AYC TA Log - SAPR & MERT data

Summarizing Counts by Date & Setting

Josephine McKelvy

February 08, 2021

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

Click the **+c Insert** button to insert code chunks (or functions). Add documentation outside of the code chunks as reminders of what your code does.

When you click the **Knit** button, it compiles all of the code chunks and generates a document that includes both content as well as the output of any embedded R code chunks within the document. Use the **gear** icon to adjust settings (e.g., “show output only”) for each code chunk.

The shortcut key for “<-” is ALT + [dash] CTRL + ENTER will run the highlighted line(s) of code. The hashtag is code that is commented out.

# 1. Find/Set the working directory, where you’ll import/export files.

getwd()

## [1] "C:/Users/jmckelvy/OneDrive - SHIFT NC/Documents/AYC/ayc/2021-02-08 TA Log"

# Wherever this code is saved, the file(s) for your dataset(s) should be there too

# 2. Name & import your data frame, or set of information (like a .csv file of survey responses), as factor variables.

* I created a dataframe object (which I named “raw.df”) for each dataset that will read in the respective .csv files and convert categorical variables into factor variables. (Copy and paste the name of the file. There’s a hidden character in the SurveyMonkey exports.)

rawio.df <- read.csv("AYC TA Log 2020-2023.csv", stringsAsFactors = TRUE)  
rawcl.df <- read.csv("AYC QI TTA Log 2020-2023.csv", stringsAsFactors = TRUE)   
  
# Export .CSV from your survey platform and confirm numeric values, NOT choice text.   
  
# Clean variable names in advance (e.g., insert a row of Q#s). Do NOT start variable names with numbers.

# 3. Clean the analytic dataset.

* I created an analytic dataset for clinics (which I named “tacl.df”) and one for IOs (which I named “taio.df”) that will subset the data columns and remove informational header rows that are not survey responses.

(**1-based index**: R starts with the first row of obs. being “1”; 3:nrow means that your responses/obs. start on row 4 until the Nth row. Check your dataset to see which row the actual observations start.)

tacl.df <- rawcl.df[2:nrow(rawcl.df),c(10,12:19)]  
names(tacl.df) <- c("Trainer/TA Provider",  
 "Training/TA",  
 "Date of T/TA",  
 "Minutes of T/TA",  
 "Mode of T/TA",  
 "B.Peds",  
 "BCHC",  
 "Charles Drew HC",  
 "ACHD"  
 #col. 21-34 are check-all-that-apply best practices  
 #col. 35-43 are check-all-that-apply training topics  
 )  
  
taio.df <- rawio.df[2:nrow(rawio.df),c(10,12:28,63:72)]   
names(taio.df) <- c("Trainer/TA Provider",  
 "Training/TA",  
 "Date of T/TA",  
 "Minutes of T/TA",  
 "Mode of T/TA",  
 "Alamance Achieves",  
 "ACHD",  
 "CGDC",  
 "CrossRoads",  
 "ACDSS",  
 "CHS",  
 "PAYC",  
 "Elon",  
 "EC",  
 "FAS",  
 "SABGC",  
 "Partners Present",  
 "Other Adults Present",  
 #col. 31-62 were the old Check-all-that-apply GTO steps  
 "GTO-01",  
 "GTO-02",  
 "GTO-03",  
 "GTO-04",  
 "GTO-05",  
 "GTO-06",  
 "GTO-07",  
 "GTO-08",  
 "GTO-09",  
 "GTO-10"  
 #col. 73-96 are check-all-that-apply training topics  
 )  
  
# Confirm the number of obs on SurveyMonkey.

* Recode missing data:

tacl.df[tacl.df==""] <- NA  
tacl.df[tacl.df=="N/A"] <- NA  
  
taio.df[taio.df==""] <- NA  
taio.df[taio.df=="N/A"] <- NA

# 3a. Clean the nominal (categorical, dichotomous/MRdum, ordinal/Likert) variables.

* For each multiple choice statement, change the response values to factor, specify levels, and rename the value labels for each level, based on the survey/codebook: (open both the dataset and the codebook/survey with question numbers and recode values to make sure the labels and levels are correct)

labelspec <- c("Lisa Garland",  
 "Caro Welker",  
 "Josephine McKelvy",  
 "Sophia Durant",  
 "Tommy White",  
 "Danielle Sherman",  
 "Other")  
labeltta <- c("Training",  
 "Technical Assistance")  
labelmode <- c("Phone",  
 "Web-based for one organization",  
 "In-person for one organization",  
 "In-person for multiple organizations",  
 "Web-based for multiple organizations")  
  
tacl.df$`Trainer/TA Provider` <- factor(tacl.df$`Trainer/TA Provider`,  
 levels = c(1),  
 labels = c("Sophia Durant"))  
taio.df$`Trainer/TA Provider` <- factor(taio.df$`Trainer/TA Provider`,  
 levels = c(1,2,3,4,5,6,7),  
 labels = labelspec)  
  
tacl.df$`Training/TA` <- factor(tacl.df$`Training/TA`,  
 levels = c(1,2),  
 labels = labeltta)  
taio.df$`Training/TA` <- factor(taio.df$`Training/TA`,  
 levels = c(1,2),  
 labels = labeltta)  
  
tacl.df$`Mode of T/TA` <- factor(tacl.df$`Mode of T/TA`,  
 levels = c(1,2,3,4,5),  
 labels = labelmode)  
taio.df$`Mode of T/TA` <- factor(taio.df$`Mode of T/TA`,  
 levels = c(1,2,3,4,5),  
 labels = labelmode)

* Create an object (called “cleanIt”) that performs a function to a vector, which serves as a placeholder for something like a column. In that function, convert the vector to a character variable. You have to convert this factor variable to a character variable first because you may see two types of labels (e.g., “1” and “NA”), but there could be more than 2 hidden levels in your factor variable. Where the variable is “NA” or a blank, replace with zero (0). Then convert the vector to a numeric variable that can be summed.

cleanIt <- function(vec){  
 chars <- as.character(vec)  
 chars[is.na(chars)] <- "0"  
 chars[chars==""] <- "0"  
 chars[chars!="0"] <- "1"  
 return(as.numeric(chars))  
}

* There is a multiple-response item/column with varying numbers of options for each clinic partner (i.e., columns 6-9 of the tacl.df) and each implementing organization (IO) (i.e., columns 7-17 of the taio.df). Use list apply (lapply) to repeat the “cleanIt” function to those columns (i.e., your vectors).

tacl.df[6:9] <- lapply(tacl.df[6:9], cleanIt) # clinic partners  
taio.df[6:16] <- lapply(taio.df[6:16], cleanIt) # IOs  
taio.df[19:28] <- lapply(taio.df[19:28], cleanIt) # GTO steps  
#ta.df[62:83] <- lapply(ta.df[62:83], cleanIt) # training topics for the old dataframe

* Create a categorical variable (i.e., mainio) in a dataframe (i.e., taio.df) out of the check-all-that-apply (multiple-response dummies) for IOs (i.e, columns 6 thru 16) across the row/case (i.e., margin of 1). Return the name of the column (i.e., the IO). If more than one IO applied (or the sum of the check-all-that-apply > 1), return “Multiple IOs”. Name the variable/IO itself IF the variable was not zero (or “No” to any of those that could apply).

taio.df$mainio <- apply(taio.df[6:16], 1, function(x) {ifelse(sum(x) > 1, "Multiple IOs", names(x[x != 0]))})  
  
#If only one IO applied (e.g., 1:1 TA), create a variable (i.e., setting) that re-categorizes the IO into its setting served: Clinic, Foster Care, Community, Higher Ed, or Multiple IOs (probably training). Then factor that variable.  
taio.df$setting <- ifelse(taio.df$mainio %in% c("ACHD"), 1, #ACHD = Clinic  
 ifelse(taio.df$mainio %in% c("ACDSS"), 2, #ACDSS = Foster Care  
 ifelse(taio.df$mainio %in% c("CHS","CGDC","CrossRoads","PAYC","Exchange","FAS","SABGC"), 3, #CHS, CGDC, CrossRoads, PAYC, Exchange, FAS, SABGC = Community  
 ifelse(taio.df$mainio %in% c("Elon"), 4, #Elon = Higher Ed  
 ifelse(taio.df$mainio %in% c("Multiple IOs"), 5, NA)))))  
  
taio.df$setting <- factor(taio.df$setting,  
 levels = c(1,2,3,4,5),  
 labels = c("Clinic TA","Foster Care TA","Community TA","Higher Ed TA","Multi-setting (training?)"))  
  
#mertsetting <- data.frame(table(ta.df$setting))  
#names(mertsetting) <- c("Setting","Count") #counted logs submitted  
  
#library(sjPlot)  
#tab\_df(mertsetting,  
# title = "TTA logs for AYC",  
# file = "MERT - TTA Logs Table.doc")  
#tab\_xtab(var.row = ta.df$setting, var.col = ta.df$Mode.of.Training.or.TA,  
# title = "TTA logs by mode",  
# file = "MERT - TTA Mode Table.doc")

# 3b. Convert column/variable to “date” class (%m = 2-digit month; %d = 2-digit day; %[capital]Y = 4-digit year)(<https://www.statmethods.net/input/dates.html>; <https://www.statology.org/subset-by-date-range-in-r/>)

tacl.df$`Date of T/TA` <- as.Date(tacl.df$`Date of T/TA`,"%m/%d/%Y")  
taio.df$`Date of T/TA` <- as.Date(taio.df$`Date of T/TA`,"%m/%d/%Y")

# 3c. Convert continuous variables from factor to numeric.

(Factor variables are stored as integer codes–not numeric value–to create levels (e.g., responses ranging from 10 to 12 are three levels). Skipping “as.character” will save the level, not the label, as the value. (<https://stackoverflow.com/questions/6328771/changing-values-when-converting-column-type-to-numeric>; <https://www.geeksforgeeks.org/convert-factor-to-numeric-and-numeric-to-factor-in-r-programming/>) Convert to character and then numeric.)

tacl.df$`Minutes of T/TA` <- as.numeric(as.character(tacl.df$`Minutes of T/TA`))  
taio.df$`Minutes of T/TA` <- as.numeric(as.character(taio.df$`Minutes of T/TA`))  
  
tacl.df$Hours <- tacl.df$`Minutes of T/TA`/60  
taio.df$Hours <- taio.df$`Minutes of T/TA`/60  
  
taio.df$`Partners Present` <- as.numeric(as.character(taio.df$`Partners Present`))  
  
taio.df$GTO1hrs <- taio.df$`GTO-01` \* taio.df$Hours  
taio.df$GTO2hrs <- taio.df$`GTO-02` \* taio.df$Hours  
taio.df$GTO3hrs <- taio.df$`GTO-03` \* taio.df$Hours  
taio.df$GTO4hrs <- taio.df$`GTO-04` \* taio.df$Hours  
taio.df$GTO5hrs <- taio.df$`GTO-05` \* taio.df$Hours  
taio.df$GTO6hrs <- taio.df$`GTO-06` \* taio.df$Hours  
taio.df$GTO7hrs <- taio.df$`GTO-07` \* taio.df$Hours  
taio.df$GTO8hrs <- taio.df$`GTO-08` \* taio.df$Hours  
taio.df$GTO9hrs <- taio.df$`GTO-09` \* taio.df$Hours  
taio.df$GTO10hrs <- taio.df$`GTO-10` \* taio.df$Hours

* Combine QI & EBP TA Logs

tacl1.df <- tacl.df[,c(1:4,10)]  
names(tacl1.df) <- c("Trainer/TA Provider",  
 "Training/TA",  
 "Date of T/TA",  
 "Minutes of T/TA",  
 "Hours")  
  
taio1.df <- taio.df[,c(1:4,31)]   
names(taio1.df) <- c("Trainer/TA Provider",  
 "Training/TA",  
 "Date of T/TA",  
 "Minutes of T/TA",  
 "Hours")  
  
#columns need to match to bind two datasets  
ta.df <- rbind(tacl1.df,taio1.df)

# 4. Subset data by date ranges (e.g., quarters)

Update begin/finish dates & ggsave file names below

begin <- "2020-07-01" #options: "2020-07-01", "2021-01-01", "2021-07-01", "2022-01-01", "2022-07-01", "2023-01-01"  
finish <- "2021-06-30" #options: "2020-12-31", "2021-06-30", "2021-12-31", "2022-06-30", "2022-12-31", "2023-06-30"  
  
tasapr <- ta.df[ta.df$`Date of T/TA` >= begin & ta.df$`Date of T/TA`<= finish,]  
tamert <- taio.df[taio.df$`Date of T/TA` >= begin & taio.df$`Date of T/TA` <= finish,]  
  
# can create specific dataframes (below) but will need to update the dataframe 6 times for the tabulations  
#year <- ta.df[ta.df$Date.of.Training.or.TA >="2020-07-01" & ta.df$Date.of.Training.or.TA <="2021-06-30"]  
#qtr1 <- ta.df[ta.df$Date.of.Training.or.TA >="2021-07-01" & ta.df$Date.of.Training.or.TA <="2021-09-30",]  
#qtr2 <- ta.df[ta.df$Date.of.Training.or.TA >="2021-10-01" & ta.df$Date.of.Training.or.TA <="2021-12-31",]  
#qtr3 <- ta.df[ta.df$Date.of.Training.or.TA >="2022-01-01" & ta.df$Date.of.Training.or.TA <="2022-03-31",]  
#qtr4 <- ta.df[ta.df$Date.of.Training.or.TA >="2022-04-01" & ta.df$Date.of.Training.or.TA <="2022-06-30",]

* How many adults (from IOs, CAG) were trained this quarter (by T vs TA)?

#ta.df$`Partners Trained` <- as.numeric(as.character(ta.df$`Partners Trained`))  
#sum(ta.df$`Partners Trained`) #will contain duplicates and across time; see M:\...\Facilitator Roster instead  
  
# https://www.datasciencemadesimple.com/sum-function-in-r/  
aggregate(x = tamert$`Partners Present`,  
 by= list(tamert$`Training/TA`),  
 FUN=sum) #will contain duplicates; see M:\...\Facilitator Roster instead

## Group.1 x  
## 1 Training 161  
## 2 Technical Assistance 92

* How many hours of T/TA were provided this reporting period? (for SAPR/AER) by setting? (for MERT)

#new <- ta.df[,c(2:3,31:32)] #check that the reinterpretation of TA vs Training is accurate  
  
hrsbytype <- aggregate(x=tasapr$Hours,  
 by= list(tasapr$`Training/TA`),  
 FUN=sum) #more accurate (SAPR/AER)  
names(hrsbytype) <- c("Type","Total Hours")  
  
tabyset<- aggregate(x=tamert$Hours,  
 by= list(tamert$setting),  
 FUN=sum) #TA by setting (MERT); need to add clinics TA  
names(tabyset) <- c("Setting","TA Hours")  
  
#Optional:  
tabyio <- aggregate(x = tamert$Hours,  
 by= list(tamert$mainio),  
 FUN=sum) #TA by IO (AER)  
names(tabyio) <- c("IO","TA Hours")

* Updated GTO

gto <- tamert[,c(32:41)]  
gtotot <- colSums (gto, na.rm = FALSE, dims = 1)  
View(gtotot)