf.level=NULL),
           Z = ncCI <- list(Lower.Limit = (x - qnorm((1-CL)/2, lower.tail = FALSE)),
                            Upper.Limit = (x + qnorm((1-CL)/2, lower.tail = FALSE)))
    )
    return(cbind(ncp = x,
                 ncp.lo = ncCI$Lower.Limit,
                 ncp.hi = ncCI$Upper.Limit)
    )
}
```