**MergeData.R Documentation**

Table of Contents

[Purpose 1](#_Toc521417508)

[Overview 1](#_Toc521417509)

[How to Use 2](#_Toc521417510)

[Detailed Explanation of Data Tidying and Transformation 3](#_Toc521417511)

[User Inputs 3](#_Toc521417512)

[Data Import: 3](#_Toc521417513)

[MCS: 3](#_Toc521417514)

[DM: 4](#_Toc521417515)

[FP: 4](#_Toc521417516)

[Merging 4](#_Toc521417517)

[Added Columns 4](#_Toc521417518)

[Limitations and Future Iterations 4](#_Toc521417519)

Purpose**:** The R script merges all three data sets from the manufacturing process, MCS, Matrix Reader, and Inspection. The merged data connects hot end data to cold end data for analysis.

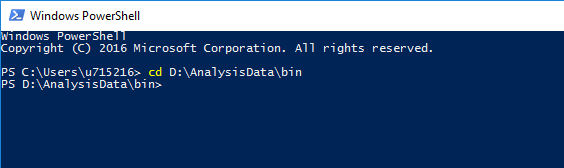
Overview**:** The script reads in all three datasets from the VM’s Import folder for the date that the user specifies upon input. Then, the three datasets undergo a process of tidying and transformation which is explained in detail in “Detailed Explanation” section. After this, the data undergoes a merging process, explained in the following paragraph.

The Matrix Reader and Inspection Reader data are combined by column/row for row, since it is assumed that the bottles go through each reader in the same order. Then, the combined data is merged with the MCS data by the CreateTime and Gob variables. The CreateTime variable in the MCS data is the read time from the MCS machine and the CreateTime variable from the Datamatrix reader is the creation timestamp extracted from the serial number. We merge the datasets upon the assumption that the bottle’s creation in the Datamatrix Reader data happens at the same (within the minute) that the bottle was recorded by the MCS Machine.

After the three datasets are merged, the columns ShldrRatio, BodyRatio, HeelRatio and SplitFinishReject are created for use in analysis process. Finally, the merged data is exported to the *Export* folder in the VM as a .csv file.

# How to Use

1. Open Windows PowerShell and type **cd D:\AnalysisData\bin** to change to the correct directory:



1. Execute the Rscript by typing this line of code and press [Enter]:

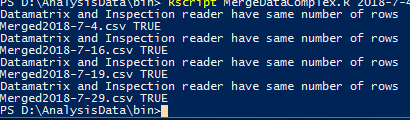
**Rscript MergedData.R date1 date2 date3 … etc.**

The dates should be in the format of *yyyy-mm-dd* and separated by a single space. You can reference the correct format with the screenshot below. Once you have typed this with the desired dates for merging, hit enter.



Note: The necessary R packages for this program should already be installed and stored in the “R-Packages” folder. If you do not see this folder in the D drive or run into errors attributed to packages while executing the program, refer to server setup.

1. Once you enter the PowerShell command, the output on the screen should look like this:



* The first line tells the user if any rows from the Datamatrix or Inspection data were deleted. The message in the example tells us that no rows were deleted.
* If the execution was successful, the program will return “Merged[date].csv TRUE” . This message tells us that the merged data file is exported to the Export folder on the Serialization(D:) drive.

# Detailed Explanation of Data Tidying and Transformation

User Inputs: The arguments input to the script from PowerShell are stored as a vector of strings in the variable date within R. Then each argument stored in date is called to set the directory to the corresponding folder of data for that input. This is done by pasting the date string to the full path to the Import folder. Note that since the full path is specified in the Rscript, if the hierarchy of the folders in the VM are changed, the Rscript must be altered so the path to the Import folder is correct. The same goes for the Export folder.

Data Import: The program identifies the names of all the files stored in the dated folder with the function list.files(). Then uses string manipulating functio/ns to import the log files from the Datamatrix folder identified with “DM”, inspection files identified with “FP” and hot end files identified “MCS1” or “MCS2”. Since the Datamatrix reader data exists across multiple log files, the merger combines all the files into one dataset when being read in.

MCS: The MCS data is stored in separate datasets organized by attribute. While being read in, the first two lines are skipped, since they yield a conflicting number of columns, and the column names are assigned. Once the data is read in, all duplicate column names with null values are removed from each of the MCS datasets. Then, the three sets are merged together by time. However, the column names gob specific, e.g. “S64 GA-DIA” and “S64 GB-DIA” . This will not merge well with the cold end data, since each row of cold end data are gob specific. Therefore, the MCS data should be transformed to the rows rather than columns being gob specific so it can merge with cold end data.

To start this process, the newly combined MCS is split into two separate datasets again, but this time it is by the gob specified in column names rather than by attributes like diameter and length . Each new MCS by gob dataset should include Time, F-BTM and all variables with the correct gob. After that, a gob column is added to each and populated with the correct gob indicator (A, or B) and “GA-“ etc. are removed from the column names. Once all of this is finished the A or B gob specific MCS datasets are bound by row or, in other words, ‘stacked’ upon one another. Now we have a workable MCS dataset to merge with the cold end data.

DM: The empty fields are removed and the column names are assigned to the data during the read in process. In the DM section, the serial number is formatted to remove quotes and to assign it a character class. In addition, the ‘True’ and ‘False’ elements of the Read Value column are changed to all caps (‘TRUE’ and ‘FALSE’) so that it is recognized as binary. Then, all of the information contained in the Serial number is extracted into their own columns, and from this, a timestamp is made to indicate the time that the bottle was created.

FP: The unique reader number from the Datamatrix data is found and is used to filter the FP data for the corresponding FP machine. The merger currently only accounts FP3. If more should be added, the Rscript must be edited. Then, the Section and Gob are extracted from the Position variable to create their own columns.

Merging : First, the DM and FP datasets are merged into a table called Temp. The DM and FP are most likely of different lengths, so the difference between the lengths of the tables are calculated. And this value is passed through some if statements to determine which set is bigger. The bigger dataset is subset to the size of the smaller. Then the two are ready to merge. During the merging of these two, the time column from the FP is removed since two columns of the same name prevent the datasets from being combined by column. If this is an issue, the Rscript can be altered to change the names of the timestamp columns prior to merging DM and FP. But the SQL table will have to be adjusted to account for this. After that, a full right join is performed to merge the temp table and the MCS data by CreateTime and Gob. After the merged data is created, the columns for analysis are added.

Added Columns: The columns that are added to the data are:

* AvgShldr = (OTG\_Shldr\_Min\_Thick + OTG\_Shldr\_Max\_Thick)/2
* AvgHeel = (OTG\_Heel\_Min\_Thick + OTG\_Heel\_Max\_Thick)/2
* ShldrRatio = OTG\_Shldr\_MaxThick/OTG\_Shldr\_MinThick
* BodyRatio = OTG\_Body\_MaxThick/OTG\_Body\_MinThick
* HeelRatio = OTG\_Heel\_MaxThick/OTG\_Heel\_MinThick
* SplitFinishRejects : equals 1 if the variables CHECK\_VERT\_1 or CHECK\_VERT\_2 equals 1.

All of the new columns are utilized during the analysis process in BI, with the exception of AvgShldr and AvgHeel. These columns were catered to a previous iteration of analysis but were kept in case they would be of interest in the future. If there are more metrics that need to be added in analysis, columns can be created and calculated in BI. New metrics could also be calculated in R, but the SQL table will also have to be adjusted.

# Limitations and Future Iterations

Overall, this merger script was designed to process data from the Sao Paolo plant running through FP machine 3. Therefore, the merging process is limited to processing metrics collected for this plant from these specific machines. This Rscript will work for future data as long as they are in the format of the three files do not change and the assumptions on the relationship of the data remain true. The complications of these topics and how they alter the effectiveness of the merge are discussed in the following paragraphs.

The current merger relies on two assumptions: MCS read time and timestamp from the serial number correspond to one another, and the FP and Matrix Reader data are connected row for row due to same order. However there are no methods that mitigate the potential issue of these assumptions being false. The primary issue of the two is the relationship between the Matrix Reader and FP data. There are possibilities of a bottle being read more than once by the FP machine after having passed once through the Matrix Reader. In addition, there is also the very slim chance that a bottle is removed between going through the Matrix Reader and the Inspection Machine. To improve this program, the logic for merging of the three datasets should accommodate for these events. And in turn, the code in the Rscript to be altered to match the improved logic.

During the data reading and tidying, the names of the MCS and the Datamatrix Reader are being assigned within the script. The MCS names are assigned in the script because there are two conflicting lines of headers. By skipping the first two lines, R is able to create a data frame with the correct columns. However, by skipping these two lines, the column names must be added back in which is done by assigning names that the program reads just the first line of the MCS. Note that when the first line of the MCS file is read, the last character has always appeared to be NA, so the program takes that out when assigning the column names. The only error that is foreseen as an issue is if NA is not the last element of the first line. If this happens, the Rscript must be edited. For the Datamatrix Reader, none of the headers are included in the txt file, so the column names must be assigned by the script. Therefore, if the order of the columns, or the column names get changed in the Datamatrix Reader data, there is no way for the script to automatically accommodate for it. Rather, someone will have to re-code those sections in the script to change the column names, or the order of the columns.

This script is meant to process just gobs A and B for the purposes of the first iteration. However, a second script named *MergeDataComplex.R* can also be found in the bin folder. The purpose of this script is to expand the capabilities of the *MergeData.R* script. It can process up to four MCS files and four gob types(A-D). Whereas the *MergeData.R* script processes exactly two MCS files and two gobs, A and B for the purpose of the first iteration. The areas of the code that differ between the two files are the Read Data and the Combine MCS sections. In Read Data, the number of MCS files in the dated import folder are automatically detected and the corresponding files are read in. Then in the Combine MCS section there are added capabilities to provide data tidying and transformation to the correct amount of MCS files being processed. Finally, the Combine MCS section correctly identifies which gobs are represented in the MCS data and creates the correct gob separated tables. Other than these two sections, the process and limitations of the program are the same as the *MergeData.R* script. To run the *MergeDataComplex.R,* use the same steps of the tutorial, but instead of Rscript MergeData.R type the command Rscript MergeDataComplex.R with the desired dates as arguments.