Standard Functions for Basic Statistical Analysis

code in standardfunctions V2.R

Bruno Fischer Colonimos

4 june 2016

This is the code, presented as a notebook It should help readability

# Helper functions

## constants and Default options settings

# sfinitdefaults sets some constants in the data structure and returns a  
# function that gets and sets global options  
sfinitdefaults <- function () {  
 # structure definition, with some constants already set up  
 defaultvalue <- list( #namesum  
 namesumeng = c("n", "Mean", "St.dev",  
 "Min.", "1st Qu.","Median", "3rd Qu.", "Max.",  
 " NA's"),  
 namesumfrench = c("n", "Moyenne", "Ecart-type",  
 "Min.", "Q1","Médiane", "Q3", "Max.",  
 " NA's"),  
 namesum ="",  
 language = "",  
 filldefault = "steelblue")  
 # access function  
 function(name=NULL, value = NULL){  
 if(is.null(name)) {  
 warning("initdefaults : You must supply a name",  
 immediate. = TRUE, call. = TRUE)  
 NULL  
 } else {  
 if (!is.null(value)) {  
 defaultvalue[[name]] <<- value  
 defaultvalue[[name]]  
 } else {defaultvalue[[name]]}  
 }  
 }  
}

## set up access function and some defaults

sfdefault <- sfinitdefaults()  
  
sfdefault("language","french")

## [1] "french"

if(sfdefault("language") == "french") {  
 sfdefault("namesum", sfdefault("namesumfrench"))  
} else {  
 sfdefault("namesum", sfdefault("namesumeng"))  
}

## [1] "n" "Moyenne" "Ecart-type" "Min." "Q1"   
## [6] "Médiane" "Q3" "Max." " NA's"

sfdefault("reportNA", FALSE) # report number of NA's in a variable?

## [1] FALSE

sfdefault("digits", 2)

## [1] 2

## Simple and multiple summary tables

# (nb of cases, mean, stdev, five-number-summary, optionally nb of NA's)

vector of summaries for 1 quant variable

sumvector <- function (var, dnames = sfdefault("namesum"), reportNA = sfdefault("reportNA")) {  
 if (length(var) == 0) {  
 sapply(numeric(length = 9), function(x) NA)  
 }else {# construct a more complete summary vector  
 s <- summary(var)  
 if (length(s) < 7) {s <- c(s, rep(0, times=7-length(s)))}  
 ret <- numeric(3)  
 ret[1] <- sum(!is.na(var))  
 ret[2] <- s["Mean"]  
 ret[3] <- sd(var, na.rm = TRUE)  
 s <- c(ret, s[-4])  
 names(s) <- dnames  
 if (reportNA) {s} else {s[1:(length(s) - 1)] }  
 }  
}

Combined summaries for different variables in a dataframe, for all individuals

cbsummaries <- function (dataf, vnames) {  
 # vnames = a vector of variable names (each a numeric variable of dataf)  
 lsum = lapply(vnames, function(nam) sumvector(dataf[[nam]]))  
 df <- do.call(what = data.frame, args = lsum)  
 colnames(df) <- vnames  
 # rownames(df) <- namesum  
 df  
}

Combined summaries for one variable, *conditional* to the values of a factor

condsummaries <- function (dataf, vname, fname) {  
 # vname = the variable name  
 # fname = the factor name  
 # levels: if not factor, make it a factor and take the levels  
 if (is.factor(dataf[[fname]])) {  
 lv <- levels(dataf[[fname]])  
 } else {  
 lv <- levels(factor(dataf[[fname]]))  
 }  
 lsum = lapply(lv ,  
 FUN=function(lev) {  
 dt <- dataf[dataf[[fname]]==lev , ]  
 sumvector(dt[[vname]])  
 } )  
 df <- do.call(what = data.frame, args = lsum)  
 colnames(df) <- lv  
 # rownames(df) <- namesum # rownames are preserved  
 df  
}

## Frequency tables

# new . fonctionne avec des tbl\_df aussi  
condfreqtable <- function(dataf, nomfact1, nomfact2, useNA = "no") {  
 if (useNA == "no") {  
 dataf <- dataf[!is.na(dataf[[nomfact1]]) & !is.na(dataf[[nomfact2]]) , ]  
 }  
 dt <-prop.table(table(dataf[[nomfact1]], dataf[[nomfact2]] , useNA = useNA),  
 margin = 1)  
 dt2 <- as.data.frame(dt)  
 names(dt2) <- c(nomfact1, nomfact2, "perc") #compatibilité avec la def ancienne  
 dt2  
}

## reordering factors

# new definition seems ok for both data frame and tbl\_df  
orderfact <- function(dataf, nomfact, orderfreq = TRUE, orderdesc = TRUE,  
 ordervar = "c..nt", orderval = NA, orderfun = sum,  
 nlevels = NULL) {  
 if (is.null(nlevels)) {  
 direction <- ifelse(orderdesc,-1, 1)  
  
 if (orderfreq & ordervar == "c..nt") {  
 dataf$c..nt <- c(1)  
 }  
 if (is.na(orderval) & ordervar == "c..nt") {  
 dataf$c..nt <- c(1)  
 } else if (is.na(orderval) & ordervar != "c..nt") {  
 # dataf$c..nt <- ifelse(is.na(dataf[, ordervar]), 0, 1)  
 #  
 # ordervar <- "c..nt"  
 # ne rien faire ??  
 } else {  
 dataf$c..nt <-  
 ifelse(is.na(dataf[[ordervar]]),  
 0 ,  
 ifelse(dataf[[ordervar]] == orderval,  
 1, 0))  
 ordervar <- "c..nt"  
 }  
 # réordonner le facteur  
 xx <- dataf[[nomfact]]  
 xxx <- direction \* dataf[[ordervar]]  
 resfact <- reorder(xx, xxx, orderfun, na.rm = TRUE)  
  
 } else {  
 resfact <- factor(dataf[,nomfact], levels = nlevels)  
 }  
 # retour  
 resfact  
}

## Statistical Testing functions

# identify a warning  
is.warning <- function(x) {"warning" %in% class(x)}  
  
  
# try.chisq.test ==> essaye un test du chi2, et si il génère un warning  
# (conditions approximation du chi2 non satisfaites), alors, calculer la  
# p-valeur par simulation  
# si keep-all, retourne les 2 tests (chi2 et  
# simulation, une valeur logique indiquant le warning, et le warning lui-même).  
# Le test préférée est alors listé comme test1  
  
try.chisq.test <- function(..., keep.all = TRUE) {  
 ww <- tryCatch(chisq.test(...),  
 error = function(e) {e},  
 warning = function(w) w )  
  
 if (is.warning(ww)) {  
 if (keep.all) {  
 list(test1 = chisq.test(..., simulate.p.value = TRUE),  
 test2 = chisq.test(...),  
 warning = TRUE,  
 warningmsg = ww )  
 } else {  
 list(test1 = chisq.test(..., simulate.p.value = TRUE))  
 }  
 } else {  
 if (keep.all) {  
 list(test1 = chisq.test(...),  
 test2 = chisq.test(..., simulate.p.value = TRUE),  
 warning = FALSE,  
 warningmsg = "" )  
 } else {  
 list(test1 = chisq.test(...))  
 }  
 }  
}

# Fonctions d'analyse simples

**Résultat désiré:**

* **une variable**
  + cat1 1 facteur
  + num1c 1 variable continue
  + num1d 1 variables discrete
* **deux variables**
  + cat2 2 facteurs
  + cat1num1
  + num2

## cat1

# new definition OK  
cat1 <- function(dataf, nomfact, useNA = "no",  
 orderfreq = TRUE, orderdesc = TRUE, ordervar = "c..nt",  
 orderval = NA, orderfun = sum,  
 rfreq = TRUE, digits = 2, cfill = "steelblue") {  
 # useNA = "always, "ifany" or "no",  
 # orderfreq = TRUE or FALSE,  
 # descorder =TRUE or FALSE  
 # ordervar = variable to use for ordering,  
 # orderval = value if the ordering variable is the frequency of ordervar == value  
  
 # reordering the levels:  
 dataf[[nomfact]] <-  
 orderfact(dataf, nomfact,  
 orderfreq, orderdesc, ordervar, orderval, orderfun)  
  
 # make table as dataframe  
 tbl <- table(dataf[[nomfact]], useNA = useNA)  
 tbl <- data.frame(num = tbl, rfreq = tbl / sum(tbl))  
 tbl <- tbl[, c(1,2,4)]  
 names(tbl) <- c(nomfact, "num", "rfreq")  
 tbl$numlabs <- paste0("n=" ,tbl$num)  
 tbl$perclabs <- paste0(100 \* round(tbl$rfreq, digits),"%")  
 tbl$index <- ave(1:nrow(tbl), FUN = function(x) 1:length(x)) # rank  
 num <- sum(tbl$num)  
  
 # Goodness-of-Fit chi-square test for a uniform distribution  
 uchisq <- try.chisq.test(tbl[["num"]])  
  
 # bar chart with ggplot2  
 # the data  
 dataf1 <- if (useNA == "no") {  
 dataf[which(!is.na(dataf[[nomfact]])), ]  
 } else {  
 dataf  
 }  
 # base ggplot  
 pt <- if (rfreq) {  
 ggplot(dataf1,  
 aes\_(as.name(nomfact), quote(100 \* ..count.. / sum(..count..))))  
 } else {  
 ggplot(dataf1,  
 aes\_(as.name(nomfact)))  
 }  
 # geom  
 pt <- pt + geom\_bar(fill = cfill)  
 # ylabel  
 if (rfreq) {pt <- pt + ylab(label = "percent")}  
  
 # return a list of values  
 list( name = nomfact,  
 levels = levels(dataf[[nomfact]]),  
 table = tbl, num = num,  
 uchisq = uchisq,  
 plot = pt  
 )  
}

## num1d = 1 numeric d(iscrete)

# new definition:  
num1d <- function(dataf, nomvar, useNA ="no", digits = 2, sumdigits = 2,  
 rfreq = TRUE, width = .5, cfill = "steelblue") {  
 # make a table (with Frequency = nb of rows)  
 tb <- table(dataf[[nomvar]])  
 num <- sum(tb)  
 tbf <- tb/sum(tb)  
 tbflabs <- paste0(100\* round(tbf,digits), "%")  
 tbl <- data.frame(tb, tbf, tbflabs)  
 tbl <- tbl[ , c(1,2,4,5)]  
 colnames(tbl) <- c(nomvar, "num", "rfreq", "perclabs")  
 tbl$numlabs <- paste0("n=", tbl$num)  
 tbl$index <- ave(1:nrow(tbl), FUN = function(x) 1:length(x)) # rank  
 # print(tbl) #dbg  
  
 s <- sumvector(dataf[[nomvar]])  
  
 # Goodness-of-Fit chi-square test for a uniform distribution  
 uchisq <- try.chisq.test(tbl[["num"]])  
  
 # bar chart  
 # data+aes  
 if (useNA == "no") {dataf <- dataf[which(!is.na(dataf[[nomvar]])), ]}  
 if (rfreq) {  
 pt <- ggplot( dataf,  
 aes\_(as.name(nomvar),  
 quote(100 \* ..count.. / sum(..count..))) )  
 } else {  
 pt <- ggplot( dataf,  
 aes\_(as.name(nomvar)) )  
 }  
 # geom  
 pt <- pt + geom\_bar(width = width, fill = cfill )  
 # ylabel  
 if (rfreq) {pt <- pt + ylab("percent")}  
  
 # return a list of values  
 list( name = nomvar,  
 summaries = s,  
 table = tbl,  
 num = num,  
 uchisq = uchisq,  
 plot = pt)  
}

## num1c = 1 numeric c(ontinuous)

# another helper function  
# make class labels from bins vector  
mkclabs <- function(bins, sep = " - ") {  
 left <- head(bins, length(bins) - 1)  
 right <- tail(bins, length(bins) - 1)  
 mapply(function(x,y){paste(x,y,sep = sep)},  
 left, right, SIMPLIFY =TRUE)  
}  
  
# another helper: NA's remover  
nonavect <- function(vect) {vect[which(!is.na(vect))]}  
  
  
  
# num1c  
num1c <- function(dataf, nomvar, usedensity = FALSE, plot\_density = FALSE,  
 fillhist = "steelblue", color\_density = "red", digits = 2, # à modifier  
 bins = NULL, ...) { # ... = addtl arguments for geom\_hist  
 if (plot\_density) {usedensity <- TRUE} # plot\_density overrides usedensity  
 # bins = Null, integer, or a function name : "nclass.Sturges", "nclass.FD" , "nclass.scott"  
 if (!is.null(bins)) {  
 if ("character" %in% class(bins) ) {  
 bins <- do.call(bins, list(nonavect(dataf[[nomvar]])))  
 } else {bins <- NULL  
 warning("bins is not a function", call. = TRUE)}  
 }  
  
 p <- ggplot(dataf, aes\_(as.name(nomvar))) +  
 if (usedensity) {geom\_histogram(aes(y=..density..),  
 bins = bins, fill = fillhist,...)  
 } else {geom\_histogram(bins = bins, fill = fillhist, ...)}  
  
 if (plot\_density) {p <- p + geom\_density(color=color\_density) }  
  
 # make summaries vector + get number of cases  
 s = sumvector(dataf[[nomvar]])  
 num = s["n"] # number of cases  
  
 # get the frequency table from ggplot  
 tb <- ggplot\_build(p)$data[[1]][ , 1:8]  
 # add columns to it  
 tb$rfreq <- tb$count/num  
 tb$numlabs <- paste0("n=", tb$count)  
 tb$perclabs <- paste0(100\* round(tb$rfreq, digits), "%")  
 tb$index <- ave(1:nrow(tb), FUN = function(x) 1:length(x)) # rank  
 # get binwidth  
 cbinw <- unique(round(tb$xmax-tb$xmin,digits))  
 # get bins vector  
 cbins <- with(tb, c(xmin[1],xmax))  
 # make class lablels  
 clabs <- mkclabs(cbins)  
 # make a printable table  
 ptb <- data.frame(  
 class = clabs,  
 center = tb$x,  
 freq = tb$count,  
 rfreq = tb$rfreq  
 )  
  
 # Uniform Chi2 test  
 uchisq <- try.chisq.test(tb$count)  
 if (length(cbinw) >= 2) {  
 warning(paste0("Unif chi2 test ",  
 nomvar,  
 " called with different class widths!",  
 call. = TRUE)) }  
  
 # return named list  
 list(name = nomvar,  
 summaries = s,  
 tables = list(ggtable = tb,  
 binwidths = cbinw,  
 bins = cbins,  
 ptable = ptb),  
 num = num,  
 uchisq = uchisq,  
 plot = p)  
}

## cat2 = 2 categorical vars

# definition  
cat2 <- function(dataf, nomfact1, nomfact2, useNA = "no",  
 orderfreq1 = TRUE, orderdesc1 = TRUE, ordervar1 = "c..nt",  
 orderval1 = NA, orderfun1 = sum, nlevel1 =NULL,  
 orderfreq2 = TRUE, orderdesc2 = TRUE, ordervar2 = "c..nt",  
 orderval2 = NA, orderfun2 = sum, nlevel2 =NULL,  
 rfreq = TRUE, digits = 2, cfill = "steelblue"){  
 # useNA = "always, "ifany" or "no", orderfreq = TRUE or FALSE,  
 # descorder =TRUE or FALSE  
 # ordervar = variable to use for ordering  
  
 # reordering the levels:  
 # nomfact2 first  
 dataf[[nomfact2]] <- orderfact(dataf, nomfact2, orderfreq2, orderdesc2,  
 ordervar2, orderval2, orderfun2, nlevel2)  
 # nomfact1  
 if(orderfreq1 == TRUE &  
 ordervar1 == nomfact2 & !is.na(orderval1)){ # fréquences conditionnelles!  
 #print("Frequ cond")  
 tbl <- condfreqtable(dataf, nomfact1, nomfact2, useNA = "no")  
 #print("apres Frequ cond table")  
 tbl <- tbl[tbl[[nomfact2]] == orderval1, ]  
 # print("tbl") #dbg  
 # print(tbl) #dbg  
 tbl[[nomfact1]] <- orderfact(tbl, nomfact1,  
 orderfreq1, orderdesc1,  
 ordervar = "perc",  
 orderfun = orderfun1) #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 dataf[[nomfact1]] <- orderfact(dataf, nomfact1,  
 nlevels = levels(tbl[,nomfact1]))  
 } else { # autres cas  
 dataf[[nomfact1]] <- orderfact(dataf, nomfact1, orderfreq1,  
 orderdesc1, ordervar1, orderval1,  
 orderfun1, nlevel1)  
 }  
  
 # print(levels(dataf[[nomfact1]])) #debug  
 # print(levels(dataf[[nomfact2]])) #debug  
 # make table as dataframe  
 tblcrois <- table(dataf[[nomfact1]], dataf[[nomfact2]], useNA = useNA)  
  
 tbl <- as.data.frame(tblcrois)  
 colnames(tbl) <- c(nomfact1,nomfact2,"num")  
 # print(tbl) #debug  
 num <- sum(tbl$num)  
  
 tbl1 <- summarize\_(group\_by\_(tbl,as.name(nomfact1)),  
 num=quote(sum(num))) # shit with non-standard eval  
 tbl2 <- summarize\_(group\_by\_(tbl,as.name(nomfact2)),  
 num=quote(sum(num))) # shit with non-standard eval  
  
 # supplement tbl1  
 tbl1$numlabs = paste0("n=", tbl1$num)  
 if (!is.na(orderval1)){  
 tbl1$numval <- tblcrois[ ,orderval1] # keep it, not a df  
 tbl1$percval <- tbl1$numval / tbl1$num  
 tbl1$perclabs <- paste0(100 \* round(tbl1$percval, digits), "%")  
 }  
 tbl1$index <- ave(1:nrow(tbl1), FUN = function(x) 1:length(x)) # rank  
  
 # Chi-square test for independence  
 ichisq <- try.chisq.test(tblcrois)  
  
 # bar chart with ggplot2  
 # data  
 dataf2 <- if (useNA == "no") {  
 dataf[which(!is.na(dataf[[nomfact1]]) &  
 !is.na(dataf[[nomfact2]])), ]  
 } else {dataf  
 }  
 # plot  
 pt <- ggplot(dataf2) +  
 geom\_bar(aes\_(as.name(nomfact1), fill = as.name(nomfact2)),  
 position = "Fill") +  
 guides(fill = guide\_legend(reverse = TRUE)) +  
 ylab("percent")  
  
 #retourner les éléments  
 list(name = c(nomfact1, nomfact2),  
 levels = list(levels1 =levels(dataf[[nomfact1]]),  
 levels2 =levels(dataf[[nomfact2]]) ),  
 tables =list(tbl=tbl, tblcrois=tblcrois, tbl1=tbl1, tbl2=tbl2),  
 num = num,  
 ichisq = ichisq,  
 plot = pt  
  
 )  
}  
  
# cat1num1c

#  
# fonctions de détermination du nombre de classes dabs un histogramme  
# nclass.Sturges(mpg$hwy)  
# nclass.FD(mpg$hwy)  
# nclass.scott(mpg$hwy)  
#  
#

### fonctions de generation de graphiques

# continuous x factor boxplot & jitter plot  
cbyfboxjit <- function(dataf, varf, varc, useNA = "no",  
 labellayer = "", labelall = "All values", labelgroups = "by goup") {  
  
 if (useNA == "no") {  
 dataf <- dataf[!is.na(dataf[[varf]]) & !is.na(dataf[[varc]]), ]  
 }  
 ggplot(dataf,aes\_(as.name(varf) , as.name(varc), color=as.name(varf))) +  
 geom\_boxplot(aes(group = 1, fill = labelall), outlier.colour = "gray") +  
 geom\_boxplot(aes(fill = labelgroups), varwidth = TRUE, outlier.colour = "gray") +  
 geom\_jitter( width =.5, alpha=.5) +  
 labs(fill = labellayer)  
}  
  
  
# continuous x discrete boxplot & jitter plot  
cbydboxjit <- function(dataf, vard, varc, useNA = "no",  
 labellayer = "", labelall = "All values", labelgroups = "by goup") {  
 # dataf <- as.data.frame(dataf) # same problem with tbl\_df  
 if (useNA == "no") {  
 dataf <- dataf[!is.na(dataf[[vard]]) & !is.na(dataf[[varc]]), ]  
 }  
 dataf$fact\_vard. <- factor(dataf[[vard]])  
 ggplot(dataf,aes\_(as.name(vard) , as.name(varc), color=quote(fact\_vard.))) +  
 geom\_boxplot(aes(group = 1, fill = labelall), outlier.colour = "gray") +  
 geom\_boxplot(aes(fill = labelgroups), varwidth = TRUE, outlier.colour = "gray") +  
 geom\_jitter( width =.5, alpha=.5) +  
 labs(fill = labellayer)  
}