Using a Jupyter Notebook to perform a reproducible scientific analysis over semantic web sources

Alasdair J G Gray (ORCID:0000-0002-5711-4872 (http://orcid.org/0000-0002-5711-4872))
Heriot-Watt University, Edinburgh, UK

Abstract: In recent years there has been a reproducibility crisis in science. Computational notebooks, such as Jupyter, have been touted as a solution to this problem. However, when executing analyses over live SPARQL endpoints, we get different answers depending upon when the analysis in the notebook was executed. In this paper, we identify some of the issues discovered in trying to develop a reproducible analysis over a collection of biomedical data sources and suggest some best practice to overcome these issues.

Keywords: Reproducibility, Computational Notebooks

1. Introduction

In recent years there has been a reproducibility crisis in science (Baker; 2016). Computational notebooks such as Jupyter (Kluyver et al; 2016) that combine analysis with narrative have been touted as a solution to this problem. The approach is known as Literate Programming (Knuth; 1984). A key idea of Literate Programming is that the documentation of the analysis code is kept up to date with the code, as they are edited in the same environment. When writing the analysis, the author is forced to think about how others can understand and interpret the analysis, thus writing analysis in a way to aid understanding of the code (Piccolo, Frampton; 2016). Ideally, the computation notebook becomes the publication, as this notebook attempts. Specifically, in this notebook we investigate the feasibility of publishing a reusable data analysis paper as a Jupyter Notebook in the Semanatic Web community.

There is existing work on publishing computational workflows via notebooks (Kluyver et al; 2016). Indeed, in their paper Kluyver et al recognise some of the limitations of academic publishing within a Jupyter notebook, particularly in the lack of support for citations. In this work, we are looking at performing an analysis over pharmacology Linked Data. The contents of this notebook provide a good starting point for those who want to do an analysis of the content of the chemical substances known in some of the leading chemisty databases used to support early stage pharmacology research, and supports the understanding of the evolution of their content. The analysis will compare the compound count between ChEBI (Hastings et al; 2016), ChEMBL (Gaulton et al; 2017), DrugBank (Wishart et al; 2018), and Guide to Pharmacology (Harding et al; 2018).

In the rest of this notebook, we will first setup our computational environment in Section 2. We will then perform a simple, but crucial, analysis in Section 3 that does a crude comparison over four major pharmacology datasets. We will then discuss and conclude the paper in Section 4.

2. Method

2.1 Computational Environment

This notebook was prepared using Jupyter server (http://jupyter.org/) version 5.0.0 running on Python 3.6.3 distributed by Anaconda, Inc. (https://anaconda.org/) (default, Oct 6 2017, 12:04:38) [GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)].

2.2 Configuring the Environment

The following cells load various support libraries and define functions. The functions enable the running of queries over SPARQL endpoints and printing the results.

First we will pull in various libraries that we will use. At the time of publication, the following versions of these libraries were used:

- SPARQLWrapper (https://rdflib.github.io/sparqlwrapper/doc/1.8.1/): 1.8.1
- JSON (simplejson (https://pypi.org/project/simplejson/)): 3.15

In [1]:

```
from SPARQLWrapper import SPARQLWrapper, JSON
```

Ideally at this point we would print the version number of each module imported automatically. However the `print (SPARQLWrapper.__version__)` command does not work when using the `from SPARQLWrapper` import mechanism.

2.3 Configuring SPARQL endpoints

The endpoints used in this workbook:

- DrugBank: http://bio2rdf.org/sparql (http://bio2rdf.org/sparql)
- EBI: https://www.ebi.ac.uk/rdf/services/sparql (https://www.ebi.ac.uk/rdf/services/sparql)
- Guide to Pharmacology: https://rdf.guidetopharmacology.org/sparql (https://rdf.guidetopharmacology.org/sparql)

Note that the EBI SPARQL endpoint is used to query both ChEBI and ChEMBL (Jupp et al; 2014). DrugBank is not originally published as RDF, so we use the Bio2RDF conversion and endpoint (Callahan et al; 2013).

2.3.1 DrugBank Endpoint

In [2]:

```
#Define DrugBank SPARQL endpoint and function to run queries over it
dbSparql = SPARQLWrapper("http://bio2rdf.org/sparql")
dbSparql.setReturnFormat(JSON)
def queryDrugBank(query):
    dbSparql.setQuery(query)
    results = dbSparql.queryAndConvert()
    return results
```

While the Bio2RDF documentation (https://github.com/bio2rdf/bio2rdf-scripts/wiki/Bio2RDF-Dataset-Provenance#querying-the-provenance-graph) includes a query to extract the VoID metadata for the dataset, the query only provides results for the DisGeNet dataset, not DrugBank in which we are interestered.

In [3]:

```
query = """
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX void: <http://rdfs.org/ns/void#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT * WHERE {
?dataset rdf:type void:Dataset .
?dataset rdfs:label ?datasetLabel .
?dataset dcterms:created ?creationDate .
?dataset dcterms:creator ?creationScript .
?dataset dcterms:rights ?license .
?dataset prov:wasDerivedFrom ?parentDataSet .
?dataset void:dataDump ?downloadLink .
?dataset void:sparqlEndpoint ?endpointLink .
}
"""
queryDrugBank(query)
```

Out[3]:

```
{'head': {'link': [],
  'vars': ['dataset',
   'datasetLabel',
   'creationDate',
   'creationScript',
   'license',
   'parentDataSet',
   'downloadLink',
   'endpointLink']},
 'results': {'bindings': [], 'distinct': False, 'ordered': True}}
```

According to the Bio2RDF website (http://download.bio2rdf.org/files/release/3/release.html), the version of DrugBank loaded dates from 2014-07-25. The original DrugBank version is not stated.

2.3.2 EBI SPARQL Endpoint

In [4]:

```
#Define EBI SPARQL endpoint and function to run queries over it
#EBI endpoint can be used to query a variety of datasets including ChEMBL and Ch
EBI
ebiSparql = SPARQLWrapper("https://www.ebi.ac.uk/rdf/services/sparql")
ebiSparql.setReturnFormat(JSON)
def queryEBI(query):
    ebiSparql.setQuery(query)
    results = ebiSparql.queryAndConvert()
    return results
```

The following cell executes a query to determine what version of ChEMBL is currently loaded into the EBI triplestore. At the time of writing, the verion was 24.0, which was last modified on 5 January 2018.

In [5]:

```
#Query to determine the version of ChEMBL loaded
query = """
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX pav: <http://purl.org/pav/>

SELECT  ?version ?date
where {
  <http://rdf.ebi.ac.uk/dataset/chembl> pav:hasCurrentVersion ?currentVersion.
  ?currentVersion pav:version ?version ;
                  dcterms:hasDistribution ?dist.
  ?dist pav:lastUpdateOn ?date
}
"""
queryEBI(query)
```

Out[5]:

```
{'head': {'vars': ['version', 'date']},
 'results': {'bindings': [{'date': {'datatype': 'http://www.w3.org/2
001/XMLSchema#dateTime',
     'type': 'typed-literal',
     'value': '2018-01-05T00:00:00.0'},
    'version': {'type': 'literal', 'value': '24.0'}}]}}
```

2.3.3 Guide to Pharmacology SPARQL Endpoint

In [6]:

```
#Define Guide to Pharmacology SPARQL endpoint and function to run queries over i
t
gtpSparql = SPARQLWrapper("https://rdf.guidetopharmacology.org/sparql")
gtpSparql.setReturnFormat(JSON)
def queryGtPdb(query):
    gtpSparql.setQuery(query)
    results = gtpSparql.queryAndConvert()
    return results
```

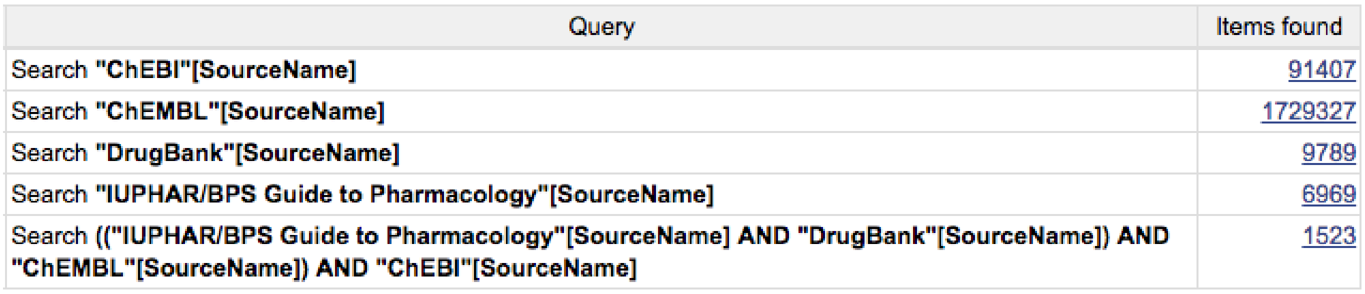
Currently, the Guide to Pharmacology does not support querying of their metadata. A VoID file is available through their website (http://www.guidetopharmacology.org/download.jsp#rdf) but this has not yet been loaded into their triplestore. The latest version of the Guide to Pharmacology data is listed as `2018.2`.

2.3.4 Discussion

It would be ideal to query each of the endpoints to programmatically determine the version of the dataset that was loaded at the time of execution, as we did with the EBI endpoint. This can be achieved using the VoID (https://www.w3.org/TR/void/) vocabulary, although acquiring the knowledge to the structure of the metadata and the graph in which it is loaded is often reliant on the documentation of the data.

3. Analysis: Compare the number of compound structures known in GtoPdb, ChEMBL, DrugBank, and ChEBI

This question arose from discussions with Chris Southan (https://scholar.google.co.uk/citations?user=y1DsHJ8AAAAJ&hl=en), University of Edinburgh, whilst creating the Guide to Pharmacology RDF data. Chris currently performs his investigation on PubChem (https://pubchem.ncbi.nlm.nih.gov/). I recreated Chris's search queries (15/3/2018) and generated the output shown in the figure below:



Query	Items found
Search **"ChEBI"[SourceName]**	91407
Search **"ChEMBL"[SourceName]**	1729327
Search **"DrugBank"[SourceName]**	9789
Search **"IUPHAR/BPS Guide to Pharmacology"[SourceName]**	6969
Search **(("IUPHAR/BPS Guide to Pharmacology"[SourceName] AND "DrugBank"[SourceName]) AND "ChEMBL"[SourceName]) AND "ChEBI"[SourceName]**	1523

Each of the queries can be rerun by the following links:

- ChEBI: "ChEBI"[SourceName] (https://www.ncbi.nlm.nih.gov/pccompound?term=%22ChEBI%22%5BSourceName%5D&cmd=DetailsSearch)
- ChEMBL: "ChEMBL"[SourceName] (https://www.ncbi.nlm.nih.gov/pccompound?term=%22ChEMBL%22%5BSourceName%5D&cmd=DetailsSearch)
- DrugBank: "DrugBank"[SourceName] (https://www.ncbi.nlm.nih.gov/pccompound?term=%22DrugBank%22%5BSourceName%5D&cmd=DetailsSearch)
- Guide to Pharmacology: "IUPHAR/BPS Guide to Pharmacology"[SourceName] (https://www.ncbi.nlm.nih.gov/pccompound?term=%22IUPHAR/BPS Guide to Pharmacology%22%5BSourceName%5D&cmd=DetailsSearch)
- Intersection of all sources: (("IUPHAR/BPS Guide to Pharmacology"[SourceName] AND "DrugBank"[SourceName]) AND "ChEMBL"[SourceName]) AND "ChEBI"[SourceName] (https://www.ncbi.nlm.nih.gov/pccompound?term=%28%28%22IUPHAR/BPS%20Guide%20to%20Pharmacology%22%5BSourceName%5D%20AN

We will now recreate the same data by querying the SPARQL endpoints.

In [7]:

```
# Define function to extract count result from JSON SPARQL result set
def extract_count(results):
    """
    Extract the count result from the JSON format
    """
    for result in results["results"]["bindings"]:
        return result["count"]["value"]

#Initialise counts dictionary to store count for each dataset in
counts = {}

# Find the number of InChI Keys stored in ChEBI
query = """
SELECT  (count(?inchikey) as ?count)
WHERE {
    ?ligand <http://purl.obolibrary.org/obo/chebi/inchikey> ?inchikey.
}
"""
results = queryEBI(query)
counts['ChEBI'] = extract_count(results)

# Find the number of InChI Keys stored in ChEMBL
query = """
SELECT (COUNT(?substance) as ?count)
FROM <http://rdf.ebi.ac.uk/dataset/chembl>
WHERE {
?substance a <http://semanticscience.org/resource/CHEMINF_000059>
}
"""
results = queryEBI(query)
counts['ChEMBL'] = extract_count(results)

#Find the number of InChI Keys stored in DrugBank
query = """
    SELECT (COUNT(?s) as ?count)
    WHERE {?s a <http://bio2rdf.org/drugbank_vocabulary:InChIKey>}
"""
results = queryDrugBank(query)
counts['DrugBank'] = extract_count(results)

# Find the number of InChI Keys stored in GtPdb
query = """
    PREFIX gtpo: <http://rdf.guidetopharmacology.org/ns/gtpo#>
    SELECT  (count(?inchikey) as ?count)
    WHERE {
        ?ligand gtpo:inChIKey ?inchikey.
    }
"""
results = queryGtPdb(query)
counts['GtPDb'] = extract_count(results)

# Print results
print (counts)
```

```
{'ChEBI': '184393', 'ChEMBL': '1820035', 'DrugBank': '6810', 'GtPD
b': '7065'}
```

As can already be seen, by June 2018 the content of the SPARQL endpoints differs from that in the PubChem snapshot from March 2018. See table below.

Dataset	PubChem	SPARQL
ChEBI	91,407	184,393
ChEMBL	1,729,327	1,820,035
DrugBank	9,789	6,810
Guide to Pharmacology	6,969	7,065
Intersection	1,523	--

In general, the datasets contain more data. This is to be expected as the datasets are adding more content. The exception to this is DrugBank, which has seen a decrease. This is due to the Bio2RDF endpoint offering an older version of the dataset.

On PubChem we were able to calculate the intersection of the datasets, i.e. we could count the number of substances that are contained in all the datasets. This has not been possible with the SPARQL endpoints due to the distributed nature of the endpoints and the unreliability of federated queries. There are also complications due to the way that the substance structures (InChI Keys) are represented in the various datasets. The following queries demonstrate the output of the first 5 results of each dataset.

In [8]:

```
# Show 5 InChI Keys stored in ChEBI
query = """
SELECT  ?inchikey
WHERE {
    ?ligand <http://purl.obolibrary.org/obo/chebi/inchikey> ?inchikey.
} LIMIT 5
"""
queryEBI(query)
```

Out[8]:

```
{'head': {'vars': ['inchikey']},
 'results': {'bindings': [{'inchikey': {'datatype': 'http://www.w3.o
rg/2001/XMLSchema#string',
     'type': 'typed-literal',
     'value': 'NVKAWKQGWWIWPM-ABEVXSGRSA-N'}},
   {'inchikey': {'datatype': 'http://www.w3.org/2001/XMLSchema#strin
g',
     'type': 'typed-literal',
     'value': 'NVKAWKQGWWIWPM-ABEVXSGRSA-N'}},
   {'inchikey': {'datatype': 'http://www.w3.org/2001/XMLSchema#strin
g',
     'type': 'typed-literal',
     'value': 'ATHGHQPFGPMSJY-UHFFFAOYSA-N'}},
   {'inchikey': {'datatype': 'http://www.w3.org/2001/XMLSchema#strin
g',
     'type': 'typed-literal',
     'value': 'ATHGHQPFGPMSJY-UHFFFAOYSA-N'}},
   {'inchikey': {'datatype': 'http://www.w3.org/2001/XMLSchema#strin
g',
     'type': 'typed-literal',
     'value': 'JKLRIMRKZBSSED-UHFFFAOYSA-N'}}]}}
```

In [9]:

```
# Show 5 InChI Keys stored in ChEMBL
query = """
SELECT ?inchiKey
FROM <http://rdf.ebi.ac.uk/dataset/chembl>
WHERE {
?substance a <http://semanticscience.org/resource/CHEMINF_000059>;
  <http://semanticscience.org/resource/SIO_000300> ?inchiKey .
} LIMIT 5
"""
queryEBI(query)
```

Out[9]:

```
{'head': {'vars': ['inchiKey']},
 'results': {'bindings': [{'inchiKey': {'type': 'literal',
     'value': 'NUNJVAPBPSWAAF-AATRIKPKSA-N'}},
   {'inchiKey': {'type': 'literal', 'value': 'JJMSWOAYYIOHPY-UHFFFAO
YSA-N'}},
   {'inchiKey': {'type': 'literal', 'value': 'PGPLULVIABTRIF-YNHSGCS
HSA-N'}},
   {'inchiKey': {'type': 'literal', 'value': 'QLZXDQMITOAIAU-KCHLEUM
XSA-N'}},
   {'inchiKey': {'type': 'literal', 'value': 'ODLHCKMRHURGDZ-MIBBOOF
ZSA-N'}}]}}
```

In [10]:

```
# Show 5 InChI Keys stored in DrugBank
query = """
    SELECT ?inchi
    WHERE {?s a <http://bio2rdf.org/drugbank_vocabulary:InChIKey>;
        <http://bio2rdf.org/drugbank_vocabulary:value> ?inchi.
    } LIMIT 5
"""
queryDrugBank(query)
```

Out[10]:

```
{'head': {'link': [], 'vars': ['inchi']},
 'results': {'bindings': [{'inchi': {'datatype': 'http://www.w3.org/
2001/XMLSchema#string',
     'type': 'typed-literal',
     'value': 'InChIKey=PMATZTZNYRCHOR-IMVLJIQENA-N'}},
   {'inchi': {'datatype': 'http://www.w3.org/2001/XMLSchema#string',
     'type': 'typed-literal',
     'value': 'InChIKey=SFKQVVDKFKYTNA-YVGXZPIDNA-N'}},
   {'inchi': {'datatype': 'http://www.w3.org/2001/XMLSchema#string',
     'type': 'typed-literal',
     'value': 'InChIKey=DEQANNDTNATYII-RRCPSWKPSA-N'}},
   {'inchi': {'datatype': 'http://www.w3.org/2001/XMLSchema#string',
     'type': 'typed-literal',
     'value': 'InChIKey=NGVDGCNFYWLIFO-UHFFFAOYSA-N'}},
   {'inchi': {'datatype': 'http://www.w3.org/2001/XMLSchema#string',
     'type': 'typed-literal',
     'value': 'InChIKey=SEKGMJVHSBBHRD-WZHZPDAFSA-M'}}],
  'distinct': False,
  'ordered': True}}
```

As can be seen from the results, additional string processing would be required to compare the InChI Key string from DrugBank with the other datasets. This would be a costly, but not impossible operation. However, one that would be best suited to centralising all the data first and then performing the SPARQL queries.

4. Conclusions

In this notebook, we have conducted a very simple analysis, counting the number of compounds in different pharmacology datasets. However, it has raised several questions such as:

1. How should you present the computational environment used?
2. How should you present the results of the original computation against those of the live results when the content of datasets is evolving?

For the first question, you would ideally use the computational environment to report about itself. However, this is not always straightforward. Libraries don't necessarily report their versions, and SPARQL endpoints all have different approaches (or perhaps none) for providing metadata about their datasets.

The second question focuses on recognising that sources change over time, often gaining more date. The publication of this notebook captures a point in time, or indeed two. The first is the time when the query was run over the PubChem database and the second when the SAPRQL queries were executed. This allowed us to compare the results. In this notebook we chose to copy the results of the computation for publication into the text in order to allow others reusing the notebook to be able to compare their answers. We hope that this proves to be a useful approach.

We have deliberately not used any Jupyter extensions in this notebook, believing that this will aid reproducibility into the future. However, this complicated the generating of the paper in this notebook, as there is no inbuilt support for academic references. If the use of computational notebooks is to grow, then referencing needs to become a core feature. In the future, it would be interesting to investigate the additional effort required in making the notebook itself a semantic resource.

References

Baker, M. 1,500 scientists lift the lid on reproducibility. Nature 533, 452–454 (2016).
DOI: 10.1038/533452a (https://doi.org/10.1038/533452a)

Callahan A., Cruz-Toledo J., Ansell P., Dumontier M. Bio2RDF Release 2: Improved Coverage, Interoperability and Provenance of Life Science Linked Data. ESWC 2013: 200-212. (2013)
DOI: 10.1007/978-3-642-38288-8_14 (https://doi.org/10.1007/978-3-642-38288-8_14)

Gaulton A, Hersey A, Nowotka M, Bento AP, Chambers J, Mendez D, Mutowo P, Atkinson F, Bellis LJ, Cibrián-Uhalte E, Davies M, Dedman N, Karlsson A, Magariños MP, Overington JP, Papadatos G, Smit I, Leach AR. The ChEMBL database in 2017. Nucleic Acids Research, 45(D1) D945-D954. (2017)
DOI: 10.1093/nar/gkw1074 (https://doi.org/10.1093/nar/gkw1074)

Harding SD, Sharman JL, Faccenda E, Southan C, Pawson AJ, Ireland S, Gray AJG, Bruce L, Alexander SPH, Anderton S, Bryant C, Davenport AP, Doerig C, Fabbro D, Levi-Schaffer F, Spedding M, Davies JA; NC-IUPHAR. The IUPHAR/BPS Guide to PHARMACOLOGY in 2018: updates and expansion to encompass the new guide to IMMUNOPHARMACOLOGY. Nucleic Acids Research, 46 (Issue D1): D1091-D1106 (2018).
DOI: 10.1093/nar/gkx1121 (https://doi.org/10.1093/nar/gkx1121)

Hastings J, Owen G, Dekker A, Ennis M, Kale N, Muthukrishnan V, Turner S, Swainston N, Mendes P, Steinbeck C. ChEBI in 2016: Improved services and an expanding collection of metabolites. Nucleic Acids Research (2016).
DOI: 10.1093/nar/gkv1031 (https://doi.org/10.1093/nar/gkv1031)

Jupp S, Malone J, Bolleman J, Brandizi M., Davies M., Garcia L., Gaulton A., Gehant S., Laibe C., Redaschi N., Wimalaratne S.M., Martin M., Le Novère N., Parkinson H., Birney E. and Jenkinson A.M. The EBI RDF Platform: Linked Open Data for the Life Sciences Bioinformatics 30 1338-1339. 2014.
DOI: 10.1093/bioinformatics/btt765 (https://doi.org/10.1093/bioinformatics/btt765)

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B.E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J.B., Grout, J., Corlay, S. and Ivanov, P. Jupyter Notebooks-a publishing format for reproducible computational workflows. In ELPUB, 87-90 (2016).
DOI: 10.3233/978-1-61499-649-1-87 (https://doi.org/10.3233/978-1-61499-649-1-87)

Knuth, D. E. Literate Programming. The Computer Journal 27, 97–111 (1984).
DOI: 10.1093/comjnl/27.2.97 (https://doi.org/10.1093/comjnl/27.2.97)

Piccolo, S. R. & Frampton, M. B. Tools and techniques for computational reproducibility. Gigascience 5, 30 (2016).
DOI: 10.1186/s13742-016-0135-4 (https://doi.org/10.1186/s13742-016-0135-4)

Wishart DS, Feunang YD, Guo AC, Lo EJ, Marcu A, Grant JR, Sajed T, Johnson D, Li C, Sayeeda Z, Assempour N, Iynkkaran I, Liu Y, Maciejewski A, Gale N, Wilson A, Chin L, Cummings R, Le D, Pon A, Knox C, Wilson M. DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic Acids Research, 2017.
DOI: 10.1093/nar/gkx1037 (https://10.1093/nar/gkx1037)