HSAPS Data Cleaning Documentation

2024-10-18

The cleaning process begins with importing the most recent data set. It is located in "//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports to date", with the file name "^hsaps_cleaned_YYYY-MM-DD.xlsx". This data set would also be referred as "existing data(set)" or "older data(set)" from now on in the document. Descriptions of all the data sets mentioned in this documentation can be found here as well: Home Safe Data Processing and Cleanup

Gate Keeping

Key variables used

County (Reporting_Agency), First Name, Last Name, Date of Birth, Case Start Date

Background

Initial concept of gate keeping code was to be conservative towards the new data while assuming that the existing (i.e older) data has the most accurate information. Therefore, we have been focusing on flagging and excluding new data rows.

However, it turned out that initially compiled quarterly reports (from the beginning of the program till FY 22-23 Q3) already had accumulated data issues, and it was impossible to reach out and rely on grantees to help us with every single issue. Moreover, we realized grantees update their county data (including some key variable information) in their most recent quarterly report, colliding with our initial assumption that the existing data would be more accurate. Due to this nature, even though we performed data investigation a few times (massive outreach to grantees to help us figure out which information is correct between older and newer information of a participant), we realized that the nature of the problem would not go away as long as the data continues to flow in. Since we reached the conclusion that it would not be sustainable to keep reaching out to the grantees on a regular basis, we decided to use gate keeping code to not only flag the issues from the newest data, but also address some data issues from the existing data set.

Code Description: Functions

The function, **hsaps_gate_keeping**, was generated to perform the task described above (flagging the issues from the newest data as well as addressing data issues from the existing data). It begins with importing the most recent compiled quarterly report (from RADD) and formatting the imported data set by defining the column types, cleaning the names of the variables, and standardizing the key variables. This was done through the function called **hsaps_exporting_data_with_proper_col_types**.

```
hsaps_exporting_data_with_proper_col_types = function(dt_name,report_date){
  library(readxl)
```

```
  library(stringr)
  cat("Processing the data for sheet:", dt_name, "\n")
  dt = read_excel(paste0("//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports to date/",dt_name),sheet = "Data")
  coltype = as.character(dt[1,])
  coltype = tolower(coltype)
  coltype = gsub("new","text",coltype)
  coltype = gsub("4 digit ","",coltype)
  coltype = ifelse(is.na(coltype),"text",ifelse(grepl("integer",coltype),"numeric",coltype))
  dt_new = read_excel(paste0("//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports to date/",dt_name),sheet = "Data",col_types=coltype)[-1,]
  names(dt_new)[1:2] = as.character(dt[2,1:2])
  dt_new = dt_new[-1,]
  dt_new = data.frame(dt_new,check.names = F)
  names(dt_new) = gsub("^Item [0-9]+:\r\n","",names(dt_new))
  names(dt_new) = gsub("\r\n","",names(dt_new))
  names(dt_new) = gsub("^Item [0-9]+: ","",names(dt_new))
  names(dt_new) = gsub(" \\(optional\\)$","",names(dt_new))
  names(dt_new) = gsub("^Home Safe ","",names(dt_new)) #to match with var name "Case Start Date"
  names(dt_new) = gsub("^HSAPS ","",names(dt_new))
  names(dt_new) = gsub("Home Safe ","",names(dt_new))
  names(dt_new) = gsub("\\?$","",names(dt_new))
  names(dt_new) = gsub("Home Safe$","HSAPS",names(dt_new))
  names(dt_new) = gsub(" to HSAPS$","",names(dt_new))
  names(dt_new) = gsub("-up","-Up",names(dt_new))
  names(dt_new) = gsub("  -"," -",names(dt_new))
  names(dt_new) = gsub("/\\s","/",names(dt_new))
  names(dt_new) = gsub("\\s\\s","\\s",names(dt_new))
  names(dt_new) = gsub("by Other-","by Other -",names(dt_new))
  names(dt_new) = gsub("from be","from Be",names(dt_new))
  names(dt_new) = gsub("Reported Incident","Report",names(dt_new))
  dt_new$Report_Month = report_date
  dt_new$Reporting_Agency = str_to_title(dt_new$Reporting_Agency)
  dt_new$`First Name` = str_to_title(dt_new$`First Name`)
  dt_new$`Last Name` = str_to_title(dt_new$`Last Name`)
  return(dt_new)
}
```

Then, the code formats a few more date variables to make it consistent and clean. The gate_keeping function code from importing the compiled data to formatting additional variables is shown below:

```
cat("Running the gate-keeping function...\n")
new_dt = hsaps_exporting_data_with_proper_col_types(new_dt_name,new_dt_quarter)
new_dt = new_dt[!is.na(new_dt$Reporting_Agency),]
```

```
cat("Formatting date fields...\n")
new_dt$`Date of Birth` = gsub(" [0-9]+:[0-9]+:[0-9]+$","",as.character(new_dt
$`Date of Birth`))
new_dt$`Case Start Date` = gsub(" [0-9]+:[0-9]+:[0-9]+$","",as.character(new_
dt$`Case Start Date`))
new_dt$`Case Closure Date` = gsub(" [A-Z,a-z]+$","",as.character(new_dt$`Case
 Closure Date`))
new_dt$`APS Report Date` = gsub(" [0-9]+:[0-9]+:[0-9]+$","",as.character(new_
dt$`APS Report Date`))
```

After all the initial formatting, the function performs two main checks. The descriptions and decisions made for each part are following:

1. The function goes through each row of the existing data without case start date and looks for the key variable information match (county, first name, last name, date of birth) to the newest compiled data. This is to identify any data rows where the case start dates in the existing data set are missing, but appeared in the new data set.

```
  #part 1: sorting through cases btw no case start dates to yes case star
  t dates
  cat("Sorting through cases with updated Project Start Dates...\n")
  csd_updated = c()
  dt_no_csd = dt[is.na(dt$`Case Start Date`),]
  dt_yes_csd = dt[!is.na(dt$`Case Start Date`),]
  cat("Checking for matches with missing Project Start Dates...\n")
  for(i in 1:nrow(dt_no_csd)){
    match = which(tolower(paste0(new_dt$Reporting_Agency,new_dt$`Last Nam
  e`,new_dt$`First Name`,new_dt$`Date of Birth`)) %in% tolower(paste0(dt_
  no_csd$Reporting_Agency[i],dt_no_csd$`Last Name`[i],dt_no_csd$`First Na
  me`[i],dt_no_csd$`Date of Birth`[i])))
    if(length(match)==0){
      #new participant without case start date!
      csd_updated = c(csd_updated,i)
    }
    else if(length(match)>1){
      if(dt_no_csd$Report_Month[i]>max(new_dt$Report_Month[match])
         & !is.na(dt_no_csd$`Case Closure Date`[i])
         & all(is.na(new_dt$`Case Closure Date`[match]))){
        csd_updated = c(csd_updated,i)
      }
    }
    else{
      if(dt_no_csd$Report_Month[i]>new_dt$Report_Month[match]
         & !is.na(dt_no_csd$`Case Closure Date`[i])
         & is.na(new_dt$`Case Closure Date`[match])){
        csd_updated = c(csd_updated,i)
      }
    }
  }
```

```
if(length(index_to_include)>0){
  cat("Updating the dataset with cases having missing Project Start Dat
es...\n")
  dt_no_csd_removed = rbind(dt_yes_csd,dt_no_csd[csd_updated,])
}
```

- When there is no key variable information match between the existing and the new data ("length(match)==0"), this either means the participant is brand new, or one or more key variable information (other than case start date) of the same participant have changed.
  - Since it is very rare as well as hard to clearly determine with generalized algorithm whether the participants are the same when two or more key variables differ, we decided to include this data regardless of whether the participant is brand new or not. Until we come up with a better method, we will manually eliminate the older data rows where two or more key variables were updated in the newer data.
- If there is at least one match of key variable information between the existing and the new data, this means the case start date was updated in the new the data. Therefore, the data row without case start date from existing data set should be removed, in order to be replaced with the new data.

This first check ends with an updated existing data set where older data rows without case start date was removed when there are newer data rows with updated case start dates.

2. The next main check is to determine if any new cases are the same as the existing ones even when they have different case start dates. The function goes through the each row of the newest data set and looks for the key variable information match (county, first name, last name, date of birth) to the existing data set. This is to compare cases from the newest data with the existing one.

```
#part 2: gate keeping
cat("Performing gate-keeping checks on the new dataset...\n")
match_to_exclude = c()
index_to_include = c()
index_to_exclude = c()
for(i in 1:nrow(new_dt)){
  match_target = tolower(paste0(new_dt$Reporting_Agency[i],new_dt$`Firs
t Name`[i],new_dt$`Last Name`[i],new_dt$`Date of Birth`[i]))
  match = which(tolower(paste0(dt_no_csd_removed$Reporting_Agency,dt_no
_csd_removed$`First Name`,dt_no_csd_removed$`Last Name`,dt_no_csd_remov
ed$`Date of Birth`))==match_target)

  #if there is no match
  if(length(match)==0){
    #sorting through cases with updated dob's (either Jan 1st to an upd
ated date or no date to an updated date)
```

```
    match_wo_dob = which(tolower(paste0(dt_no_csd_removed$Reporting_Age
ncy,dt_no_csd_removed$`First Name`,dt_no_csd_removed$`Last Name`,dt_no_
csd_removed$`Case Start Date`))==tolower(paste0(new_dt$Reporting_Agency
[i],new_dt$`First Name`[i],new_dt$`Last Name`[i],new_dt$`Case Start Dat
e`[i])))
    #if there is match_wo_dob but no match
    if(length(match_wo_dob)>0){
      #before doing so, however, need to check if dob was updated
      for(m in 1:length(match_wo_dob)){
        if((!grepl("01-01$",new_dt$`Date of Birth`[i]) & grepl("01-01$"
,dt_no_csd_removed$`Date of Birth`[match_wo_dob[m]]))
           | (is.na(dt_no_csd_removed$`Date of Birth`[match_wo_dob[m]])
 & !is.na(new_dt$`Date of Birth`[i]))){
          dt_no_csd_removed$`Date of Birth`[match_wo_dob[m]]=new_dt$`Da
te of Birth`[i]
        }
      }
    }
  }

  #after updating DOB, let's see if there is a match
  match = which(tolower(paste0(dt_no_csd_removed$Reporting_Agency,dt_no
_csd_removed$`First Name`,dt_no_csd_removed$`Last Name`,dt_no_csd_remov
ed$`Date of Birth`))==match_target)
  #if there still isn't a match,
  if(length(match)==0){
    #need to check if there is typo
    if((tolower(paste0(new_dt$Reporting_Agency[i],new_dt$`First Name`[i
],new_dt$`Date of Birth`[i])) %in%
        tolower(paste0(dt_no_csd_removed$Reporting_Agency,dt_no_csd_rem
oved$`First Name`,dt_no_csd_removed$`Date of Birth`)))
       | (tolower(paste0(new_dt$Reporting_Agency[i],new_dt$`Date of Bir
th`[i],new_dt$`Last Name`[i])) %in%
          tolower(paste0(dt_no_csd_removed$Reporting_Agency,dt_no_csd_r
emoved$`Date of Birth`,dt_no_csd_removed$`Last Name`)))
       | (tolower(paste0(new_dt$Reporting_Agency[i],new_dt$`First Name`
[i],new_dt$`Last Name`[i])) %in%
          tolower(paste0(dt_no_csd_removed$Reporting_Agency,dt_no_csd_r
emoved$`First Name`,dt_no_csd_removed$`Last Name`)))){
      index_to_exclude = c(index_to_exclude,i)
    }
    #include new participants!
    if(!(i %in% index_to_exclude)){
      index_to_include = c(index_to_include,i)
    }
  }

  #if there is at least one match btw the newest report and the clean d
ata set
  else if(length(match)>0){
```

```
    for(m in 1:length(match)){
      #check if the cases are the same even with different case start d
ates -> remove the older row
      if((!is.na(new_dt$`Case Start Date`[i])
          & !is.na(dt_no_csd_removed$`Case Closure Date`[match[m]])
          & !is.na(new_dt$`Case Closure Date`[i])
          & dt_no_csd_removed$`Case Start Date`[match[m]]!=new_dt$`Case
 Start Date`[i]
          & dt_no_csd_removed$`Case Closure Date`[match[m]]==new_dt$`Ca
se Closure Date`[i])
         | (!is.na(new_dt$`Case Start Date`[i])
            & !is.na(dt_no_csd_removed$`APS Report Date`[match[m]])
            & !is.na(new_dt$`APS Report Date`[i])
            & dt_no_csd_removed$`Case Start Date`[match[m]]!=new_dt$`Ca
se Start Date`[i]
            & dt_no_csd_removed$`APS Report Date`[match[m]]==new_dt$`AP
S Report Date`[i])
         | (!is.na(new_dt$`Case Start Date`[i])
            & dt_no_csd_removed$`Case Start Date`[match[m]]!=new_dt$`Ca
se Start Date`[i]
            & !is.na(dt_no_csd_removed$`Case Closure Date`[match[m]])
            & new_dt$`Case Start Date`[i]<dt_no_csd_removed$`Case Closu
re Date`[match[m]])
         | (!is.na(new_dt$`Case Start Date`[i])
            & dt_no_csd_removed$`Case Start Date`[match[m]]!=new_dt$`Ca
se Start Date`[i]
            & is.na(dt_no_csd_removed$`Case Closure Date`[match[m]])
            & !is.na(new_dt$`Case Closure Date`[i]))){
        # case_start_date_disc = c(case_start_date_disc,i)
        match_to_exclude = c(match_to_exclude,match[m])
      }
    }
    index_to_include = c(index_to_include,i)
  }
}
```

- If there is no key variable match between the existing and the new data (“length(match)==0”), before concluding that the participant is brand new, we need to check if there is any typos or updates in any of the key variables from the new data set.
  - For DOB, if the DOB response from the existing data is missing or January 1st (default, space-filling response), while the newer DOB is something else, replace the DOB. If the DOB response from the existing data is something else other than January 1st, flag the new DOB as a key information discrepancy.
  - As for other key variables (county, first name, last name), compare matching combinations of 3 out of 4 variables (county, first name, last name, dob) and if the match exists, it indicates the one variable not used for matching should be

flagged for possible typos. index_to_exclude() vector keeps track the indices of the new data set that fall into this condition.

- If there is at least one key variable set match, we compare the newest case to each of the corresponding existing ones to determine whether those two cases are the same cases even with different case start dates. Below is the list of conditions we came up to conclude that two cases (with the same set of key variable information) are the same. match_to_exclude() vector keeps track the indices of the existing data set that match with one of these conditions.
  - When two cases have different non-missing case start dates while having the same non-missing case closure date
  - When two cases have different non-missing case start dates while having the same non-missing APS report date
  - When two cases have different non-missing case start dates, but the case start date of the newer case is before the case closure date of the older case
  - When two cases have different non-missing case start dates, but the case closure date of the older case is missing while it isn't in the newer case

After identifying rows to include and flag, final data sets are consolidated.

```
cat("Finalizing included and excluded datasets...\n")
if(length(match_to_exclude)>0){
  dt_no_csd_removed = dt_no_csd_removed[-match_to_exclude,]
}
new_dt_include = new_dt[unique(index_to_include),]
new_dt_exclude = new_dt[unique(index_to_exclude),]
# new_dt_case_start_date_disc = new_dt[unique(case_start_date_disc),]
return(list(dt_no_csd_removed,new_dt_include,new_dt_exclude))
```

1. data rows from the existing data set were removed if match_to_exclude() has more than one indices.
2. Data rows to be included and excluded were separated into two data tables, "new_dt_include" and "new_dt_exclude".
3. The function outputs the list of these three tables.

We also recently identified cases where participants were prematurely exited and then enrolled back within the quarter. This can be either data entry or program management issues. We will most likely flag those cases soon.

Code Description: Cleaning Script

Since FY 22-23 Q4, HSAPS Cleaning Script uses the gate_keeping() function. (As "Background" section above also briefly mentioned, all the monthly/quarterly Home Safe reports up to FY 22-23 Q3 were compiled and processed as the very initial cleaned data set.) The code within the cleaning script is shown below:

```
##post FY22-23 Q3 compiling
#make sure to change the dataset to the most recent one
```

```
hsaps_backup = read.xlsx("//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports to date/hsaps_cleaned_2024-07-29.xlsx",sep.names = " ")
#please make sure to use the most recent quarterly report
hsaps_new_data = hsaps_gate_keeping(hsaps_backup,"HSAPS FY23-24 2024-8-23.xlsm","2024-04-01")
hsaps_backup = hsaps_new_data[[1]]
hsaps_new_data_include = hsaps_new_data[[2]]
hsaps_new_data_exclude = hsaps_new_data[[3]]
hsaps = unique(rbind.fill(hsaps_backup,hsaps_new_data_include))
```

- **hsaps_backup**: the most recent version of cleaned data set
- After applying gate_keeping function, each output table from the function was redefined/assigned:
  - **hsaps_backup**: updated existing data set after the data rows needed to
  - **hsaps_new_data_include**: list of data rows from the newest report that need to be included to the existing data set
  - **hsaps_new_data_exclude**: list of data rows from the newest report that should NOT be included to the existing data set as it possibly contains typos from the key variable information
  - **hsaps**: combined data set from hsaps_backup and hsaps_new_data_include

Formatting Prior to De-duplication

Code Description: Cleaning Script

After importing the data sets, the cleaning script then formats the data before de-duplicating it.

1. Changing all the date values to “YYYY-MM-DD” form (as some were “YYYY-MM-DD HH:MM:SS” form)

```
#fix all the date vars to format "yyyy-mm-dd"
for(j in grep("date",tolower(names(hsaps)))){
  if(length(grep(" [0-9]+:[0-9]+:[0-9]+.*",hsaps[,j]))>0){
    hsaps[,j] = gsub(" [0-9]+:[0-9]+:[0-9]+.*","",hsaps[,j])
  }
  hsaps[,j] = as.character(as.Date(hsaps[,j]))
}
```

2. Splitting the data set into two, missing and non-missing case start dates, to deal with data deduplication process separately

```
#flagging the data with no case start date
hsaps_no_start_date = hsaps[which(is.na(hsaps$`Case Start Date`)),]
hsaps = hsaps[which(!is.na(hsaps$`Case Start Date`)),]
```

De-duplication

Pt 1: Data without Case Start Date

Key variable used

County (Reporting_Agency), First Name, Last Name, Report_Month

Background/Decisions

Since this data (**hsaps_no_start_date**) does not have case start dates, it won't be counted towards most of the reporting numbers that involve time period such as # of enrolled cases. However, since intervention reporting numbers are based on intervention dates, not case start dates, we decided to include in the final data set if the data row has any intervention information.

Code Description: Cleaning Script

```
#dealing with the dataset without case start date
hsaps_no_start_date_duplicate = hsaps_no_start_date[(duplicated(hsaps_no_star
t_date[c("Reporting_Agency","Report_Month","First Name","Last Name")])
& duplicated(hsaps_no_start_date[c("Reporting_Agency","Report_Month","First N
ame","Last Name")],fromlast=T)
& duplicated(hsaps_no_start_date[c("Reporting_Agency","Report_Month","First N
ame","Last Name")],fromlast=T)
& duplicated(hsaps_no_start_date[c("Reporting_Agency","Report_Month","First N
ame","Last Name")],fromlast=T)),c("Reporting_Agency","Report_Month","First Na
me","Last Name")]

#getting the list of duplicate pairs without case start date within the same
reporting month
hsaps_no_start_date_duplicate = unique(merge(hsaps_no_start_date_duplicate,hs
aps_no_start_date,all.x=T))

#removing the list of duplicate pairs without case start date within the same
 reporting period
hsaps_no_start_date = hsaps_duplicate_data_remove(hsaps_no_start_date,hsaps_n
o_start_date_duplicate)

#order the rest of the duplicate pairs by reporting month
hsaps_no_start_date = hsaps_no_start_date[order(hsaps_no_start_date$Report_Mo
nth,decreasing = T),]

#getting the most recent data
hsaps_no_start_date_unique = hsaps_no_start_date[!(duplicated(hsaps_no_start_
date[c("First Name","Last Name","Reporting_Agency")])
& duplicated(hsaps_no_start_date[c("First Name","Last Name","Reporting_Agency
")],fromlast=T)
```

```
& duplicated(hsaps_no_start_date[c("First Name","Last Name","Reporting_Agency
")],fromlast=T)),]
```

- When the data rows have the same county, first name, last name, AND Report_Month values: flag out as a duplicate (the list was stored in **hsaps_no_start_date_duplicate**)
- When the data rows have the same county, first name, last name values, but NOT Report_Month value: take the data with the most recent Report_Month value (the list was stored in **hsaps_no_start_date_unique**)

After de-duplicating the data, we checked each data row within hsaps_no_start_date_unique whether it has any intervention information.

```
#trying to see which participants have intervention data even without start d
ate -> include in the data pool
hsaps_no_start_date_unique$no_intervention = 0
for(i in 1:nrow(hsaps_no_start_date_unique)){
  hsaps_no_start_date_unique$no_intervention[i] = ifelse(all(is.na(hsaps_no_s
tart_date_unique[i,grep("Intervention [0-9]{1} - Type",names(hsaps_no_start_d
ate_unique))])) | all(tolower(hsaps_no_start_date_unique[i,grep("Intervention
 [0-9]{1} - Type",names(hsaps_no_start_date_unique))])=="no intervention"),1,
hsaps_no_start_date_unique$no_intervention[i])
}
```

Only the rows with the intervention information (no_intervention = 0) would be incorporated into to the final data set.

Pt 2: Data with Case Start Date

Key variable used

County (Reporting_Agency), First Name, Last Name, Date of Birth, Case Start Date, Report_Month

Background/Decisions

- Grantees continuously update the case information to the newest report, so we concluded that the case information from the newest report would be the most comprehensive.
- If there happen to be duplicate data within the same report (due to the reasons such as double data entries from grantees), manual sorting is necessary.

Code Description: Cleaning Script

De-duplication for the data with case start dates consists of two parts:

1. De-duplicating the data across different reporting quarters
2. De-duplicating the data within the same reporting quarter

```
hsaps$`Case Start Date` = gsub("\\.[0-9]+$","",hsaps$`Case Start Date`)
hsaps_duplicate_data = hsaps[(duplicated(hsaps[c("Reporting_Agency","First Na
```

```
me","Last Name","Date of Birth","Case Start Date")])
& duplicated(hsaps[c("Reporting_Agency","First Name","Last Name","Date of Bir
th","Case Start Date")],fromlast=T)
& duplicated(hsaps[c("Reporting_Agency","First Name","Last Name","Date of Bir
th","Case Start Date")],fromlast=T)
& duplicated(hsaps[c("Reporting_Agency","First Name","Last Name","Date of Bir
th","Case Start Date")],fromlast=T)
& duplicated(hsaps[c("Reporting_Agency","First Name","Last Name","Date of Bir
th","Case Start Date")],fromlast=T)),c("Reporting_Agency","First Name","Last
Name","Date of Birth","Case Start Date")]

#finding all the data rows within hsaps that have the same first/last name, d
ob, and date pairs
hsaps_duplicate_data = unique(merge(hsaps_duplicate_data,hsaps,all.x=T))

#flaggihng all out
hsaps = hsaps_duplicate_data_remove(hsaps,hsaps_duplicate_data)

#determining which one to add back in -> the most recent rows
hsaps_duplicate_data_ordered = hsaps_duplicate_data[order(hsaps_duplicate_dat
a$Report_Month,decreasing = T),]

#duplicate data within the same reporting period
hsaps_duplicate_data_same_period = hsaps_duplicate_data_ordered[(duplicated(h
saps_duplicate_data_ordered[c("Report_Month","First Name","Last Name","Date o
f Birth","Case Start Date")])
& duplicated(hsaps_duplicate_data_ordered[c("Report_Month","First Name","Last
 Name","Date of Birth","Case Start Date")],fromlast=T)
& duplicated(hsaps_duplicate_data_ordered[c("Report_Month","First Name","Last
 Name","Date of Birth","Case Start Date")],fromlast=T)
& duplicated(hsaps_duplicate_data_ordered[c("Report_Month","First Name","Last
 Name","Date of Birth","Case Start Date")],fromlast=T)
& duplicated(hsaps_duplicate_data_ordered[c("Report_Month","First Name","Last
 Name","Date of Birth","Case Start Date")],fromlast=T)),c("Reporting_Agency",
"Report_Month","First Name","Last Name","Date of Birth","Case Start Date")]

#finding all the data rows within hsaps that have the same first/last name, d
ob, and date pairs
hsaps_duplicate_data_same_period = unique(merge(hsaps_duplicate_data_same_per
iod,hsaps_duplicate_data_ordered,all.x=T))

#removing duplicate data from the same reporting period
hsaps_duplicate_data_ordered = hsaps_duplicate_data_remove(hsaps_duplicate_da
ta_ordered,hsaps_duplicate_data_same_period)

#removing the "Remove" list from the duplicate data
hsaps_duplicate_data_same_period = hsaps_duplicate_data_same_period[-grep("re
move",tolower(hsaps_duplicate_data_same_period$`Remove Case`)),]
#de-duplicating to identify the data needed for manual sorting
```

```
hsaps_duplicate_sort_needed = hsaps_duplicate_data_same_period[(duplicated(hs
aps_duplicate_data_same_period[c("Reporting_Agency","First Name","Last Name",
"Date of Birth","Case Start Date")])
& duplicated(hsaps_duplicate_data_same_period[c("Reporting_Agency","First Nam
e","Last Name","Date of Birth","Case Start Date")],fromlast=T)
& duplicated(hsaps_duplicate_data_same_period[c("Reporting_Agency","First Nam
e","Last Name","Date of Birth","Case Start Date")],fromlast=T)
& duplicated(hsaps_duplicate_data_same_period[c("Reporting_Agency","First Nam
e","Last Name","Date of Birth","Case Start Date")],fromlast=T)
& duplicated(hsaps_duplicate_data_same_period[c("Reporting_Agency","First Nam
e","Last Name","Date of Birth","Case Start Date")],fromlast=T)),
c("Reporting_Agency","First Name","Last Name","Date of Birth","Case Start Dat
e")]
hsaps_duplicate_sort_needed = merge(hsaps_duplicate_sort_needed,hsaps_duplica
te_data_same_period,all.x=T)
#whatever row that was duplicated with "Remove" entry can be added back into
the final data set
hsaps_duplicate_sort_include = hsaps_duplicate_data_remove(hsaps_duplicate_da
ta_same_period,hsaps_duplicate_sort_needed)
#removing duplicate list where case information was continued
hsaps_duplicate_case_continued = unique(subset(read.xlsx("//cdss/feed/Central
 Office/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports
 to date/Cleaning In Progress/hsaps_duplicate_data_same_period_sorted.xlsx",s
heet="duplicates_case_continued",sep.names = " ")[,c(1,3:6)],!grepl("^Include
",Reporting_Agency)))
hsaps_duplicate_case_continued = subset(merge(hsaps_duplicate_case_continued,
hsaps_duplicate_sort_needed,all.x=T),Report_Month=="2024-04-01")
hsaps_duplicate_case_continued$case_continued = "Yes"
hsaps_duplicate_sort_needed = hsaps_duplicate_data_remove(hsaps_duplicate_sor
t_needed,hsaps_duplicate_case_continued)

if(nrow(hsaps_duplicate_sort_needed)!=0){
  write_xlsx(hsaps_duplicate_sort,"hsaps_duplicates_same_period_sort_needed.x
lsx")
}
```

- **hsaps_duplicate_data**: List of all the duplicate case data (based on counties, first names, last names, birth dates, and case start dates) across all reporting periods, except the duplicate pairs from the same reporting period
- **hsaps_duplicate_data_ordered**: hsaps_duplicate_data ordered by the reporting period (from the most recent quarter to the oldest)
- **hsaps_duplicate_data_same_period**: List of all the duplicate case data (based on counties, first names, last names, birth dates, and case start dates) from the SAME reporting period (indicated by Report_Month)
- **hsaps_sort_needed**: Subset list of duplicate cases from hsaps_duplicate_data_same_period that needed to be manually sorted

- **hsaps_duplicate_sort_include**: list of data rows that are not truly duplicated, but the data were duplicated with the rows with "Remove" responses in "Remove Case" element
- **hsaps_duplicate_case_continued**: list of data rows that are not truly duplicated, but the data are duplicated because grantees entered extra row(s) for continued intervention information

The code above describes the process below:

1. Creating hsaps_duplicate_data by filtering out all the duplicate data from hsaps data set (both across different reporting period and within the same reporting period) based on key variables listed above except Report_Month
2. Creating hsaps_duplicate_data_ordered by ordering data rows in hsaps_duplicate_data by Report_Month variable (from the most recent to the least recent)
3. Creating hsaps_duplicate_data_same_period by filtering out the duplicate data from hsaps_duplicate_data based on all key variables listed above (including Report_Month)
4. Creating hsaps_sort_needed by removing all the duplicate data pairs where at least one of the data row within the pair has "Remove" response in "Remove Case" variable
5. Removing all the duplicate data pairs where they are part of hsaps_duplicate_case_continued
6. Saving hsaps_sort data set in the local folder as "hsaps_duplicates_same_period_sort_needed.xlsx"

Manual Sorting for Duplicate Data within The Same Report

The process of manual sorting for hsaps_sort is below:

1. Go to the file directory, \cdssOfficeProgramsSafeand EvaluationReports to dateIn Progress, and open the file, "**hsaps_duplicate_data_same_period_sorted.xlsx**". The spreadsheet contains 5 sheets.
    - **duplicates_sort_needed**: all the list of duplicate data needed to be sorted
    - **duplicates_to_be_included**: subset list of data from duplicates_sort_needed that would need to be included in the final data set
    - **duplicates_case_continued**: list of data from duplicates_sort_needed that are not truly duplicated, but the data are duplicated because grantees entered extra row(s) for continued case information
    - **duplicates_tbd**: subset list of data from duplicates_sort_needed that we need reach out to counties for
2. In the "duplicates_sort_needed" sheet, on the very bottom row, label "Included MM/DD/YY" where MM/DD/YY as the date of manual sorting
3. Copy all the data rows from "hsaps_duplicates_same_period_sort_needed.xlsx" to "duplicates_sort_needed" sheet

4. Go through each duplicate pair and compare the info between the duplicate data rows to determine which row to include back into the final data set
    - If the rows within the duplicate pair mostly have similar/same information, copy (not cut) the row with more information to “duplicates_to_be_included” sheet
    - If the rows within the duplicate pair mostly have different information, copy both rows of the duplicate pair to “duplicates_tbd” sheet
    - If the rows within the duplicate pair mostly have similar/same information, while the intervention information seems to continue, copy both rows of the duplicate pair to “duplicates_case_continued” sheet
    - Whenever it is not fully clear which data within the duplicate pair to include, please put the data pair into “duplicates_tbd” sheet, to receive help from county workers.

Code Description: Cleaning Script - Continued

After sorting through the list in hsaps_sort manually (putting the rows into different sheets), the code continues with incorporating the de-duplicated data rows back into the original data set.

```
#after going through some list of duplicates within the same reporting period
, run below
hsaps_duplicate_data_same_period_include = read.xlsx("//cdss/feed/Central Off
ice/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports to
date/Cleaning In Progress/hsaps_duplicate_data_same_period_sorted.xlsx",sheet
="duplicates_to_be_included",sep.names = " ")[,1:79]
hsaps_duplicate_data_same_period_include = hsaps_duplicate_data_same_period_i
nclude[hsaps_duplicate_data_same_period_include$Report_Month=="2024-04-01",]
hsaps_duplicate_case_continued = subset(read.xlsx("//cdss/feed/Central Office
/HHCRB/HHB/Housing Programs/Home Safe/Data and Evaluation/Data Reports to dat
e/Cleaning In Progress/hsaps_duplicate_data_same_period_sorted.xlsx",sheet="d
uplicates_case_continued",sep.names = " ")[,1:79],!grepl("^Include",Reporting
_Agency)&grepl("2024-04-01",Report_Month))
hsaps_duplicate_case_continued$case_continued = "Yes"

#incorporating de-duplicated data from the same reporting period back to the
ordered duplicate data list
hsaps_duplicate_data_ordered = unique(rbind(hsaps_duplicate_data_ordered,hsap
s_duplicate_data_same_period_include,hsaps_duplicate_sort_include))
hsaps_duplicate_data_ordered = hsaps_duplicate_data_ordered[order(hsaps_dupli
cate_data_ordered$Report_Month,decreasing=T),]

#using the duplicated() functions again to filter out all the older data
hsaps_duplicate_data_include = hsaps_duplicate_data_ordered[!(duplicated(hsap
s_duplicate_data_ordered[c("First Name","Last Name","Date of Birth","Case Sta
rt Date")])
& duplicated(hsaps_duplicate_data_ordered[c("First Name","Last Name","Date of
 Birth","Case Start Date")],fromlast=T)
& duplicated(hsaps_duplicate_data_ordered[c("First Name","Last Name","Date of
 Birth","Case Start Date")],fromlast=T)
```

```
& duplicated(hsaps_duplicate_data_ordered[c("First Name","Last Name","Date of
 Birth","Case Start Date")],fromlast=T)),]

#including data rows
hsaps = unique(rbind(hsaps,hsaps_duplicate_data_include,hsaps_no_start_date_u
nique[hsaps_no_start_date_unique$no_intervention==0,-ncol(hsaps_no_start_date
_unique)]))
hsaps$case_continued = "No"
hsaps = rbind(hsaps,hsaps_duplicate_case_continued)
```

- **hsaps_duplicate_data_same_period_include**: list of de-duplicated data rows after manual sorting process
- **hsaps_duplicate_data_include**: list of de-duplicated data rows from hsaps_duplicate_data_ordered (free from all duplicate data)

The code above goes through the procedures below:

1. Creating hsaps_duplicate_data_same_period_include by importing the list of manually de-duplicated data
2. Creating hsaps_duplicate_case_continued by importing the list of manually sorted cases where the cases were continued
3. Incorporating the sorted duplicate data within the same period (hsaps_duplicate_data_same_period_include & hsaps_duplicate_sort_include) back to the hsaps_duplicate_data_sorted table
4. Creating hsaps_duplicate_data_include by de-duplicating the data across different reporting periods (from hsaps_duplicate_data_ordered)
5. Incorporating hsaps_duplicate_data_include (de-duplicated data with case start dates) and hsaps_no_start_date_unique (de-duplicated data without case start dates with at least one intervention information) back to hsaps data set
6. Incorporating hsaps_duplicate_case_continued after labeling all the rest of the data as non-continued cases

Further Clean Up and Flagging

After obtaining the final list of data rows, the data set goes through further clean-up and flagging process.

Code Description: Cleaning Script

The code first flagged and saved the list of data from hsaps data set that are older than two quarters. We are taking into account of late/missing submissions for the given quarter, and this is why we are giving two-quarter threshold.

```
#make sure to adjust the date below ("as.Date(date)") every quarter -> have a
t least two quarter discrepancy (to account for non-submission)
hsaps_noncumulative = subset(hsaps,as.Date(Report_Month)<as.Date('2023-10-01'
))[,c(grep("Reporting_",names(hsaps)),grep("Report_Mon",names(hsaps)),grep("
```

```
Name$",names(hsaps)),grep("of Birth$",names(hsaps)),grep("Case Start",names(h
saps)))]
hsaps_noncumulative = merge(hsaps_noncumulative,quarter[,c("Report_Month","Qu
arter")],all.x=T)
write_xlsx(hsaps_noncumulative,"//cdss/feed/Central Office/HHCRB/HHB/Housing
Programs/Home Safe/Data and Evaluation/Data Reports to date/Cleaning In Progr
ess/hsaps_dropped.xlsx")
```

Then, it removed the data rows where “Remove Case” element’s response says “remove” or “Remove”.

```
#removing entries with "Remove" mark
if(length(grep("remove",tolower(hsaps$`Remove Case`)))>0){
  hsaps = hsaps[-grep("remove",tolower(hsaps$`Remove Case`)),]
}
```

Next, it goes through data elements and clean up and/or consolidate responses.

```
hsaps$`Remove Case` = gsub("blank","",tolower(hsaps$`Remove Case`))
for(j in grep("Name$",names(hsaps))){
  hsaps[,j] = str_to_title(hsaps[,j])
}
for(j in c(grep("^Location of P",names(hsaps)):grep("^Living Situation Upon E
ntry",names(hsaps)),grep("^Client Homeless",names(hsaps)):grep("^Discharge fr
om",names(hsaps)),grep("Other - Financial",names(hsaps)):grep("^Previous APS"
,names(hsaps)),grep("CES",names(hsaps)),seq(grep("1 - Type$",names(hsaps)),gr
ep("6 - Type",names(hsaps)),by=4),grep("at Exit$",names(hsaps)),grep("up - me
thod",tolower(names(hsaps))),grep("Up - Living Situation$",names(hsaps)),grep
("Up - Homeless",names(hsaps)))){
  hsaps[,j] = tolower(hsaps[,j])
  hsaps[,j] = gsub("does not","doesn't",hsaps[,j])
  hsaps[,j] = gsub(" /","/",hsaps[,j])
  hsaps[,j] = gsub("/ ","/",hsaps[,j])
  hsaps[,j] = gsub(" and/or ","/",hsaps[,j])
  hsaps[,j] = gsub(", ","/",hsaps[,j])
  hsaps[,j] = gsub(" or ","/",hsaps[,j])
  hsaps[,j] = gsub("or ","",hsaps[,j])
  hsaps[,j] = gsub("hetere","hetero",hsaps[,j])
  hsaps[,j] = gsub("temporary -","temporary-",hsaps[,j])
  hsaps[,j] = gsub("permanent -","permanent-",hsaps[,j])
  hsaps[,j] = gsub("african-american","african american",hsaps[,j])
  hsaps[,j] = gsub("w/","with ",hsaps[,j])
  hsaps[,j] = gsub("\r\r\n","",hsaps[,j])
  hsaps[,j] = gsub("\r$","",hsaps[,j])
  hsaps[,j] = gsub("the last 3","the last three",hsaps[,j])
  hsaps[,j] = gsub("three years/longer","three years or longer",hsaps[,j])
  hsaps[,j] = gsub("prito","prior to",hsaps[,j])
  hsaps[,j] = gsub("prior home safe","prior to home safe",hsaps[,j])
  hsaps[,j] = gsub("–","-",hsaps[,j])
  hsaps[,j] = gsub("^relocation assistance$","relocation assistance/storage",
```

```
hsaps[,j])
  hsaps[,j] = gsub("ukno","unkno",hsaps[,j])
  hsaps[,j] = gsub("unko","unkno",hsaps[,j])
  hsaps[,j] = gsub("no homeless","not homeless",hsaps[,j])
  hsaps[,j] = gsub("^refused$","client refused",hsaps[,j])
  hsaps[,j] = gsub("yes-before","yes - before",hsaps[,j])
  hsaps[,j] = gsub("facilitiy","facility",hsaps[,j])
  hsaps[,j] = gsub("medical professional","medical personnel",hsaps[,j])
  hsaps[,j] = gsub("mental health professional","mental health personnel",hsa
ps[,j])
  hsaps[,j] = gsub("owners","owner",hsaps[,j])
  hsaps[,j] = gsub("not rights","no rights",hsaps[,j])
  hsaps[,j] = gsub("lives with","with",hsaps[,j])
  hsaps[,j] = gsub("paying ","",hsaps[,j])
  hsaps[,j] = gsub("not rent","no rent",hsaps[,j])
  hsaps[,j] = gsub("inervention","intervention",hsaps[,j])
  hsaps[,j] = gsub("verified- program","verified - program",hsaps[,j])
  hsaps[,j] = gsub(" \\[i.e./doubled up\\]","",hsaps[,j])
  hsaps[,j] = str_to_title(hsaps[,j])
  hsaps[,j] = gsub("Or","or",hsaps[,j])
}
```

ID and Case Generation

After further clean-up and flagging process, the code assigns ID's and case numbers to participants.

Code Description: Cleaning Script

```
#anonymizing the data through generating participant id's and case numbers
hsaps_participant = unique(hsaps[,c(grep("Reporting_Age",names(hsaps)),grep("
 Name$",names(hsaps)):grep("Date of B",names(hsaps)))])
hsaps_participant = unique(merge(hsaps_participant,hsaps_backup[,c(1:4,grep("
^id$",tolower(names(hsaps_backup))))],all.x=T))
hsaps_id_assign = unique(hsaps_participant[is.na(hsaps_participant$ID),1:4])
hsaps_id_assign$ID = as.character(sample(c(1000000:9999999)[-which(c(1000000:
9999999) %in% hsaps_participant$ID)],nrow(hsaps_id_assign)))
hsaps_participant = rbind(hsaps_participant[-which(paste0(hsaps_participant$R
eporting_Agency,hsaps_participant$`First Name`,hsaps_participant$`Last Name`,
hsaps_participant$`Date of Birth`)
%in% paste0(hsaps_id_assign$Reporting_Agency,hsaps_id_assign$`First Name`,hsa
ps_id_assign$`Last Name`,hsaps_id_assign$`Date of Birth`)),],hsaps_id_assign)
hsaps = merge(hsaps,hsaps_participant,all.x=T)

#assigning case numbers
hsaps = hsaps[order(hsaps$Reporting_Agency,hsaps$ID,hsaps$`Case Start Date`),
]
hsaps$case = ifelse(is.na(hsaps$`Case Start Date`),NA,1)
for(i in 2:nrow(hsaps)){
  if(!is.na(hsaps$case[i])){
```

```
    if(hsaps$ID[i]==hsaps$ID[i-1]){
      hsaps$case[i] = hsaps$case[i-1]+1
    }
  }
}
hsaps$case = as.character(hsaps$case)

#assigning 1 to new cases without case start date
hsaps_case = hsaps[is.na(hsaps$`Case Start Date`)&!is.na(hsaps$Reporting_Agen
cy),]
hsaps = hsaps[!is.na(hsaps$`Case Start Date`),]
hsaps_case$case = ifelse(hsaps_case$ID %in% hsaps$ID,hsaps_case$case,1)
hsaps = rbind(hsaps,hsaps_case)
rm(hsaps_case)
```

- Checking and importing previously assigned ID's from hsaps_backup
- Assigning ID's to new participants
- Assigning case numbers to all data rows after ordering the data by ID and Case Start Date (so that the earlier case of a participant would be assigned to an earlier case number)
- Ensuring the case number is assigned as 1 for a new participant without Case Start Date

Archive Final Data Set & Extra Processing for Tableau Data Set

Archiving Data Set

Finally, the data is ready to be stored in "\cdssOfficeProgramsSafeand EvaluationReports to date" with the naming convention, "^hsaps_cleaned_YYYY-MM-DD.xlsx" where "YYYY-MM-DD" is the date of the data being archived.

```
#saving the data for those running the cleaning script
write_xlsx(hsaps[order(hsaps$Reporting_Agency,hsaps$`Last Name`),c(1:grep("Co
mment",names(hsaps)),grep("^id$",tolower(names(hsaps))):grep("^case$",names(h
saps)))],paste0("//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home S
afe/Data and Evaluation/Data Reports to date/^hsaps_cleaned_",Sys.Date(),".xl
sx"))
```

Tableau Data Set

Background/Decisions

We noticed that, when we did not clean up the responses within the final data set that are not aligned with Home Safe ACIN, Tableau displays all the possible options as filters, which is inconvenient for Tableau users. Therefore, we decided to clean up responses that are not aligned with Home Safe ACIN to proper responses (ex. "Data not collected", "Unknown")

Code Description: Functions

The function, **hsaps_categorical_fix**, was used to carry out the decision.

```
#for tableau dataset
hsaps_categorical_fix = function(dt,var_name,category,assign){
  #compare all the responses to tolower(category)
  #var in dt[,var] form
  dt[,var_name] = ifelse(tolower(dt[,var_name]) %in% category,dt[,var_name],a
ssign)
  return(dt)
}
```

It takes the variable (**"var_name"**) within a data table (**"dt"**) and changes into an assigned (**"assign"**) value if any of the response is not part of the given category (**"category"**).

Code Description: Cleaning Script

Below is the corresponding code in the cleaning script.

```
##further processing for Tableau dataset
#setting up categories
na = c("client doesn't know","client refused","data not collected")
gi = c("female","male","gender other than female/male","transgender","questio
ning","unknown/not provided",na)
race1 = tolower(c("american indian/alaskan Native/Indigenous","Asian/asian am
erican","Asian","american indian/alaskan native","Black/African American/Afri
can","Native Hawaiian/Pacific Islander","pacific islander/native hawaiian","o
ther","White",na,"Unknown/Not provided","black/african american"))
ethn = c("non-hispanic/latin(a)(o)(x)","hispanic/latin(a)(o)(x)","cuban","pue
rto rican","mexican/chicano","other hispanic/latino","not hispanic/latino","u
nknown/not provided",na)
ms = c("married","not married/living with partner","divorced","separated","wi
dowed","never married","data not collected","unknown/not provided")
so = c("straight/heterosexual","gay/lesbian","bisexual","questioning",na,"unk
nown/not provided")
pl = c("english","spanish","mandarin/cantonese","vietnamese","tagalog","korea
n","other","data not collected")
veteran_cl3yrs_evics_discharge = c("yes","no",na,"unknown")
ls_entry = c("homeless","temporary housing","temporary- residential program",
"permanent- residential program","rent leaseholder","owner","other permanent
housing","other","data not collected","with others rent","with others no rent
","homeless unsheltered","homeless sheltered","hotel no rights","hotel with r
ights","owner lives alone","owner with others rent","owner with others no ren
t","skilled nursing facility","residential care facility","board and care fac
ility")
abuse_neglect_prevaps_oldcols = c("yes","no","unknown")
rs = c("professional service provider","educator","financial service provider
","law enforcement","medical personnel","mental health personnel","mental hea
lth","institutional employee","social worker","unknown","other community prof
essional","community professional","clergy","self","family member","no relati
```

```
onship","anonymous")
ces = c("no","yes - before home safe","yes - after home safe","yes - prior to
 home safe")
interv_type = c("no intervention","enhanced case management","mortgage paymen
t","rent back-pay","rent payment","housing navigation","temporary housing","e
mergency shelter","security deposit","utilities","relocation assistance/stora
ge","home habitability","legal services","caregiver services/respite care","o
ther","deep cleaning/hoarding assistance","deep cleaning","home repair","exte
rnal housing navigation")
ls_exit_fu_ls = c(ls_entry,"deceased","unknown","not exited")
fu_method = c("unable to verify","hmis","aps system","verified - program staf
f","verified - external staff")
fu_homeless = c("yes","no","client doesn't know","unknown","not applicable")

hsaps_tableau = hsaps_categorical_fix(hsaps,"Gender Identity",gi,"Data Not Co
llected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Race 1",race1,"Data Not
Collected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Ethnicity",ethn,"Data No
t Collected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Current Marital Status",
ms,"Data Not Collected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Sexual Orientation",so,"
Data Not Collected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Preferred Language",pl,"
Data Not Collected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Veteran Status",veteran_
cl3yrs_evics_discharge,"Data Not Collected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Living Situation Upon En
try",ls_entry,"Data Not Collected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Client Homeless Within t
he Last Three Years",veteran_cl3yrs_evics_discharge,"Data Not Collected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Previous Evictions or Fo
reclosures",veteran_cl3yrs_evics_discharge,"Data Not Collected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Current Eviction or Fore
closures",veteran_cl3yrs_evics_discharge,"Data Not Collected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Discharge from Instituti
on in the Last Six Months",veteran_cl3yrs_evics_discharge,"Data Not Collected
")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Abuse by Other - Financi
al",abuse_neglect_prevaps_oldcols,"Unknown")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Abuse by Other - Non-Fin
ancial",abuse_neglect_prevaps_oldcols,"Unknown")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Self-Neglect",abuse_negl
ect_prevaps_oldcols,"Unknown")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Reporting Source",rs,"Un
known")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Previous APS Involvement
",abuse_neglect_prevaps_oldcols,"Unknown")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Client Referred to CES",
```

```
ces,"Data Not Collected")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Intervention 1 - Type",i
nterv_type,"No Intervention")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Intervention 2 - Type",i
nterv_type,"No Intervention")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Intervention 3 - Type",i
nterv_type,"No Intervention")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Intervention 4 - Type",i
nterv_type,"No Intervention")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Intervention 5 - Type",i
nterv_type,"No Intervention")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Intervention 6 - Type",i
nterv_type,"No Intervention")
hsaps_tableau = hsaps_categorical_fix(hsaps_tableau,"Intervention 6 - Type",i
nterv_type,"No Intervention")
hsaps_tableau$`Living Situation at Exit` = ifelse(!is.na(hsaps_tableau$`Livin
g Situation at Exit`) & !(tolower(hsaps_tableau$`Living Situation at Exit`)%i
n%ls_exit_fu_ls),"Data Not Collected",hsaps_tableau$`Living Situation at Exit
`)
hsaps_tableau$`Six Month Follow-Up - Living Situation` = ifelse(!is.na(hsaps_
tableau$`Six Month Follow-Up - Living Situation`)  & !(tolower(hsaps_tableau$
`Six Month Follow-Up - Living Situation`) %in% ls_exit_fu_ls),"Data Not Colle
cted",hsaps_tableau$`Six Month Follow-Up - Living Situation`)
hsaps_tableau$`Twelve Month Follow-Up - Living Situation` = ifelse(!is.na(hsa
ps_tableau$`Twelve Month Follow-Up - Living Situation`)  & !(tolower(hsaps_ta
bleau$`Twelve Month Follow-Up - Living Situation`) %in% ls_exit_fu_ls),"Data
Not Collected",hsaps_tableau$`Twelve Month Follow-Up - Living Situation`)

write_xlsx(hsaps_tableau[order(hsaps_tableau$Reporting_Agency,hsaps_tableau$`
Last Name`),c(1,grep("^id$",names(hsaps_tableau)),grep("^case$",names(hsaps_t
ableau)),grep("^age$",names(hsaps_tableau)),grep("Case Start",names(hsaps_tab
leau)),grep("^Location of",names(hsaps_tableau)):grep("Comment",names(hsaps_t
ableau)))],paste0("//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/Home
 Safe/Data and Evaluation/Data Reports to date/^HSAPS Data for Tableau_",Sys.
Date(),".xlsx"))
```

- Generating the lists of responses aligned with ACIN (to be used as “category” in the function)
- Applying hsaps_categorical_fix function to each data element in the final data seT
- Storing the data set in “\cdssOfficeProgramsSafeand EvaluationReports to date” with the naming convention, “^HSAPS Data for Tableau_YYYY-MM-DD.xlsx” where “YYYY-MM-DD” is the date of the data being archived.