Paper 60-2019

# Fuzzy-Matches Made Easy: Designing shareable SAS® code to accurately and efficiently identify exact and close genetic matching tuberculosis isolates

Evan Timme, MPH, Arizona Department of Health Services

## ABSTRACT

The Tuberculosis Genotyping Information Management System (TB GIMS) is a genetic data source available to public health programs. We attempted to design universally sharable SAS® code to efficiently analyze TB GIMS data for the purpose of detecting exact and close matching isolates to an identified active TB case of interest (IATCI). The Arizona TB GIMS extract contains 2,900+ isolates. Coding was designed in base SAS® and created for easy adaptability by another health department. Modifiable macro thresholds set inclusion criteria for matches. Conventional assessments require an exact match on one 15-digit number and two 12-character alphanumeric variables. Genetic changes overtime are common and fuzzy-matches would accommodate such changes. Using a simple DO LOOP and SUBSTR function counter, we were able to count the total number of place-value matches in the 15-digit and two 12-character variables for each IATCI to all other isolates. Only fuzzy-matches meeting or exceeding the macro set thresholds are retained. Fuzzy-matches were then assessed for distance [ZIPCITYDISTANCE function] and the time [years between isolate collections]. Output reports include a dot-plot (time x-axis, distance y-axis), a single .XML with unique sheets for each IATCI, and a single .PDF with individual reports for each IATCI. When combined with epidemiologic data, the outputs were helpful for understanding potential transmission clustering. The utility of this code is helpful and complementary to epidemiologic-linking data. User defied thresholds allow for simple and easy code adaption, while being an efficient and effective use of staff time and resources. These findings are encouraging and warrant further exploration.

## INTRODUCTION

*Mycobacterium tuberculosis* complex (TB) is a bacterial disease most commonly infecting the lung of a person. The transmission of the bacteria occurs through prolonged period(s) of shared air. Although the vast majority of persons infected with bacteria never become sick and unable to transmit the bacteria, it is important to understand disease transmission and how to prevent transmission from occurring. In the United State and other supported public health regions, genetic testing for TB is progressing towards universal whole genome sequencing (WGS), which assessing about 90% of the bacterial genome; however, the current method of spacer oligonucleotide (spoligotype) and 24-loci mycobacterial interspersed repetitive units-variable number of tandem repeats (MIRU-VNTR) assesses about 1% of the bacterial genome. Until the transition to universal WGS, we rely on comparing the one 15-digit spoligotype and two 12-character MIRU of each genetic isolate to others isolates for identifying matches.

In this paper we will discuss one main component within the fuzzy-match TB GIMS coding; the utility of using a DO LOOP and a SUBSTR function to count the similarities between two variables. By creating a counter variable, the user can predetermine the number of needed matches to meet a threshold and retain only matches meeting or exceeding the threshold. Exact matches where the spoligotype and two MIRUs of one isolate identically match another isolate is easy to identify. Unfortunately, it is more difficult to identify close or fuzzy-matches where there is one or two differences between two isolates.

## MAKING ALL ISOLATE COMBINATIONS

Before any assessment of similarity can be performed, it is important to identify all combinations. There are several methods to devising combinations. It is important for the user to determine which method they feel is most appropriate for their data at-hand.

### SET, DO, SET, END

The application of incorporating a second SET statement into a DO LOOP was an advantageous approach for the TB GIMS dataset. In this dataset each isolate has a unique identification variable called the GIMSID. An A-copy and B-copy of the GIMSID variable were created and separated into two separate datasets [work.forAA with var GIMSID_A and work.forBB with var GIMSID_B]. By using a DO LOOP after the initial SET statement, the dataset work.forBB is matched to the first entry in work.forAA. This process continues for each subsequent entry in work.forAA until all combinations are created. Self matches where variable GIMSID_A is the same as variable GIMSID_B and repeated matches are excluded from the output:

```
data work.forAA (keep=GIMSID_A);
    set work.tbgims001 (keep= gimsid);
    GIMSID_A = GIMSID;
run;

data work.forBB (keep=GIMSID_B);
    set work.tbgims001 (keep= gimsid);
    GIMSID_B = GIMSID;
run;

data work.every_combination;
    set work.forAA;
  do i=1 to n;
    set work.forBB point=i nobs=n;
    if GIMSID_A < GIMSID_B then output;
  end;
run;
```

Since the GIMIDS is a numeric variable, establishing the less-than IF statement prevents the duplicated combinations where isolate 12345 and isolate 56789 are joined and kept, but isolate 56789 and isolate 12345 is not kept in the final output. By knowing the dataset and its variables, half of all combinations are removed and allows for faster processing [~4.2 million vs. ~8.4million combinations].

Once all desired combinations are established, a SQL procedure is used to join the GIMSID_A and GIMSID_B isolate specific information from the original TB GIMS dataset:

```
PROC SQL;
   CREATE TABLE WORK.for001 AS
   SELECT t1.GIMSID_A,
          t2.Date_of_Specimen_Collection as DTColl_A,
          t2.State_Case_Number as StateCaseNumber_A,
          t2.Spoligotype as Spoligotype_A,
          t2.MIRU as MIRU1_A,
          t2.MIRU2 as MIRU2_A,
          t2.GENType as GENType_A,
          t2.PCRType as PCRType_A,
          t2.Genotyping_Lineage as Lineage_A,
          t2.Zipcode as Zip_A,
          t1.GIMSID_B,
          t3.Date_of_Specimen_Collection as DTColl_B,
          t3.State_Case_Number as StateCaseNumber_B,
          t3.Spoligotype as Spoligotype_B,
          t3.MIRU as MIRU1_B,
          t3.MIRU2 as MIRU2_B,
          t3.GENType as GENType_B,
          t3.PCRType as PCRType_B,
          t3.Genotyping_Lineage as Lineage_B,
          t3.Zipcode as Zip_B
```

```
   FROM WORK.every_combination t1,
        WORK.tbgims001 t2,
        WORK.tbgims001 t3
   WHERE (t1.GIMSID_A = t2.GIMSID AND t1.GIMSID_B = t3.GIMSID);
QUIT;
```

## DO DO DO COUNTER

With each isolate combination and their respective A-copy and B-copy genetic information [e.g. spoligotype, MIRU1, and MIRU2] on a single row in the dataset, a DO LOOP and the SUBSTR function were used to count the number of matching place-values. This process is performed for the 15-digit spoligotype and each of the two MIRU 12-character variables. The counter variables for the matches are created before the first DO LOOP.

### CREATING COUNTER VARIABLES

Prior to the DO LOOP statement, three counter variables are created and set to a zero count. The maximum number of matches for a spoligotype is 15 and maximum number of matches for each MIRU is 12. All three variables will possess a format of best2.:

1. SPtot: number of place-value matches between `Spoligotype_A` and `Spoligotype_B`
2. MIRU1tot: number of place-value matches between `MIRU1_A` and `MIRU1_B`
3. MIRU2tot: number of place-value matches between `MIRU2_A` and `MIRU2_B`

```
data work.for002;
    set work.for001;
    SPtot = 00; MIRU1tot = 00; MIRU2tot = 00; MIRUtot = 00;
```

### USING SUBSTR FUNCTION IN DO LOOP

The DO LOOP for the spoligotype must process 15 times because there are 15 unique place-values within the spoligotype variables. Therefore, the DO LOOP is created to start at 1 and end after the 15$^{th}$ iteration. By using the SUBSTR function within the DO LOOP, the place-value 1 for variables `Spoligotype_A` and `Spoligotype_B`, are assessed if they are similar. If this is true, then counter increases by one:

```
do i=1 to 15;
    if substr(Spoligotype_A,i,1) = substr(Spoligotype_B,i,1)
    then SPtot +1;
end; drop i;
```

If there is no match in the place-values, the counter remains the same and the DO LOOP will move to the next place-value comparison. After 15 iterations the ending qualification is met and the spoligotype DO LOOP is complete.

The second DO LOOP begins by assessing the place-value matches of the first 12-character MIRU:

```
do i=1 to 12;
    if substr(MIRU1_A,i,1) = substr(MIRU1_B,i,1)
    then MIRU1tot +1;
end; drop i;
```

The MIRU1 DO LOOP processes in the same manner as the spoligotype, where each combination is assessed and the respective counter increases by one when a place-value match is identified.

The third and final DO LOOP assesses the sedond 12-character MIRU:

```
do i=1 to 12;
    if substr(MIRU2_A,i,1) = substr(MIRU2_B,i,1)
    then MIRU2tot +1;
end; drop i;
```

Collective, the three DO LOOP sections work well in identifying the number of place-value matches for each isolate combination. The integration of the counters allows for easy inclusion or exclusion of isolate combination for further review. In its entirety, the DO LOOP SUBSTR counter is simple and adaptable to other problem solving solutions:

```
data work.for002;
    set work.for001;
          SPtot = 00; MIRU1tot = 00; MIRU2tot = 00; MIRUtot = 00;
          do i=1 to 15;
                if substr(Spoligotype_A,i,1) = substr(Spoligotype_B,i,1)
                then SPtot +1;
          end; drop i;
          do i=1 to 12;
                if substr(MIRU1_A,i,1) = substr(MIRU1_B,i,1)
                then MIRU1tot +1;
          end; drop i;
          do i=1 to 12;
                if substr(MIRU2_A,i,1) = substr(MIRU2_B,i,1)
                then MIRU2tot +1;
          end; drop i;
run;
```

### MACRO THRESHOLD SETTING

The last component is setting limits to determine which combinations meet or exceed your fuzzy-match requirement. Since the spoligotype has a maximum of 15 matches and each MIRU has a maximum of 12 matches, MACRO variables are used for simple threshold setting. Only those combinations meeting or exceeding the MACRO variables are kept in the final dataset:

```
%let SPthreshold = 14; /* val b/t 00 and 15 */
%let MIRU1threshold = 11; /* val b/t 00 and 12*/
%let MIRU2threshold = 11; /* val b/t 00 and 12 */

data work.TBGIMSreviewfile001;
    set WORK.for002;
    where SPtot GE &SPthreshold
      and MIRU1tot GE &MIRU1threshold
      and MIRU2tot GE &MIRU2threshold;
run;
```

## CONCLUSION

Incorporating a SUBSTR function within a DO LOOP is an easy and efficient method to count the number of one-for-one place-value matches between two variables. The utility of the DO LOOP replaces using brute force and allows for similar code structure to be applied to the assessment of several paired variables. Marco variable thresholds are helpful to allow for simple adjustment of the

## REFERENCES

Austin, David, J. April, 2005. “A Clever Demonstration of the SAS® SUBSTR Function.” Proceedings of the SUGI 2005, Available at https://support.sas.com/resources/papers/proceedings/proceedings/sugi30/059-30.pdf

## ACKNOWLEDGMENTS

Thanks to Deepa and Ben for being the OG programming crew; Deb would be proud.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Evan Timme
Arizona Department of Health Services
602-364-3849
Evan.Timme@azdhs.gov
azhealth.gov