NASTAD

TOOLKIT

Dataset Matching

JANUARY 2023

ACKNOWLEDGEMENTS

NASTAD is grateful for the development of this resource by:

Shauna Onofrey, Rita Isabel Lechuga, Zakiya Grubbs, Fozia Ajmal and the following persons that facilitated the review whose contributions and support made this publication possible: Henry Roberts; Laurie Barker; Kathleen Ly; Danica Kuncio; Lindsey Sizemore; Nicola D. Thompson; Clarisse Tsang ; Sarah New; Tony Fristachi

DEDICATION

In memory of Tony Fristachi

Contents

Introduction

Congratulations! You are taking the first step towards understanding how your hepatitis data fits into a bigger picture! Maybe you are looking at co-infection of hepatitis B and C or trying to find out how many of the hepatitis cases in your registry gave birth this year. Regardless of your focus and the datasets you are using, this toolkit is designed to help you along the way. Here you will find information on paperwork you should consider before you start your match, how to prepare the data once you have it, and considerations across three software options for matching. Both exact matches and probabilistic or 'fuzzy' matching is covered here. If this guide does not provide information, you are looking for, please reach out to NASTAD via hepatitis@nastad.org for further assistance.

What is data matching?

Data matching, also called record linkage, is the process for identifying an entity across different data sources. In our situation, the entity is usually a person, but it could also be a hospital, service center, or other entity for which we collect data. When we match datasets, we can identify when that entity does or does not appear in both datasets and combine the data on that entity from both datasets to answer questions about the entity that you cannot answer from the data in one dataset.

Why match?

Matching may provide data you cannot otherwise get through standard surveillance processes. For example, you may not have a way to capture mortality data on your HCV cases, but this is data you want to capture in the care cascade. A match with vital records could provide you with that data.

Particular to PS21-2103, matching may provide solutions for the following:

Outcome 1.2.2

Improve monitoring of burden of disease and trends in hepatitis A, acute hepatitis B and acute hepatitis C infections

How matching can help: Matching with other disease or outcome datasets can be used to increase completeness of age, gender, race/ethnicity, and county of resident data.

Outcome 1.2.4

Improved monitoring of hepatitis C continuum of cure (CoC)

How matching can help: As mentioned above, matching can provide the data on mortality required by CDC in year 3. Matching may provide data about other outcomes or treatment data as well.

Outcome 1.2.5

Improved development and utilization of viral hepatitis surveillance data reports

How matching can help: Including data from matches, including co-infections among viral hepatitis types and with other syndemic illnesses in surveillance reports provides more insight into the diseases in question and may make the data reports of interest to a wider audience.

Before You Match

Before you match your data, there are some steps you should take that will make the process easier.

Define your purpose.

Before you embark on matching data, you should define the purpose of your match. Is there a particular hypothesis you want to test? Are you using the match for case finding? Is this a subset of a larger project? Use the questions, below, to define your match.

Questions to answer before matching (all examples using match with death record data):

1. What is the question you are trying to answer?

(Ex: Case finding: How many death records with HCV listed as a cause of death were never reported to our registry)

(Ex: Care continuum: How many cases of HCV with confirmatory tests in the last 5 years died following confirmatory test)

2. What will be the outcome of answering the question?

Does this have policy implications? Do you want to pursue publication? Are you adding the data to surveillance?

3. What datasets will you use to answer this question?

Dataset 1: *(Ex HCV registry)*

Dataset 2: *(Ex: Vital Records - Deaths)*

4. What time period will you need for each dataset? How are you defining time?

Dataset 1: *(Ex HCV registry in full, HCV registry cases with confirmatory test (NAT) between 2017-2022)*

Dataset 2: *(Ex Deaths with date of death in 2022)*

5. Are there timing considerations between the two datasets?
(Does one event have to have happened first? Is there a period of time that needs to have passed between the two events)?
Which dataset will anchor the match?

6. Which variables are required for the match?

From Dataset 1 *(Ex: Last_name, First_name, DOB, Address, Maiden_Name)*

From Dataset 2 *(Ex: LName, Gname, Birth_date, Street, City)*

7. What other variables do you need to answer your question? If variables are included in both datasets (Ex: race, ethnicity) which dataset's variable will you use?

From Dataset 1 (Ex: Race, Ethnicity, Age)

From Dataset 2 (Ex: Causes of Death 1-22)

8. How will this data be used?
(Ex: Cases identified that were not in the registry will be added, and reporting hospital, if available, will be contacted for medical records. OR Aggregate data will be used to create a new bar in the HCV care continuum produced for and available on our website)

9. How readily available is each dataset?

10. Once you are ready to do the match, how long will it take you to get your data ready?

11. How long will it take to get the dataset for matching?

12. How often will the match be repeated?

(Ex: One time only, once a year)

Data use agreements

What is it?

A data use agreement is a formal agreement between two parties that have ownership of datasets. It outlines how a cut of a dataset will be used, protected, and eventually destroyed by the party that is asking for the data. The questions, above, should help you outline most of the information you will need to complete a data use agreement. Other things you should consider are the timeline you need to complete the match, who will be accessing the data, where and how they will be accessing the data. It is wise to put a timeline in your data agreement that accounts for potential setbacks, and to include at least two people from your team who can access the data in case of turnover, illness, etc.

Do I need one?

That answer depends on a few things. Who 'owns' the data you want to match? If it is a dataset you have access to for your day-to-day work, you may not require a data use agreement. Datasets such as death records are public and may also not require a data use agreement. However, if the dataset falls under another division or organization, you should plan to fill out a data use agreement.

It also depends on the rules of your organization or the owner of the other dataset. Your organization may require all data matches to have a data use agreement in place.

Even if it is not required, having a data use agreement may not be a bad idea. Having a document that outlines the details of your matching project allows for managing everyone's understanding and expectations from the beginning.

What should I expect?

Be prepared to provide detailed thinking about your project. Just as you may feel like no one understands the caveats of the hepatitis data like you do, so may these other groups feel about their data.

Some groups are more restrictive on the use of their data. You may need to compromise. Consider if you would be comfortable providing them with the data to match and to receive back aggregate data. If there is hesitation to that, or if you need person-level data, consider if you might start by matching a smaller subset (a month's worth of data) to see what the volume might be.

Do not expect that everyone will be immediately happy with your plans. Prepare to talk through and support why you want to look at what you want to look at.

The data use agreement, itself, describes who, what, when, where, why and how of the actual data match.

Who:

- Who will be doing the match?
- Who will have access to the dataset before its matched, during the match process, and after the match?
- It's wise to make sure at least two people in an organization are listed as being able to access the data at all points, in case someone is unable to continue the process, and for troubleshooting during the process.

What:

- What tools are you using to do the match?
- Which variables will be used to match the two datasets?
- What additional variables do you need?

When:

- When will the match be conducted?
- How long do you expect the process to take?
- How long will you have access to the data?
- What is the time period of the dataset you need?

Why:

- What is the purpose of your match?
- What do you plan to do with the data once it is matched?

How:

- How will data be shared?
- How will it be stored?
- How will it be destroyed at the end of the process?

The questions posed in above will all be helpful in completing a data use agreement.

Institutional Review Board

You may need to put your project through the Institutional Review Board or IRB, if you are matching to a dataset that has protected data (ex: birth registry) or if you are matching to a dataset that came from research. Most matches, even once going through IRB, will be found to be 'exempt', but you may need to go to the process anyways. The IRB form will ask for the information reviewed, above.

The decision as to if your match requires IRB approval is specific to your institution. You should plan for the review to take at least 2 weeks if it is necessary, and possibly longer depending on how often the IRB for the institution meets.

Software Options and Data Preparation

Now that you have thought through your match and gotten in place all the appropriate agreements and permissions, you are ready to begin manipulating the data to conduct the match.

Software Options

If you are already using a statistical software package, such as SAS or R, methods are available using these packages to prepare your data and to do your match. These will be detailed in the coming pages. If your data is output into Excel, information is provided on how to prepare your data for matching in Excel before moving it into another system for matching. If your statistical software is not detailed here, the information provided under SAS and R should provide terms you can search for in your particular software.

If you do not have a statistical software package you typically use, or if your datasets are particularly large and you are concerned that they may not run well in your usual software, cancer registries have come up with a few options. The most recent at the time of writing this is a software called Match*Pro, which can be found here: https://seer.cancer.gov/tools/matchpro

Match*Pro, and its predecessor LinkPlus (https://www.cdc.gov/cancer/npcr/tools/registryplus/lp.htm) were created for Cancer registries, but can be used with any fixed-width, delimited, or NAACCR XML File. Even if your data is not currently in a fixed width or delimited format, it can likely be put in the correct format to use with Match*Pro.

Match*Pro can be downloaded, for free, here https://seer.cancer.gov/tools/matchpro/download. To register, you must provide a valid e-mail address and your affiliation. A link for the download is sent instantly. Installation takes only a few minutes. It is a java-based program and can only be used on Mac Computers with a Windows emulator. The download includes a 181 page manual on how to use the software. This documentation is constantly updated, and such, the details of how to conduct a match using Match*Pro will not be detailed here. However, the steps below will prepare your data for matching using any software.

Preparing data for matching

Regardless of the software you decide to use to match your data, and the variables on which you are matching, the data will need to be prepared in various ways. As you already know if you are familiar with working with data, while we may view NASTAD, nastad, and N A S T A D as the same word, most software do not. You will want to ensure variables are correctly formatted as character or numeric, and for character variables, remove excess spaces, consider removing punctuation, and change all characters to upper case to avoid missing matches. Here are some steps you will want to take to prepare your data in Excel, SAS or R:

Variable format (Character/Numeric)

A common frustration in merging data can be an error indicating your data are not in the same format. Even if you are using a variable that is populated with numeric data, like a date, your software may read it as text. Checking your data format at the beginning can help you avoid this issue.

IN EXCEL

Highlight the column in question and right click to choose the 'format cells' menu. There you can see how the variable is already formatted and change the format.

Be aware, however, in pulling an excel spreadsheet into SAS, blank values in the 'guessing rows' (the top rows in the spreadsheet that SAS uses to determine appropriate format) can change the format in the resulting dataset.

IN SAS

Run a 'proc contents' on each dataset to see the format of the variables and determine if any changes need to be made.

If you have a variable that is stored as character, but should be numeric, use the 'input' function

new_variable = input(original_variable, informat.);

Informat should include how you would like the number formatted, including date formats

If you have a numeric variable that you think would be easier to convert to character for the purpose of the match, use the 'put' function to change it to character

new_variable = put(original_variable, format.);

IN R

```
# Creating a new data set using the existing "iris" data in R
df1 <- iris

# Run the 'sapply()' or 'lapply()' or 'str()' function to determine data elements and their types
sapply(df1, class)
lapply(df1, class)
str (df1)

# to convert numeric to character variable using 'as.character()' function
df1$char_Var2 <- as.character(df1$Sepal.Width)

# to convert character to numeric using the 'as.numeric()' function
df1$num_Var2 <- as.numeric(df1$char_Var2)
```

Removing excess spaces

Excess spaces can be hard to identify but will affect matching. Removing excess spaces can be done through a 'trim' function in various software

IN EXCEL

use the formula trim(Cell). For example, putting the formula =TRIM(A1) in cell C1 will result in a trimmed value of cell A1.

IN SAS

use the COMPRESS function *newvariable=compress(oldvariable)*

IN R

```
# Creating a string variable
str <- " 001 0000 AB 01 "

# Method 1: Remove all spaces in a string using the base R "gsub()" function
str1 <- (gsub(" ","",str))
str1

# Method 2: Remove all spaces in a string using the Stringr package's str_remove_all() function
library("stringr")
str2 <- str_remove_all(str," ")
str2
```

Removing punctuation

Punctuation in names such as “O’Brien” or “Smith-Jones” is often missed or mis-placed in data entry and can lead to missed matches

IN EXCEL

There are two ways to handle this in Excel

1. You can use the “find” and “replace” functions to remove the punctuation. Do this by
    a. Press CTRL+F to open the ‘Find’ and ‘Replace’ functions. Choose ‘Replace’
    b. Enter the first type punctuation, such as “-”
    c. Leave the ‘replace with’ field blank
    d. Choose ‘Replace All’
    e. Repeat with next punctuation
2. You can create a custom function as described here: https://www.extendoffice.com/documents/excel/3296-excel-remove-all-punctuation.html

IN SAS

Again, the easiest way to do this is using the COMPRESS function, with the modifier ‘P’ for punctuation.

Example: *newvariable=compress(oldvariable, , ‘P’);*

IN R

```
s <- "I am a,# new 1 comer,!to ,please $ help,me:out,here;
a!$%bbbêéè)(/&ß2"

# Remove punctuations in stringr package’s str_replace_all or the
base R gsub
install.packages("stringr")
library(stringr)

# Method1: uses str_replace_all() punct option
s1 <- str_replace_all(s, "[[:punct:]]", "")
s1

# Method2: uses str_replace_all()  alnum option (removes
whitespaces as well)
s2 <- str_replace_all(s, "[^[:alnum:]]", "")
s2

#Method3: uses the base R gsub() function (allows choosing which
punctuation to remove)
s3 <- gsub("[?.;!¡¿·$%&#,'/:()]", "", s)
s3
```

Changing Text to Uppercase

Having text variables in different combinations of cases can be easily overlooked and will keep your matches from being found. Luckily, it's an easy problem to fix.

IN EXCEL

Use the formula UPPER

Example: enter into cell =UPPER(A1)

IN SAS

Use the function UPCASE

Example: newvariable=UPCASE(oldvariable);

In R

```
s <- "I am a new comer to this work, please help me out here"

# Change case to upper case using the toupper() function in base R
s1 <- toupper(s)
s1
```

File format

If you are using SAS or R, you will likely be familiar with the format your files should be in. If you are using Match*Pro, your data will need to be one of three file formats, sometimes referred to as 'flat files':

- A **Fixed Width** file has data organized into columns in fixed positions within the file. For example, in a fixed width file, Last name takes the first 30 characters in the file, and town name is the next 20 characters.
- A **Delimited file** has data where the columns are separated by a delimiter, such as a comma or a tab.
- A **NAACCR XML File** is a file format specific to Cancer registries

From most software, you should be able to export your dataset to a fixed width (text) or delimited (formatted text, CSV) file. This may be through an 'export' or 'save as' function, depending on the software you are using. If the solution is not obvious, search for how to export your data as a CSV or text file.

Conducting the Match

Now that your data is prepared, you are ready to perform the actual match. However, there are still many options to consider to ensure the most complete and accurate match possible. This section will describe how to perform both exact matches and probabilistic, or 'fuzzy' matches and provide considerations for both.

*Matching using Match*Pro*

Match*Pro provides excellent documentation for using the software. Refer to the help document that accompanies your download to perform your match using Match*Pro. The information, below, may be helpful in understanding some of the terms referenced in Match*Pro.

Using Statistical Packages (SAS and R)

Statistical packages such as SAS and R are useful tools for matching datasets. Once matches are completed, you can move directly into analysis without importing or exporting additional data. If these are software packages with which you are already familiar, they may be an ideal place to start for matching your datasets. Different processes for matching are detailed here, with example code for using in these processes for both SAS and R. These processes likely exist in other programs, including SPSS and STATA, and searching for the topics listed below may help you uncover the details of how to use these statistical packages for matching your datasets.

Types of matches

We have mentioned throughout this document different types of matches, including exact and probabilistic matches. Here we will define those terms and provide sample code for conducting the matches.

Exact matches

What is an exact match?

Exact matching is looking for the exact same values in both datasets. For datasets where you have the same identification number in both datasets, exact matching may be more straightforward. However, when you are looking for matching names, you may find that exact matching is more of a challenge. As mentioned in the 'preparing data for matching' section, above, computer software, specifically statistical software packages, are not built to ignore spaces, punctuation, and capitalization in character variables. The exact match may be your first indication that you need to spend some additional time on data preparation.

In SAS

You are likely very familiar with the exact match process in SAS, if you work with datasets often. Another name for the exact match would be 'merge'.

To complete this process, make sure the variables you are looking to match in each dataset are named the same. For comparison purposes, you may want to keep a duplicate of those variables in each dataset with different names, so you can identify differences between the variables and which dataset may need changing if there are issues. Each dataset will need to be sorted by the variables of interest. In the merge statement, the variables of interest should be named as the 'by' variables.

If using names, you may also wish to consider running a secondary exact match on 'last name' in one dataset matched to the 'given name' in another dataset, if you think the two may often be switched.

Exact matching is often the first step in the matching process. You can feel very confident that these matches are 'true' matches, but repeating the matching process using some of these additional tools will give you more confidence that you have identified as many matches as you can.

EXAMPLE:

*Merge a hepatitis C dataset with vital death records.

*Create a new dataset with the hepatitis C data with the merging variables of last name, first name, and birth date, renamed to match the vital records dataset;

```
DATA DATASET1;
SET HEPC;
FIRST_NAME=GNAME;
LAST_NAME=LNAME;
DOB=BIRTH_DATE;
RUN;
```

*Create a second dataset for merging with the vital records, keeping a copy of the required variables with new names to monitor for errors in matching;

```
DATA DATASET2;
SET VITALS_DEATHS;
DEATH_FNAME=FIRST_NAME;
DEATH_LNAME=LAST_NAME;
DEATH_DOB=DOB;
RUN;
```

*Sort datasets by required variables;

```
PROC SORT DATA=DATASET1;
BY FIRST_NAME LAST_NAME DOB;
RUN;
PROC SORT DATA=DATASET2;
BY FIRST_NAME LAST_NAME DOB;
RUN;
```

*Merge datasets. Here we only want to keep the matches found in both datasets, and use the 'in' indicators to keep only those that are found in both;

```
DATASET EXACTMATCHES;
MERGE DATASET1 (IN=A) DATASET2 (IN=B);
BY FIRST_NAME LAST_NAME DOB;
IF A AND B;
RUN;
```

Note only those records where all three matches are exactly the same will be identified as matches. Check the logs to determine how many exact matches you have found.

In R

```
# Creating a sample dataframe set Hep C
HepC <- data.frame(
  firstname= c("Carsten","Gerd","Robert","Stephen","Ralf", "Ravi"),
  lastname = c("Meier","Buaer","Hartmana","Wolff","Krueger", "Chander"),
  dob = c(1949, 1968, 1930, 1957, 1966, 1971),
  nationality = c("US","US","UK","US","Poland", "India"),
  stringsAsFactors=FALSE)

# Creating a sample dataframe set death
death <- data.frame(
  firstname = c("Carsten", "Gred", "Robert", "Stephn", "Ralff", "Ravee"),
  lastname = c("Meier","Buaer","Hartaman","Wolff","Krueger", "Chander"),
  dob = c(1949, 1968, 1930, 1957, 1966, 1970),
  death = c("Yes", "No", "Yes", "No", "No", "Yes"),
  stringsAsFactors=FALSE)

# Merge the two datasets using exact matching in R
exact_match <- merge (HepC, death, by=c("firstname","lastname", "dob"))
exact_match
```

Fuzzy Matching

'Fuzzy Matching' is known by a number of other names, including approximate string matching or probabilistic matching. Where exact matches finds values or strings where there is 100% matches, fuzzy matching provides a likelihood that two values match. There are many matching techniques available for fuzzy matching. Most of these can be categorized as either a phonetic technique or an edit distance measurement. Phonetic techniques use knowledge about sounds to determine if two values are likely matches. Edit distance measurements look a the number of changes or keystrokes that would be needed to turn one value into another value. We will provide an example of each category of technique.

Regardless of technique used, fuzzy matching will result in some false matches. Human review can help to limit the number of false matches. You may also want to increase the variables you are including in your review. For example, lets say you are going to use fuzzy matching only on individuals who share a date of birth, and you get two names that look like they may be likely matches. If you have the city of residence from both datasets and find that, in addition to the same date of birth, they also list the same city of residence, you may feel more confident in your match.

Fuzzy matching also takes time, both computing and human. You will need to make decisions about how many more potential matches you want to look for, and how long it may take the computer to find them. One important step to take is to remove your exact matches first, if possible. Since you already know those matches exist, you do not need to take processing time to search for those matches again.

SOUNDEX

Soundex is a phonetic algorithm that helps to identify words that sound alike in English, but look different. By using SOUNDEX, you can identify names that may have minor spelling differences in your records, because their soundex will be the same. Here, your software will assign a 'soundex' to each variable you are interested in, and will compare it across the two datasets.

SAS

(Continuing the example using same datasets from above);

```
DATA SOUNDHCV;
SET HEPC;
SLNAME=SOUNDEX(LNAME)
SGNAME=SOUNDEX(GNAME);
DOB=BIRTH_DATE;
RUN;

DATA SOUNDDEATH;
SET VITALS_DEATHS;
SLNAME=SOUNDEX(LAST_NAME)
SGNAME=SOUNDEX(FIRST_NAME);
DEATH_DOB=DOB;
RUN;

PROC SORT DATA=SOUNDHCV; BY SLNAME SGNAME DOB; RUN;
PROC SORT DATA=SOUNDDEATH; BY SLNAME SGNAME DOB; RUN;

* This merge is based on those soundex values;

DATA SOUNDMATCH;
MERGE SOUNDHCV (IN=A) SOUNDDEATH (IN=B)
BY SLNAME SGNAME DOB;
IF A AND B;
RUN;
```

R

```
#Fuzzy matching using Soundex method:

# Creating a sample data frame HepC1

HepC1 <- data.frame(
  firstname=  c("Carsten","Gerd","Admore","Stephen","Ralf", "Ravi"),
  lastname =  c("Meier","Buaer","Patrick","Wolff","Krueger", "Chander"),
  dob = c(1949, 1968, 1930, 1957, 1966, 1971),
  nationality = c("US","US","UK","US","Poland", "India"),
  stringsAsFactors=FALSE)
```

```
# Creating a sample data frame death
death1 <- data.frame(
  firstname = c("Carsten", "Gred", "Robert", "Stephn",  "Ralff", "Ravee"),
  lastname =  c("Meier","Buaer","Hoffman","Wolff","Krueger", "Chander"),
  dob = c(1949, 1968, 1930, 1957, 1966, 1970),
  death = c("Yes", "No", "Yes", "No", "No", "Yes"),
  stringsAsFactors=FALSE)

#Install required packages
#install.packages ("data.table")
#install.packages ("dplyr")
#install.packages ("fuzzyjoin")

#Attach packages
library(data.table)
library (dplyr)
library (fuzzyjoin)

# Pasting together the key matching variables in a single column
cols <- c( 'firstname' , 'lastname' , 'dob' )

# Creating a new column `x` including the three matching variables
in a single column
HepC1$x <- apply( HepC1[ , cols ] , 1 , paste , collapse = "-" )
death1$x <- apply( death1[ , cols ] , 1 , paste , collapse = "-" )

#Fuzzy matching using Soundex method
fuzzy_soundex <- stringdist_join(HepC1, death1,
        by=c("x"),
        mode='left', #use left join
        method = "soundex", #use soundex distance metric
        max_dist=0,
        distance_col='dist') %>%
  group_by(x.x) %>%
  slice_min(order_by=dist, n=1)

fuzzy_soundex
```

Distance Measurements

There are several ways to measure the 'edit distance' between two variables. Edit distance is the term for the measurement of how many operations are required to change one 'string' or word into another. Different standards of measure allow different combinations of deletions, insertions, substitutions, and transpositions. Many of these standards of measure can be used in fuzzy matching. For the example, below, in SAS, we are using the COMPLEV function which calculates the Levenshtein edit distance between two text strings, which is the number of insertions, deletions, or replacements of single characters that are required to convert one string to the other. There are other distance measures, but they are not built-in functions of SAS.

For this method, you set a cutoff distance, where you would want to review anything with a distance less than that cutoff. You can choose this cutoff in a few ways. You could review your data, if its short enough, starting with a higher cutoff value and determine where you stop seeing matches. You could also run a frequency on your cutoff values, and set a cutoff based on that graph.

It should be noted that to run this code, SAS is merging every possible pairing between your two datasets. Depending on the size of your datasets, this can take considerable processing time, and may crash your computer. SQL code that improves the efficiency of this process has been included here as well, for larger datasets. Remember to remove exact matches and variables not required for the match (but keep an identifier from each dataset!).

SAS

Here, the distance measures are actually calculated in the merge step. Remember to remove any matches you have already identified. For this code, you would also need to have renamed birth day as DOB. We are calculating 2 edit distances - one for first name and one for last name, and then combining the two for an overall value. The cutoff for printing was based on review of the data. The modifier 'ILN' shown in first name does the following:

i or I	ignores the case in *string-1* and *string-2*.
l or L	removes leading blanks in *string-1* and *string-2* before comparing the values.
n or N	removes quotation marks from any argument that is an n-literal and ignores the case of *string-1* and *string-2*.

```
DATA FUZZY;
MERGE HCV (IN=A) VITALS_DEATHS (IN=B);
BY DOB;
IF A AND B;
FIRST_COMPLEV=COMPLEV(FIRST_NAME, GNAME, 'ILN');
LAST_COMPLEV=COMPLEV(LAST_NAME, LNAME);
TOTALCOMPLEV=FIRST_COMPLEV+LASTCOMPLEV;
RUN;
PROC PRINT DATA=FUZZY; WHERE TOTALCOMPLEV<5; RUN;
```

SQL

Using the datasets as listed above

```
Proc sql noprint;
Create table matrix2 as
Select a.gname, b. first_name, a.lname, b.last_name a.dob, b.dob,
a.caseid, b.vitalid;
COMPLEV(a.gname, b.first_name, 'ILN) as compfirstname,
COMPLEV(a.lname, b.last_name) as complastname,
Calculated compfirstname+complastname as Combocomp
From HEPC a  vitals_deaths b;
On a.dob=b.dob AND
COMPLEV(a.gname, b.first_name, 'ILN) <= 5 OR
COMPLEV(a.lname, b.last_name)<5

Group by a.lname
Order by a.lname;
quit;
```

R

```
#Fuzzy matching using the Levenshtein method

# Creating a sample data frame HepC1
HepC1 <- data.frame(
  firstname=  c("Carsten","Gerd","Admore","Stephen","Ralf", "Ravi"),
  lastname =  c("Meier","Buaer","Patrick","Wolff","Krueger", "Chander"),
  dob = c(1949, 1968, 1930, 1957, 1966, 1971),
  nationality = c("US","US","UK","US","Poland", "India"),
  stringsAsFactors=FALSE)

# Creating a sample data frame death
death1 <- data.frame(
  firstname = c("Carsten", "Gred", "Robert", "Stephn",  "Ralff", "Ravee"),
  lastname =  c("Meier","Buaer","Hoffman","Wolff","Krueger", "Chander"),
  dob = c(1949, 1968, 1930, 1957, 1966, 1970),
  death = c("Yes", "No", "Yes", "No", "No", "Yes"),
  stringsAsFactors=FALSE)

#Install required packages
#install.packages ("data.table")
#install.packages ("dplyr")
#install.packages ("fuzzyjoin")

#Attach packages
library(data.table)
library (dplyr)
library (fuzzyjoin)

# Pasting together the key matching variables in a single column
cols <- c( 'firstname' , 'lastname' , 'dob' )

# Creating a new column `x` including the three matching variables in a single
column
HepC1$x <- apply( HepC1[ , cols ] , 1 , paste , collapse = "-" )
death1$x <- apply( death1[ , cols ] , 1 , paste , collapse = "-" )
```

```
#Fuzzy matching using the Levenshtein (a.k.a. edit difference) method
fuzzy_lv <- stringdist_join(HepC1, death1,
                by=c("x"),
                mode='left', #use left join
                method = "lv", #use Levenshtein distance metric
                max_dist=5,
                distance_col='dist') %>%
  group_by(x.x) %>%
  slice_min(order_by=dist, n=1)
fuzzy_lv

#Fuzzy matching:
# Creating a sample data frame HepC1
HepC1 <- data.frame(
  firstname=  c("Carsten","Gerd","Admore","Stephen","Ralf", "Ravi"),
  lastname =  c("Meier","Buaer","Patrick","Wolff","Krueger", "Chander"),
  dob = c(1949, 1968, 1930, 1957, 1966, 1971),
  nationality = c("US","US","UK","US","Poland", "India"),
  stringsAsFactors=FALSE)

# Creating a sample data frame death
death1 <- data.frame(
  firstname = c("Carsten", "Gred", "Robert", "Stephn",  "Ralff", "Ravee"),
  lastname =  c("Meier","Buaer","Hoffman","Wolff","Krueger", "Chander"),
  dob = c(1949, 1968, 1930, 1957, 1966, 1970),
  death = c("Yes", "No", "Yes", "No", "No", "Yes"),
  stringsAsFactors=FALSE)

#Install required packages
#install.packages ("data.table")
#install.packages ("dplyr")
#install.packages ("fuzzyjoin")

#Attach packages
library(data.table)
library (dplyr)
library (fuzzyjoin)

# Pasting together the key matching variables in a single column
cols <- c( 'firstname' , 'lastname' , 'dob' )

# Creating a new column `x` including the three matching variables
in a single column
HepC1$x <- apply( HepC1[ , cols ] , 1 , paste , collapse = "-" )
death1$x <- apply( death1[ , cols ] , 1 , paste , collapse = "-" )

#Perform fuzzy matching using the Jaro-Winkler (JW) method
fuzzy_jw <- stringdist_join(HepC1, death1,
                by=c("x"),
                mode='left', #use left join
                ignore_case  = TRUE,
                method = "jw", #use jw distance metric
                max_dist=0.5,
                distance_col='dist') %>%
  group_by(x.x) %>%
  slice_min(order_by=dist, n=1)
fuzzy_jw
```

A note on deduplication

It may be quickly apparent to you that the methods used here are also what is employed or can be employed to look for duplicates within a dataset. You may find duplicates through this process because of multiple 'near matches'. If you are concerned that you have many duplicates you have not identified, you may be able to use some of these techniques to look for duplicates in your dataset.

Regarding matching

The code listed here provides a starting place for probabilistic matching, but by no means covers all approaches. The descriptions give you language that you can use in additional searches, which will uncover other approaches to distance measures that may be of use in your matching programs. However, one can also lose themselves in a search for a 'complete' or 'perfect' match. This quest follows the law of diminishing returns. Anyone performing a match should accept that not all matches will be found, and determine a stopping point before they start, instead of trying to include every method they know for finding additional matches. If you are going to use multiple approaches in a code, assuming you are looking for one to one matches (one person with hepatitis in your dataset can only match to one death event for example), you may want to remove matches as you go, so that you are only looking for matches for cases that do not have a match.

Datasets for Consideration

In general, datasets are good for matching if they have the same individual level data as the dataset you want to match with. That means, if you have hospital level data, and find another dataset with hospital-level data, they will be good for matching. Similarly, if you are matching with a registry, you will likely want another dataset with individual-level data.

Datasets that do not have the same levels of data may be good for comparison, or 'triangulation'. Even if you can't make the datasets match, exactly, they still may be used to understand trends, or where there may be gaps in the available data.

The datasets listed here may be good places to start for either of these processes.

Datasets to Consider for Matching

The datasets listed here are available in many jurisdictions, and usually have person-level data. They may be useful in further exploring populations to prioritize for interventions, the current implementation levels of known interventions, and long-term outcomes of viral hepatitis.

Other Hepatitis Datasets

Matching to other viral hepatitis datasets available in your jurisdiction is a great place to start on matching. Often, these datasets may be more readily available to you, and you are aware of some of the challenges with matching with them. They may provide you with an opportunity to explore some of the methods described here without worrying about a data use agreement or data restrictions.

Matching between hepatitis viruses can provide a lot of insight into both diseases. For example, a match between hepatitis A cases and hepatitis C cases that shows a rising proportion of hepatitis A cases occurring annually among people who have ever been infected with hepatitis C may be used to increase vaccination efforts among people with hepatitis C. If more risk behavior is available among the hepatitis C cases, you may be able to detect a shift in the populations affected by hepatitis A in your jurisdiction.

Other 'Syndemic' infectious diseases

Matching to other diseases where risk behaviors overlap, including HIV, sexually transmitted diseases, and TB are another great place to start to try these techniques. Understanding where these diseases overlap may allow for smarter deployment of shared services when available. If allowed by your jurisdiction, and depending on timing of infections, these may also be opportunities for improving completeness of demographic and risk data.

Vital Records – Deaths

Death records are considered public records and should be easy to get access to. Depending on your jurisdiction and the goal of your match, you may want to ask if you can get access to 'preliminary' data. Usually, death records go through a process of coding and data cleaning that can delay the release of the final dataset, sometimes up to a year. Preliminary records may be available sooner and can be used with some caveats.

Death records can be used in a number of ways. They may be used for case-finding (deaths with hepatitis listed that were not previously reported), understanding your care cascade, looking at average age of death for people reported with hepatitis, what people with hepatitis are listed as dying from, etc.

Death records utilize International Classification of Diseases (ICD) codes to code causes of death as well as conditions contributing to deaths. Where multiple codes of death are reported, causes are put through an algorithm to determine what is the 'underlying' cause of death. It is recommended that when using the cause of death variables you look at all causes, not just the underlying cause, for this reason.

Many jurisdictions have been able to gain better access to death records due to COVID, and if COVID is captured in the same system as your hepatitis data, there may be a process in place that could be used as a template for matching these systems.

Vital Records – Births

Depending on your jurisdiction, birth records may be harder to access. Mothers and newborns are considered protected, and there are more restrictions on this data.

When matching to this dataset, consider WHO you are matching – it may be best to match women of childbearing age to women who have given birth, rather than trying to match to the babies. This would allow you to understand the proportion of people with viral hepatitis who are

Hepatitis may be listed as an underlying condition on the birth record, but it may be a checklist rather than a coded variable.

Vaccine Registry

Your state's vaccine registry is another system that may provide useful data for matching. Knowing hepatitis A and B vaccination status of people living with HCV, for example, may allow you to better target vaccination campaigns. This is also a system where matching with COVID has already been explored, and a template may exist for a match.

Many vaccine registries have more complete vaccine histories on children than adults, and the completeness can vary across jurisdictions. Having a contact who knows the system well is important for any of these matches, but for the vaccine registries you will want someone who can give you a sense of if the registry will have complete data on what you want.

Cancer Registry

Cancer registries are often accustomed to matching and staff may be able to assist you with the process. This is another dataset where you should be able to get person-level data. Cancer registry matches can provide more information for your care cascade, including a clearer understanding of long-term outcomes of viral hepatitis. You may want to match to all types of cancer, not just liver cancer.

Voter Registration

Some states have been able to match with voter registration data to improve and support the demographic data in their surveillance system.

Electronic Medical Records

Electronic Medical Records (EMRs), also called Electronic Health Records (EHRs) are the electronic records held by medical institutions. The ability for these systems to report appropriate data to public health systems is a big part of 'Meaningful use'. Matching with a single health system's EMR can help you understand where there are gaps in surveillance, including how good their reporting is, what data they have that is not being consistently reported, and what data they do not have.

Aggregate Data Sets

Even if you are not able to match person-level data, summary data may be used to provide insight into a question you would like to answer. The following datasets may be available in your jurisdiction and may provide useful information that can be included in annual reports. These datasets may also aid you in estimating denominators for your jurisdiction.

Census

US census data is likely one of the first data sources you turn to for understanding the denominators in your state or jurisdiction. It is a helpful reminder that census can provide aggregate data down to the census tract including data on select demographics.

Insurance data: All Payer Claims and Medicaid

Many states maintain 'All Payer Claims' datasets. These are claims data collected by major medical insurers in the state. These contain billing data sent to and collected from most of the insurance providers in a given state. While there may be a unique identifier that would allow understanding the number of claims per individual, most of the identifiers here are likely stripped away, making matching impossible. However, claims data can help understand other issues such as hepatitis A and B vaccines distributed or medication prescription practices.

Similar to All Payor Claims, Medicaid for each jurisdiction may also be able to provide a dataset based on claims. Medicaid data may or may not be included in the 'All Payor Claims' dataset depending on the jurisdiction.

Hospital discharge data

Many states maintain hospital discharge datasets. These datasets contain information on all discharge from acute care hospitals in the state, often at the patient level. This data can be useful for understanding hospital utilization and changes in incidence and burden of disease and injury. Information is included on select demographics, procedures, services, and diagnoses. While there is patient-level data, most identifiers are usually stripped from the available datasets, so matching is likely not possible.

BRFSS

The Behavioral Risk Factor Surveillance System (BRFSS) is a telephone-based survey administered annually across the US. BRFSS asks a multitude of questions regarding health practices, chronic conditions, and use of prevention services. Local jurisdictions are also permitted to add questions to their local BRFSS. More information, including the survey instrument, can be found here: https://www.cdc.gov/brfss/index.html

YRBSS

The Youth Risk Behavior Surveillance System (YRBSS) is a system of surveys collected by CDC, states, territories, tribal governments, and health agency focused on school-aged children. It monitors health related behaviors among youth and young adults including alcohol and drug use, sexual behaviors including information on sexual identity and sexual partners, tobacco use, dietary behaviors, physical activity, and activity leading to unintentional injury and violence. More information can be found here: https://www.cdc.gov/healthyyouth/data/yrbs/index.htm

Syndromic surveillance

Syndromic surveillance is data emergency room chief complaint information which is provided to states in real time. It is most useful for looking at patterns and changes over time in the use of said emergency departments. Many states have worked on 'syndromes' to better understand overdoses in their state.

Emergency Medical Services (EMS) Data

While syndromic surveillance data provides emergency department data, EMS data contains information gathered from ambulances reports. This is another data source that has been used to deepen understanding of opioid use.

Drug treatment

Drug treatment data may belong in both overarching categories. An individual drug treatment facility would have patient-level data but working with that data would likely be especially challenging due to federal regulations which protect the identities of people seeking drug treatment. However, the area of your health department that provides addiction services may have publicly available data in aggregate regarding the utilization of drug treatment facilities by individuals in different areas of your jurisdiction. They may also provide some information regarding the demographics of the individuals utilizing these services.

The US Department of Housing and Urban Development (HUD)

HUD provides data by State on shelter and unsheltered homeless individuals through their data exchange. This may be another denominator worth understanding, particularly considering hepatitis A outbreaks in recent years. https://www.hudexchange.info/programs/hdx/