COVID-19 Specimen Data Processing: manifest_code.R

IMPORTANT: All manifest file names should follow the same formatting.

<SourceLocation>_<YYYYMMDD>_<#>.csv

SourceLocation = Where the file is from, should correspond to the "Manifests" subfolders [CBR, Martin, CSTP]
YYYYMMDD = When the manifest and samples on the list were received
# = 1, 2, 3, ..., n ; If more than one manifest & batch of samples are received on the same day, this should differentiate the lists

Libraries Needed:
library(tidyverse)
library(lubridate)
library(janitor)
library(openxlsx)
library(withr)

Overview: This code file pulls in all manifest files received from sample sources. Data checks are completed, and additional columns are added.

Fill in **starting_path**. This is the path from your own machine to Box/DropBox

Ex. "C:/Users/juliegil/Box Sync/"

Fill in Manifest folder paths (there should be a separate folder for each location that sends sample manifests)

cbr_manifest_fp = "SampleMetadataOrganization/Manifests/CBR"
martin_manifest_fp = "SampleMetadataOrganization/Manifests/Martin"
cstp_manifest_fp = "SampleMetadataOrganization/Manifests/CSTP"

List of all the manifest folder paths [**manifest_folder_list**] is created. This is iterated over later, to ensure all files from all locations are compiled.

Fill in **outputLOC**, the output location of the manifest file compilation

"SampleMetadataOrganization/Manifests/ManifestsComplete"

Created as empty dataframe first, then gets filled in later in our process.

manifest_storage

Iterate through every folder in **manifest_folder_list**

Store the name of every .csv file in the folder in **file_list**

IMPORTANT: This will need to be altered if manifests begin coming in a different file format (.xlsx, .tsv, etc.)

Iterate through every file in **file_list**

Read in the file as a dataframe [**file_in**] with all columns as character data type

We're trying to avoid losing leading zeros in our subject_ids (MRNs and/or UMIDs)

Any cells that contain only "" or " " are converted to NA, so that remove_empty() works to take away any completely blank rows

This sometimes occurs with .csv files that have been opened/manipulated in Excel

Store the names of the columns of **file_in** inside **column_name_check**

Acting under the assumption that the manifest files have been pre-screened for column names; **true_columns** = "position", "sample_id", "subject_id", "coll_date", "flag"

IF true_columns and column_name_check are identical

Yes

Nothing needs to be done

No

IF they aren't identical, but there are still 5 columns

Yes

Rename the columns of **file_in** with **true_columns**

Acting under the assumption that the names may have spaces in them, or are mis-spelled, rather than being in the incorrect order

No

IF they aren't identical, and there are not 5 columns

Yes

The difference between **true_columns** and **column_name_check** will be printed to the console, along with a warning and a file location

Code will STOP executing if this occurs

No

IF any **sample_id**'s are NA

Yes

Code will STOP executing and print a warning to the console

No

IF any **subject_id**'s are NA

Yes

Code will STOP executing and print a warning to the console

No

IF any **coll_date**'s are NA

Yes

Code will STOP executing and print a warning to the console

No

IF test_date_format turns into NA when attempting to coerce it into a number

Yes

Assume the **coll_date** column is in "%m/%d/%y" format and coerce it into R date format (%Y-%m-%d)

The first 4 characters of the cell in row 1, column 4 of **file_in** are stored in **test_date_format**

If the manifest file structure changes order, this will need to be altered; This should correspond with the first row entry of **coll_date**

This may need additions if more alternate date structures are found in the manifests

No

The file name is split on "_" and the second item is kept as **rec_date**, a character string with all leading and trailing whitespace removed

rec_date is then reformatted into [first four characters]-[characters 5 and 6]-[characters 7 and 8] to form a date

rec_date then gets assigned as the value for every row in a new column of **file_in**, called **received_date**

The file name is again split on "_" and the first item is kept as **rec_source**, a character string with all leading and trailing whitespace removed

rec_source then gets assigned as the value for every row in a new column of **file_in**, called **received_source**

file_in is row bound to **manifest_storage** and kept as **manifest_storage** to continue to collect new rows

Only unique, distinct rows are kept in **manifest_storage**

At this point in the code, we have all our unique specimen receipt rows from all our sources, compiled into manifest_storage. Following this, we're going to flag some rows that are potentially of concern, write out our final compiled manifest receipt list, and write out a report of what we've got for record-keeping

Store the number of unique combinations of **sample_id**, **subject_id**, and **coll_date** from **manifest_storage** in **unique_ids**

IF unique_ids is not equal to the number of rows in **manifest_storage**

Yes

Get the count of each combination of the three variables, and use those to identify the duplicate entries as **duplicate_ssc**.

Alter **flag** for these rows in **manifest_storage** to mark them with "Duplicate Sample - Subject - Collection"

If we assume that every row is a unique instance of **sample_id**, **subject_id**, and **coll_date**, then we'd expect the row count of **manifest_storage** to match **unique_ids**. If they don't match, we need to identify the duplicates

No

Create **subject_id_length** on **manifest_storage**, to look at the number of characters in each **subject_id**

Current assumptions are that samples from CBR should be 9 characters long (as MRNs) and samples from CSTP should be 8 characters long (as UMIDs). If more information is learned about the samples, additional checks should be added for these next steps.

Add "MRN < 9 digits + leading 0s restored" to **flag** where **received_source** is "CBR" and **subject_id_length** is less than 9

Leading zeros are added to **subject_id** for those marked rows

Add "UMID < 8 digits + leading 0s restored" to **flag** where **received_source** is "CSTP" and **subject_id_length** is less than 8

Leading zeros are added to **subject_id** for those marked rows

manifest_storage is written out as a csv at the **outputLOC** as sample_full_manifest_list.csv

manifest_output_report_template.xlsx is read in from **outputLOC**

The date, number of columns of **manifest_storage**, and number of rows of **manifest_storage** are added to the SUMMARY tab. If there are duplicate rows, then **duplicate_ssc** is written out to the DUPLICATES tab. Any instances of added zeros to MRNs or UMIDs are written out to the RESTORE_ZEROS tab.

The populated report is then written out to **outputLOC** as manifest_output_report_YYYYMMDD.xlsx.