Use of Optional Data Curation Features by Harvard Dataverse Users

Data Cleaning and Analysis Workflow for Harvard Dataverse Inventory

Author: C. M. Boyd, ceilyn.boyd@simmons.edu, ceilyn_boyd@harvard.edu

- **Created:** 8 June 2020
- **Updated:** 6 July 2020

About the Data: Two files serve as raw data for this project: `raw_filedata.csv` and `raw_dataset_data.csv`. These two files reflect the inventory of the Harvard Dataverse repository on 28 October 2019 and 29 October 2019.

- `raw_filedata.csv`: Using an SQL query, a file containing a list of all files associated with datasets was created on 28 October 2019. This inventory identified **383,687** files associated with **29,295** datasets.
- `raw_dataset_data.csv`: A second SQL query was made on 29 October 2019 to get at list of all datasets at that time. There were **29,997** datasets in the inventory on that day.

Note: There is 702 item discrepancy between the number of datasets reflected in these two files. Data cleaning and analysis was performed using Python code and Excel in order to generate descriptive statistics about the Harvard Repository on those two days. Ultimately, **29,295** datasets and **383,685** files were identified.

Systems

- **Python Environment:** Anaconda Navigator 1.6.9, Python 3.6
- **Jupyter Notebook:** Version 5.0.0
- **MacOS:** Catalina, 10.15.4 and Mojave, 10.14.6
- **Excel for MacOS:** 16.37
- **DB Browser for SQLite:** 3.11.2 (https://sqlitebrowser.org (https://sqlitebrowser.org)) (Used to review SQL and inspect output of SQL statements)
- **SPSS for MacOS:** Version 26

Files

Input Files

These two raw input files were created by an IQSS database analyst who generated the SQL queries on the database to capture raw information about datasets and files in Harvard Dataverse. These two files were then cleaned, processed, and analyzed by C. M. Boyd to generate descriptive statistics about the data.

- `raw_filedata.csv`: File containing a list of all files associated with datasets was created on 28 October 2019 using an SQL query. This inventory identified **383,687** files associated with **29,296** datasets.
- `raw_dataset_data.csv`: File containing a list of all datasets in Harvard Dataverse on 29 October 2019 created using an SQL query. There were **29,997** datasets in the inventory on that day.

Generated Files

- `dataset_data_subset.tsv`: The file containing the subset of datasets that were referenced in cleaned_filedata.tsv and all additional fields needed to support statistical analysis using SPSS

Other Output Files

- `dataset_data_subset.xlxs`: The manually create, resaved file `dataset_data_subset.tsv`; operation is performed because there is a complex record in the file that contains nested quotes that confuses the SPSS tab-separated-values and comma-separated-values parsers.
- `hdofu_dataset_data_subset.sav`: Manually created SPSS project file, result of importing `dataset_data_subset.xlsx` and performing statistical processes upon it.

Excel

Clean Raw Data Files

1. Clean `raw_dataset_data.csv`

Upon inspection, I discovered one poorly formatted entry in `raw_dataset_data.csv` for `dataset_id = 2676050`. Therefore, using Excel, I manually removed this record from `raw_dataset_data.csv` and saved the resulting file as `cleaned_dataset_data.tsv`. Additional cleaning to address NA values is performed before this file is ingested into SQLite3.

2. Clean `raw_filedata.csv`

To reflect the deletion of the record in `raw_dataset_data.csv` where `dataset_id = 2676050`, I also use Exel to remove all file records associated with `dataset_id = 2676050` from `raw_filedata.csv` to create the new tab delimited file `cleaned_filedata.tsv`. Additional cleaning to address NA values is performed before this file is ingested into SQLite3.

Python

Global Variables

File-related Variables

In [2]:
```
# Import os module for interacting with the operating system
import os

# Establish base directory for data, code, and jupyter notebook associ
ated with the project
curr_dir = os.getcwd()
dir_array = curr_dir.split('/')
dir_array.pop(len(dir_array)-1)
base_dir = '/'.join(dir_array)

# Directory for the data quality project
project_dir = base_dir

# Directory for the SQLite3 database
db_dir = project_dir + '/db'

# Database name
db_name = 'hdofu.db'

# Database file path
db_path = db_dir + '/' + db_name

# Directory for the data quality project data (input and output)
data_dir = project_dir + '/data'

# Cleaned dataset data file
cleaned_dataset_data_tsv = data_dir + '/cleaned/cleaned_dataset_data.t
sv'

# Cleaned filedata file
cleaned_filedata_tsv = data_dir + '/cleaned/cleaned_filedata.tsv'

# Target output file for subset of datasets used in SPSS analyses
dataset_data_subsets_tsv = data_dir + '/dataset_data_subset.tsv'
```

Import Python Libraries

In [3]:
```
# Import Pandas library, v. 1.0.3
import pandas as pd

# Import sQLite3: DB-API 2.0 interface for SQLite databases
import sqlite3

# Import csv
import csv
```

Process Data Files

1. Create the `hdofu` database and read in the file `cleaned_dataset_data.tsv` to create the `cleaned_dataset_data` table

In [34]:
```
# Create or open the dataquality database
db = sqlite3.connect(db_path)

# Use Pandas to read in cleaned_dataset_data.tsv
df = pd.read_csv(cleaned_dataset_data_tsv, delimiter='\t')

# Create and/or replace data to cleaned_dataset_data table
df.to_sql('cleaned_dataset_data', db, if_exists="replace")

# Close the connection
db.close()
```

2. Create database and read in `cleaned_filedata.tsv` to create the `cleaned_filedata` table

In [35]:
```
# Open the dataquality database
db = sqlite3.connect(db_path)

# Use Pandas to read in cleaned_filedata.tsv
df = pd.read_csv(cleaned_filedata_tsv, delimiter='\t')

# Fill download_count field with zeros, downcast float to int
df.fillna(0,inplace=True,downcast={'int64'})

# Create and append data to cleaned_filedata table
df.to_sql('cleaned_filedata', db, dtype={'download_count':'INTEGER'},
if_exists="replace")

# Close the connection
db.close()
```

2.1 Add `cleaned_filedata.is_PSF` column, with values, to `cleaned_filedata` table for possible supplemental files and update `cleaned_filedata.has_PSF` with values from filedata

In [36]:
```
# Open the dataquality database
db = sqlite3.connect(db_path)

# Get the cursor
c = db.cursor()

# Alter table sql statement, to add is_PSF field to dataset
sql = 'ALTER TABLE cleaned_filedata ADD is_PSF INTEGER;'

# Set initial values in the new is_PSF column to 0
sql = sql + 'UPDATE cleaned_filedata SET is_PSF = 0;'

# Update with possible supplemental files
sql = sql + 'UPDATE cleaned_filedata SET is_PSF = 1 WHERE cleaned_file
data.filename LIKE "%codebook%" OR cleaned_filedata.filename LIKE "%co
de book%" OR cleaned_filedata.filename LIKE "%readme%";'


# Execute the sql
c.executescript(sql)

# Close the connection
db.close()
```

3. Summarize `cleaned_filedata` as view. Group `cleaned_filedata` table by `dataset_id`.

In [37]:
```
# Open the dataquality database
db = sqlite3.connect(db_path)

# Get the cursor
c = db.cursor()

# Create summarized view
sql = 'CREATE VIEW filedata_summary_view AS SELECT dataset_id, SUM(dow
nload_count) AS download_count, SUM(is_PSF) AS num_PSF, SUM(restricted
) AS num_RF FROM cleaned_filedata GROUP BY dataset_id;'

# Execute the sql
c.execute(sql)

# Close the connection
db.close()
```

4. Create `dataset_data_subset` by combining `cleaned_dataset_data` table with `filedata_summary_view` view.

The `cleaned_dataset_data` table has more unique `dataset_ids` than `filedata_summary`. Join the two tables to return a combined table that only contains matching records that appear in the `filedata_summary_view` view. Note that we create `dataset_data_subset` as a table, rather than a view, because we will make additions to the table (e.g. add columns) to support statistical analyses.

In [38]:
```
# Open the dataquality database
db = sqlite3.connect(db_path)

# Get the cursor
c = db.cursor()

sql = 'CREATE TABLE dataset_data_subset AS SELECT filedata_summary_vie
w.download_count, filedata_summary_view.num_PSF, filedata_summary_view
.num_RF, cleaned_dataset_data.dataset_id, cleaned_dataset_data.dataset
_doi_url, cleaned_dataset_data.cc0,cleaned_dataset_data.terms_of_use,
cleaned_dataset_data.pub_year, cleaned_dataset_data.latest_version_yea
r, cleaned_dataset_data.dataset_version, cleaned_dataset_data.descript
ion_length, cleaned_dataset_data.subject, cleaned_dataset_data.keyword
_count, cleaned_dataset_data.datasets_with_relpubs, cleaned_dataset_da
ta.dataverse_id, cleaned_dataset_data.dataverse_url, cleaned_dataset_d
ata.dataverse_cat, cleaned_dataset_data.optional_metadata, cleaned_dat
aset_data.submit_for_review FROM filedata_summary_view LEFT JOIN clean
ed_dataset_data ON (filedata_summary_view.dataset_id = cleaned_dataset
_data.dataset_id);'

# Execute the sql
c.execute(sql)

# Get the subset from the database and save to an external file
df = pd.read_sql('SELECT * from dataset_data_subset', db)
df.to_csv(dataset_data_subsets_tsv, sep='\t')

# Close the connection
db.close()
```

5. Add new columns to `dataset_data_subset` to hold information relevant to performing analyses related to optional feature use.

These variables may be assigned the value of either 1 (TRUE) or 0 (FALSE). The following variables are added, and set to 0 (FALSE) initially:

1. **`has_PSF`:** dataset has possible supplemental files
2. **`has_OMB`:** dataset has one or more optional metadata blocks
3. **`has_ATOU`:** dataset has additional terms of use
4. **`has_KW`:** dataset has one or more keywords
5. **`has_DESC`:** dataset has description of one or more words
6. **`has_RPUBS`:** dataset has related publications
7. **`has_RF`:** dataset has one or more restricted files
8. **`has_REVIEW`:** dataset requires a review before submission
9. **`ofus`:** optional feature use score, the sum of the following variables: `has_OMB`, `has_ATOU`, `has_KW`, `has_DESC`, and `has_PSF`.

In [39]:
```
# Open the dataquality database
db = sqlite3.connect(db_path)

# Get the cursor
c = db.cursor()

# Alter table sql statement, to add has_PSF field to dataset
# Variable: has possible supplemental files
sql = 'ALTER TABLE dataset_data_subset ADD has_PSF INTEGER;'

# Set initial values in the new has_PSF column to 0
sql = sql + 'UPDATE dataset_data_subset SET has_PSF = 0;'

# Alter table sql statement, to add has_OMB field to dataset
# Variable: has one or more optional metadata blocks
sql = sql + 'ALTER TABLE dataset_data_subset ADD has_OMB INTEGER;'

# Set initial values in the new has_OMB column to 0
sql = sql + 'UPDATE dataset_data_subset SET has_OMB = 0;'

# Alter table sql statement, to add has_ATOU field to dataset
# Variable: has additional terms of use
sql = sql + 'ALTER TABLE dataset_data_subset ADD has_ATOU INTEGER;'

# Set initial values in the new has_ATOU column to 0
sql = sql + 'UPDATE dataset_data_subset SET has_ATOU = 0;'

# Alter table sql statement, to add has_KW field to dataset
# Variable: has keywords
```

```
sql = sql + 'ALTER TABLE dataset_data_subset ADD has_KW INTEGER;'

# Set initial values in the new has_KW column to 0
sql = sql + 'UPDATE dataset_data_subset SET has_KW = 0;'

# Alter table sql statement, to add has_DESC field to dataset
# Variable: has description
sql = sql + 'ALTER TABLE dataset_data_subset ADD has_DESC INTEGER;'

# Set initial values in the new has_DESC column to 0
sql = sql + 'UPDATE dataset_data_subset SET has_DESC = 0;'

# Alter table sql statement, to add has_RPUBS field to dataset
# Variable: has related publications
sql = sql + 'ALTER TABLE dataset_data_subset ADD has_RPUBS INTEGER;'

# Set initial values in the new has_RPUBS column to 0
sql = sql + 'UPDATE dataset_data_subset SET has_RPUBS = 0;'

# Alter table sql statement, to add has_RF field to dataset
# Variable: has restricted files
sql = sql + 'ALTER TABLE dataset_data_subset ADD has_RF INTEGER;'

# Set initial values in the new has_RF column to 0
sql = sql + 'UPDATE dataset_data_subset SET has_RF = 0;'

# Alter table sql statement, to add has_REVIEW field to dataset
# Variable: requires a review prior to submission
sql = sql + 'ALTER TABLE dataset_data_subset ADD has_REVIEW INTEGER;'

# Set initial values in the new has_REVIEW column to 0
sql = sql + 'UPDATE dataset_data_subset SET has_REVIEW = 0;'

# Alter table sql statement, to add ofus field to dataset
# Variable: sum of has_OMB, has_ATOU, has_KW, has_DESC, and has_PSF.
sql = sql + 'ALTER TABLE dataset_data_subset ADD ofus INTEGER;'

# Set initial values in the new ofus column to 0
sql = sql + 'UPDATE dataset_data_subset SET ofus = 0;'

# Execute the sql
c.executescript(sql)

# Close the connection
db.close()
```

6. Populate the new columns with values drawn from other columns

In [41]:
```
# Open the dataquality database
db = sqlite3.connect(db_path)

# Get the cursor
c = db.cursor()

# Variable: has possible supplemental files
# Update for datasets that have one or more possible supplemental file
s
sql = 'UPDATE dataset_data_subset SET has_PSF = 1 WHERE dataset_data_s
ubset.num_PSF >=1;'

# Variable: has one or more optional metadata blocks
# Update for datasets that have one or more optional metadata blocks
sql = sql + 'UPDATE dataset_data_subset SET has_OMB = 1 WHERE dataset_
data_subset.optional_metadata >=1;'

# Variable: has additional terms of use
# Update where dataset has an additional terms of use agreement (field
is not null)
sql = sql + 'UPDATE dataset_data_subset SET has_ATOU = 1 WHERE dataset
_data_subset.terms_of_use IS NOT NULL;'

# Variable: has keywords
# Update where dataset has one or more keyword
sql = sql + 'UPDATE dataset_data_subset SET has_KW = 1 WHERE dataset_d
ata_subset.keyword_count >= 1;'

# Variable: has description
sql = sql + 'UPDATE dataset_data_subset SET has_DESC = 1 WHERE dataset
_data_subset.description_length >= 1;'

# Variable: has related publications
sql = sql + 'UPDATE dataset_data_subset SET has_RPUBS = 1 WHERE datase
t_data_subset.datasets_with_relpubs >= 1;'

# Variable: has restricted files
sql = sql + 'UPDATE dataset_data_subset SET has_RF = 1 WHERE dataset_d
ata_subset.num_RF >= 1;'

# Variable: requires a review prior to submission
sql = sql + 'UPDATE dataset_data_subset SET has_REVIEW = 1 WHERE datas
et_data_subset.submit_for_review = 1;'

# Variable: sum of has_OMB, has_ATOU, has_KW, has_DESC, and has_PSF.
sql = sql + 'UPDATE dataset_data_subset SET ofus = has_OMB + has_ATOU
+ has_KW + has_DESC + has_PSF;'

# Execute the sql
c.executescript(sql)
```

```
# Close the connection
db.close()
```

7. Save `dataset_data_subset` table as an external file: `dataset_data_subset.txt` for later analysis using SPSS.

- **Note:** Use the extention `*.txt` instead of `*.tsv` because SPSS does not understand `*.tsv`

In [42]:
```
# Open the dataquality database
db = sqlite3.connect(db_path)

# Get the subset from the database and save to an external file
df = pd.read_sql('SELECT * from dataset_data_subset', db)
df.to_csv(dataset_data_subsets_tsv, sep='\t',index=False)

# Close the connection
db.close()
```

8. Use Excel to save `dataset_data_subset.tsv` as `dataset_data_subset.xlxs`

- **Note:** We need to resave the file as Excel format because there is a complex record in the file that contains nested quotes that confuses the SPSS tab-separated-values and comma-separated-values parsers.

SPSS Data Analysis

Ingest `dataset_data_subset.tsv` into SPSS

- Load `dataset_data_subset.txt` into SPSS and manually set all variables to the correct type. For instance, `has_PSF` is set to type `numeric`.
- Save SPSS dataset to `{HOME}/hdofu/spss/hdofu_dataset_data_subset.sav`

Perform statistical analyses

- Please see final paper for details

End document.