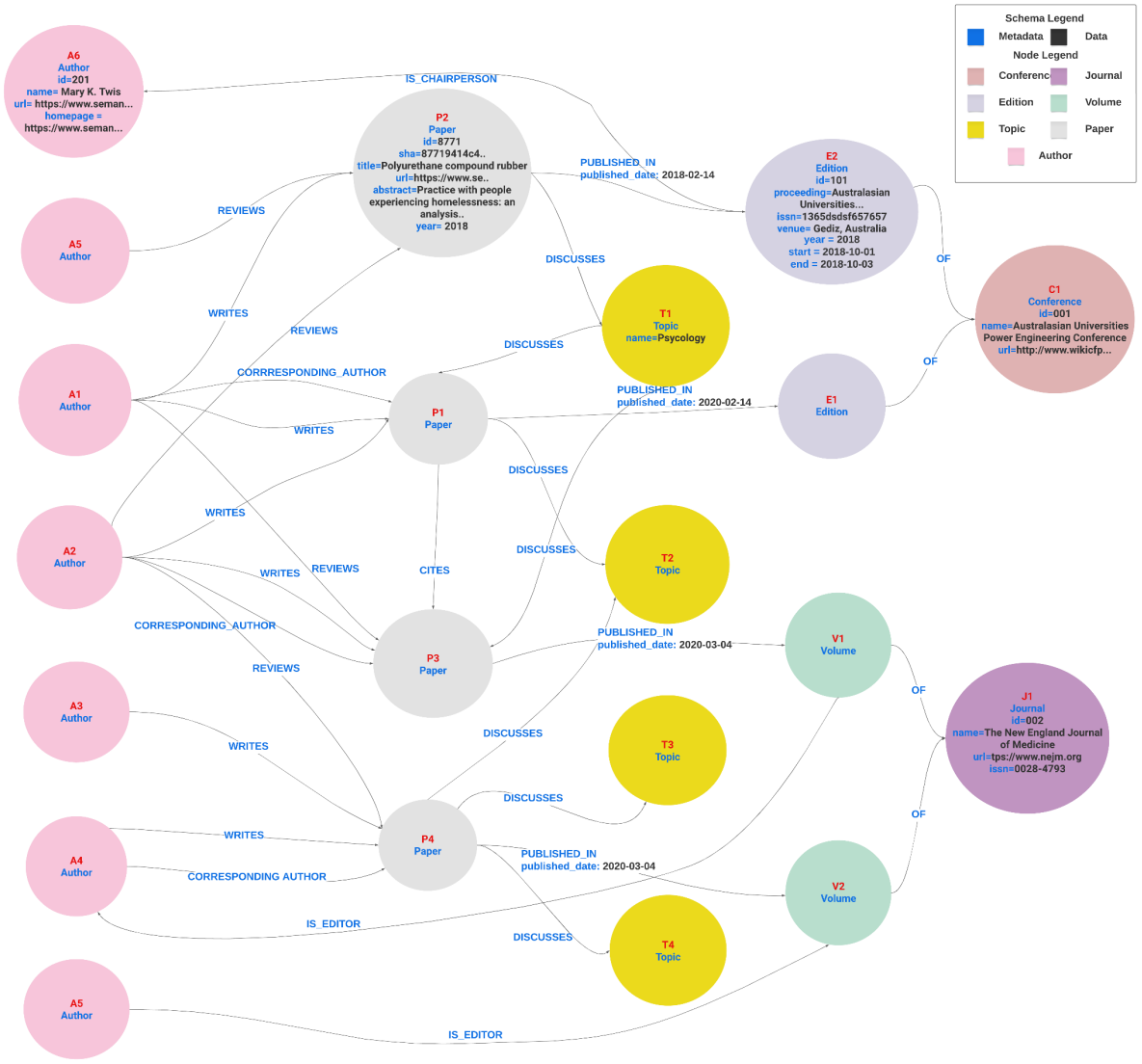
Lab 1

A. Modeling, Loading, Evolving

A.1 Modeling



Assumptions:

1. *A paper is submitted and published only in one proceeding or journal.* This is assumed since most journals, especially those that are peer-reviewed, do not allow submissions that have already been published or submitted to other journals (Wager, 2009).
2. *Conferences have one edition per year and similarly journals have one volume per year.* This is because scientific journals are usually organized once a year
3. *Editions start and end within the same year.* This is for easier imputation since collected data is sparse.
4. *Keywords are considered as topics.* Both pertain to what a paper mainly discusses about and are thus used interchangeably for the purpose of this report.
5. *Editors of journal volumes and chairpersons of conference editions can also be authors or reviewers of papers submitted.* This is because they usually have a decorated scientific career and would have authored at least one paper in their lifetime (Hassink, 2013).
6. *Both conferences and workshops are assumed to have similar formats.*

Design Decisions:

Node types in the graph : Paper, Author, Conference, Edition, Journal, Volume & Topic

a. **Distinction between Conference and Journal :** It is necessary as there is also a clear distinction between the two in real life which are reflected in the queries as well.
   i. Impact factor of journals
   ii. Acceptance rate of conference
b. **One to Many relationships leading to distinction between entities :**
   i. Conference and Editions
   ii. Journal and Volumes
   iii. Topic and Papers
   iv. Edition/Volume and Papers
   v. Paper and Authors

   This is to reduce data redundancy and increase performance of the queries.
   As an example consider case (i),
   Option I: If Conference details are included in Edition, then these details will have to be redundantly included in Edition nodes. Also in order to get editions of a conference all the Edition nodes will have to be traversed.
   Option II: If editions are represented by edges between Conference and Paper, then the edition details will have to be redundantly included on the edges
c. **Proceeding is included as a property in Edition** : Property and Edition has a one to one relationship. Even though the two appear to be separate entities, considering the performance of the queries (If proceeding is a node, it adds one more hop to the path between Conference and Paper), they are merged.

*Properties of each **node** :*

Conference: id, name, url
Edition: id, proceeding, issn, venue, year, start, end
Journal: id, name, url,issn
Volume: id, name, year
Paper: id, sha, title, url, abstract, year
Author: id, name, url, homepage
Topic: name

a. In general, node ids are kept to allow for easier transition from csv to loading the data in neo4j. This especially comes in handy when updating and defining new edges since nodes may have more than one attribute differentiating it from others.
b. **Year** property value is casted to integer at data loading to improve query performance (in Edition, Volume and Paper)
   Example : Query with arithmetic comparison - Total publications of two consecutive volumes of a journal
c. Publication **year** is added to the paper redundantly in order to improve query performances. (also in the submitted edition/volume)
   Example : Papers of a certain year - Without year as a property in paper, linked edition/volume needs to be traversed in order to get the published year. Hence adding year as a property to paper means that one hop will be reduced in similar queries which are frequent.
d. **Start and End** property values are casted to Date types at data loading to improve query performance
e. Except for the properties mentioned above, all other property values are stored as strings.
f. **Venue** is not split into a hierarchy (i.e. city, country) as queries related to geographic location are not that frequent in this use case
g. **Proceeding** is a property in Edition following the reasoning in (c) part of previous section

Edge types in the graph :

a. **Representing One to Many relationships** :
   WRITES, OF, PUBLISHED_IN, IS_EDITOR, IS_CHAIRPERSON, DISCUSSES
b. **CORRESPONDING_AUTHOR as an edge :**
   i. Author already exists as a node. Adding corresponding author as a property in node would be redundant
   ii. Increase query performance. Consider how many times/ in which papers has an author been the corresponding author of the papers published by him
c. **Representing Many to Many relationships :**
   i. REVIEWS ii. CITES
   If each relationship is to be represented as properties of the nodes it will be redundant, reduce query performance and sometimes reduce maintainability.
   Example :
   In order to represent CITES relationship, two arrays will have to be added into the related Paper nodes to indicate the referenced papers and cited papers. This would be redundant. Moreover two nodes will have to be updated in order to indicate a single relationship

Properties of edges : PUBLISHED_IN edge - published_date

A.2 Instantiating/Loading

Source: Semantic Scholar
Pipeline:
The following process was taken in the construction of the research publication dataset, in preparation for loading into Neo4j:

1. **Paper-id dataset** was downloaded from Semantic Scholar
2. For each paper id (corpusid) **the Paper API was queried** for a json entry with complete details.
3. Queried data in JSON format was **converted to CSV** using python and separated into different files, one for each node or edge, as described in the graph database model in Section A.1.
4. Initial **cleaning** was done for each csv file, in which the following were addressed:
   a. Drop duplicates
   b. Remove null values for id and name (ex: Volumes with null ids and names)
   c. Type casts (ex: Cast float values to integers - years, numeric ids)
5. **Synthesize** and impute missing data: **(synthesize.py)**
   a. *Corresponding author :* Inferred as the first author in the author list per publication.
   b. *Start and End date for Conference Editions :* Consider maximum and minimum published dates of the papers of an edition
   c. Generate synthetic nodes: papers, editions/volumes, and their corresponding edges: of, submitted to. : To be able to find communities for queries in Part B, new editions were created. Fake papers and their properties were also randomly generated together with

connections to existing authors. For more comprehensive data, this process was repeated for journals.

d. *Generate editors and chairpersons for journals and conferences respectively :* It was assumed that chairpersons and editors are not authors of papers in the same conference or journal they are currently organizing. For each journal/conference, an author is randomly assigned with an 80% chance to get an existing author, 20% synthetic.
e. *Generate reviewers and their corresponding reviews per paper :* For each paper, an author who was not an author of the paper nor was an author in the same volume or edition the paper was submitted to, was assigned. For each REVIEW relationship, random paragraphs were generated as review text and the decision was set to 0 or 1 if rejected or accepted, respectively with 80% chance of acceptance. This rate was chosen so that more papers would be accepted and query results would be populated. If a paper is accepted, 2 or 3 of the reviewers will be randomly selected to have decision = 1. If rejected, 0 or 1 reviewers were randomly chosen.
f. *Generate additional citations between papers :* As many queries rely on citation relationships between papers, more CITED relationships were synthesized. Only published papers with their publication date less than the submission date of the paper citing it were cited.
g. *Generate venues for each edition :* Randomly assigned a city to each conference edition based on the world city list as provided in the world-cities Github repository (*World-Cities*, n.d.)
h. *Generate current institutions and affiliations per author :* For each institution listed in the world-universities Github repository (Stutern, LLC., n.d.), a probability was assigned using the Dirichlet distribution. Based on these probabilities, institutions were randomly assigned to each author.
i. *Generate additional topics, specifically for the database community :* Keywords related to the database community, as enumerated in Part D, were added to the node list of topics and randomly assigned to papers.

6. **Data was loaded** using Neo4j driver for Python **(PartA.2_BondocGanepola.py)**, using the LOAD CSV WITH HEADERS predicate from Cypher. For this to work, csv data must be in the assigned import library of the Neo4j project. The following nodes and edges were initially loaded:
   - Nodes: Paper, Author, Conference, Journal, Topic, Edition, Volume
   - Edges: Writes, Corresponding Author, Reviews, Is Chairperson, Is Editor, Of, Submitted to, Discusses, Cites

Note: Edit input.py file with the correct input data as described in the file before any execution.

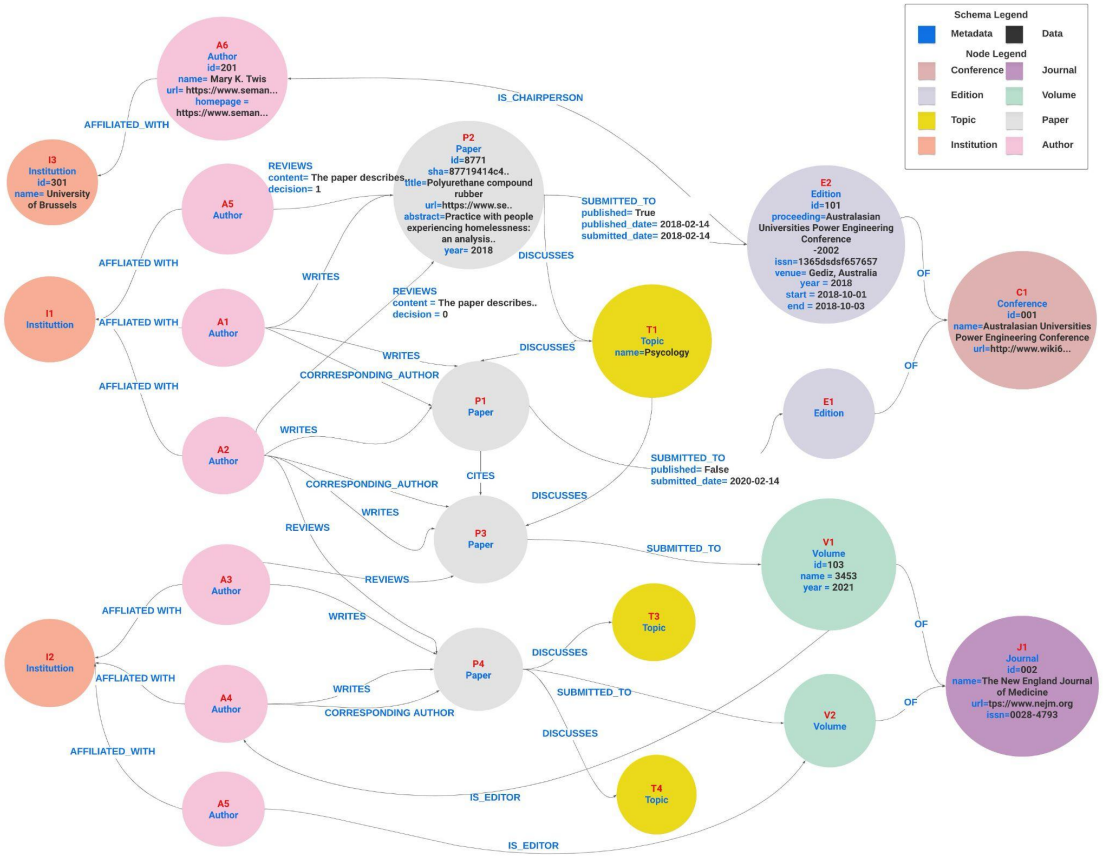
A3. Evolving the Graph (PartA.3_BondocGanepola.py)

Store reviews and decision sent by each reviewer

- New properties “content” (paragraph of review) and “decision” (1 = accept, 0 = reject) were added to the graph.
- The “PUBLISHED_IN” edge was converted to “SUBMITTED_TO” in order to include the submitted but rejected papers. The inclusion of rejected papers allows for a richer graph in which queries such as acceptance rate of a journal can be queried in the future.
- A new derived property called “published” (True if sum of decisions in reviews per paper is >1, otherwise False) was added to the edge “SUBMITTED_TO”, to improve query performance where only published papers are considered.

Affiliations of authors

- A new node for **Institution** was created, with its corresponding id and name and was connected to authors with a new edge called "AFFILIATED_WITH". *Note that only the author's current affiliation is stored so one author can only be affiliated with one university.* This is to facilitate queries where universities want to know the past work of the people/authors who are currently affiliated with them.



B. Querying (PartB_BondocGanepola.py)

1. Find the top 3 most cited papers of each conference.

```
MATCH (:Paper)-[i:CITES]->(p:Paper)
-[:SUBMITTED_TO {published:True}]->(:Edition)-[:OF]->(c:Conference)
WITH COUNT(i) as cite_cnt, p, c
ORDER BY cite_cnt DESC
WITH c, COLLECT([p, cite_cnt])[..3] AS topPapers
ORDER BY c.name
UNWIND topPapers AS a
RETURN c.name as Conference, a[0].title AS Paper, a[1] AS citation_count
```

2. Finding the community for each conference: i.e., those authors that have published papers on that conference in, at least, 4 different editions.

```
MATCH (c:Conference)<-[:OF]-(e:Edition)<-[:SUBMITTED_TO
{published:True}]-(:Paper)<-[:WRITES]-(a:Author)
```

```
WITH c.name as conf, a.name as auth, COUNT(DISTINCT(e)) as editionCount
WHERE editionCount >=1
RETURN conf as conference, COLLECT(auth) as community
```

3. Finding the impact factors of journals in the graph

Assumption: For the first two volumes of a journal, the impact factor is considered as 0 and since the 3rd volume the impact factor is calculated according to the following formula from Wikipedia:

$$Impact\ Factor\ =\ \frac{Citations_x}{Publications_{x-1}+Publications_{x-2}}$$

```
CALL{MATCH (j:Journal)<-[:OF]-(v:Volume)<-[:SUBMITTED_TO {published:TRUE}]-
(:Paper)<-[c:CITES]-(:Paper)-[:SUBMITTED_TO{published:TRUE}]-()
   WITH j, (v.year) as yr0, count(c) as citations
   MATCH (j:Journal)<-[:OF]-(v1:Volume{year:(yr0-1)})<-[s:SUBMITTED_TO
{published:TRUE}]-(:Paper)
   WITH j, citations, yr0, v1.year as yr1, count(s) as pub1
   MATCH (j:Journal)<-[:OF]-(v2:Volume{year:(yr1-1)})<-[s2:SUBMITTED_TO
{published:TRUE}]-(:Paper)
   WITH j, yr0, citations, yr1, pub1, v2.year as yr2, count(s2) as pub2
   ORDER BY j.name, yr0
   RETURN j.name as Journal, yr0 as Year, toFloat(citations)/toFloat(pub1+pub2) as impactFactor
   UNION ALL
   MATCH (j:Journal)<-[:OF]-(v:Volume)
   RETURN j.name as Journal, v.year as Year, 0 as impactFactor}
RETURN Journal, Year, max(impactFactor) as impactFactor
```

4. Finding the h-indexes of authors, based on Wikipedia's formula (Wikipedia, n.d.)

Assumption: Papers will not be cited unless they were published. Hence the `published=TRUE` condition is not checked.

```
MATCH (a:Author)-[:WRITES]->(p:Paper)<-[c:CITES]-(:Paper)
WITH a,p, count(c) as citation
ORDER BY citation DESC
WITH a, collect([citation]) as col
WITH a , RANGE(0, SIZE(col)-1) AS indexes, col
UNWIND indexes As i
WITH a, COLLECT([i+1,col[i][0]]) as citIndex
WITH a ,([x in citIndex WHERE x[0]<=x[1] | x[0]])[-1] as hIndex
ORDER BY hIndex DESC
RETURN a.name as author , hIndex
```

C. Recommender system (PartC_BondocGanepola.py)

Point 1: Find and define research communities

```
MATCH (t)
WHERE ANY (topic IN t.name WHERE topic IN ['Data Management', 'Indexing',
'Data Modeling','Big Data', 'Data Processing', 'Data Storage', 'Data Querying'])
RETURN t
```

Point 2: Find conferences and journals related to the database community

```
MATCH (t)<-[:DISCUSSES]-(p:Paper)-[:SUBMITTED_TO{published:TRUE}]->()-[:OF]->(c)
WHERE ANY (topic IN t.name WHERE topic IN ['Data Management', 'Indexing',
'Data Modeling','Big Data', 'Data Processing', 'Data Storage', 'Data Querying'])
WITH c, COUNT (DISTINCT p.id) as potentialMembers
MATCH (p1:Paper)-[:SUBMITTED_TO{published:TRUE}]->()-[:OF]->(c)
WITH c, potentialMembers, COUNT (DISTINCT p1.id) as allMembers
WHERE potentialMembers/allMembers >= 0.9
RETURN c.name AS name, labels(c)[0] as label
```

Point 3: Identify the top 100 papers of these conferences/journals.

```
MATCH (t)<-[:DISCUSSES]-(p:Paper)-[:SUBMITTED_TO{published:TRUE}]->()-[:OF]->(c)
WHERE ANY (topic IN t.name WHERE topic IN ['Data Management', 'Indexing',
'Data Modeling','Big Data', 'Data Processing', 'Data Storage', 'Data Querying'])
WITH c, COUNT (DISTINCT p.id) as potentialMembers
MATCH (p1:Paper)-[:SUBMITTED_TO]->()-[:OF]->(c)
WITH c, potentialMembers, COUNT (DISTINCT p1.id) as allMembers
WHERE potentialMembers/allMembers >= 0.9
MATCH (p1:Paper)-[:SUBMITTED_TO{published:TRUE}]->()-[:OF]->(c)
WITH COLLECT ( p1) AS papersInCommunity
CALL gds.graph.project.cypher(
  'databaseCommunity',
  'UNWIND $nodes AS n RETURN id(n) AS id, labels(n) AS labels',
  'MATCH (n)-[c:CITES]->(m)
    WHERE (n IN $nodes) AND (m IN $nodes)
    RETURN id(n) AS source, id(m) AS target, type(c) AS type',
  { parameters: { nodes: papersInCommunity} })
 YIELD graphName, nodeCount AS nodes, relationshipCount AS rels
 RETURN graphName, nodes, rels
```

```
CALL gds.pageRank.stream('databaseCommunity', {maxIterations: 20, dampingFactor: 0.85})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).title AS name, score
ORDER BY score DESC, name ASC
LIMIT 100
```

Point 4: Find potential reviewers and identify gurus

Assumption: For identifying gurus who are potential reviewers for top conferences, it is assumed that they can be an author in at least two top papers in either journals or conferences.

```
CALL gds.pageRank.stream('databaseCommunity', {maxIterations: 20, dampingFactor: 0.85})
YIELD nodeId, score
WITH gds.util.asNode(nodeId) AS topPaper, score
LIMIT 100
MATCH (a:Author)-[:WRITES]->(topPaper)
RETURN DISTINCT a.name AS author , COUNT(topPaper)>=2 AS Guru
ORDER BY Guru DESC
```

D. Graph Algorithms (PartD_BondocGanepola.py)

The **Betweenness centrality** measure was used among papers and their citations, specifically for papers that discuss Data Management. It was used to identify the top 10 most influential papers that would be most efficient for finding papers within the community. Particularly, they could act like a broker in which users looking for more information about a topic will be able to find more papers efficiently just by checking the citations of these papers with high betweenness.

```
CALL gds.graph.project.cypher('papers',
  'MATCH (t)<-[:DISCUSSES]-(p:Paper) WHERE ANY (topic IN t.name WHERE topic IN ["Data
Management"]) RETURN id(p) as id',
  'MATCH (t)<-[:DISCUSSES]-(n:Paper)-[r:CITES]->(m:Paper)-[:DISCUSSES]->(t) WHERE ANY (topic
IN t.name WHERE topic IN ["Data Management"]) RETURN id(n) AS source, id(m) AS target')
YIELD graphName AS graph, nodeQuery, nodeCount AS nodes, relationshipQuery, relationshipCount AS
rels
```

```
CALL gds.betweenness.stream('papers')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).title AS PaperTitle, score as BetweennessScore
ORDER BY score DESC
LIMIT 10
```

Node Similarity (default similarity metric: Jaccard (Noe4j. n.d.)**)** is used in the below case to find papers which refer to similar subject matter. The notion is that if two papers cite the same set of papers, and if *this overlapping ratio is at least 50% of all the papers cited by them,* they would be similar in terms of the subject matter that they are related to. Hence node similarity algorithm is used to find the similarity of two papers based on their outgoing CITE edges.

```
CALL gds.graph.project.cypher('citedPapers',
'MATCH (p:Paper) RETURN id(p) as id,labels(p) as labels',
'MATCH (n:Paper)-[c:CITES]->(m:Paper) RETURN id(n) AS source, id(m) AS target, type(c) AS type')
YIELD graphName, nodeCount AS nodes, relationshipCount AS rels
RETURN graphName, nodes, rels
```

```
CALL gds.nodeSimilarity.stream('citedPapers')
YIELD node1, node2, similarity
WITH gds.util.asNode(node1).title AS Paper1, gds.util.asNode(node2).title AS Paper2, similarity
WHERE similarity >= 0.5
RETURN Paper1, Paper2, similarity
ORDER BY similarity DESCENDING, Paper1, Paper2
```

References

Hassink, L. (2013, July 1). *How do publishers choose editors, and how do they work together?* Elsevier. Retrieved March 16, 2023, from https://www.elsevier.com/connect/how-do-publishers-choose-editors-and-how-do-they-work-together

Noe4j. (n.d.). *Node Similarity*. Neo4j. Retrieved March 16, 2023, from https://neo4j.com/docs/graph-data-science/current/algorithms/node-similarity/#algorithms-node-similarity-syntax

Stutern, LLC. (n.d.). *world-universities/world-institutions.csv at master · Stutern/world-universities*. GitHub. Retrieved March 16, 2023, from https://github.com/Stutern/world-universities/blob/master/world-institutions.csv

Wager, E. (2009, Dec 30). Why You Should not Submit Your Work to More than One Journal at a Time. *National Library of Medicine*, *2010*(7 (2)), 160–161. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3021155/

Wikipedia. (n.d.). *h-index*. Wikipedia. Retrieved March 16, 2023, from https://en.wikipedia.org/wiki/H-index

Wikipedia. (n.d.). *Impact factor*. Wikipedia. Retrieved March 16, 2023, from https://en.wikipedia.org/wiki/Impact_factor

world-cities. (n.d.). GitHub. Retrieved March 16, 2023, from https://github.com/datasets/world-cities/blob/master/data/world-cities.csv