HDAP Data Cleaning Documentation

2024-10-22

The cleaning process begins with importing the most recent data set. It is located in "\CdssOfficeProgramsDataPII 21 Reports", with the file name "^hdap_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: HDAP Data Processing and Cleanup

Gate Keeping

Key variables used

County, First Name, Last Name, Date of Birth, Project 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 Q4) 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, **hdap_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 **hdap_exporting_data_with_proper_col_types**.

```r
hdap_exporting_data_with_proper_col_types = function(dt_name,sheet_name){
  library(readxl)
  library(stringr)
  cat("Processing the data for sheet:", sheet_name, "\n")
  if(sheet_name=="Data"){
    dt = read_excel(paste0("//cdss/feed/Central Office/HHCRB/HHB/Housing Prog
rams/HDAP/HDAP Data/HDAP PII 21 Reports/Compiled Reports/",dt_name),sheet = s
heet_name)
    coltype = as.character(dt[1,])
    coltype = tolower(coltype)
    coltype = gsub("general","date",coltype)
    coltype = gsub("integer","numeric",coltype)
    coltype = gsub("ssn","text",coltype)
    cat("Handling missing column types...\n")
    if(length(which(is.na(coltype)))>0){coltype[which(is.na(coltype))] = c("t
ext","date","date","text")}
    cat("Reading the data again with proper column types...\n")
    dt_new = read_excel(paste0("//cdss/feed/Central Office/HHCRB/HHB/Housing 
Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Compiled Reports/",dt_name),sheet
 = sheet_name,col_types=coltype)[-1,]
    names(dt_new) = as.character(dt[2,])
    dt_new = dt_new[-1,]
  }
  else if(sheet_name=="PII"){
    cat("Reading the PII data...\n")
    coltype = tolower(as.character(read_excel("//cdss6psas2/RADD/Housing & Ho
melessness/Reports/HDAP PII/05_Deliverables/HDAP Compiled Data for Housing/HD
AP PII FY21-22 Q1 2022-02-08.xlsx",sheet = "Data")[1,1:68]))
    names = as.character(read_excel("//cdss6psas2/RADD/Housing & Homelessness
/Reports/HDAP PII/05_Deliverables/HDAP Compiled Data for Housing/HDAP PII FY2
1-22 Q1 2022-02-08.xlsx",sheet = "Data")[2,1:68])
    coltype = gsub("general","date",coltype)
    coltype = gsub("integer","numeric",coltype)
    coltype = gsub("ssn","text",coltype)
    if(length(which(is.na(coltype)))>0){coltype[which(is.na(coltype))] = c("t
ext","date","date")}
    dt_new = read_excel(paste0("//cdss6psas2/RADD/Housing & Homelessness/Repo
rts/HDAP PII/05_Deliverables/HDAP Compiled Data for Housing/",dt_name),sheet 
= sheet_name,col_types = coltype)[-1,]
    names(dt_new) = names
    dt_new = dt_new[-1,]
    names(dt_new) = gsub("^REPORT PERIOD","Quarter",names(dt_new))
    names(dt_new) = gsub("START","Start",names(dt_new))
    names(dt_new) = gsub("END","End",names(dt_new))
  }
  cat("Finalizing column names and data formatting...\n")
  dt_new = data.frame(dt_new,check.names=F)
  names(dt_new) = gsub("  "," ",names(dt_new))
  names(dt_new) = gsub("^Item [0-9]+:\r\n","",names(dt_new))
  names(dt_new) = gsub("^Item [0-9]+: ","",names(dt_new))
  names(dt_new) = gsub("^\r\n","",names(dt_new))
  names(dt_new) = gsub("^ ","",names(dt_new))
  names(dt_new) = gsub("\\)-","\\) -",names(dt_new))
  names(dt_new) = gsub("First Benefit Appeal","Subsequent Benefit Appeal",nam
es(dt_new))
  names(dt_new) = gsub("/ ","/",names(dt_new))
  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 = hdap_exporting_data_with_proper_col_types(new_dt_name,sheet_name)
new_dt = new_dt[!is.na(new_dt$County),]

cat("Formatting date fields...\n")
new_dt$`Date of Birth` = gsub(" [0-9]+:[0-9]+:[0-9]+$","",as.character(new_dt
$`Date of Birth`))
new_dt$`Project Start Date` = gsub(" [0-9]+:[0-9]+:[0-9]+$","",as.character(n
ew_dt$`Project Start Date`))
new_dt$`Exit Date` = gsub(" [a-z,A-Z]+$","",new_dt$`Exit Date`)

cat("Merging with county codes...\n")
new_dt = merge(new_dt,county_code[,1:2],by.x="County",by.y="code",all.x=T)
new_dt$County = new_dt$county_name
new_dt$county_name = NULL
```

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 project 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 project start dates in the existing data set are missing, but appeared in the new data set.

```
  #part 1: sorting through cases btw no Project Start dates to yes Projec
t Start dates
  cat("Sorting through cases with updated Project Start Dates...\n")
  dt_no_csd = dt[is.na(dt$`Project Start Date`),]
  dt_yes_csd = dt[!is.na(dt$`Project Start Date`),]

  cat("Checking for matches with missing Project Start Dates...\n")
  csd_updated = c()
  for(i in 1:nrow(dt_no_csd)){
    match = which(tolower(paste0(new_dt$County,new_dt$`Last Name`,new_dt$
`First Name`,new_dt$`Date of Birth`))
                  %in% tolower(paste0(dt_no_csd$County[i],dt_no_csd$`Last
 Name`[i],dt_no_csd$`First Name`[i],dt_no_csd$`Date of Birth`[i])))
```

```
    if(length(match)==0){
      #new paticipant without project start date!
      csd_updated = c(csd_updated,i)
    }
    else if(length(match)>1){
      if(dt_no_csd$`Quarter Start`[i]>max(new_dt$`Quarter Start`[match])
         & !is.na(dt_no_csd$`Exit Date`[i])
         & all(is.na(new_dt$`Exit Date`[match]))){
        csd_updated = c(csd_updated,i)
      }
    }
    else{
      if(dt_no_csd$`Quarter Start`[i]>new_dt$`Quarter Start`[match]
         & !is.na(dt_no_csd$`Exit Date`[i])
         & is.na(new_dt$`Exit Date`[match])){
        csd_updated = c(csd_updated,i)
      }
    }
  }

  if(length(csd_updated)>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[unique(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 project 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 project start date was updated in the new the data. Therefore, the data row without project 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 project start date was removed when there are newer data rows with updated project 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 project 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.

```r
#part 2: gate keeping
match_to_exclude = c()
index_to_include = c()
index_to_exclude = c()
case_start_date_disc = c()
for(i in 1:nrow(new_dt)){
  match_target = tolower(paste0(new_dt$County[i],new_dt$`First Name`[i]
,new_dt$`Last Name`[i],new_dt$`Date of Birth`[i]))
  match = which(tolower(paste0(dt_no_csd_removed$County,dt_no_csd_remov
ed$`First Name`,dt_no_csd_removed$`Last Name`,dt_no_csd_removed$`Date o
f Birth`))==match_target)
  #if there is no match
  if(length(match)==0){
    match_wo_dob = which(tolower(paste0(dt_no_csd_removed$County,dt_no_
csd_removed$`First Name`,dt_no_csd_removed$`Last Name`,dt_no_csd_remove
d$`Project Start Date`))==tolower(paste0(new_dt$County[i],new_dt$`First
 Name`[i],new_dt$`Last Name`[i],new_dt$`Project Start Date`[i])))
    #sorting through cases with updated dob's (either Jan 1st to an upd
ated date or no date to an updated date)
    if(length(match_wo_dob)>0){
      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$County,dt_no_csd_remov
ed$`First Name`,dt_no_csd_removed$`Last Name`,dt_no_csd_removed$`Date o
f Birth`))==match_target)
  #if there still isn't a match,
  if(length(match)==0){
    if((tolower(paste0(new_dt$County[i],new_dt$`First Name`[i],new_dt$`
Date of Birth`[i])) %in%
             tolower(paste0(dt_no_csd_removed$County,dt_no_csd_removed$
`First Name`,dt_no_csd_removed$`Date of Birth`)))
       | (tolower(paste0(new_dt$County[i],new_dt$`Date of Birth`[i],new
_dt$`Last Name`[i])) %in%
          tolower(paste0(dt_no_csd_removed$County,dt_no_csd_removed$`Da
te of Birth`,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 Project Star
t dates
      if((!is.na(new_dt$`Project Start Date`[i])
          & !is.na(dt_no_csd_removed$`Exit Date`[match[m]])
          & !is.na(new_dt$`Exit Date`[i])
          & dt_no_csd_removed$`Project Start Date`[match[m]]!=new_dt$`P
roject Start Date`[i]
          & dt_no_csd_removed$`Exit Date`[match[m]]==new_dt$`Exit Date`
[i])
         | (!is.na(new_dt$`Project Start Date`[i])
            & dt_no_csd_removed$`Project Start Date`[match[m]]!=new_dt$
`Project Start Date`[i]
            & !is.na(dt_no_csd_removed$`Exit Date`[match[m]])
            & new_dt$`Project Start Date`[i]<dt_no_csd_removed$`Exit Da
te`[match[m]])
         | (!is.na(new_dt$`Project Start Date`[i])
            & dt_no_csd_removed$`Project Start Date`[match[m]]!=new_dt$
`Project Start Date`[i]
            & is.na(dt_no_csd_removed$`Exit Date`[match[m]])
            & !is.na(new_dt$`Exit 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 project 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 project start dates while having the same non-missing exit date
    - When two cases have different non-missing project start dates, but the project start date of the newer case is before the exit date of the older case
    - When two cases have different non-missing project start dates, but the exit 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 = dt[unique(match_to_exclude),]
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.

<u>Code Description: Cleaning Script</u>

Since FY 23-24 Q1, HDAP Cleaning Script uses the gate_keeping() function. (As "Background" section above also briefly mentioned, all the quarterly HDAP reports up to FY 22-23 Q4 were compiled and processed as the very initial cleaned data set.) The code within the cleaning script is shown below:

```
##post FY22-23 compiling
cat("Reading backup dataset...\n")
hdap_backup = read.xlsx("//cdss/feed/Central Office/HHCRB/HHB/Housing Program
s/HDAP/HDAP Data/HDAP PII 21 Reports/hdap_cleaned_2024-08-12.xlsx",sep.names
= " ")
#pls change the report name to the most recent one
cat("Compiling new data...\n")
hdap_new_data = hdap_gate_keeping(hdap_backup,"FY 23-24/HDAP PII FY23-24 Q4 2
024-09-03.xlsx","Data")
hdap_backup = hdap_new_data[[1]]
hdap_new_data_include = hdap_new_data[[2]]
hdap_new_data_exclude = hdap_new_data[[3]]
hdap_new_data_exclude = hdap_new_data_exclude[-grep("duplicate",tolower(hdap_
new_data_exclude$`HMIS ID (Optional)`)),]
cat("Merging old and new data...\n")
hdap = unique(rbind.fill(hdap_backup,hdap_new_data_include))
```

- **hdap_backup**: the most recent version of cleaned data set
- After applying gate_keeping function, each output table from the function was redefined/assigned:
    - **hdap_backup**: updated existing data set after the data rows needed to
    - **hdap_new_data_include**: list of data rows from the newest report that need to be included to the existing data set
    - **hdap_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
    - **hdap**: combined data set from hdap_backup and hdap_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(hdap)))){
  if(length(grep(" [0-9]+:[0-9]+:[0-9]+.*",hdap[,j]))>0){
    hdap[,j] = gsub(" [0-9]+:[0-9]+:[0-9]+.*","",hdap[,j])
  }
  hdap[,j] = as.character(as.Date(hdap[,j]))
}
```

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

```
#flagging the data with no case start date
cat("Flagging data without case start date...\n")
```

```
   hdap_no_start_date = hdap[which(is.na(hdap$`Project Start Date`)),]
   hdap = hdap[which(!is.na(hdap$`Project Start Date`)),]
```

De-duplication

Pt 1: Data without Project Start Date

Key variable used

County, First Name, Last Name, Quarter Start

Background/Decisions

Since this data (**hdap_no_start_date**) does not have project start dates, it won't be counted towards most of the reporting numbers that involve time period such as # of enrolled cases.

Code Description: Cleaning Script

```
#dealing with the dataset without case start date
cat("Processing duplicates without case start date...\n")
hdap_no_start_date_duplicate = hdap_no_start_date[(duplicated(hdap_no_start_d
ate[c("County","Quarter Start","First Name","Last Name")])
& duplicated(hdap_no_start_date[c("County","Quarter Start","First Name","Last
 Name")],fromlast=T)
& duplicated(hdap_no_start_date[c("County","Quarter Start","First Name","Last
 Name")],fromlast=T)
& duplicated(hdap_no_start_date[c("County","Quarter Start","First Name","Last
 Name")],fromlast=T)),c("County","Quarter Start","First Name","Last Name")]
if(nrow(hdap_no_start_date_duplicate)>0){
  #getting the list of duplicate pairs without case start date within the sam
e reporting month
  hdap_no_start_date_duplicate = unique(merge(hdap_no_start_date_duplicate,hd
ap_no_start_date,all.x=T))
  #removing the list of duplicate pairs without case start date within the sa
me reporting period
  hdap_no_start_date = hdap_duplicate_data_remove(hdap_no_start_date,hdap_no_
start_date_duplicate)
  #order the rest of the duplicate pairs by reporting month
  hdap_no_start_date = hdap_no_start_date[order(hdap_no_start_date$`Quarter S
tart`,decreasing = T),]
  #getting the most recent data
  hdap_no_start_date_unique = hdap_no_start_date[!(duplicated(hdap_no_start_d
ate[c("First Name","Last Name","County")])
& duplicated(hdap_no_start_date[c("First Name","Last Name","County")],fromlas
t=T)
& duplicated(hdap_no_start_date[c("First Name","Last Name","County")],fromlas
t=T)),]
}
write_xlsx(hdap_no_start_date_duplicate,"//cdss/feed/Central Office/HHCRB/HHB
/Housing Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Cleaning in Progress/hda
p_no_start_date_duplicate.xlsx")

if(nrow(hdap_no_start_date_duplicate)==0){
  hdap_no_start_date_unique = hdap_no_start_date
}
```

- When the data rows have the same county, first name, last name, AND Quarter Start values: flag out as a duplicate (the list was stored in **hdap_no_start_date_duplicate**)
- When the data rows have the same county, first name, last name values, but NOT Quarter Start value: take the data with the most recent Quarter Start value (the list was stored in **hdap_no_start_date_unique**). This was done only when hdap_no_start_date_duplicate has at least one row.
  - If there is no data in hdap_no_start_date_duplicate, we don't need to de-duplicate hdap_no_start_date data. In this case hdap_no_start_date_unique is equivalent to hdap_no_start_date.

Pt 2: Data with Project Start Date

Key variable used

County, First Name, Last Name, Date of Birth, Project Start Date, Quarter Start

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 project 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

```
#checking duplicate data
#flagging the list of first name and last names with the same birth date and
project start date pairs
cat("Processing duplicate data...\n")
hdap$`Project Start Date` = gsub("\\.[0-9]+$","",hdap$`Project Start Date`)
hdap_duplicate_data = hdap[(duplicated(hdap[c("County","First Name","Last Nam
e","Date of Birth","Project Start Date")])
& duplicated(hdap[c("County","First Name","Last Name","Date of Birth","Projec
t Start Date")],fromlast=T)
& duplicated(hdap[c("County","First Name","Last Name","Date of Birth","Projec
t Start Date")],fromlast=T)
& duplicated(hdap[c("County","First Name","Last Name","Date of Birth","Projec
t Start Date")],fromlast=T)
& duplicated(hdap[c("County","First Name","Last Name","Date of Birth","Projec
t Start Date")],fromlast=T)),c("County","Last Name","First Name","Date of Bir
th","Project Start Date")]
#finding all the data rows within hdap that have the same first/last name, do
b, and date pairs
hdap_duplicate_data = unique(merge(hdap_duplicate_data,hdap,all.x=T))
#flaggihng all out
hdap = hdap_duplicate_data_remove(hdap,hdap_duplicate_data)
#determining which one to add back in -> the most recent rows
hdap_duplicate_data_ordered = hdap_duplicate_data[order(hdap_duplicate_data$`
Quarter Start`,decreasing = T),]
#duplicate data within the same reporting period
hdap_duplicate_data_same_period = hdap_duplicate_data_ordered[(duplicated(hda
p_duplicate_data_ordered[c("Quarter Start","First Name","Last Name","Date of 
Birth","Project Start Date")])
& duplicated(hdap_duplicate_data_ordered[c("Quarter Start","First Name","Last
 Name","Date of Birth","Project Start Date")],fromlast=T)
& duplicated(hdap_duplicate_data_ordered[c("Quarter Start","First Name","Last
 Name","Date of Birth","Project Start Date")],fromlast=T)
& duplicated(hdap_duplicate_data_ordered[c("Quarter Start","First Name","Last
 Name","Date of Birth","Project Start Date")],fromlast=T)
& duplicated(hdap_duplicate_data_ordered[c("Quarter Start","First Name","Last
 Name","Date of Birth","Project Start Date")],fromlast=T)),
c("County","Quarter Start","First Name","Last Name","Date of Birth","Project 
Start Date")]
#finding all the data rows within hdap that have the same first/last name, do
b, and date pairs
hdap_duplicate_data_same_period = unique(merge(hdap_duplicate_data_same_perio
d,hdap_duplicate_data_ordered,all.x=T))
#removing duplicate data from the same reporting period
hdap_duplicate_data_ordered = hdap_duplicate_data_remove(hdap_duplicate_data_
ordered,hdap_duplicate_data_same_period)
```

- **hdap_duplicate_data**: List of all the duplicate case data (based on counties, first names, last names, birth dates, and project start dates) across all reporting periods, except the duplicate pairs from the same reporting period
- **hdap_duplicate_data_ordered**: hdap_duplicate_data ordered by the reporting period (from the most recent quarter to the oldest)
- **hdap_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 Quarter Start)

The code above describes the process below:

1. Creating hdap_duplicate_data by filtering out all the duplicate data from hdap data set (both across different reporting period and within the same reporting period) based on key variables listed above except Quarter Start

2. Creating hdap_duplicate_data_ordered by ordering data rows in hdap_duplicate_data by Quarter Start variable (from the most recent to the least recent)
3. Creating hdap_duplicate_data_same_period by filtering out the duplicate data from hdap_duplicate_data based on all key variables listed above (including Quarter Start)

The code then continues to sort through the duplicate data pairs within the same reporting period based on these two rules:

1. If one row within the duplicate pair has more information (i.e number of non-missing data elements) than the other row, discard the row with less information.
2. If the number of non-missing data elements for each row is the same, then flag both rows to be verified by the county representatives which row is accurate.

```
#sorting through which rows among the data above to include -> the least numb
er of blanks; if the number of blanks the same, flag
#identifying number of blank elements in each row
hdap_duplicate_data_same_period$na_num = 0
for(i in 1:nrow(hdap_duplicate_data_same_period)){
  hdap_duplicate_data_same_period$na_num[i] = length(which(is.na(hdap_duplica
te_data_same_period[i,])))
}

#sorting through whether to include the data to the final data set or flag
hdap_duplicate_data_same_period$include = ''
for(i in 1:nrow(hdap_duplicate_data_same_period)){
  if(!is.na(hdap_duplicate_data_same_period$`First Name`[i])&!is.na(hdap_dupl
icate_data_same_period$`Last Name`[i])){
    index = which(hdap_duplicate_data_same_period$County==hdap_duplicate_data
_same_period$County[i]
& hdap_duplicate_data_same_period$`First Name`==hdap_duplicate_data_same_peri
od$`First Name`[i]
& hdap_duplicate_data_same_period$`Last Name`==hdap_duplicate_data_same_perio
d$`Last Name`[i])
  }
  else if(is.na(hdap_duplicate_data_same_period$`First Name`[i])){
    index = which(hdap_duplicate_data_same_period$County==hdap_duplicate_data
_same_period$County[i]
& hdap_duplicate_data_same_period$`Last Name`==hdap_duplicate_data_same_perio
d$`Last Name`[i]
& hdap_duplicate_data_same_period$`Date of Birth`==hdap_duplicate_data_same_p
eriod$`Date of Birth`[i])
  }
  else if(is.na(hdap_duplicate_data_same_period$`Last Name`[i])){
    index = which(hdap_duplicate_data_same_period$County==hdap_duplicate_data
_same_period$County[i]
& hdap_duplicate_data_same_period$`First Name`==hdap_duplicate_data_same_peri
od$`First Name`[i]
& hdap_duplicate_data_same_period$`Date of Birth`==hdap_duplicate_data_same_p
eriod$`Date of Birth`[i])
```

```
  }
  if(i != index[1]){
    next
  }
  min_na = which.min(hdap_duplicate_data_same_period$na_num[index])+i-1
  hdap_duplicate_data_same_period$include[min_na]='y'
  if(min_na == which.max(hdap_duplicate_data_same_period$na_num[index])+i-1){
    hdap_duplicate_data_same_period$include[min_na:(min_na+length(index)-1)]
= 'flag'
  }
}
```

After sorting through the list in hdap_duplicate_data_same_period, 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
hdap_duplicate_data_same_period_include = subset(hdap_duplicate_data_same_per
iod,include=="y")
hdap_duplicate_data_same_period_tbd = subset(hdap_duplicate_data_same_period,
include=="flag")
hdap_duplicate_data_ordered = unique(rbind(hdap_duplicate_data_ordered,hdap_d
uplicate_data_same_period_include[,names(hdap_duplicate_data_ordered)]))
hdap_duplicate_data_ordered = hdap_duplicate_data_ordered[order(hdap_duplicat
e_data_ordered$`Quarter Start`,decreasing=T),]
# write_xlsx(hdap_duplicate_data_ordered,"HDAP/Cleaning/hdap_duplicate_data_o
rdered.xlsx")

#using the duplicated() functions again to filter out all the older data
hdap_duplicate_data_include = hdap_duplicate_data_ordered[!(duplicated(hdap_d
uplicate_data_ordered[c("First Name","Last Name","Date of Birth","Project Sta
rt Date")])
& duplicated(hdap_duplicate_data_ordered[c("First Name","Last Name","Date of
Birth","Project Start Date")],fromlast=T)
& duplicated(hdap_duplicate_data_ordered[c("First Name","Last Name","Date of
Birth","Project Start Date")],fromlast=T)
& duplicated(hdap_duplicate_data_ordered[c("First Name","Last Name","Date of
Birth","Project Start Date")],fromlast=T)),]

#including data rows
cat("Finalizing and archiving data...\n")
hdap = unique(rbind(hdap,hdap_duplicate_data_include,hdap_no_start_date_uniqu
e))
```

- **hdap_duplicate_data_same_period_include**: list of de-duplicated data rows from hdap_duplicate_data_same_period, where the code determined to include to the final data set

- **hdap_duplicate_data_same_period_tbd**: subset list of duplicate pairs from hdap_duplicate_data_same_period, where further help is needed from county representatives to determine which data within the pair is accurate
- **hdap_duplicate_data_include**: list of de-duplicated data rows from hdap_duplicate_data_ordered (free from all duplicate data)

The code above goes through the procedures below:

1. Creating hdap_duplicate_data_same_period_include and hdap_duplicate_data_same_period_tbd by filtering from hdap_duplicate_data_same_period
2. Incorporating the de-duplicated data within the same period (i.e hdap_duplicate_data_same_period_include) back to the hdap_duplicate_data_ordered table
3. Creating hdap_duplicate_data_include by de-duplicating the data across different reporting periods (from hdap_duplicate_data_ordered)
4. Incorporating hdap_duplicate_data_include (de-duplicated data with case start dates) and hdap_no_start_date_unique (de-duplicated data without case start dates with at least one intervention information) back to hdap data set

Numeric Responses to Corresponding Categorical Responses

After obtaining the final list of data rows, the data set goes through each data elements and replace the numeric responses to corresponding categorical responses.

Code Description: Functions

The function, **hdap_numeric_responses_to_texts**, was generated to perform what was described above.

```
hdap_numeric_responses_to_texts = function(dt,var_list,text_category){
  for(v in var_list){
    for(j in grep(v,tolower(names(dt)))){
      for(i in 1:nrow(dt)){
        dt[i,j] = ifelse(any(grepl(paste0('^',dt[i,j],'-'),text_category)),text_category[grep(paste0('^',dt[i,j],'-'),text_category)],ifelse(any(grepl(dt[i,j],gsub("^[0-9]+-","",text_category))),dt[i,j],NA))
      }
      dt[,j] = gsub("^[0-9]+-","",dt[,j])
    }
  }
  return(dt)
}
```

- text_category: the list of drop-down responses based on ACL
- var_list: list of variables that has text_category as a proper list of responses
- dt: data table of interest

Code Description: Cleaning Script

The code first assigns lists of categorical responses based on ACL. These will be used as "text_category" in the function.

```r
#changing numeric categorical responses to texts
unknown = paste0(8:9,"-",c("Client Does Not Know","Client Refused"))
na = "99-Data Not Collected"
mf = paste0(0:1,"-",c("Female","Male"))
race = c(paste0(1:6,"-",c("American Indian/Alaska Native","Asian","Black/Afri
can American","Native Hawaiian/Other Pacific Islander","White","Multiracial")
),unknown)
eth = c(paste0(1:2,"-",c("Non-Hispanic/Non-Latino","Hispanic/Latino")),unknow
n)
gender = c(mf,"2-Transgender Female","3-Transgender Male","4-Non-Binary",unkn
own,"10-Another Gender Identity")
sex = c(mf,"2-Non-Binary","9-Decline to State")
so = c(paste0(0:4,"-",c("Straight/Heterosexual","Gay/Lesbian","Bisexual","Que
er","Another Sexual Orientation")),unknown)
target = paste0(0:1,"-",c("No","Yes"))
vet = c(target,unknown) #same as disabled et al
ls = c(paste0(c(1:7,12:16,18:27),'-',c("Emergency Shelter","Transitional Hous
ing","Permanent Housing (except RRH) for Formerly Homeless Persons","Psychiat
ric Hospital/Facility","Substance Abuse Treatment Facility/Detox Center","Hos
pital/Residential Medical Facility","Jail/Prison/Juvenile Facility","Living w
ith Family","Living with Friend","Hotel/Motel Paid without Emergency Shelter 
Voucher","Foster Care/Group Home","Place Not Meant for Habitation","Safe Heav
en","Rental with VASH","Rental with Other Housing Subsidy (including RRH)","O
wner with Housing Subsidy","Rental without Housing Subsidy","Owner without Ho
using Subsidy","Long-Term Care Facility/Nursing Home","Rental with GPD TIP","
Residential Project/Halfway House with No Homeless Criteria","Interim Housing
")),unknown)
previous_stay = c(paste0(c(2:5,10:11),"-",c("Between One week and One Month",
"Between One Month and 90 Days","Between 90 days and 1 Year","1+ Year","One N
ight or Less","2-6 Nights")),unknown,na)
homeless_count = c(paste0(1:4,"-",c("Once","Twice","Three Times","4+ Times"))
,unknown,na)
homeless_length = c(unknown,na,"101-One Month",paste0(102:113,'-',c(2:12,"12+
")," Months"))
disabl_type = paste0(1:4,"-",c("SSI/SSP","SSDI","CAPI","Veteran's Benefits"))
denial_reason = paste0(0:11,"-",c("Loss of Contact","Capable to Re/Enter Work
force","Insufficient Medical Evidence","Lack of Follow-Through with Treatment
 Plan","Lack of Follow-Through with Application Process","Prior Denial of Ben
efits","Did Not Meet Disability Criteria","Lack of Work Credits","Not Disable
d Prior to Last Insured","Excess Resources","Unknown","Other"))
exit = c(paste0(c(1:7,10:25,28:32),"-",c("Emergency Shelter","Transitional Ho
using","Permanent Housing (except RRH) for Formerly Homeless Persons","Psychi
atric Hospital/Facility","Substance Abuse Treatment Facility/Detox Center","H
ospital/Residential Medical Facility","Jail/Prison/Juvenile Facility","Rental
 without Housing Subsidy","Owner without Housing Subsidy","Living with Family
 Temporarily","Living with Friend Temporarily","Hotel/Motel Paid without Emer
gency Shelter Voucher","Foster Care/Group Home","Place Not Meant for Habitati
on","Other","Safe Heaven","Rental with VASH","Rental with Other Housing Subsi
dy","Owner with Housing Subsidy","Living with Family Permanently","Living wit
h Friend Permanently","Deceased","Long-Term Care Facility/Nursing Home","Rent
al with GPD TIP","Residential Project/Halfway House with No Homeless Criteria
","No Exit Interview Completed","Rental with RRH/Equivalent Subsidy","Retaine
d Housing")),unknown)
```

Then, it uses hdap_numeric_responses_to_texts function to replace numeric responses of data elements to corresponding text responses.

```
hdap = hdap_numeric_responses_to_texts(hdap,"race",race)
hdap = hdap_numeric_responses_to_texts(hdap,"ethni",eth)
hdap = hdap_numeric_responses_to_texts(hdap,"gender i",gender)
hdap = hdap_numeric_responses_to_texts(hdap,"sex ",sex)
hdap = hdap_numeric_responses_to_texts(hdap,"sexual o",so)
hdap = hdap_numeric_responses_to_texts(hdap,c("target pop","experiencing home
less","exit due to"),target)
hdap = hdap_numeric_responses_to_texts(hdap,c("veter","disabling","chronicall
y hom"),vet)
hdap = hdap_numeric_responses_to_texts(hdap,"living situation",ls)
hdap = hdap_numeric_responses_to_texts(hdap,"previous st",previous_stay)
hdap = hdap_numeric_responses_to_texts(hdap,"homeless count",homeless_count)
hdap = hdap_numeric_responses_to_texts(hdap,"length of hom",homeless_length)
hdap = hdap_numeric_responses_to_texts(hdap,"type applied",disabl_type)
hdap = hdap_numeric_responses_to_texts(hdap,"denial$",denial_reason)
hdap = hdap_numeric_responses_to_texts(hdap,"destin",exit)
```

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

```
#assigning ID's
hdap_participant = unique(hdap[,c(grep("^County$",names(hdap)),grep(" Name$",
names(hdap)),grep("Date of B",names(hdap)))])
hdap_participant = unique(merge(hdap_participant,hdap_backup[,c(1:3,grep("Dat
e of Bir",names(hdap_backup)),grep("^id$",tolower(names(hdap_backup))))],all.
x=T))
hdap_id_assign = unique(hdap_participant[is.na(hdap_participant$id),1:4])
hdap_id_assign$id = as.character(sample(c(1000000:9999999)[-which(c(1000000:9
999999) %in% hdap_participant$id)],nrow(hdap_id_assign)))
hdap_participant = rbind(hdap_participant[-which(paste0(hdap_participant$Coun
ty,hdap_participant$`First Name`,hdap_participant$`Last Name`,hdap_participan
t$`Date of Birth`)
                                                    %in% paste0(hdap_id_assign
$County,hdap_id_assign$`First Name`,hdap_id_assign$`Last Name`,hdap_id_assign
$`Date of Birth`)),],hdap_id_assign)
hdap = merge(hdap,hdap_participant,all.x=T)

#assigning case numbers
hdap$case = 1
hdap = hdap[order(hdap$County,hdap$id,hdap$`Project Start Date`),]
for(i in 2:nrow(hdap)){
  if(hdap$id[i]==hdap$id[i-1]){
    hdap$case[i] = hdap$case[i-1]+1
  }
}
hdap$case = as.character(hdap$case)
hdap$Quarter = NULL

#removing rows
if(length(which(grepl("n/a",tolower(hdap$`First Name`))
                & grepl("n/a",tolower(hdap$`Last Name`))
                & is.na(hdap$`Date of Birth`))>0)){
  hdap = hdap[-which(grepl("n/a",tolower(hdap$`First Name`))
                   & grepl("n/a",tolower(hdap$`Last Name`))
                   & is.na(hdap$`Date of Birth`)),]
}
```

- Checking and importing previously assigned ID's from hdap_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)
- Removing data rows where first name and last name are "N/A"s

Archive Final Data Set

Finally, the data is ready to be stored in "\\Cdss\feed\Central Office\HHCRB\HHB\Housing Programs\HDAP\HDAP Data\HDAP PII 21 Reports" with the naming convention, "^hdap_cleaned_YYYY-MM-DD.xlsx" where "YYYY-MM-DD" is the date of the data being archived.

```
##MAKE SURE YOU PUT THE EXISTING DATA FILE TO "ARCHIVE" FOLDER BEFORE SAVING
THIS NEW DATASET
write_xlsx(hdap[order(hdap$County,hdap$id,hdap$`Last Name`,hdap$`First Name`)
,],paste0("//cdss/feed/Central Office/HHCRB/HHB/Housing Programs/HDAP/HDAP Da
ta/HDAP PII 21 Reports/^hdap_cleaned_",Sys.Date(),".xlsx"))
#saving datasets needed for dqr generation
write_xlsx(hdap_new_data_exclude,"//cdss/feed/Central Office/HHCRB/HHB/Housin
g Programs/HDAP/HDAP Data/HDAP PII 21 Reports/Cleaning In Progress/hdap_new_d
ata_exclude.xlsx")
write_xlsx(hdap_noncumulative,"//cdss/feed/Central Office/HHCRB/HHB/Housing P
rograms/HDAP/HDAP Data/HDAP PII 21 Reports/Cleaning In Progress/hdap_dropped.
xlsx")
```