Appendix A to "Cost per Use in Power BI using Alma Analytics and a Dash of Python"

A Step-by-Step Guide to Building an Alma to Power BI Pipeline for Cost Per Use

1. Establish prerequisites
    a. To replicate our approach, you will need access to, knowledge of, and some skills in Alma, Alma Analytics, COUNTER 5, APIs, SUSHI, Python, and Power BI.
    b. Power BI Desktop requires a Windows PC.
    c. It is helpful to have some elementary familiarity with programming with SQL in Alma Analytics and M and DAX languages in Power BI, in order to make adjustments that work better for your needs and workflow.
2. Set up vendors with SUSHI API in Alma with tr_j1 reports types enabled
    a. Follow the "Managing SUSHI Accounts" section of Ex Libris documentation "Managing COUNTER-Compliant Usage Data". [1]
    b. Set up vendors manually in Alma.
        i. Identify administrative usernames and passwords for the content providers and the next layer of credentials for SUSHI: the customer ID, requestor ID, and occasionally, API key. Please note:
        - Some SUSHI credentials can be found on the content provider's website, but others cannot. It may be necessary to contact customer service to obtain them.
        - Many credentials changed during the switch from COUNTER 4 to 5.
        - Credentials can change at any time without notice.
    c. Establish the base URL for the SUSHI API string. Please note:
        - Commonly used vendors are already established in our consortial instance of Alma, and most of the time the base URLs were correct.
        - If the "Vendor URL" doesn't work in Alma, experiment with the "Override URL" field. One may need to add "/sushi", "/sushi/", "/reports", or "/reports/" to the end of the base URL to pull a report successfully.
        - "Requester" ID is spelled with an "e" in Alma and is spelled "Requestor", with an "o" in COUNTER API strings. This is an important detail if you test the API strings in a browser.
    d. Under "Usage Report Types" in Alma, click "Add Report Type" to see the list of report types. Select tr_j1 to follow our example.
    e. Select "Test with Response" to test whether the connection is working and see an example of the harvest that Alma returns.
3. Set up auto-harvest of SUSHI in Alma
    a. Follow the "Managing SUSHI Harvesting" section of Ex Libris documentation "Managing COUNTER-Compliant Usage Data". [1]
    b. Once the vendors are set up, tested, and verified, enable the auto-harvest.
        i. Select a harvest date.
            1. After experimenting with different Alma preset dates, we settled on the 18th of the month to collect new usage reports.
        ii. Enable the function to receive the monthly SUSHI harvest report by email once the harvest has been completed.
            1. The monthly SUSHI harvest report will list completed and failed attempts, with errors. Handling errors is described in step 12.a.

2. This email signals that the Power BI model can be refreshed to intake updated usage statistics, described in step 12.c.

4. Create an Alma Analytics report on costs
    a. Experiment with Ex Libris' out-of-the-box reports or create your own. We did not find an Out-of-the-box report that met our specific needs.
    b. To follow our example, use Library Name, Fiscal Period Description, Transaction Expenditure Amount, Fund Code, Title (Normalized), ISSN, and PO Line Reference details including Item Description and filter on "continuous" continuity and an "active" status.
        i. Item Description provided the best results for us to be able to match titles with Normalized Title in the Usage Data subject area of Analytics, but this may vary with your cataloging workflow.
        ii. Edit the column formula of Item Description to turn all item descriptions into lower case, like so - `"LOWER("PO Line"."Item Description")` - to begin normalizing the data. Removing proceeding articles occurs in a later step, in Power BI. It is also an option to lower the case in Power BI to do all the normalization steps in one environment. This may be the preferred method if Analytics formulas aren't familiar.
        iii. Write both Analytic reports to display the fiscal year as "FY-XXXX" to enable matching between tables.
    c. Test the report and adjust as necessary for your data and institution.
5. Create an Alma Analytics report on COUNTER 5 tr_j1 use
    a. We did not find an out-of-the-box report that met our specific needs.
    b. To follow our example, use TR_J1 – Unique Item Requests, TR_J1 – Total Item Requests, Usage Date Fiscal Year, Usage Date Year, Usage Date Month Key, Usage Date Month, Usage Date Quarter, Normalized Title, Normalized ISSN, Normalized EISSN, Origin ISSN, Origin EISSN, Normalized Publisher, Publisher, Normalized Platform, and Load File COUNTER Report Type.
    c. This report uses the normalized title, so no additional transformations are necessary.
    d. Write both Analytic reports to display the fiscal year as "FY-XXXX" to enable matching between tables.
    e. Test the report and adjust as necessary for your data and institution.
6. Obtain permissions to use the Alma Analytics API key for Acquisitions
    a. Work with your institution's Alma developer to get permissions and the API Key to use the "Retrieve Analytics Report" API, with secrets held in the Alma Developer Network.
        i. Follow Gandalf's advice to "keep it secret, keep it safe." [2]
    b. Build the path for each Analytics report with the root URL, Analytics report path, and Acquisitions API Key.
        i. Our path begins with `https://api-na.hosted.exlibrisgroup.com/almaws/v1/analytics/reports?path=` but it may vary for your Alma instance.
        ii. Open either the cost or use report in Analytics. In the browser, copy the portion of the link for the Analytics report that begins after the equal sign. Add the copied text after the equal sign of the path you are building. Notice that when you copy and paste, blank spaces are replaced by "%20". (Figure 1)

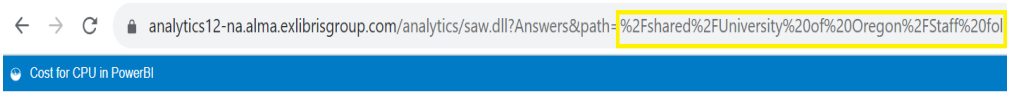

Figure 1

iii. Add "`&apikey=`" and then the API Key for Acquisitions to the end of the path.

c. Repeat the steps in 6.b to build the second report URL.

d. Test each report.

i. There are many options for testing APIs, including putting the whole string in the browser.

ii. Another way to test this is to set the API as the data source in Power BI.

iii. Review the data. If your data is more than 1000 rows, you will need to use a resumption token to retrieve the full set in Power BI. We were unable to get the resumption token to work using Power Query (M) directly, so we used a Python script to retrieve the report and pass in the resumption token, as described in the next step.

7. Write a Python script to query the Alma Analytics API, including resumption tokens to retrieve more than 1000 results

a. We based our Python query on a well-documented set of instructions for how to query the Open Alex API created by Eric Schares and Sandra Mierz. [3]

b. Feel free to copy our code, which is available in several different versions. [4]

c. Because the Alma Analytics API has access to patron information, we maintained a high level of security. We initially did this by creating a Python script tracked in GitHub that used the `configparser` library to pull in our secrets (API key and report paths) from a separate `config.ini` file not tracked in GitHub.

d. We recommend testing your Python code outside of Power BI using your preferred Python interpreter with debugger since Power BI doesn't have a Python debugger built in.

e. However, because Power BI does not allow the Python code to consult other files (such as a `config.ini` file) we manually created a version that no longer used `configparser` to reference a separate file with secrets, but instead included all secrets hard coded into the Python script. In the next section, we discuss how we added this hard-coded script to Power BI, and how we modified it to allow each secret to be a Power BI parameter.

8. Establish Power BI connection cost and use reports using Python script

a. To add a Python script to Power BI, first follow the instructions in Microsoft documentation to install Python on the desktop (a virtual environment is recommended), import any necessary Python libraries (we used `pandas`, `xml.etree.ElementTree`, `requests`, and `time`), configure Python access in Power BI, and enable permissions. As of this writing, all Python data sources need to be set to "Public" to work with the Power BI service.

b. Once permissions are configured, in Power BI Desktop, select "Get Data" and scroll down to select "More" to open up a new box. Scroll down to select "Python script" and then click "Connect". Paste the Python script in the text box, then click "OK". If the script runs correctly, you will be prompted to load data into a Power Query table. The first applied step for this data will be "Source," and a gear icon to

the right of the step allows you to open and view the Python script as originally entered.

c. To see the Python script as it appears loaded into a Power Query (M) script, open “Advanced editor" in the Power Query ribbon. Note that Power BI has translated the script by wrapping it with a command to run in Python and adding `#(lf)` where each line ending was in the original Python.

d. We recommend turning the hard coded secrets (API key and report path) into Power BI “Parameters” that can be more easily updated by those less proficient with Python or M. To do this, first follow Microsoft’s instructions to create your parameters. For our code we created four: `url, api_key, use_data_path` and `cost_data_path`. Be sure to specify data type “Text” and enter each parameter with quotation marks around it. For example, the `api_key` parameter will look something like “`k3lk2ke4536kjklk4`” (quotes included).

e. Now return to the “Advanced editor” window in Power Query and replace the hard coded API key, which will be something like “`k3lk2ke4536kjklk4`”, with “`& api_key &`”, where `api_key` is the name of the parameter, and the replacement code begins and ends with double quotes.

f. To see our Python code in the Power BI context, open the Power BI template (.pbit) file available in our GitHub repository, https://github.com/uodataservices/cost_per_use/. [5]

i. If no appropriate Alma Analytics reports are configured, enter the real url, api_key, use_data_path and cost_data_path values. If not, enter any placeholder value into the fields (like “test”) to access our Power BI model. Either way, click “Load”.

ii. You may need to give permission to Power BI to run the script.

iii. There will be a delay as Power BI tries to make the API calls.

iv. If parameters have “test” entries, you will see a popup window warning of Python script errors. Go ahead and close that error box.

v. Despite more error messages, you are now in our Power BI file and can explore every step of our data transformations in Power Query (M) and DAX.

vi. Find the Python code by choosing “Transform Data” and then clicking on the "Advanced editor” for either the usage_data or cost_data tables.

The following steps will explain how we worked with our data once it was imported into Power BI. You can look for these steps in the Power BI environment to go with the steps described.

9. Match cost and use data on ISSNs and Titles using M and DAX languages

a. Once a connection is established, transformations will need to be done in Power Query using the M language to open the tables, normalize data, and prepare it for matching.

i. Starting with the cost data, open the data tables by clicking on the arrows that point simultaneously left and right until all the column carrots point down and you have drilled down to the columns so that they match the Alma Analytic on costs. (Figure 2)

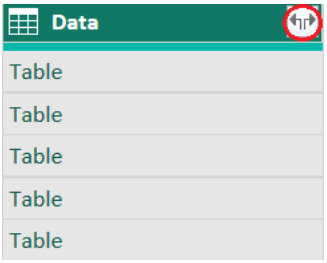

Figure 2

ii. You may need to change data types (e.g. convert numbers to currency), remove unnecessary columns, rename columns for clarity, etc. so that your data makes sense and functions well in Power BI. This is an interactive process that you will need to customize for your data and needs.

iii. Recall that we lowered the case of the Item Description in Alma Analytics. At this point use M code in Power Query to remove the beginning "the" and "a" to further normalize and match these titles with the titles that will appear in the use table.

1. Code for removing "the":

```
= Table.ReplaceValue(#"Renamed Columns", each
[Title Normalized], each if
Text.StartsWith([Title Normalized],"the ")
then Text.RemoveRange([Title Normalized],0,4)
else [Title Normalized], Replacer.ReplaceText,
{"Title Normalized"})
```

2. There may still be differences in the cost table versus usage table between "and" and "&", punctuation in the middle of a title, and other unknown differences.

iv. The field ISSN contains multiple ISSNs. There were as many as six ISSNs, but we decided that two would be enough to match on. We split the column by a delimiter to separate the first two ISSNs into two columns.

1. Code for splitting by delimeter:

```
= Table.SplitColumn(#"Replaced Value", "ISSN",
Splitter.SplitTextByDelimiter("; ",
QuoteStyle.Csv), {"ISSN.1", "ISSN.2"})
```

v. Optionally, add a conditional column for Fund Code Display and group fund codes by broader category to make this information more digestible to subject librarians. Anecdotally, our funding schema is more complicated than most libraries, so you may not need this step.

vi. Move to the usage data, and open up the data tables until you have drilled down to the columns so that they match the Alma Analytic on use, like in step 8.a.1.

vii. Normalize titles here as well, removing the beginning "the" and "a" so that they will match with the titles in the cost table.

viii. Again, you may need to change data types, remove columns, rename columns, etc. so that your data makes sense and functions well in Power BI. This is an interactive process that you will need to customize for your data and needs.

ix. Navigating out of Power Query, and into the tables, at this point you should have one table for cost and one table for use. Our table for cost has all the columns we built in the Alma Analytic on cost as well as the split ISSN columns and the Fund Code Display column.

b. Create a use summary table. Cost data is available per fiscal year, while usage data is available aggregated by month. To simplify joining information, we summarized usage data in a new table and matched the summaries with cost data on ISSNs or normalized titles.

i. Here is the DAX code for the usage summary table including the sum of Total Requests:

```
usage_summary = SUMMARIZE('usage_data',
usage_data[Usage Date Fiscal Year],
usage_data[Normalized Title],
usage_data[Normalized Platform],
usage_data[Normalized Publisher],
usage_data[Normalized EISSN],
usage_data[Normalized ISSN], "Total Requests",
SUM(usage_data[Total Requests]))
```

c. With the usage summary table established, match on ISSN and title.

i. We started with ISSN1 of the cost table and matched it against the Normalized EISSN of the use table. Here is the DAX code:

```
ISSN1 by Normalized EISSN = CALCULATE (SUM (
'usage_summary'[Total Requests] ), FILTER (
'usage_summary',
AND(AND('usage_summary'[Normalized EISSN] =
'cost_data'[ISSN1], 'usage_summary'[Usage Date
Fiscal Year]='cost_data'[Fiscal Period
Description]), IF(AND('cost_data'[ISSN1] = "",
usage_summary[Normalized EISSN] = ""), FALSE,
TRUE)))
```

ii. We made a new column for ISSN2 by Normalized EISSN, and so on, as well as Title by Title.

d. Because of data irregularities, we chose to calculate Cost Per Use in a fuzzy way using the most expensive year and the highest use year. It will not be exact for data in a specific year. Since this information is meant to discover outliers, this level of detail is sufficient for our needs. To calculate Cost Per Use we added columns for maximum use and maximum cost per use:

i. Electronic Use Max:

```
Electronic Use MAX = ROUND(MAX('cost_data'[Title
by Title],MAX(MAX('cost_data'[ISSN1 by Normalized
EISSN],'cost_data'[ISSN1 by Normalized
ISSN]),MAX('cost_data'[ISSN2 by Normalized EISSN],
'cost_data'[ISSN2 by Normalized ISSN]))),0)
```

1. The employment of the ROUND function is to resolve troubles with a fraction of a use.

ii. Electronic Max CPU:

```
Electronic Max CPU = IF([Electronic Use
MAX]=0,0,'cost_data'[Transaction Expenditure
Amount]/[Electronic Use MAX])
```

e. Recognizing that borrowing material is also an option in libraries, we included a formula for the Cost Beyond ILL Threshold, and a column for how that formula translated into whether to consider ILL instead of subscription.

i. This is our formula for Cost Beyond ILL Threshold, assuming documents through Reprints Desk cost $35 and we can request an item 20 times before having to pay for it [5]:

```
Cost Beyond ILL Threshold =
IF('cost_data'[Electronic Use MAX]-20<0,0,
('cost_data'[Electronic Use MAX]-20)*35)
```

ii. The Consider ILL column is a simple True / False statement:

```
ConsiderILL = IF(AND('cost_data'[Cost Beyond ILL
Threshold] < 'cost_data'[Transaction Expenditure
Amount],cost_data[Electronic Use MAX] > 0 ), TRUE,
FALSE)
```

iii. Create a table for ILL vs Subscriptions so that it can be shown as a visualization. Here is the code:

```
ILL_vs_subscription = SUMMARIZE('usage_summary',
usage_summary[Normalized Title],
usage_summary[Normalized Platform],
usage_summary[Normalized Publisher],
usage_summary[Normalized EISSN],
usage_summary[Normalized ISSN], "Max Yearly
Requests", MAX(usage_summary[Total Requests]))
```

10. Create visualizations based on cost and use

a. The point of all of that data wrangling was to be able put together a cost per use visualization, and now that all that hard work is done, the visualization is relatively straightforward. For Cost Per Use, we chose a stacked bar chart, used Title Normalized for the y-axis and the sum of Electronic Use Max CPU (step 7.d.ii) for the x-axis, and added the Fund Code Display (which we shortened to Fund) as the Legend. We added slicers for Fund and Fiscal Year. The display can be filtered by these slicers, or it can be searched by title. (Figure 6 of main paper)

b. For a closer look at the usage data, on the second tab, we created a table drawn from the Usage Summary table's Normalized Title, Total Requests, and Usage Data Fiscal Year, and added in a slicer for Fund. This table can be filtered by fund or searched by platform, publisher, or title. (Figure 7 of main paper)

c. The third display is a table of titles that might be cheaper to obtain by ILL than subscription, based on the usage patterns and average ILL cost. (Figure 8 of main paper)

d. The last tab is an About page, covering the highlights of what is and isn't included in the data, known limitations, and the process overview.

11. Publish the Power BI web dashboard/report so you can share it with colleagues

a. In the Power BI cloud app, create a Workspace where you will share the model.

i. Name the Workspace, describe it, and add the people who you want to have access to it.

b. In Power BI Desktop, click "Publish" and select the Workspace where you want to upload it.

c. Navigate to the Report in the Workspace and then click the "Share" button to share the report colleagues.

12. Perform ongoing maintenance

a. Continually refresh vendor usernames and passwords, harvest credentials, API keys, changes to platforms and base URLs as needed.
    i. There are a variety of errors to work through on the monthly SUSHI harvest report, ranging from the vendor's usage dates not being available yet, to failure to connect, to retrieving unwanted reports. It is helpful to have Alma's monthly SUSHI harvest report emailed the person managing the accounts to identify and troubleshoot errors.
    ii. SUSHI errors are described by Project Counter. [6]
    iii. Alma / SUSHI errors are well described by CARLI. [7]
b. Add new vendors as they enable COUNTER 5 reporting.
c. Monthly, after Alma performs the SUSHI harvest, refresh the data in Power BI.
    i. Open the Power BI model in Power BI Desktop
    ii. Click "Refresh" to bring in both the updated cost data and usage statistics.
    iii. Click "Publish" to overlay the currently published model with the updated one.
    iv. Click "Save" to save the updated workbook.
    v. A message will pop up about overlaying the currently published data set. Agree, and click "Replace".

Appendix Notes

[1] "Managing COUNTER-Compliant Usage Data": https://knowledge.exlibrisgroup.com/Alma/Product_Documentation/010Alma_Online_Help_(English)/020Acquisitions/090Acquisitions_Infrastructure/010Managing_Vendors/Managing_COUNTER-Compliant_Usage_Data

[2] See the movie "The Lord of the Rings: The Fellowship of the Ring"

[3] OpenAlex-CitedReferences: https://github.com/eschares/OpenAlex-CitedReferences

[4] Available at our project Github repository (https://github.com/uodataservices/cost_per_use) and archived in Zenodo at https://zenodo.org/records/10426231.

[5] This formula is based on a formula created by our colleague David Ketchum, Director of Access Services at UO Libraries.

[6] Appendix F: Handling Errors and Exceptions https://www.projectcounter.org/appendix-f-handling-errors-exceptions/

[7] Alma SUSHI Harvesting Status Examples https://www.carli.illinois.edu/products-services/i-share/electronic-res-man/sushierror