# Overview

The following is the run book for ingesting, transforming and then performing the analysis. You will need to run this on the AWS machine that is configured properly with access to the data that is supplied from Learning Emergence Partners (LJP).

The information about the Anaconda environment is located in this GitHub Repository - <https://github.com/RoryIAngus/CIC-Visualisation/tree/master/2_Environment>. This code is written in the base environment that is using a call to the MongoDB Environment, both are running using the commands found in the environment start-up script - <https://github.com/RoryIAngus/CIC-Visualisation/blob/master/2_Environment/startEnv-Copy_from_root.sh>.

[Overview 1](#_Toc10553934)

[Instructions 2](#_Toc10553935)

[Data Extraction 2](#_Toc10553936)

[Script 01 – Load Results 3](#_Toc10553937)

[Script 02 – Load Coach Relationships 5](#_Toc10553938)

[Script 03 – Load Users 6](#_Toc10553939)

[Script 04 – Load Groups 7](#_Toc10553940)

[Script 05 – Create Journeys 8](#_Toc10553941)

[Script 06 – Cleaning Export 9](#_Toc10553942)

[Script 11 – Create Visualisations 19](#_Toc10553943)

[Export the results 20](#_Toc10553944)

# Instructions

|  |  |
| --- | --- |
| Data Extraction |  |
| Get the data | The data currently needs to be manually extracted from the LJP by a technician. They have a script that contains SQL and instructions to generate four JSON files.  A summary version of their document can be read by contacting LJP.  It is worth noting that the document also functions as a data dictionary. |
| Request Data Extract | The long form version of the above document has been provided to LJP and they have agreed that the SQL extraction scripts will be run on request. |
| Place the Data | At the moment the data is stored on an AWS Unix machine, which also hosts a standalone MongoDB environment and an Anaconda setup running Python 3. These instructions will only work if you have localhost redirection working to the remote environment.  Place the files that are received here. Note their file names. |
| Script 01 – Load Results |  |
| Open the first script | 01\_CLARA\_to\_MongoDB\_RealData\_ClaraResults\_SSODataModel.ipynb  This is currently located in <https://github.com/RoryIAngus/CIC-Visualisation/tree/master/3_Code> |
| Description | This code reads the individual CLARA results and loads them into the MongoDB. The results are one per survey and linked to the journey and the user.  At the moment the code drops the entire table and reloads it, the code is not configured to run incrementally at the moment. The biggest change will be changing the variable ***first\_run*** to be ***False*** and ensuring the data is correct. I think it will work after that. |
| Change the variable that contains the file location |  |
| Restart and rerun the kernel |  |
| Click Restart and Run All Cells | Note: This step will not be shown again, it is assumed that you can do this without prompting. |
| Ensure that it finishes | Check MongoDB to make sure the data has loaded into CLARA.raw\_data\_user\_results |
| Script 02 – Load Coach Relationships |  |
| Open the second script | 02\_CLARA\_to\_MongoDB\_RealData\_CoachingRelationship\_SSODataModel.ipynb  This is currently located in <https://github.com/RoryIAngus/CIC-Visualisation/tree/master/3_Code> |
| Description | This code reads the intersection table between coach and coachee and loads them into the MongoDB. The results are one per line per relationship with a start and end date if valid. This means that there could be duplicates of you ignore the date aspect of the record.  At the moment the code drops the entire table and reloads it, the code is not configured to run incrementally at the moment. The biggest change will be changing the variable ***first\_run*** to be ***False*** and ensuring the data is correct. I think it will work after that. |
| Change the variable that contains the file location |  |
| Restart and rerun the kernel |  |
| Ensure that it finishes | Check MongoDB to make sure the data has loaded into CLARA.raw\_data\_coach\_coachee |
| Script 03 – Load Users |  |
| Open the Third script | 03\_CLARA\_to\_MongoDB\_RealData\_Users\_SSODataModel.ipynb  This is currently located in <https://github.com/RoryIAngus/CIC-Visualisation/tree/master/3_Code> |
| Description | This code reads the individual users and their key ***ID*** fields that are used by the LJP and loads them into the MongoDB. The system uses two types of ID’s, one being a large random hash type string the other being an incremental integer.  At the moment the code drops the entire table and reloads it, the code is not configured to run incrementally at the moment. The biggest change will be changing the variable ***first\_run*** to be ***False*** and ensuring the data is correct. I think it will work after that. |
| Change the variable that contains the file location | The data in this file ***...\_LivePlatform\_ListOfUsersPAM.json*** contains a single record for Pam Ryan and does not need to be refreshed. It is included as this user is not part of the UTS organisation, but she is linked to UTS users as a coach. This is due to migration issues and cannot be resolved without creating a new user in UTS for Pam and then linking those students to her as the coach and having the dates manually set back (this is just easier to do). It is added to ensure that students linked to her as a coach can be selected.  If any other users are like this, then LEP needs to be contacted to help resolve the issue. |
| Restart and rerun the kernel |  |
| Ensure that it finishes | Check MongoDB to make sure the data has loaded into CLARA.raw\_data\_claraUsers |
| Script 04 – Load Groups |  |
| Open the Fourth script | 04\_CLARA\_to\_MongoDB\_RealData\_UserGroup\_SSODataModel.ipynb  This is currently located in <https://github.com/RoryIAngus/CIC-Visualisation/tree/master/3_Code> |
| Description | This code reads the group/user intersection table that identifies which groups users are in. It is also date and time bound with a start and end date. The code does not currently take this into account.  At the moment the code drops the entire table and reloads it, the code is not configured to run incrementally at the moment. The biggest change will be changing the variable ***first\_run*** to be ***False*** and ensuring the data is correct. I think it will work after that. |
| Change the variable that contains the file location |  |
| Restart and rerun the kernel |  |
| Ensure that it finishes | Check MongoDB to make sure the data has loaded into CLARA.raw\_data\_group\_user |
| Script 05 – Create Journeys |  |
| Open the Fifth script | 05\_Raw\_Data\_Combine\_Diagnose\_Measure\_to\_Single\_Row.ipynb  This is currently located in <https://github.com/RoryIAngus/CIC-Visualisation/tree/master/3_Code> |
| Description | This code loads the results data from Mongo and processes it to ensure that the correct elements are brought together. The end result is a single line that contains a journey, which has the results from the Diagnose and Measure survey.  It is important to note that my understanding of the data model has changed since the last code was written. The journey can have many steps and Clara survey results linked to it. In some instances, I have seen up to 8 different Clara results with a single journey. Checking with Shaofu, this is apparently normal, and so I am having to adjust this code to automate the gathering of a single set of data.  That is one user -> one journey -> one Clara result for the Diagnose step -> (optional) one Clara result for the Measure step  If there are more than two Clara results for step, then a decision needs to be made about which one to keep. I am not sure what that should be. I am guessing the first one for Diagnose and the latest one for Measure. This needs to be verified and certainly, it needs to be flagged in the reporting (new field).  Another new field is the duration that the tests took as well as the amount of elapsed time between the two tests. |
| These are the collections that the code is using | Read - raw\_data\_user\_results  Write - raw\_data\_combined\_user\_results |
| Check for errors | In the section - Multiple Surveys Error Reporting  If the following message appears there is an error with the data that needs to be investigated. The data in question is presented after the error message.  There is some data that you need to look at to work out which records to keep. The data in question is stored in the data frame called dfError and is presented here for ease. |
| Script 06 – Cleaning Export |  |
| Open the Sixth script | 06\_MongoDB\_to\_CSV\_RealData\_ClaraResults\_SSODataModel\_CoachingandGroups.ipynb  This is currently located in <https://github.com/RoryIAngus/CIC-Visualisation/tree/master/3_Code> |
| Description | This code reads the data from the MongoDB and uses it to interact with the user to create and then save a CSV file of data.  This is because there is a manual cleaning step that needs to happen and it is easier to do in Excel. |
| These are the collections that the code is using | Read:  coachDataCollection = db.raw\_data\_coach\_coachee  groupDataCollection = db.raw\_data\_group\_user  resultsDataCollection = db.raw\_data\_combined\_user\_results  usersDataCollection = db.raw\_data\_claraUsers  Write:  A CSV output file that needs to be used in the next step after being cleaned. |
| Restart and rerun the kernel |  |
| Scroll to the end of the notebook | In the last cell, there is an interactive component. |
| Select a Group |  |
| Select a Coach |  |
| Click on the Pick Group + Coach Button |  |
|  | Read the messages to confirm the expected output which determines the number of records for each element (Group and Coach) and the intersection of those two and displays the information along with a sample of the data |
| Just select a Group | Select the blank line for the Coach and then select the group and click the button |
| The same goes when just selecting a Coach |  |
| Note the messages | This explains the outcome of the different selections that have been made. Note that not every user has CLARA results and some may have more than one. That is why in this example there are only 31 records. |
| Type in the name of the file and click Save File | Once you are happy with the selection, save the file. |
| Open the Saved File from the previous step and download the file | Take note of the location of the file from the previous step.  ~/datasets/CLARA/UserSaved/32601\_Adv\_ProjMgmt.csv |
| Open the file in Excel or your spreadsheet of choice. | This is where you will need to clean the data. The best outcome is to have a single journey for each student and that journey to have a diagnose and measure result. It may be required to work with the individual tutors and lecturers at this point.   1. Delete the journeys that do not have Diagnose results. 2. Check to see if students have more than one journey.    1. If they do, it could be a test journey or it could be that they have done the Measure step as part of a new journey so it has ended up in the diagnose section    2. Copy the results from Diagnose to Measure if that is the case, remember to look at the dates and copy those as well as calculate the durations, Completed Second Survey flag etc.    3. Remove unneeded journeys 3. Check to see if the results make sense. In some cases all the values are 1, this is clearly not a valid survey. 4. Look at survey duration, did it take them a few minutes to answer 80 questions. This is also a good indication of the survey results being compromised    1. It is important to note that there is a bug in the platform that allows surveys to be completed in a very short amount of time. Shaofu is investigating this. 5. Examine the duration between the Diagnose and Measure survey.    1. Was it too short for it to be a valid measure of change? |
| Save the CSV and upload it to the platform |  |
| Add a suffix so that it doesn’t overwrite the original file |  |
| Files ready to be used |  |
| Script 11 – Create Visualisations |  |
| Open the script file | 11\_ReportGenerationCode\_LiveData\_ImprovedCharts\_Generic  This is located here: <https://github.com/RoryIAngus/CIC-Visualisation/tree/master/3_Code> |
| Description | This code reads the data from the CSV File and uses it to create the visualisations of the data. |
| Change the readLoc variable to point to the new file. |  |
| Restart and rerun the kernel |  |
| Export the results |  |
|  | The following will create a PDF of the charts etc, but will delete all of the code.  This was adapted from <https://stackoverflow.com/questions/34818723/export-notebook-to-pdf-without-code> |
|  | Run Jupyter notebook and download the notebook in the browser:  File->Download as->HTML and you will get a HTML page with code and output. |
| Open the file | Open the exported HTML with browser |
| Open the console | Activate the browser developer console by pushing the key F12 |
| Remove Code Content | Run following command in the console:  document.querySelectorAll("div.input").forEach(function(a){a.remove()})    Press enter to execute.  The code removes all input div DOM which basically contains the code sections from the workbook. It leaves behind the data and charts and Markup boxes. |
| Save the new file | Click the right mouse button and chose "Save As..." |
| Save as type | Select Webpage, Complete |
| Specify the File name |  |
| Open the file | Open the folder where the file was saved and retrieve the file |
| Compression | You may want to Zip it to email it as it is a HTML file and it is quite large. |
| Email attachment |  |