NZReport.R Documentation

Table of Contents

[Overview 1](#_Toc522089469)

[How to Use 2](#_Toc522089470)

[1. Open File 2](#_Toc522089471)

[2. Packages (Optional) 2](#_Toc522089472)

[3. User Input 2](#_Toc522089473)

[Set path: 3](#_Toc522089474)

[Rejected Bottles (Table 2) input: 3](#_Toc522089475)

[4. Run the code: 3](#_Toc522089476)

[Detailed explanation of code 4](#_Toc522089477)

[Read Data 4](#_Toc522089478)

[Table 1 Read Totals : 4](#_Toc522089479)

[Table 2 Rejection Tracking : 5](#_Toc522089480)

[Table 3 DMX Code Check : 5](#_Toc522089481)

[Table 4 IPU vs. Reader : 5](#_Toc522089482)

[Export : 6](#_Toc522089483)

[Copying file path 6](#_Toc522089484)

[Method 1 6](#_Toc522089485)

[Method 2 7](#_Toc522089486)

[Editing Raw Files 8](#_Toc522089487)

[Navigating Rstudio 8](#_Toc522089488)

[Set-Up 8](#_Toc522089489)

[Features 9](#_Toc522089490)

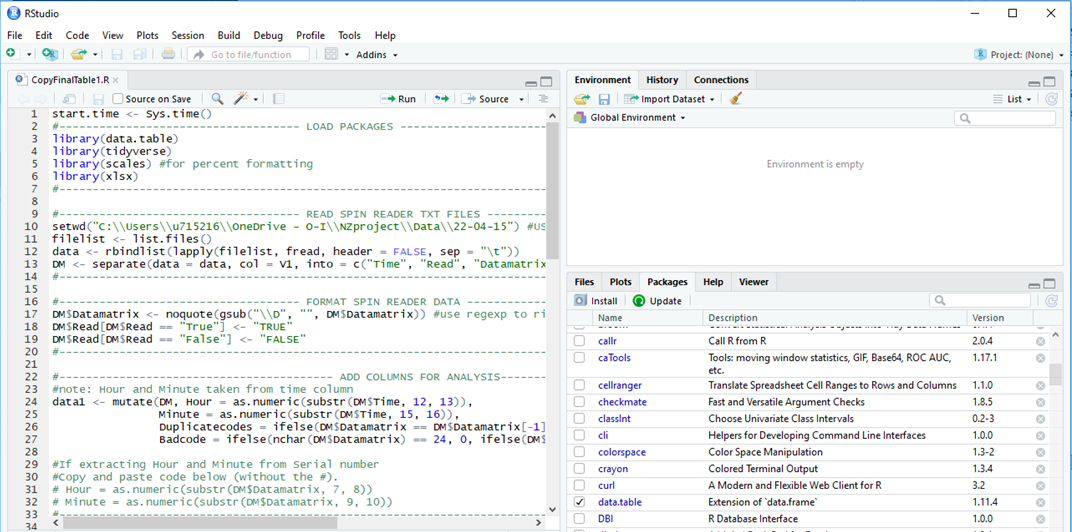
Overview: This program is designed to automate analysis reports for the performance of the Inspection and Reader machines for the New Zealand plant. The main focus is to analyze the reads on the serial numbers. The program reads one day’s worth of data from the spin reader and the IPU and outputs an excel workbook report on read rates, rejection tracking, code checks, and mismatching codes. The user inputs required are paths to files, file names, and bottle criteria for rejection tracking. The program was written in R language using Rstudio/R. The packages that this program uses are data.table, tidyverse, scales, xlsx.

How to Use:

1. Open File : First, open the Rstudio app.

Then go to Files->Open File, choose CombinedReport.R and click open.

1. Packages (Optional) : Check that the packages needed for the program are installed. You can look at the Packages list on the bottom right section of the app. If you need to install the packages do so at this step and see the *Installing Packages* documentfor additional help.

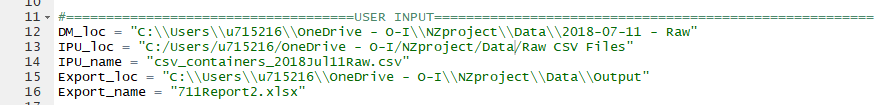


1. User Input : This section requires input of file name/location and bottle criteria for the rejection tracking table. The input for file information requires you to get the directory to your files. Some methods for this are demonstrated in the [*Copy File Path*](#_Copying_file_path)section of this document.

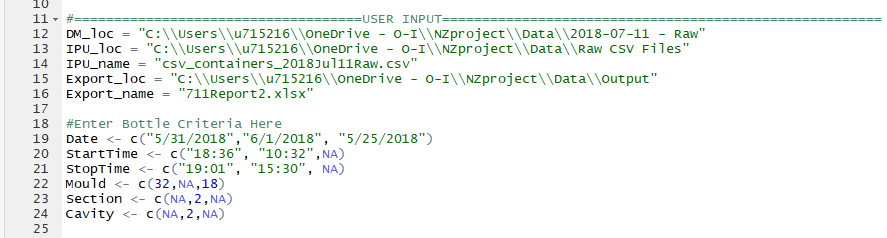
Set path: Copy and paste the directory into the following variables

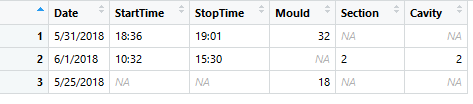
* **DMdata\_loc** : location of the reader data logs
* **IPUdata\_loc** : location of IPU data
* **IPUname :** Name of the IPU .csv file.
* **Export\_loc:** location for storing the exported excel workbook
* **Export\_name:** Name of .xlsx file

Important note: When inserting a directory (e.g. “C:\Users\u715216\OneDrive - O-I\NZproject\Data\22-04-15 “) you must change all \ characters to \\ or /. This is because on windows machines, R will read \ as an escape character.



Rejected Bottles (Table 2) input: Insert your queries in the format shown by the example below. All values separated by commas. The grouping of bottle criteria is by order of entry. All first entries are a set of bottle criteria, second entries are another set, etc. In other words, if you were to make these inputs into a table, all first entries would create a row and the program filters for bottles that fit the criteria of that row.

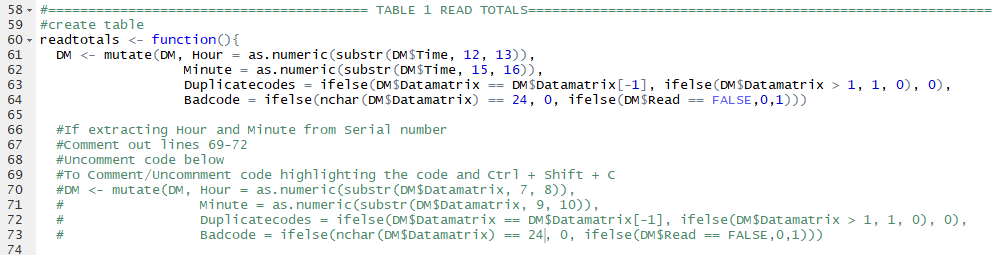




For formatting, Dates and Times in quotations in *mm/dd/yyyy* and *hh:mm*, respectively. If you have nothing to fill in, type *NA.*

1. Run the code: After completing the previous steps you can run the code with this keyboard shortcut Ctrl + Alt + R or use the toolbar: Code -> Run Region -> Run All

Note: The Hour and Minute Extraction for Table1 is default to taking the Hour and Minute from the Time variable. But if you would like these measurements to be extracted from the Data matrix code instead, edit the “Table 1 Read Totals” section and follow the instructions within the code.



Detailed explanation of code:

Read Data : All of the bottle criteria is stored in its own input table. Then, the Data matrix logs are imported by the script from the folder location that the user set. It reads all the logs in order, skipping the ones that are empty and appends them into one dataset. Then the data is formatted to get rid of the quotes around the serial number, makes true/false all caps, and sets the empty serial numbers as 0. After that, the IPU data is imported according to the name and location that the user provides. The empty serial numbers are set to 0 and the information contained in the serial number is extracted and put into their own columns.

Table 1 Read Totals : The creation of the table occurs within the *readtotals* function to keep the transformations and reformatting of the Data matrix data exclusive to this section. First, the Data matrix data is mutated to add *Hour*, *Minute*, *Duplicatedcodes*, and *Badcode* columns. Note that *Duplicatedcodes* and *Badcode* serve as metrics to calculate the remaining variables, but are not displayed in the final table. The hour and minute are defaulted to extract from the Time variable, but additional code is provided in the Rscript if these should be extracted from the data matrix code instead. Then, a for loop is used to populate the columns: *Date*, *GoodReads*, *BadReads*, *DuplicateReads*, *BadCodes* with their totals per Hour. The for loop runs its calculations for every hour in between the first hour read in the data to the last hour read in the data. Next, the columns TotalReads and RawReadRate are calculated and added to the table. The table below details how each of these added columns are calculated.

|  |  |
| --- | --- |
| **Variable** | **Calculation** |
| Duplicatedcodes | 0 if consecutive pair of serial numbers are unique, 1 otherwise. |
| Badcode | 0 if code is 24 digits, 1 otherwise |
| GoodReads | Count of read value TRUE |
| BadReads | Count of read value FALSE |
| DuplicateReads | Sum of *Duplicatedcodes* |
| TotalReads | *GoodReads* + *BadReads* |
| RawReadRate | (*GoodReads + DuplicateReads + BadCodes)/TotalReads* |

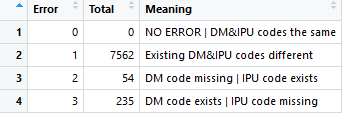
Table 2 Rejection Tracking : This table is created with a function that takes the bottle criteria input as its argument. Then the function loops through each row of input criteria to filter for the bottles with the given attributes.

The attributes are filtered in order of input into a temporary data frame named *temp*. First, the IPU data is filtered for the date. It is assumed that the date will always be provided, so unlike the rest of the attributes, there is no accommodation for a missing date in the code. Second, the function filters for time, however if start and stop time are not provided it will skip to the next attribute. Furthermore, it is expected that when times are provided, it will always be a pair of start/stop times. So if only one time (start or stop) is provided, the program will not filter for time. Third, the data is filtered for the correct *Mould*, which in the IPU table is identified under the variable *Cavity*. If the input for *Mould* is NA, this filtering step is skipped. Finally, the data is filtered for *Section* and *Cavity* which in the IPU are the variables *Section* and *Gob* that were extracted from the data matrix code. Similar to the Time input, if one or both are NA, then the filtering for these attributes is skipped.

After the IPU data is filtered for the correct attributes, it records the totals for *Inspection,* *Rejection, Error, Nocavity* and *AutoRejected*. The table displays the user input criteria along with the calculated totals.

Table 3 DMX Code Check : This table filters through the Inspection (IPU) data to return a summary of the total reads that reported an empty code and reported 1 for the columns *DmNoPacket, DmBadRead, DmCodeError.*

Table 4 IPU vs. Reader : This table is also made within a function to keep its transformations to the datasets contained to this section only. The Inspection and Reader data are reformatted to rename their *Time* and *Datamatrix* columns to be unique to the dataset it originated from and truncates the Reader time to seconds to match format of the Inspection times. In addition, just the *Time* and *Datamatrix* columns from the Inspection reader are selected. Then, the program compares the lengths of the datasets and deletes the last rows of the longer dataset to match the length of the shorter. Furthermore, the program outputs a message stating how many rows were deleted from which dataset. After that, the Inspection and Reader data are combined row for row so that their data matrix codes can be compared. Finally, the program creates two columns, *MisMatch* and *Error*. *MisMatch* returns a 0 if the codes are the same and a 1 if they are different. *Error* returns a number 0-3 to categorize the type of miss-match that occurred. The table 4 outputs this created table along with a legend that provides an explanation of the Error categories and a sum of rows for each error. Example of the legend is shown below.



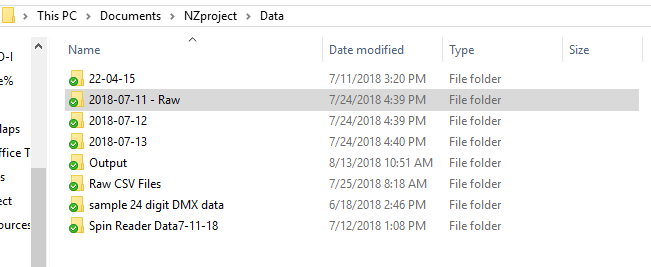
Export : The working directory is reset to the location that the report of all the tables will be exported to. Then, each table is written to its own excel sheet which will be exported in the one excel workbook.

# Copying file path

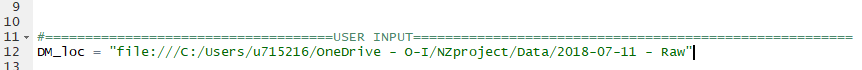
This section covers some different ways you can get the full path to copy and paste into the user input. None of these are necessarily more efficient than the other. Choose whichever is comfortable or straightforward. For each method I will be using the retrieval of the folder containing all the data matrix logs as an example.

## Method 1

1. Go to your file directory and locate the folder that you want the path for.



1. Right click on the folder and choose **Copy**
2. In the Rscript, go to the User Input section and **Paste** into the DM\_loc variable.



Notice that the path already has the separator character changed to the forward slash. This saves you the trouble of replacing all the \ to / or \\. However, Notice that the file:///before C:

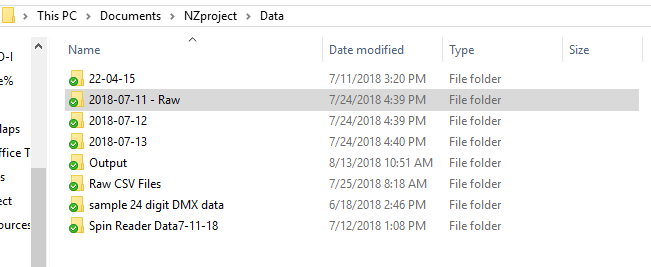
1. Highlight the file:/// and delete.



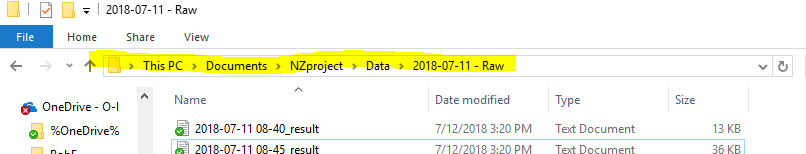
Now you have successfully defined the path to the folder for the data matrix logs.

## Method 2

1. Go to your file directory and locate the folder that you want the path for.



1. Enter the folder and locate the address bar.



1. Right click on the address bar and choose **Copy address** or **Copy address as text**
2. In the Rscript, go to the User Input section and **Paste** into the DM\_loc variable.



1. Change all the \ characters to \\ or /





Now you have successfully defined the path to the folder for the data matrix logs.

# Editing Raw Files

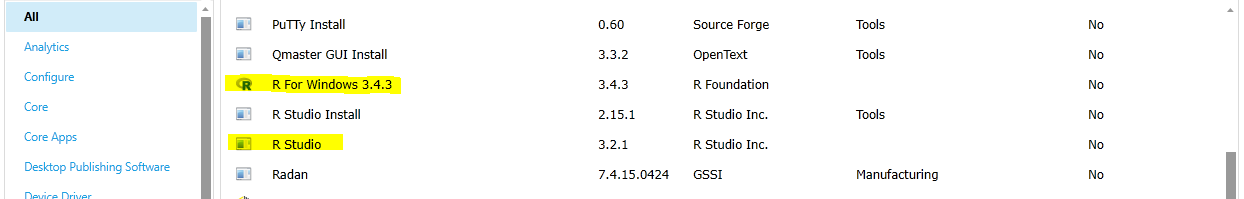
When editing files, be aware that when you delete lines the new line character might also be deleted in the process. Therefore, ruining the program’s ability to read the files correctly. If you see that the program is struggling to read a file that has been edited, try editing the raw file again and indent a new line at the part of the data where the program fails to read properly. The best way to avoid this issue is to use a more advanced text editor when editing the data files like Notepad++, VSCode or other equivalent editors.

# Navigating Rstudio

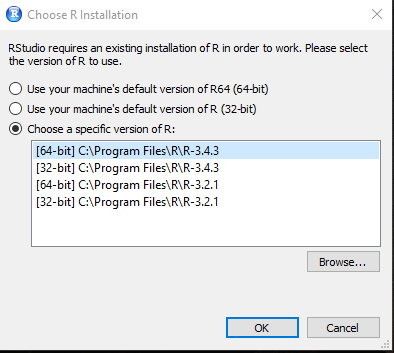
## Set-Up

I wrote this documentation under the assumption that the user already has the software, but if they do not, don’t fret! Here is a guide to getting everything set up to run the script.

First, download R and Rstudio from the O-I catalog (highlighted in picture below). It is found on page 14 of the ALL section.



Rstudio should automatically be synced to the most updated version of R that you have downloaded, but if you want to check, hold down Ctrl while opening the Rstudio app. The following window will pop up, and then select the most recent version of R.

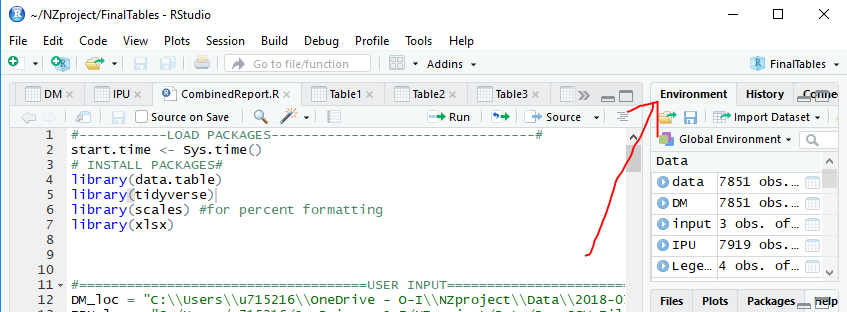


After this, Rstudio is ready for use. I would suggest installing the packages to R during the set up process, so that missing packages are not an issue when running the program. The packages needed for the program are listed in Overview at the beginning of this document. But an additional package named ‘bit64’ is needed for R to process large numbers. This is not called into the program, but it works in the background so that large numbers like the serial codes can be displayed and processed in full.

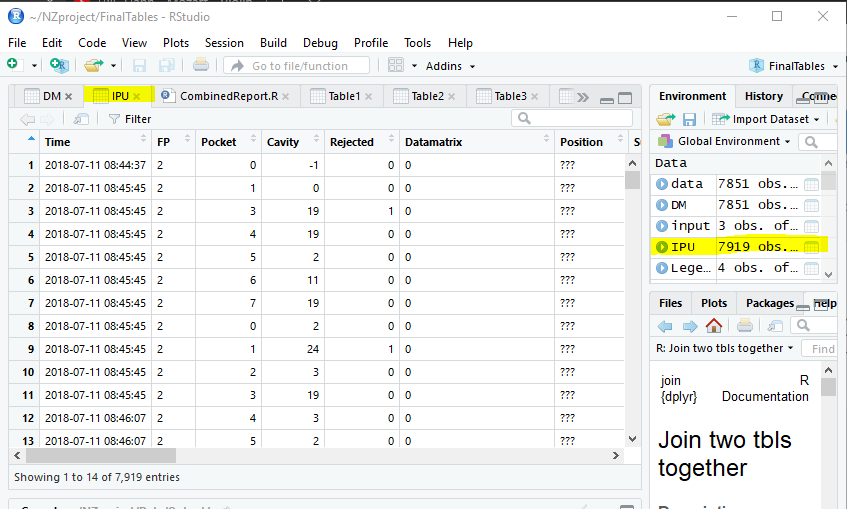
## Features

Although you will most likely only use Rstudio to give user inputs and run the code, you can also utilize this app to have your own look at the data and the code. Two important features that might be useful are Environment and Document Outline.

Environment is located in the upper right hand corner of the console.

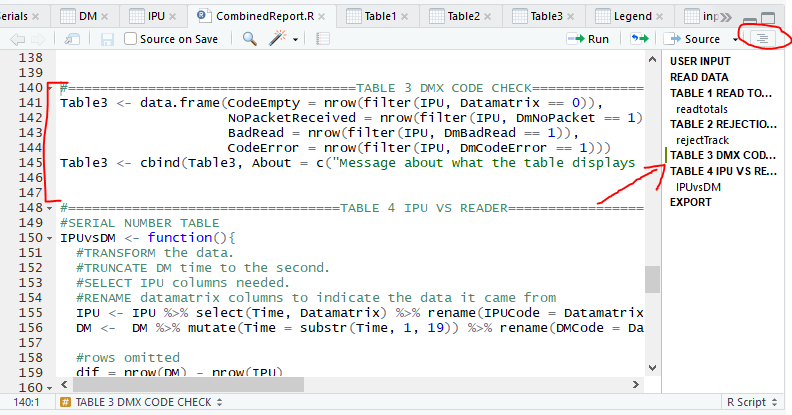


Notice that the first section in the Environment window is Data. It lists all the datasets that you ran through the program along with their dimensions. For example, let’s look at IPU data. It says that there are 7919 observations or rows of the data. To view it, all you have to do is click on the row that lists the IPU data and it will be displayed to the left.



The Document Outline serves a Table of Contents for the R script. The script has sections according to the type of operation it is carrying out to create the report (ex. Importing data, creating table2…). Although the script is not too long, being able to use the document outline can take you right to a section of interest. For example, perhaps you would like to see code for how Table 3 is created.

Click on the button with stacked lines next to source and it will display a list of sections in bold. Click on Table 3, and it will take you to that seciton in the script.



The same action can be done by using the bottom border as pictured below.

