Paper 161-2023

Processing Large Volumes of Communicable Disease Data: SAS Techniques from the COVID-19 Experience in San Diego County

Whitney Webber, Nathaly Moran, Jacob Ritz, Fatema Sakha, Jacquelyn Ho, Jennifer A. Nelson and Jeffrey Johnson

ABSTRACT

San Diego County is one of very few counties in California that maintains its own electronic communicable disease reporting and surveillance system. Epidemiologists routinely extract and analyze data from the system to identify and report on the status of diseases in the county. The large influx of COVID-19 lab reports during the pandemic challenged the system and surveillance operations and the ability of epidemiologists to provide daily counts of COVID-19 cases, hospitalizations, and deaths in a timely, efficient manner. This paper will examine how the COVID-19 surveillance epidemiologists in San Diego County, primarily new users of SAS, reimagined how to manage and analyze COVID-19 data with Base SAS to meet the information needs of leaders (for priority-setting and policy decisions) and the public. SAS techniques from the COVID-19 experience in San Diego County include PROC APPEND, PROC DATASETS, macros and arrays, KEEP= option with Colon (:) Modifier, ODBC libraries, PROC IMPORT, FIRST. and LAST. processing, IF-THEN-DO statements, LAG, and INTCK. This paper will also explore the successes and challenges of using SAS for COVID-19 surveillance during the pandemic, and how SAS is now being leveraged to respond to current communicable disease outbreaks.

INTRODUCTION

The Epidemiology and Immunization Services Branch in the Public Health Services department of the County of San Diego Health and Human Services Agency was at the forefront of COVID-19 disease response and surveillance in San Diego County during the COVID-19 pandemic. The County maintains its own communicable disease registry system called WebCMR (a County-contracted system hosted by Clinisys) to register all disease reports submitted to Public Health Services and document disease investigations. In 2019, the County registered approximately 30,000 disease reports with nearly 10,000 requiring public health investigation; in 2021 due to COVID-19, those numbers skyrocketed to over 6 million registered laboratory and disease reports (including negative COVID-19 lab results, reporting of which was mandated by California Department of Public Health) and 440,000 investigations.

For reporting COVID-19 data, epidemiologists needed to develop programming to isolate COVID-19 cases based on test results and clean the data to make sure it was as accurate and complete as possible. As part of routine surveillance, epidemiologists were accustomed to extracting and processing all necessary data from WebCMR in near real-time to produce their reports, which was manageable for small numbers being reported out weekly or even less frequently. Under COVID-19 pandemic conditions, this process became unsustainable. To handle such unprecedented volumes of data, address data quality issues, and meet the demands of daily reporting, epidemiologists hastened to reconfigure their processes and systems all while learning new tools to appropriately manage COVID-19 data.

SAS software soon became one of these important tools in the COVID-19 disease response. This paper will explain the main SAS techniques that epidemiologists applied for exporting and reading in COVID-19 data tables, transforming data, and assuring quality data with checks for missing, inaccurate, and duplicate data. This paper will specifically highlight: ODBC libraries and PROC IMPORT for importing data; macros for defining dates and libraries, transposing data, reading in files, running quality assurance checks, and optimizing storage of stored variables (%SQUEEZE); KEEP= option with the Colon (:) Modifier for efficiently keeping specified variables with the same prefix; arrays for computing new variables; PROC APPEND for appending the values of one dataset to another; PROC DATASETS for saving specific SAS datasets; FIRST. and LAST. processing to create a running count of the number of records within a group and indicate the first and last records within this same group; IF-THEN-DO statements to flag duplicates, and LAG and INTCK functions for calculating the difference in days between two consecutive rows of data.

DESCRIPTION OF COVID-19 CASE DATA

We use a communicable disease registry system called WebCMR to register all positive and negative COVID-19 results that we receive from laboratories for San Diego County. Most results were sent via electronic laboratory reporting (ELR), but especially in the early days of the pandemic, many results were also entered manually by a large surge team with consequent impacts to data quality. Self-administered tests were not reported to the County and were not included in COVID-19 data reporting. Investigators were assigned to interview eligible COVID-19 cases as soon as they were received. Even for cases that were not or could not be interviewed, the following fields were required to be entered in WebCMR: address, race, ethnicity, gender, case classification (confirmed or probable), was patient hospitalized, was patient admitted to an ICU, did patient die, lab result, interview status, and investigator notes.

To begin our reporting on COVID-19 data, we exported the data out of WebCMR as Microsoft Access databases. (A back-end SQL connection for accessing data is not configured for WebCMR.) We used six types of Access databases for COVID-19 reporting containing demographics and laboratory information, clinical and epidemiologic information, hospitalization details, exposure settings, and outbreak data. To avoid crashing WebCMR or timing out an export query, we broke up each Access database by date range. Therefore, as the volume of COVID-19 cases increased so did the number of tables that were utilized. To date, there are nearly 150 relational tables that are used for COVID-19 reporting, representing thousands of fields and upwards of a million records.

COVID-19 REPORTS PRODUCED

All processing was done in SAS to produce SAS output that we used to make COVID-19 reports, generate Excel files that were used to create Tableau dashboards, or produce SAS datasets that could be used to produce many different data products. We produced a multitude of reports during the COVID-19 pandemic. The types of reports and the information they contained evolved over time to meet the ever-changing data needs of County policymakers, response leadership, and the public. We shared reports publicly, internally, or with the California Department of Public Health to assist with operations. For COVID-19 reporting in 2020 and 2021, we produced static PDF reports. By early 2022, we were also producing and publishing online dynamic reports using Tableau. Figure 1 is an example of the dynamic Tableau dashboard produced showing COVID-19 confirmed cases by episode date. Many of the reports are still accessible on the county's webpage. The following list summarizes some of the main topics covered by the public-facing reports:

- Cumulative counts of cases, hospitalizations, deaths, and outbreaks
- Trends over time of cases, hospitalizations, deaths, and outbreaks
- Age, sex, and race/ethnicity of cases, hospitalizations, and deaths
- Vaccination status of cases, hospitalizations, and deaths
- Cases among persons experiencing homelessness
- Cases by zip code and city of residence

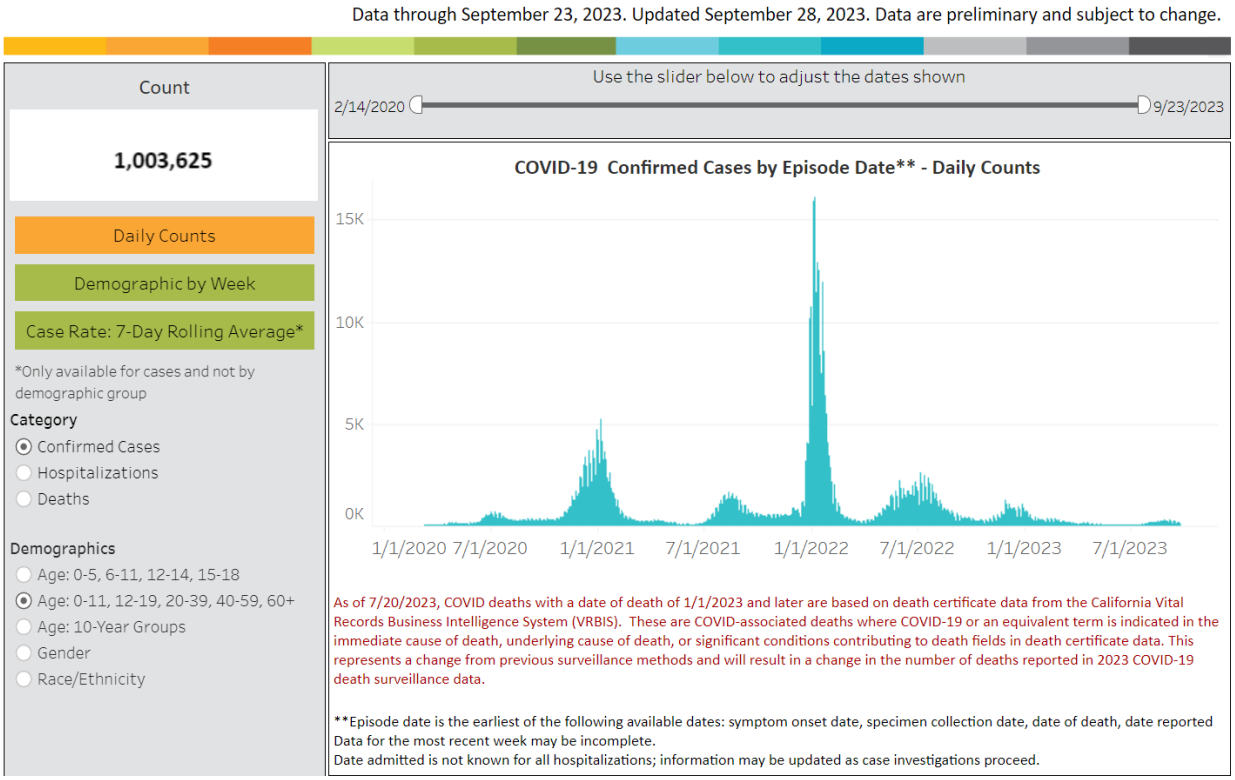

Figure 1. COVID-19 Tableau Data Dashboard Example

COVID-19 DATA PROCESSING EXPLAINED

The initial process to produce COVID-19 data reports involved four main steps described below:

1. Export and read in tables – In this step, we exported all COVID-19 lab results from WebCMR, read them in, transposed them, and merged them.
2. Data transformations – In scanning through different lab results, we merged and classified COVID-19 cases and created variables for hospitalized, ICU, resident status, inmate, K-12 schools, vaccination status, symptomatic, healthcare workers, and housing status. We also recoded race/ethnicity and computed different age groups that could be used across different analyses/reports.
3. Quality assurance – To ensure high quality data, we implemented quality assurance checks for duplicate outbreaks, out-of-range dates, resident status, case classification (confirmed or probable), and vaccination status. We hard coded any QA issues that we identified. We also ran scripts to identify and remove duplicates based on exact name and date of birth.
4. Analysis – For this final step, we created three SAS datasets—one that was everything prior to de-duplication; one for all COVID-19 investigations, including non-residents and those investigations that were determined to not meet case criteria; and one for investigations among all San Diego County residents. The case files were later used to prepare the reports described above and by other teams to create additional products, respond to data requests, and produce other QA reports.

Processing a large amount of COVID-19 data daily was challenging. The process was very lengthy and time consuming, averaging five hours to complete, with most of the time being spent waiting on reading in files and classifying the cases based on their lab results. There was also over-processing of data; all data from the beginning of the pandemic to the present was read in and analyzed each day. Further, the process allowed a lot of room for human error; for instance, the dates in the file names were not automated.

Amidst these challenges, we strived to meet our daily reporting deadlines. To meet our goals of saving time and improving data quality, we reimagined the process using SAS. Some, but not all of us, had basic proficiency in SAS and were aware that data processing would be faster in SAS compared to the current programming language. Despite being primarily new users of SAS, we achieved success with the new process. Processing time went from five hours to roughly two hours, a 56% improvement!

SAS TECHNIQUES TO SAVE TIME AND IMPROVE DATA QUALITY

The following sections outline the SAS techniques we used that imparted the most time savings and improved the quality of the data, which were used in the first three process steps. Figure 2 shows the SAS workflow that reflects the key SAS techniques.

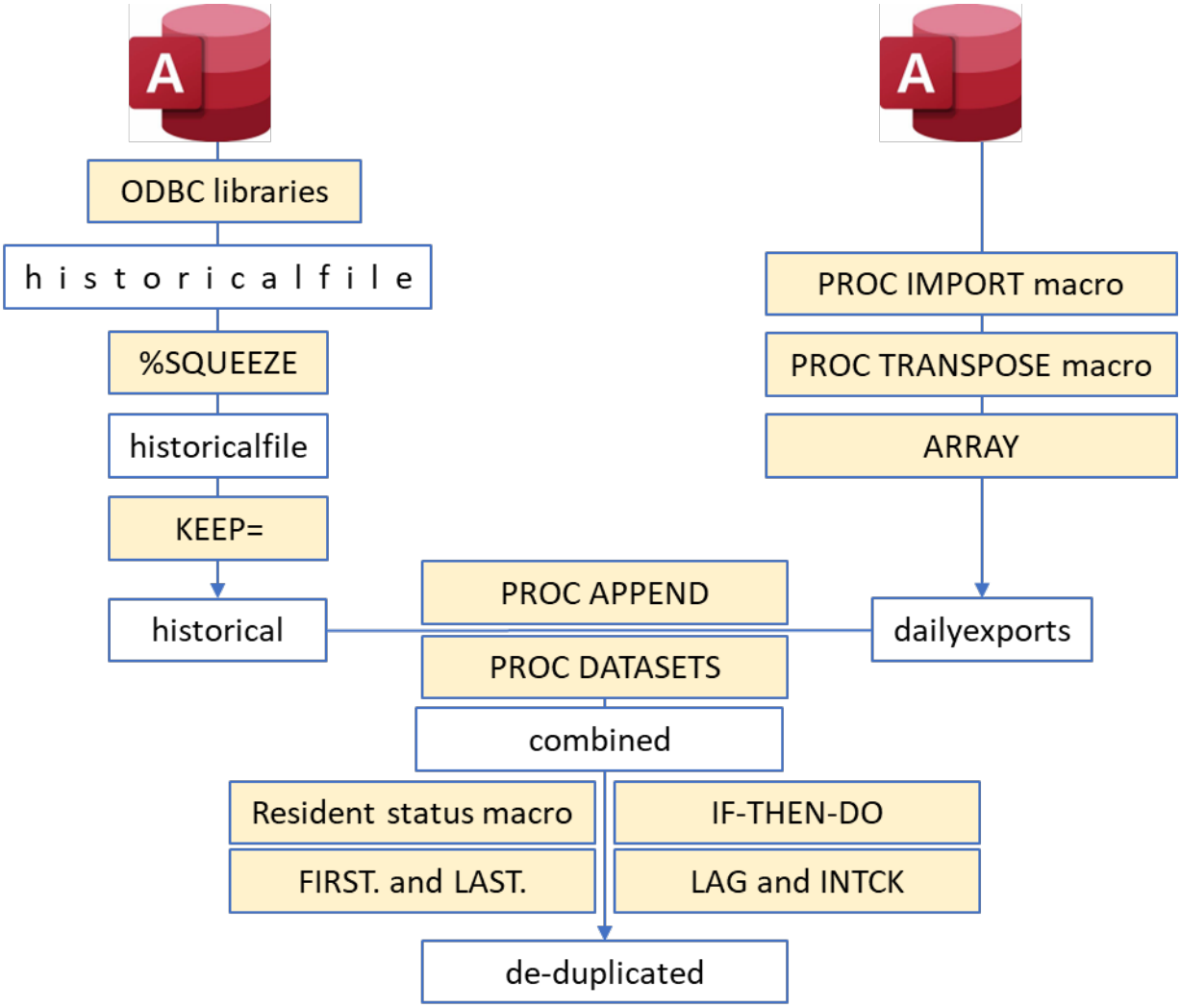

Figure 2. SAS Workflow

ODBC LIBRARIES FOR READING FILES IN FASTER AND SMARTER

The biggest change was to limit daily exports to the most recent month of data and append the recent data to a historical data file, which was refreshed weekly. We had to frequently update our data files because previous cases were often still under investigation and data updates to past cases were still occurring. We used a LIBNAME statement for the ODBC Engine to read tables in with, which was faster than the PROC IMPORT procedure for creating the historical file. PROC IMPORT was also crashing SAS

due to the size and number of files being imported, and the LIBNAME approach was an alternative that worked and did not take a lot of processing time. The following example illustrates the SAS code to import the three tables ('Disease Incident Export', 'Demographics – DEMADD', 'Laboratory Information (system)'), associated with the Microsoft Access database containing demographic and laboratory information. We used macros to define the location of the export folder and weekly export date. As referenced earlier, we broke up the exported databases into date ranges by creation date; the example below shows the earliest of the exports. The code ends with a statement to close the library.

```
%let exportfolder = C:\HISTORICAL FILE;


%let exportdate = 06072022;

libname savesdb1 odbc required="driver=Microsoft Access Driver (*.mdb,
*.accdb);
dbq= &exportfolder\disinc_01012020_10142020_create_exp&exportdate..accdb;";


data covidexp1;
set savesdb1."Disease Incident Export"n;

data demadd1;
set savesdb1."Demographics - DEMADD"n;

data covidsyslab1;
set savesdb1."Laboratory Information (system)"n;
run;

libname savesdb1;
```

%SQUEEZE MACRO FOR OPTIMIZING SPACE FOR STORED VARIABLES

We applied the %SQUEEZE macro to find the minimum lengths required for both numeric and character variables in the SAS historical dataset and use those minimum lengths for the variables to reduce the size of the dataset (Bettinger, 2007). We integrated the %SQUEEZE macro into the SAS program for creating the historical data file after all the historic files were merged together and before the data transformations. This had the two-fold benefit of reducing processing time for the weekly refresh file and helping us work with a much smaller dataset when it came time to append the daily exports to the historical file. The following example demonstrates how we used %SQUEEZE on our input SAS dataset (coviddata_qa) resulting in an output SAS dataset (covidsqueeze). We did not omit variables from the minimum-length computation process, which is why there are no variables listed after the NOCOMPRESS option:

```
proc contents data=coviddata_qa ; run ;
%SQUEEZE( coviddata_qa, covidsqueeze, NOCOMPRESS= )
proc contents data=covidsqueeze ; run ;
```

KEEP= OPTION WITH COLON (:) MODIFIER FOR SELECTING NECESSARY VARIABLES

SAS options such as KEEP= were also critical for reducing the size of the SAS historical dataset. At the start of processing, we read in the historical file and ran a KEEP statement on it. There were over 1,000 variables in the historical data file, and nearly 70% were preserved for later analysis by us and other teams using a KEEP statement. It was essential to use a Colon (:) Modifier for many of the variables so we did not have to keep track of the growing list of variables. The colon following a variable name prefix refers to any variable that starts with that prefix. A practical use of the Colon (:) Modifier involved selecting multiple lab results that were being added to the same patient record if they were not a reinfection; hence, some cases had as many as 48 lab collection dates and corresponding results contained in their patient record, especially when workplaces were requiring regular testing. Sample code is shown below, limited to only a few variables for example purposes.

```
data historicalfile (keep = age sex id occupation primary_language
```

```
provider race_ethnicity residentstatus collectiondate: street_address);
set historicalfile;
run;
```

MACROS FOR PROC IMPORT TO SIMPLIFY THINGS

For reading in the daily exports, we were able to use PROC IMPORT given that the data were limited to one month of records. We used macros to simplify the code and reduce human error. The example shown below references the same type of Microsoft Access database and tables shown in the ODBC libraries example.

```
%let startdate= 01082023;
%let enddate= 02132023;
%let exportdate= 02142023;

%let disinc = disinc_&startdate._&enddate._create_exp&exportdate..accdb;


%macro readaccess(exportname = , inname = , outname = );
PROC IMPORT OUT= &outname
            DATATABLE= "&inname"
            DBMS=ACCESS REPLACE;
     DATABASE="&exportfolder\&exportname";
     SCANMEMO=YES;
     USEDATE=YES;
     SCANTIME=YES;
RUN;
%mend;


%readaccess(exportname =&disinc , inname = Disease Incident Export, outname
= covid);
%readaccess(exportname =&disinc , inname = Demographics - DEMADD, outname =
demadd);
%readaccess(exportname =&disinc , inname = Laboratory Information (system),
outname = covidsyslab);
```

TRANSFORMING DATA USING MACROS AND ARRAYS

We transposed many of the COVID-19 data tables to be able to merge them using patient ID to create the daily export and historical files. With so many variables contained in some of the tables, it became necessary to develop macros for transposing.

Macro Example for Transposing

One of the tables that we transposed was the table containing 15 fields for hospitalization details, including hospital address, discharge and admit dates, and whether the patient was admitted to the hospital and ICU. This table was called NCOVPUIHospDtl. Because investigators indicated if a patient was admitted to the ICU by selecting “Yes” in either of two fields (NCOVPUIHospDtlICU and NCOVPUISxICUadmit) contained in two separate tables (NCOVPUIHospDtl and NCOVPUISx), we needed to derive a variable to indicate if the patient was in the ICU. We used an array to compute this variable. We renamed the field NCOVPUIHospDtlICU to hospicu so it did not share a prefix with any of the other fields in our dataset. If NCOVPUIHosplICU was used in the array, the array would try to process any field with the prefix NCOVPUIHospDtl. Here is the macro to transpose the hospitalization information:

```
data hospital3;
set hospital2;
rename NCOVPUIHospDtlICU=hospicu;
run;
```

```
%macro tdata;
%let vars= NCOVPUIHospDtlName NCOVPUIHospDtlAddr NCOVPUIHospDtlCity
NCOVPUIHospDtlState NCOVPUIHospDtlZip NCOVPUIHospDtlPhone NCOVPUIHospDtlMRN
NCOVPUIHospDtlAdmitDiagno NCOVPUIHospDtlAdmitDt NCOVPUIHospDtlDischDiagno
NCOVPUIHospDtlDischDt hospICU NCOVPUIHospDtlICUAdmitDt
NCOVPUIHospDtlICUDischDt NCOVPUIHospDtlIntub;

%local i next;
    %do i=1 %to %sysfunc(countw(&vars));
          %let next = %scan(&vars, &i);
          Proc transpose data=hospital3 out= hospitaltranspose&I
(drop=_name_ _label_) prefix=&next;
          by ID;
          var &next;
          run;
    %end;
%mend tdata;
%tdata;

data hospitaldata;
    merge hospitaltranspose:;
    by ID;
    run;
```

Array Example for Computing New Variable

We developed arrays for: creating an ICU variable; tagging migrant cases; identifying K-12 school cases; flagging hospitalized cases on mechanical ventilation; indicating outbreaks in congregate and community settings; and identifying the sequence, laboratory, collection and result dates, and accession numbers for cases sequenced for variants. Below is the example of the variable created to indicate if the patient was in the ICU described above:

```
data widehosp;
merge symptoms hospitaldata;
by id;
wasicu=0;
run;

data widehosp2;
set widehosp;
    array icudata(*) hospicu:;
    do i = 1 to dim(icudata);
    if icudata[i]="Yes" or NCOVPUISxICUadmit="Yes" then wasicu=1;
    end;
    drop i;
run;
```

PROC APPEND FOR COMBINING RECENT REPORTS WITH HISTORICAL DATA

Appending recent data to the historical file was achieved from a simple line of code using the PROC APPEND function:

```
proc append base=historicalfile data=dailyexports force;
run;
```

PROC DATASETS FOR SAVING SPECIFIC SAS DATASETS

Prior to running PROC APPEND, we also found it helpful to clear from memory any unnecessary files by saving only those files that were needed for further processing using the SAVE option with PROC

DATASETS. It is important to note that hospitalization and death files were maintained separately and read in as Excel files due to the complex nature of hospital and death investigations. Also, tables containing outbreak information were not part of the historical data file because it was manageable to export all data in real-time. Therefore, we merged COVID-19 hospitalizations, deaths, and outbreaks with the appended file after running PROC APPEND:

```
proc datasets lib=work memtype=data nodetails;
save historicalfile dailyexports hospitalizations deaths outbreaks;
run;
```

WORKING WITH MISSING, INACCURATE, AND DUPLICATE DATA

We were committed to ensuring the COVID-19 data reporting was both timely and high quality. To that end, our programming included checks for missing, inaccurate, and duplicate data. However, we had to implement most quality assurance processes outside of daily processing given time and resource constraints. Quality assurance (QA) efforts were numerous and extensive involving designated teams to handle such efforts as merging or deleting records, correcting lab information, and updating hospitalization and death records. We also sent individual investigators twice weekly reports that were flagged for any missing or inaccurate data in their patient records that they were tasked with correcting. Additionally, we developed a separate program for de-duplication that entailed probabilistic matching of patient records. These efforts are beyond the scope of this paper.

The QA checks that we were able to implement and address for daily processing included checks for: duplicate outbreak numbers; resident status; missing and inaccurate dates for date received, episode date, hospital admit dates, and lab collection dates; and exact matches for name and date of birth for deduplication and/or reinfection identification.

As a result of QA, we hard coded in some cases, hospitalizations and deaths; modified resident status; and corrected dates. We hard coded out: false positive lab results; merged records from the previous day; and some serology results. The hard codes were important so we could immediately report on the changes, but we also coordinated with the investigators to make updates to the records in WebCMR as well.

Flagging Resident Status Using Macros

We only reported COVID-19 case data among San Diego County residents. Therefore, it was necessary to verify resident status and city of residence as part of COVID-19 data processing. We constructed a variable for resident status based primarily on address information (residents: residentstatus=1, non-residents: residentstatus=2). We then constructed four macros for cities, unincorporated areas, neighborhoods, and census-designated places that accounted for common misspellings. If we had coded a patient as a non-resident but the patient's city field was found in one of the macros, we flagged the patient for non-resident QA. Alternatively, if we had coded a patient as a resident but the patient's city field was not in one of the macros or was listed as unknown, not provided, not applicable, confidential, or homeless, we flagged the patient for resident QA. We maintained and read in an Excel list of patient IDs with verified resident status and compared these with our current dataset. If the patient was flagged for resident or non-resident QA and was not found in the verified list, we used a PROC PRINT command to list the IDs for us to check. We worked with the investigators to correct any missing or inaccurate information in WebCMR, added any IDs that we were able to verify to our Excel sheet, and hard coded resident status for some cases based on the outcome of our QA. Here's a modified example of what we used for our resident status QA check, showing one abbreviated macro only and none of the special conditions for simplicity:

```
%let cities =  'SAN DIEGO', 'CHULA VISTA', 'CORONADO', 'NATIONAL CITY',
'IMPERIAL BEACH', 'IMPERIAL BCH', 'SAN YSIDRO', 'SOLANA BEACH', 'VISTA',
'CARLSBAD', 'ENCINITAS', 'OCEANSIDE', 'CARDIFF', 'CARDIFF BY THE SEA',
'ESCONDIDO', 'POWAY', 'SAN MARCOS', 'PAUMA VALLEY', 'PALA', 'SANTA YSABEL';

PROC IMPORT OUT= verifiedlist DATAFILE= "C:\Verified_IDs_residentstatus.xlsx"
DBMS=XLSX REPLACE;
```

```
GETNAMES=YES;
RUN;

data resqacheck;
set  covidcases;
if residentstatus=2 and city in (&cities) then nonres_qa=1;
if residentstatus=1 and city not in (&cities) then sdc_qa=1;
run;

data covidcases2;
merge covidcases(in=a) verifiedlist(in=b);
by id;
residentqa=0;
if a;
if b = 1 then residentqa=1;
run;

proc print data=covidcases2 noobs;
var id;
where SDC_QA=1 and residentqa=0;
TITLE 'This is the QA for cases currently coded as residents';
run;

proc print data=covidcases2 noobs;
var id;
where NONRES_QA=1 and residentqa=0;
TITLE 'This is the QA for cases currently coded as non-residents';
run;
```

De-duplication and Reinfection Identification

The largest COVID-19 surge in January 2022 required that the County quickly automate more of the intake of COVID-19 lab reports, which had the unfortunate consequence of duplication of many patient records. We resolved de-duplication of patient records with two separate programs described above. Our SAS program was able to identify and remove exact matches on first name, last name, and date of birth among confirmed and probable cases, while keeping reinfections and preserving the variant sequencing information for confirmed cases:

- We used FIRST. and LAST. processing with the COUNT function to number the records and to indicate first and last records within each group based on first name, last name, date of birth, case classification, episode date, and patient ID.
- We used IF-THEN-DO statements to flag the duplicates.
- We computed the difference in days between episode dates using the LAG and INTCK functions to identify reinfections to keep. Reinfections are cases with episode dates more than 90 days apart.
- We created a subset of duplicates with genomic sequencing information and tagged the first sequencing result to keep. Sequencing information is only found for confirmed cases and is used to identify which variant of SARS-CoV-2 is present in the lab specimen.
- We merged the data back with the main data file. We then removed duplicates for further processing of the data.

CONCLUSION

SAS was an important tool used by epidemiologists at the County of San Diego for managing and reporting COVID-19 data during the COVID-19 pandemic. Using a variety of SAS techniques, we

overcame the challenges of being new SAS users and streamlined the process for analyzing and reporting on COVID-19 case data that enabled us to meet significant demands for COVID-19 case information. The techniques can be used generally to handle large volumes of any type of data and are scalable to future communicable disease outbreak situations. Because there continue to be new and existing public health challenges, further improvements are also being made for processing communicable disease data. The logical organization for writing code and processing data resulting from this experience provides a framework for mocking up code for other diseases, is useful for the development of dashboards, helps establish schedules for reporting, and can be adapted and manipulated quickly for further data requests.

REFERENCES

Bettinger, Ross. 2007. Sample 24804: %SQUEEZE-ing Before Compressing Data, Redux. Cary, NC: SAS Institute Inc. Available at: https://support.sas.com/kb/24/804.html.

CONTACT INFORMATION

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

Whitney Webber
whitney.webber@sdcounty.ca.gov

Nathaly Moran
nathaly.moran1@sdcounty.ca.gov

Jacob Ritz
jacob.ritz@sdcounty.ca.gov

Fatema Sakha
fatema.sakha1@sdcounty.ca.gov

Jacquelyn Ho
jacquelyn.ho@sdcounty.ca.gov

Jennifer A. Nelson
jennifer.nelson2@sdcounty.ca.gov

Jeffrey Johnson
jeffrey.johnson@sdcounty.ca.gov