


Analyse des Open-Access-Anteils bei Zeitschriftenartikeln: Anleitung und Dokumentation eines Python-Skripts

Eva Bunge*, Michaela Voigt†

Technische Universität Berlin, Universitätsbibliothek

Stand: 20. September 2016

Das Python-Skript (BSD 3-Clause) und die vorliegende Anleitung (CC BY 4.0) sind online über das GitHub-Konto der Universitätsbibliothek der TU Berlin verfügbar:
<https://github.com/tuub/oa-eval>

 Dieses Material steht unter der Creative-Commons-Lizenz Namensnennung 4.0 International, Lizenztext s. <https://creativecommons.org/licenses/by/4.0/>.

Inhaltsverzeichnis

| | | |
|----------|--|-----------|
| 1 | Hintergrund | 2 |
| 2 | Funktionsweise des Python-Skripts | 3 |
| 2.1 | Aufbau und einzelne Funktionen | 3 |
| 2.2 | Eine neue Institution ansetzen | 7 |
| 2.3 | Eine Datenbank von der Analyse ausschließen | 9 |
| 2.4 | Eine neue Datenbank aufnehmen | 9 |
| 2.5 | Potentielle Fehlerquellen | 10 |
| 2.6 | Mögliche Erweiterungen | 11 |
| 3 | Eine Analyse durchführen | 12 |
| 3.1 | Systemanforderungen | 12 |
| 3.2 | Handhabung des Skripts | 13 |
| 3.3 | Abfrage der Datenbanken und Aufbereitung der Daten | 14 |
| | Anhang | 22 |

*ORCID:0000-0002-5587-5934

†ORCID:0000-0001-9486-3189

1 Hintergrund

In Vorbereitung auf die Antragstellung bei der Deutschen Forschungsgemeinschaft (DFG) zur Förderung eines Open-Access-Publikationsfonds wurde an der Technischen Universität Berlin (TU Berlin) folgendes Verfahren für die Analyse des Publikationsaufkommens entwickelt. Dieses Verfahren wurde weiterentwickelt für die Analyse des Open-Access-Anteils wissenschaftlicher Zeitschriftenartikel der Angehörigen von Berliner Bildungs- und Forschungseinrichtungen.

Für die DFG-Antragsstellung benötigt wurden Aussagen über das Aufkommen von Zeitschriftenartikeln von TU-Angehörigen für die Jahre 2014 und 2015, insbesondere über den Anteil von Artikeln in Open-Access-Zeitschriften. Für die Analyse des Berliner Publikationsaufkommens von neun verschiedenen Einrichtungen wurden die Jahre 2013 bis 2015 analysiert.

Für die Analyse wurde auf Daten aus zehn bzw. sechzehn externen Literatur- und Zitationsdatenbanken zurückgegriffen. Die gewonnenen Daten zu den Dokumententypen **Article** bzw. **Review** wurden normalisiert, aggregiert und auf Dubletten geprüft. Um Artikel aus Open-Access-Zeitschriften zu identifizieren, wurden Daten des *Directory of Open Access Journals (DOAJ)*¹ genutzt. In den verbleibenden Daten von Open-Access-Artikeln wurden im Folgenden diejenigen Artikel identifiziert, für die Angehörige der untersuchten Einrichtungen als Erst- oder Korrespondenzautoren angegeben sind.

Bei der Datenerhebung wurden folgende Datenbanken berücksichtigt: *Web of Science Core Collection*, *SciFinder (CAPlus)*, *PubMed*, *TEMA*, *Inspec*, *IEEE Xplore*, *ProQuest Social Sciences*², *Business Source Complete*, *GeoRef*, *CABAbstracts*, *CINAHL*, *Academic Search Premier*, *Embase*, *LISA*, *Scopus*, *SportDiscus*.

Zur Unterstützung der Evaluation wurde ein Python-Skript entwickelt, dessen Funktionsweise ab S. 3 beschrieben wird. Ab S. 12 wird erläutert, welche Schritte für eine eigene Analyse mithilfe dieses Skripts durchzuführen sind.

Ob mit dem hier beschriebenen Verfahren alle Open-Access-Artikel einer bestimmten Einrichtung identifiziert werden können, bleibt offen. Folgende Faktoren stellen potentielle Fehlerquellen dar:

- Artikel in Open-Access-Zeitschriften, die nicht in einer der geprüften Datenbanken indexiert sind, werden nicht berücksichtigt.

¹Directory of Open Access Journals (DOAJ) s. <http://doaj.org>

²Das Datenbankpaket *Social Sciences* besteht aus folgenden Einzeldatenbanken: Applied Social Sciences Index and Abstracts (ASSIA), British Periodicals, Digital National Security Archive, ebrary e-books, ERIC, Index Islamicus, International Bibliography of the Social Sciences (IBSS), PAIS International, Periodicals Archive Online, Periodicals Index Online, Physical Education Index, PILOTS: Published International Literature On Traumatic Stress, ProQuest Dissertations & Theses Full Text: The Humanities and Social Sciences Collection, ProQuest Dissertations & Theses Global: Social Sciences, Social Services Abstracts, Sociological Abstracts, Worldwide Political Science Abstracts

- Die Artikel werden über externe Datenbanken ermittelt; Voraussetzung für die Identifizierung ist das Erfassen der Affiliation in diesen Datenbanken. Es werden hier zwar Affiliationen für alle Autorinnen und Autoren erfasst – allerdings pro Autor bzw. Autorin meist nur eine Affiliation. Bei Mehrfachaffiliationen wird i. d. R. nur eine Institution in der Datenbank erfasst.
- Open-Access-Zeitschriften werden mithilfe des *DOAJ* identifiziert. Ist eine Zeitschrift nicht im *DOAJ* erfasst, werden Open-Access-Artikel nicht als solche erkannt. Es wird zudem nur berücksichtigt, ob die Zeitschrift zum Zeitpunkt der Analyse im *DOAJ* verzeichnet ist. Es ist also denkbar, dass die Zeitschrift zum Zeitpunkt der Publikation des Artikels noch unter einem Closed-Access-Modell operierte. Es ist ebenso möglich, dass die Zeitschrift zum Zeitpunkt der Publikation des Artikels noch als Open-Access-Zeitschrift im *DOAJ* gelistet wurde, zum Zeitpunkt der Analyse aber aus dem *DOAJ* entfernt wurde.
- Während im *Web of Science (WoS)* die Korrespondenzautorin bzw. der Korrespondenzautor als *reprint author* gesondert ausgewiesen wird, werden in anderen Datenbanken lediglich Affiliationen erfasst. Ein eindeutiger Rückschluss auf die Korrespondenzautorschaft ist für diese Daten nicht möglich. Es werden für diese Datenbanken daher lediglich die Institutionsangaben zu Erstautorinnen bzw. Erstautoren evaluiert. Nicht in allen Disziplinen aber sind Korrespondenz- und Erstautorin identisch. Open-Access-Artikel, für die Angehörige einer Einrichtung Korrespondenz- nicht aber Erstautoren sind, bleiben somit unentdeckt.

2 Funktionsweise des Python-Skripts

2.1 Aufbau und einzelne Funktionen

Das Skript ist in Python Version 2.7 geschrieben (zu Systemanforderungen vgl. Kap. 3.1 auf S. 12) und in drei Teilskripte gegliedert: `graphics.py` erzeugt die graphische Ausgabe der Ergebnisse. `cr.py` führt die Abfragen der *CrossRef*- und *DOAJ*-APIs durch. Das Kernstück des Skripts ist `main.py`, welches in zehn Teile gegliedert ist:

1. Funktionalitäten (de-) aktivieren Verschiedene Funktionalitäten des Skripts können aktiviert bzw. deaktiviert werden, zum Beispiel die Abfrage der *CrossRef*-API. Siehe die folgenden Punkte und Kap. 3.2 für eine Erläuterung der Funktionalitäten.

2. Klassen und Funktionen Es werden die Eigenschaften der untersuchten Institutionen, Datenbanken und Publikationen festgelegt und Funktionen definiert, die später

mehrfach benötigt werden: der Dublettenabgleich und die Normalisierung der verschiedenen Datenformate³.

3. Institutionen ansetzen Es wird festgelegt, welche Institutionen bei der Analyse berücksichtigt werden sollen. Es können beliebig viele Institutionen mit beliebig vielen Namensvarianten definiert werden. Im Skript sind mehrere Berliner Institutionen angelegt. Soll eine Institution nur kurzfristig von der Analyse ausgeschlossen werden, müssen alle Zeilen des jeweiligen Abschnitts ausgenommen werden, indem eine Raute (#) an den Zeilenanfang gesetzt wird. Soll eine Institution dauerhaft von der Analyse ausgeschlossen werden, sind die jeweiligen Zeilen zu löschen. Neue Institutionen können nach dem vorhandenen Muster angelegt werden (siehe auch Kap. 2.2 auf S. 7).

4. Datenbanken ansetzen und Daten einlesen Es wird festgelegt, welche Datenbanken bei der Analyse verwendet werden. Jede Datenbank erhält einen Namen und eine Datenbank-ID. Diese ID-Nummer ist beliebig, muss aber eindeutig sein. Über diese ID kann in den Ausgabedateien nachvollzogen werden, welche Artikeldaten aus welcher Datenbank stammen.

Aus den im Vorfeld erhobenen Daten werden die relevanten Informationen eingelesen, extrahiert und einheitlich strukturiert. Jede Publikation erhält dabei die folgenden Eigenschaften: Autorinnen und Autoren, Titel, DOI, Zeitschrift, ISSN, eISSN, Jahr, Affiliation, Korrespondenz- bzw. Erstautorschaft, E-Mail-Adresse, Fachgebiet laut Datenbank⁴, Angaben zu (Drittmittel-) Förderung⁵, gefundene Namensvariante, Fachgebiet laut *DOAJ*, Verlag, Lizenz, Anmerkungen und die Datenbank-ID der Datenbank, in der die Publikation gefunden wurde. Liegen keine Informationen zu einer Eigenschaft vor, so bleibt sie leer.

5. Dublettenabgleich Die Artikeldaten werden miteinander verglichen und Mehrfacheinträge eliminiert. Der Dublettenabgleich erfolgt in zwei Schritten: Es werden zunächst vorhandene DOIs verglichen. Sind keine DOIs vorhanden, werden Angaben zu Autorin bzw. Autor und Titel abgeglichen, indem die ersten vier Konsonanten der Autorennamen und die ersten 20 Konsonanten des Titels zu einem String zusammengefügt und dann verglichen werden. Die dazugehörigen Statistiken werden ausgegeben.

Nachdem der Dublettenabgleich durchgeführt wurde, werden die generierten Daten in einer Datei `finalList` abgespeichert. Um bei nochmaligen Start des Skripts Zeit zu sparen, kann im Anschluss mithilfe der Variable `doReadIn` die Funktion deaktiviert werden. Bei Skriptstart wird die Datei `finalList` eingelesen; nochmaliges Einlesen der Datenbankdaten und der Dublettenabgleich entfallen.

³Je nach Datenbank ist die ISSN unterschiedlich formatiert. Das Programm kann die folgenden Formatierungen verarbeiten: 1234-5678, 12345678, ISSN 1234-5678 und 1234-5678, 9012-3456. Bei der letzten Variante wird jedoch nur die erste ISSN erfasst.

⁴Zurzeit nur *Web of Science* und *SciFinder*

⁵Zurzeit nur *Web of Science*

6. DOAJ-Daten Die relevanten Informationen aus dem *DOAJ* werden eingelesen. Die ISSN und eISSNs werden mit der Liste der gefundenen Publikationen abgeglichen. Die so gefundenen OA-Artikel in echten OA-Zeitschriften werden in einer Liste abgespeichert und mit Daten zu Fachgebiet, Verlag und Lizenz aus dem *DOAJ* angereichert.

7. CrossRef-Daten Publikationen, die eine DOI haben, deren ISSN bzw. eISSN aber nicht im *DOAJ* gefunden werden konnte, werden über das Teilskript `cr.py` an die *CrossRef*-API⁶ geschickt. Publikationen, für die bei *CrossRef* eine Creative-Commons-Lizenz oder eine ACS-Author-Choice-Lizenz⁷ hinterlegt ist, werden identifiziert und die Lizenz-URL gespeichert. Diese Artikel erhalten außerdem die Notiz *Licence added via CrossRef*. Ziel ist, OA-Artikel in Hybridzeitschriften zu identifizieren.

Bei einigen dieser Artikel fehlen ISSN-Angaben in den Ausgangsdaten. Für über *CrossRef* ermittelte OA-Artikel werden ISSN-Angaben aus den *CrossRef*-Daten ergänzt und durch Abfrage der *DOAJ*-API wird geprüft, ob es sich eine echte Open-Access-Zeitschrift handelt. So können weitere OA-Artikel in echten OA-Zeitschriften identifiziert werden.

Die Ergebnisse der *CrossRef*- und *DOAJ*-Abfragen werden in den Dateien `CRResults` und `CRResultsDOAJ` gespeichert. Im Folgenden kann die Funktion des Skripts mithilfe der Variable `contactCR` im ersten Teil des Skripts deaktiviert werden, um die Daten aus den beiden Dateien zu laden. Die zeitaufwändige Abfrage der Schnittstellen entfällt damit.

8. Erst- bzw. Korrespondenzautorschaft Es wird untersucht, welche Publikationen eine Autorin bzw. einen Autor der gesuchten Institution(en) als Erst- bzw. Korrespondenzautorin haben. Dazu werden die von *Web of Science*, *SciFinder* und *PubMed* bereitgestellten Affiliationsangaben mit den in Abschnitt 3 des Skript definierten Namensvarianten der Institutionen abgeglichen. Zudem wird die Autorenangabe auf die ersten 200 Zeichen gekürzt. So werden Probleme durch sehr lange Autorenlisten bei der Weiterverarbeitung der Daten (z. B. in Tabellenkalkulationsprogrammen) vermieden.

Die restlichen analysierten Datenbanken liefern (in dem gewählten Exportformat) keine bzw. uneinheitlich strukturierte Informationen zur Korrespondenzautorschaft, was eine automatisierte Verarbeitung erschwert bzw. unmöglich macht. Im nächsten Schritt werden daher die restlichen Publikationen in einer Datei namens `docstobechecked.txt` gespeichert. Es handelt sich erfahrungsgemäß um etwa 10 % der OA-Publikationen, für die manuell die Affiliation der Korrespondenzautorin bzw. des Korrespondenzautors ermittelt werden muss.

⁶Die Basis-URL für diese Abfrage lautet <http://api.crossref.org/works/>. Ein API-key ist für diese Abfragen nicht erforderlich. Eine ausführliche API-Dokumentation stellt *CrossRef* online bereit, s. <https://github.com/CrossRef/rest-api-doc/>.

⁷Die American Chemical Society (ACS) vergibt z. T. selbst entwickelte Open-Access-Lizenzen, vgl. <http://pubs.acs.org/page/policy/authorchoice/index.html>.

2 Funktionsweise des Python-Skripts

Die händisch ermittelten Daten können wieder in das Skript eingelesen werden, indem sie in Form einer tab-separierten Textdatei namens `docsChecked.txt` in der Codierung UTF-8 gespeichert und die Funktionalität zum Einlesen dieser Datei mithilfe der Variable `checkToDo` im ersten Teil des Skripts aktiviert wird. In der Datei `docsChecked.txt` muss jede Zeile einer Publikation entsprechen und die drei Spalten den Titel, die DOI und die gefundene Namensvariante (in dieser Reihenfolge) enthalten.

9. APCs abschätzen Die Gesamtzahl von OA-Artikeln, für die die Erst- bzw. Korrespondenzautorschaft bei der/den untersuchten Institution(en) liegt, wird mit 1285 €⁸ bzw. 980 €⁹ multipliziert und die Resultate werden im Terminal ausgegeben. Sollen andere Durchschnittsgebühren zugrunde gelegt werden, können im Skript die Werte 1285 bzw. 980 ersetzt werden. Beide Werte liefern – ausgehend von unterschiedlichen Durchschnittskosten – einen Überschlag über die anfallenden Kosten für Artikelgebühren pro Jahr. Für eine fundierte Angabe zum Mittelbedarf sollten die Einzelartikel im Detail betrachtet werden, denn die Daten des OpenAPC-Projekts zeigen deutliche Schwankungen der durchschnittlichen Artikelgebühr: So hat die Technische Universität Dortmund im Schnitt 915 €, die Universität Heidelberg im Schnitt 1434 € pro Artikel gezahlt.¹⁰

10. Statistik und Analyse Es werden die folgenden Informationen in tab-separierte Textdateien geschrieben:

- `allPubs.txt` = Liste aller gefundenen Artikel
- `allOAPubs.txt` = Liste aller gefundenen OA-Artikel
- `allOAPubsWithCorrAuthor.txt` = Liste aller OA-Artikel mit einer Autorin bzw. einem Autor der analysierten Einrichtung(en) als Erst- bzw. Korrespondenzautor. Es fehlen die Publikationen, die in der Datei `docstobechecked` zu finden sind.
- `docsToBeChecked.txt` = Liste der Publikationen, für die es nicht möglich ist, die Erst- bzw. Korrespondenzautorschaft automatisch zu ermitteln
- `hybridArticles.txt` = Liste aller gefundenen OA-Artikel in Hybridzeitschriften

Alle Dateien sind wie in Abschnitt 4. *Datenbanken ansetzen und Daten einlesen* (vgl. S. 4) beschrieben formatiert. Diese Dateien können für detaillierte Auswertungen in Tabellenkalkulationsprogramme (bspw. *Excel*) importiert werden.

Außerdem werden die folgenden Auswertungen vorgenommen:

⁸Es handelt sich dabei um die durchschnittliche Gebühr für Artikel in OA-Zeitschriften der an OpenAPC-teilnehmenden Institutionen, vgl. <https://github.com/OpenAPC/openapc-de> (Stand 26. April 2016).

⁹Es handelt sich hierbei um die durchschnittliche Gebühr für OA-Artikel, die ausgehend von den in 2014 und 2015 publizierten Artikeln von Angehörigen der TU Berlin und den Angaben zu APC auf den Verlagswebseiten ermittelt wurde.

¹⁰Vgl. Zahlen zu Artikelgebühren der an OpenAPC-teilnehmenden Institutionen, <https://github.com/OpenAPC/openapc-de> (Stand 26. April 2016).

- Für jedes Jahr werden a) die Gesamtzahl der gefundenen Artikel sowie jeweils die Gesamtzahl und der Prozentsatz der b) OA-Artikel in Open-Access-Zeitschriften, c) OA-Artikel in Hybridzeitschriften und d) der OA-Artikel in Open-Access-Zeitschriften mit Erst- bzw. Korrespondenzautorschaft bei einer gesuchten Institution ausgegeben. Für alle Werte wird außerdem die Summe über alle untersuchten Jahre gebildet. Die Resultate werden auch in Form einer Grafik (`Figure_OANumbers.png`) ausgegeben und in der Textdatei `statistics_OA.txt` gespeichert.
- Der Prozentanteil der OA-Artikel in Open-Access-Zeitschriften, gemessen an der Gesamtzahl der Publikationen, und der Anteil der Publikationen mit Erst- bzw. Korrespondenzautorschaft einer relevanten Institution werden im Jahresvergleich in Form einer Grafik (`Figure_OAShare.png`) ausgegeben.
- Der absolute Zuwachs an Publikationen insgesamt und an OA-Artikeln in Open-Access-Zeitschriften im Vergleich zum Vorjahr wird in Form einer Grafik ausgegeben (`Figure_OAIncrease.png`).
- Die in der Datei `allOAPubsWithCorrAuthor.txt` vorhandenen Angaben zu Verlagen werden analysiert; es wird eine Statistik über die Häufigkeitsverteilung der OA-Artikel auf die vorhandenen Verlage erstellt. Die Ergebnisse der Analyse werden in der Datei `statistics_publishers.txt` gespeichert.

Weiterhin werden die folgenden Dateien ausgegeben:

- `CRResults.txt` = Liste der an *CrossRef* gesendeten DOIs und den abgefragten Informationen: ISSN, Lizenz-URL und das Feld, in dem die URL hinterlegt war. In der letzten Spalte wird auch hinterlegt, falls die Schnittstelle eine Fehlermeldung zur angefragten DOI sendete, entweder weil die DOI *CrossRef* unbekannt ist oder weil die DOI einen Fehler enthält.
- Die Datei `DOAJResults.txt` enthält eine Liste aller ISSNs, die an die *DOAJ*-API geschickt wurden und die entsprechende Antwort der Schnittstelle.
- Die Dateien `CRResults`, `CRResultsDOAJ` und `finalList` enthalten Informationen, die vom Skript wieder eingelesen werden können (s. o.) und sind nur für die interne Datenverarbeitung relevant.

2.2 Eine neue Institution ansetzen

Im Skript können beliebig viele Institutionen gleichzeitig verarbeitet werden. Für jede Institution können außerdem beliebig viel Namensvarianten angesetzt werden, nach denen in den Affiliationsangaben der Datenbanken gesucht wird. Neue Institutionen lassen sich wie folgt ansetzen:

In Abschnitt drei des Skripts die Institution nach dem vorhandenen Muster initialisieren:

```
# TU Berlin initialisieren
TUNames = [['Tech', 'Univ', 'Berlin'], ['Berlin', 'TU'],
```

2 Funktionsweise des Python-Skripts

```
['Berlin', 'Inst', 'Technol']]
TU = inst('TU', TUnames)

# HU Berlin initialisieren
HUnames = [['Humboldt', 'Univ', 'Berlin'], ['Berlin', 'HU']]
HU = inst('HU', HUnames)
```

Dabei entspricht der Ausdruck [['Humboldt', 'Univ', 'Berlin'], ['Berlin', 'HU']] der Suche (('Humboldt' AND 'Univ' AND 'Berlin') OR ('Berlin' AND 'HU')).

Die obigen Namensvarianten sind möglichst allgemein formuliert – so deckt die Buchstabenkombination **Univ** sowohl die Begriffe **Universität** und **University**, als auch die nicht unübliche Abkürzung **Univ** ab. Da die meisten Datenbanken nur eine Affiliation pro Autorin bzw. Autor verzeichnen, erleichtert die allgemeine Ansetzung der Namensvarianten die Suche. Eine Ausnahme bildet jedoch *PubMed*: Hier werden mitunter mehrere Institutionen pro Autorin bzw. Autor verzeichnet. Bei Institutionen wie der Humboldt-Universität zu Berlin, die einen relativ distinktiven Namen hat, verursacht dies keine großen Probleme. Bei Institutionen wie der Technischen Universität Berlin kann dies jedoch zu Problemen führen. Gehört eine Autorin bzw. ein Autor zum Beispiel der Technischen Universität Dresden und einer beliebigen anderen Institution in Berlin an, so würden die obigen Namensvarianten im Falle von *PubMed* eine Zugehörigkeit zur TU Berlin verzeichnen. Um dieses Problem zu umgehen, kann im Skript eine zweite Liste mit Namensvarianten angelegt werden. Dies muss ganz am Ende des dritten Abschnitts nach dem folgenden Muster geschehen:

```
TU.nameVar1 = [['Technische Universität Berlin'],
                ['Technische Universitaet Berlin'],
                ['Technische Universität Berlin'],
                ['Berlin Institute of Techn'],
                ['Tech Univ Berlin'],
                ['Berlin Univ Technol'],
                ['Univ Technol Berlin'],
                ['TU Berlin'],
                ['Tech. Univ. Berlin'],
                ['Berlin Inst Technol'],
                ['Technical University Berlin'],
                ['Technische Universitaet de Berlin'],
                ['Technical University of Berlin'],
                ['Berlin University of Technology']]
```

Falls keine zweite Liste mit Namensvarianten angegeben wird, werden in *PubMed* die ursprünglichen Namensvarianten gesucht. In dem Skript ist diese Liste mit Namensvarianten aktuell für die TU Berlin aktiv. Sie kann kurzfristig ausgenommen werden, indem eine Raute (#) an den Anfang jeder Zeile in diesem Abschnitt gesetzt wird. Um diese Option dauerhaft auszuschließen, sind die jeweiligen Zeilen im Skript zu löschen.

2.3 Eine Datenbank von der Analyse ausschließen

Jede Datenbank hat zwei Abschnitte im vierten Teil des Skripts. Der erste Abschnitt umfasst nur eine Zeile und definiert Namen und ID-Nummer der Datenbank. Im zweiten Abschnitt werden die Daten aus der Datenbank eingelesen. *Web of Science* ist ein elementarer Teil des Skripts und kann nicht von der Analyse ausgeschlossen werden. Alle anderen Datenbanken lassen sich wie folgt ausschließen:

1. Ersten Abschnitt finden. Dieser befindet sich im ersten Textblock des vierten Teils des Skripts und entspricht dem folgenden Beispiel:

```
dbLisa = Database('LISA', 15)
```

2. Zweiten Abschnitt finden. Dieser befindet sich am Ende des vierten Teils des Skripts. Der Name der Datenbank wird explizit im zugehörigen Kommentar erwähnt, zum Beispiel:

```
# Read in 'LISA' file and extract relevant information.
dbLisa.content = wosFormat('lisa20xx.txt', dbLisa.idNumber)
print 'Finished reading in LISA'
```

3. Falls die Datenbank permanent von der Analyse ausgeschlossen werden soll, können alle Zeilen dieser beiden Abschnitte gelöscht werden. Soll die Datenbank nur kurzzeitig ausgeschlossen werden, müssen alle Zeilen der beiden Abschnitte auskommentiert werden, indem man eine Raute (#) an die Zeilenanfänge setzt.

2.4 Eine neue Datenbank aufnehmen

Eine Liste der aktuell integrierten Datenbanken ist Kap. 3.3 ab S. 14 zu entnehmen. Eine neue Datenbank kann dann einfach zur Auswertung hinzugefügt werden, wenn die Daten mithilfe von *Citavi* in das *WoS*-Datenformat umgewandelt werden. Diese Methode wurde bisher z. B. für die Datenbanken *TEMA* und *ProQuest* verwendet.

Eine Datenbank lässt sich wie folgt hinzufügen:

1. Abfrage in der Datenbank durchführen und die Vorgehensweise dokumentieren.
2. Resultate exportieren (z. B. im RIS-Format) und in *Citavi* importieren.
3. Resultate aus *Citavi* im *WoS*-Datenformat exportieren und im gleichen Verzeichnis wie sonstige Datenbankergebnisse ablegen (Dateiname z. B. `testa20xx.txt`).
4. Im vierten Teil des Skripts zwei neue Abschnitte für diese Datenbank hinzufügen:
 - a) Namen, Datenbank-ID und Variablenamen für die Datenbank nach dem vorhandenen Muster am Anfang des vierten Teils des Skripts anlegen (im Textblock, der auf den Kommentar # - **Set up databases** folgt). Für eine Datenbank namens *Testa* könnte dies zum Beispiel wie folgt aussehen:

```
dbTesta = Database('Testa', 77)
```

2 Funktionsweise des Python-Skripts

Dabei entspricht `dbTesta` dem Variablennamen, der intern im Skript für die Datenbank verwendet wird, `Testa` ist der Name der Datenbank und `77` ist die Datenbank-ID¹¹. Alle drei Werte sind beliebig, müssen aber im Skript eindeutig sein.

- b) Zeilen im Skript nach dem vorhandenen Muster im vierten Teil des Skripts ergänzen, um die Daten einlesen zu lassen. Für unsere Datenbank *Testa* würde dies wie folgt aussehen:

```
# Read in 'Testa' file and extract relevant information.
dbTesta.content = wosFormat('testa20xx.txt', dbTesta.idNummer)
print 'Finished reading in Testa'
```

Vorsicht: Neue Datenbanken müssen vor den letzten Abschnitt des vierten Skriptteils (*Transform all characters in DOIs to lower case*) hinzugefügt werden.

2.5 Potentielle Fehlerquellen

Die folgenden Faktoren können Fehler in der Analyse mithilfe des hier beschriebenen Skripts verursachen:

- In *WoS* werden alle Titel ins Englische übersetzt. Ist ein Titel in einer anderen Datenbank mit dem deutschen Titel verzeichnet und hat dieser Artikel keine DOI, würde diese Publikation doppelt gezählt.
- Ist der Autorenname sehr kurz bzw. enthält nur wenige Konsonanten (z. B. Lee, Zhi), kann es zu Fehlern beim Dublettenabgleich kommen, falls eine Datenbank nur den ersten Buchstaben des Vornamens, eine andere Datenbank aber den vollständigen Vornamen erfasst.
- Es kann zu Fehlern beim Dublettenabgleich kommen, wenn der Nachname der Autorin bzw. des Autors aus zwei Wörtern (z. B. da Silva) besteht und in den Datenbanken uneinheitlich nachgewiesen ist (da Silva, H. oder Silva, H. da).
- *PubMed* verzeichnet teilweise mehrere Publikationsdaten, die sich in seltenen Fällen in der Jahresangabe unterscheiden. Im Skript wird lediglich eines der beiden Datumsfelder benutzt (DP = Date of Publication). Andere Datenbanken verzeichnen nur ein einziges Publikationsdatum. Artikel, die nur in *PubMed* verzeichnet werden, enthalten so manchmal Jahresangaben, die von der ursprünglichen Suchabfrage nicht abgedeckt sind. Wird zum Beispiel nach Artikeln aus dem Jahr 2015 gesucht, können auch Artikel enthalten und nachgewiesen werden, deren Publikationsdatum ins Jahr 2016 fällt.

¹¹Über diese ID können in den Ausgabedateien später die Artikeldaten identifiziert werden, die aus dieser Datenbank stammen.

2.6 Mögliche Erweiterungen

Bei der Erstellung des Skripts wurde auf ein möglichst effizientes Aufwand-Nutzen-Verhältnis geachtet. Daher wurden einige denkbare Erweiterungen und Verbesserungen des Skripts bislang nicht umgesetzt:

- Dublettencheck verbessern: Eine weitere denkbare Methode für den Dublettencheck besteht darin, aus ISSN, Jahrgang, Ausgabe und Seitenzahlen einen String zu bilden. So erhält man einen weiteren eindeutigen Identifikator, der zum Abgleich herangezogen werden kann.
- Ergänzung der ISSNs: Da einige Datenbanken keine ISSNs bereitstellen, könnte man routinemäßig für diese Fälle die *CrossRef*-API abfragen (solange eine DOI vorliegt) und Angaben zu ISSNs in den internen Artikelindex aufnehmen.
- Ausschlusskriterien: Um die Institutionssuche in den Affiliationsangaben zu verbessern, könnte eine Liste von Wörtern festgelegt werden, die nicht in der Affiliationsangabe vorkommen dürfen. So besteht für die TU Berlin eine gewisse Verwechslungsgefahr mit der Beuth Hochschule für Technik Berlin, insbesondere wenn gleichzeitig der englische Name (Beuth University of Applied Sciences) angegeben wird. Um dieses Problem zu umgehen, könnte man alle Artikel, deren Affiliationsangaben das Wort **Beuth** enthalten, automatisch von der Zuordnung zur TU Berlin ausschließen.
- Verarbeitung anderer Datenformate: Für viele Datenbanken ist der Export der Daten im RIS-Format und deren Konvertierung ins *WoS*-Format mithilfe von *Citavi* vorgesehen, um im Skript die Daten in einem einheitlichen Format verarbeiten zu können. Dabei wird zum einen die Abhängigkeit von der Software *Citavi* (vgl. Kap. 3.1 auf S. 12) und zum anderen ein Datenverlust bei der Konvertierung in Kauf genommen. Alternativ könnte das Skript erweitert werden, um Daten direkt im *RIS*-Format einzulesen, welches von vielen Datenbanken angeboten wird. Das *RIS*-Format hat jedoch den Nachteil, dass die Belegung der Felder von Datenbank zu Datenbank sehr unterschiedlich ist. So werden Affiliationsangaben in den Feldern **AD** = **Author Address** (*ProQuest*), **C1** = **Custom Field 1** (*TEMA*), **N1** = **Notes** (*BSC*), **PB** = **Publisher** (*EBSCO*) oder überhaupt nicht (*IEEE*) verzeichnet. Die Verarbeitung des Datenformats müsste also für jede Datenbank einzeln angepasst werden.
- OA-Chronologie berücksichtigen: Der OA-Status einer Zeitschrift wird mithilfe der Ist-Angaben im *DOAJ* geprüft. Eine höhere Genauigkeit könnte erzielt werden, indem das Jahr berücksichtigt wird, in dem eine Zeitschrift in ein Open-Access-Modell überführt wurde. Daten können aus dem *DOAJ* gezogen werden.

3 Eine Analyse durchführen

3.1 Systemanforderungen

Das Skript wurde für die Python-Version 2.7 entwickelt. Aktuelle Python-Versionen können für verschiedene Betriebssystemplattformen (u. a. Windows, Mac OS X und Linux) frei heruntergeladen werden.¹² Aufgrund der im Skript eingebundenen Pakete sollte mindestens die Python-Version 2.7.11 installiert sein. Je nach lokal vorhandenem System sind ggf. einzelne Pakete nachzuinstallieren.

Bei Python handelt es sich um Open-Source-Software, die kostenfrei heruntergeladen werden kann. Die Python Software Foundation stellt online Hinweise bzgl. Installation und Handhabung sowie eine ausführliche Dokumentation und ein FAQ bereit¹³. Allen, die noch nie ein Skript in einem Terminal oder einer Programmierungsumgebung gestartet haben, wird die Lektüre der Seite „Python for Non-Programmers“¹⁴ empfohlen.

Während das Skript selbst auf verschiedenen Betriebssystemplattformen lauffähig ist, wird empfohlen, die Dateien für das Einlesen auf einem Windows-System vorzubereiten: Es wurden Probleme beim Einlesen von Dateien festgestellt, die auf einem Mac-System¹⁵ vorbereitet wurden.

Das Skript kann Daten in zwei nativen Formaten einlesen, *Web of Science* und *PubMed*. Darüber hinaus können Daten aus den folgenden Datenbanken direkt eingelesen werden: *Web of Science Core Collection*, *Inspec* und *PubMed*. Für alle anderen Daten ist für die exportierten bibliographischen Daten eine Konvertierung in das *WoS*-Format notwendig. Dies kann mithilfe von Citavi erfolgen (getestet mit Citavi5). Bei Citavi handelt es sich um eine lizenzpflichtige Software, für die viele deutsche Forschungseinrichtungen eine Campuslizenz erworben haben. Citavi kann nativ aktuell nur auf Windows-Betriebssystemen installiert werden.¹⁶ Um Citavi auch auf anderen Plattformen nutzen zu können, kann Windows in einer virtuellen Maschine installiert werden.¹⁷

Alle Input-Dateien sollten im gleichen Ordner liegen wie das zu startende Python-Skript. In dem gleichen Ordner werden Output-Dateien abgespeichert. Daher müssen für diesen Arbeitsordner sowohl Lese- als auch Schreibrechte vorhanden sein.

¹²Download unter <https://www.python.org/downloads/>. Auf aktuellen Mac- und Linux-Systemen ist Python standardmäßig vorhanden; ggf. muss aber die Version upgedatet werden.

¹³Python-Dokumentation s. <https://docs.python.org>, Hinweise für Anfänger s. insbes. <https://www.python.org/about/gettingstarted/>

¹⁴<https://wiki.python.org/moin/BeginnersGuide/NonProgrammers>

¹⁵Getestet wurde ein Zusammenführen von Dateien (Export aus verschiedenen Datenbanken) unter Mac OS X 10.9.5 und dem Texteditor TextWrangler 5.5.1. Eine Vorbereitung der Dateien in einer Linux-Umgebung wurde bisher nicht getestet.

¹⁶Download unter <https://www.citavi.com/de/download.html>

¹⁷Hinweise, wie Citavi auf dem Mac genutzt werden kann, hält das Citavi-Handbuch bereit: <https://www.citavi.com/sub/manual5/de/index.html>

3.2 Handhabung des Skripts

Um den Open-Access-Anteil bei Zeitschriftenartikeln mithilfe des Skripts zu analysieren, müssen die im Folgenden beschriebenen Schritte durchgeführt werden. Dabei können Publikationen einer oder mehrerer Institutionen gleichzeitig analysiert werden. Eine gleichzeitige Analyse mehrerer Jahre ist ebenfalls möglich.

1. Abfrage in den gewünschten Datenbanken (Beschreibung vgl. S. 14 ff)
2. *DOAJ*-Daten herunterladen und vorbereiten (Beschreibung vgl. S. 21)
3. Im dritten Abschnitt des Skripts Namensvarianten für zu untersuchende Institutionen eintragen (Beschreibung vgl. S. 4).
4. Dateien mit Ergebnissen aus Datenbankrecherchen und *DOAJ*-Daten im gleichen Verzeichnis ablegen wie Skript.
5. Im ersten Abschnitt des Skripts gewünschte Funktionalitäten auswählen:
 - Datenauswertung (Statistik, Diagramme)
 - `doAnalysis = True`: ruft den letzten Abschnitt des Skripts für erste Datenauswertungen auf
 - `doAnalysis = False`: Funktionalität deaktivieren
 - Einlesen Artikeldaten
 - `doReadIn = True`: beim ersten Starten des Skripts wählen, liest Daten aus Datenbankrecherchen etc. ein
 - `doReadIn = False`: Funktionalität deaktivieren, um Daten aus Zwischenspeicher einzulesen (verkürzt Skriptlaufzeit)
 - API-Abfragen
 - `contactCR = 0`: Funktionalität deaktivieren
 - `contactCR = 1`: beim ersten Start des Skripts wählen, kontaktiert *Cross-Ref*- und *DOAJ*-API und fragt Informationen zu Lizenzen und ISSNs ab
 - `contactCR = 2`: für wiederholtes Starten des Skripts wählen, liest Ergebnisse aus dem Zwischenspeicher ein (verkürzt Skriptlaufzeit)
 - Manuelle Prüfung Korrespondenzautorschaft
 - `checkToDo = 0`: Funktionalität deaktivieren
 - `checkToDo = 1`: beim ersten Starten des Skripts wählen, erstellt Datei `docstobeChecked.txt`
 - `checkToDo = 2`: für wiederholtes Starten des Skripts wählen, liest Ergebnisse aus der manuell erstellten Datei `docsChecked.txt`
6. Skript in Python-Umgebung¹⁸ starten
(`doAnalysis = True, doReadIn = True, contactCR = 1, checkToDo = 1`)

¹⁸Das Skript kann in einer Python-Programmierungsumgebung (IDE) (s. u. a. <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>) oder vom Terminal des Betriebssystems aus gestartet werden (s. u. a. https://en.wikibooks.org/wiki/Python_Programming/Creating_Python_Programs).

7. Die Ausgabedatei `docstobechecked.txt` nachbereiten: Für alle Einträge in der Datei prüfen, ob für den Artikel die Korrespondenz- bzw. Erstautorschaft bei einer Autorin bzw. einem Autor der untersuchten Einrichtung(en) liegt. Ergebnisse in neuer Datei mit Dateinamen `docsChecked.txt` speichern und im gleichen Verzeichnis wie Skript ablegen (vgl. Beschreibung S. 5). In der Datei `docsChecked.txt` muss jede Zeile einer Publikation entsprechen; die drei Spalten müssen den Titel, die DOI und die gefundene Namensvariante (in dieser Reihenfolge) enthalten.
8. Skript in Python-Umgebung starten
(`doAnalysis = True, doReadIn = False, contactCR = 2, checkToDo = 2`)
9. Ausgabedateien auswerten (Beschreibung der Ausgabedateien vgl. Abschnitt 10. *Statistik und Analyse* auf S. 6 bzw. Übersicht der Dateien vgl. Anhang ab S. 22)

3.3 Abfrage der Datenbanken und Aufbereitung der Daten

Im Folgenden wird jeweils ein Weg beschrieben, wie die Abfragen in den einzelnen Datenbanken durchgeführt werden können¹⁹ und wie die Daten des *DOAJ* aufbereitet werden müssen.

Academic Search Premier

- Suchabfrage (einfache Suche):
 - Beispiel: `AF (tu berlin* OR tech* univ* berlin* OR technisch* universität berlin* OR berlin* inst* technol*)`
- Suchergebnisse filtern nach
 - Jahr
 - Dokumententyp: `publication type = Academic Journals`
- Daten exportieren:
 - `Share`
 - `Export results: E-mail a link to download exported results`
 - Format auswählen: `RIS format`
 - Formular ausfüllen (E-Mail-Adresse)
 - `Send`
 - Link in E-Mail öffnen und Datei abspeichern
- RIS umformatieren in *WoS*-Format: RIS-Datei in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateiname `ebsco20xx.txt`

¹⁹Beispielabfragen sind für ein Jahr formuliert, aber das Skript kann mehrere Jahrgänge gleichzeitig verarbeiten. Die Beispiele sind (mit Ausnahme von *TEMA*) dokumentiert für eine englischsprachige Oberfläche der jeweiligen Datenbanken.

Business Source Complete

- Suchabfrage (einfache Suche) via EBSCOhost (Anführungszeichen nutzen für exakte Suche!):
 - Beispiel: AD "tech* univ* berlin" OR AD "tu berlin" OR AD "berlin inst* tech"
- Suchergebnisse filtern nach
 - Jahr
 - Dokumententyp: publication type = Academic Journals
- Daten exportieren:
 - Share
 - Add to Folder: Results
 - Folder
 - Select all
 - Export
 - RIS format
 - Save
- RIS umformatieren in *WoS*-Format: RIS-Datei in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateiname bsc20xx.txt

CAB Abstracts

- Suchabfrage (erweiterte Suche) via OvidSP:
 - Beispiel: (tech* univ* berlin or TU* Berlin).in. OR (tu-berlin de).ma.
- Suchergebnisse filtern nach
 - Jahr
 - Dokumententyp: Publication Type = Journal Article
- Daten exportieren:
 - Range: 1-200
 - Export
 - Export To: RIS
 - Select Fields to Display: Complete Reference
 - Export Citation(s)
- Datenexport wiederholen für alle vorhandenen Datensätze (*OvidSP* erlaubt Herunterladen von max. 200 Datensätzen pro Durchgang – ggf. abhängig von Lizenz)
- RIS umformatieren in *WoS*-Format: RIS-Dateien in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateiname cab20xx.txt

Cumulative Index to Nursing & Allied Health Literature (CINAHL)

- Suchabfrage (erweiterte Suche) via EBSCOhost:
 - Beispiel: `AF tech* univ* berlin OR AF TU Berlin OR AF berlin inst* technol*`
- Suchergebnisse filtern nach
 - Jahr
 - Dokumententyp: `Publication Type = Academic Journals`
- Daten exportieren:
 - `Share`
 - `Page Options`: max. Treffermenge auswählen
 - `Add to Folder`: `Results 1-50`
 - `Folder`
 - `Select all`
 - `Export`
 - `Direct Export in RIS format`
 - `Save`
- RIS umformatieren in *WoS*-Format: RIS-Dateien in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateiname `cinahl20xx.txt`

Embase

- Suchabfrage (Expertensuche) via OvidSP:
 - Beispiel: `keyword(tech* univ* berlin.ad. OR tech* univ* berlin.in. OR tech* univ* of berlin.ad. OR tech* univ* of berlin.in. OR TU Berlin.ad. OR TU Berlin.in. OR Berlin Inst* Technol*.ad. OR Berlin Inst* Technol*.in. OR Berlin Inst* of Technol*.ad. OR Berlin Inst* of Technol*.in.)`
- Suchergebnisse filtern nach
 - Jahr
 - Dokumententyp: `Publication Type = Article, Review` (hierzu Filter über `Additional Limits` bzw. `Zusätzliche Eingrenzungen` auswählen)
- Daten exportieren:
 - `Range`: `1-200`
 - `Export`
 - `Export to`: `RIS`
 - `Select Fields to Display`: `Complete Reference`
 - `Export Citation(s)`
- Datenexport wiederholen für alle vorhandenen Datensätze (*OvidSP* erlaubt Herunterladen von max. 200 Datensätzen pro Durchgang – ggf. abhängig von Lizenz)
- RIS umformatieren in *WoS*-Format: RIS-Datei in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateiname `embase20xx.txt`

GeoRef

- Suchabfrage (einfache Suche) via EBSCOhost:
 - Beispiel: `'tech* univ* berlin' (All text)`
- Suchergebnisse filtern nach
 - Jahr
 - Dokumententyp: `source type = Academic Journals`
- Daten exportieren:
 - Share
 - Add to Folder: Results
 - Folder
 - Select all
 - Export
 - RIS format
 - Save
- RIS umformatieren in *WoS*-Format: RIS-Datei in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateiname `gf20xx.txt`

IEEE Xplore

- Suchabfrage (Anführungszeichen nutzen für exakte Suche!):
 - Beispiel: `((("Author Affiliations": tech* univ* berlin) OR "Author Affiliations": "TU Berlin") OR "Author Affiliations": "Berlin Inst* of technol*")`
- Suchergebnisse filtern nach
 - Jahr
 - Dokumententyp: `document type = Journals & Magazines`
- Daten exportieren:
 - Display all records on one page
 - Select All on Page
 - Download Citations
 - Output Format= RIS
 - Download
- RIS umformatieren in *WoS*-Format: RIS-Datei in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateiname `ieee20xx.txt`

InSpec

- Suchabfrage: Namensvarianten für die eigene Institution (Anführungszeichen nutzen für exakte Suche!) und Jahresangabe
 - Beispiel: `"tech* uni* berlin" OR "TU Berlin" (Address) and 2014 (year)`
- Suchergebnisse filtern nach
 - Dokumententyp: `document type = 'journal paper'`, dabei `'conference paper'` explizit ausschließen!
- Daten exportieren:
 - `Save to Other Formats`
 - `Number of Records = Records 1 to 500`
 - `Record Content = Full Record`
 - `File Format = Tab-delimited (Win, UTF-8)`
- Datenexport wiederholen für alle vorhandenen Datensätze (*InSpec* erlaubt Herunterladen von max. 500 Einträgen pro Durchgang)
- alle Ergebnisse in *eine* Datei kopieren (Spaltennamen der Einzeldateien nur einmal übernehmen!)
- Speichern: Dateiname `inspec20xx.txt`

Library and Information Science Abstracts (LISA)

- Suchabfrage (erweiterte Suche) via ProQuest:
 - Beispiel: `(ea(tu-berlin.de) OR af(Tech* berlin)) AND rtype.exact ("Article" OR "Journal Article") AND pd(2014)`
- Daten exportieren:
 - `Items per page: 100` (am Seitenende)
 - `Select 1-100` (wiederholen für alle Treffer)
 - `Save`
 - `Export/Save: RIS` (works with EndNote, Citavi, etc.)
 - `Continue`
 - `Speichern`
- RIS umformatieren in *WoS*-Format: RIS-Datei in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateiname `lisa20xx.txt`

ProQuest Social Sciences

- Suchabfrage (erweiterte Suche):
 - Beispiel: `'au(tech* univ* berlin) OR au(TU Berlin)'`
- Suchergebnisse filtern nach
 - `Jahr`
 - Dokumententyp: `Source type = Scholarly Journals`
- Daten exportieren:
 - `Items per page: 100` (am Seitenende)

3 Eine Analyse durchführen

- Select 1-100 (wiederholen für alle Treffer)
- Save
- Export/Save: RIS (works with EndNote, Citavi, etc.)
- Continue
- Speichern
- RIS umformatieren in *WoS*-Format: RIS-Datei in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateinamen pq20xx.txt

PubMed

- Suchabfrage: Namensvariante für die eigene Institution (Anführungszeichen nutzen für exakte Suche!) und Jahr
 - Beispiel: (((("Technische Universität Berlin" [Affiliation]) OR "TU Berlin" [Affiliation]) OR "Tech Univ Berlin" [Affiliation]) OR "Berlin Institute of Technology" [Affiliation]) OR "Berlin Inst* Technol*" [Affiliation]) AND ("2014/01/01" [Date - Publication] : "2014/12/31" [Date - Publication])
- Daten exportieren:
 - Send to: File
 - Format: MEDLINE
- Speichern: Dateiname pubmed20xx.txt

SciFinder (CAplus)

- Suchabfrage:
 - Beispiel: 'tech univ berlin' (company)
- Suchergebnisse filtern nach
 - Jahr: publication year = 20xx
 - Dokumententyp: document type = journal or review
 - Datenbank: database = CAPLUS
- Dubletten entfernen: 'Tools: remove duplicates'
- Daten exportieren:
 - Export
 - Range: 1-100
 - For: Citation Manager - Quoted Format (*.txt)
 - Details: Quote Character: "
 - Delimiter: , (Komma)
- Datenexport wiederholen für alle vorhandenen Datensätze (*SciFinder* erlaubt Herunterladen von max. 100 Einträgen pro Durchgang)
- alle Ergebnisse in *eine* Datei kopieren (Spaltennamen der Einzeldateien nur einmal übernehmen!)
- Speichern: Dateiname sf20xx.txt

Scopus

- Suchabfrage:
 - Beispiel: (AF-ID("Technische Universität Berlin"60011604) OR (AFFIL(techn* AND univ* AND berlin)) OR (AFFIL(inst* AND technol* AND berlin)) AND DOCTYPE (ar OR re) AND PUBYEAR = 2014)
- Daten exportieren:
 - Select all
 - Export
 - RIS Format
 - Choose the information to export: All available information
 - Export
- Datenexport wiederholen für alle vorhandenen Datensätze (*Scopus* erlaubt Herunterladen von max. 2000 Datensätzen pro Durchgang)
- RIS umformatieren in *WoS*-Format: RIS-Datei in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateiname `scopus20xx.txt`

Sport Discus

- Suchabfrage (Search modes – Boolean/Phrase) via EBSCOhost:
 - Beispiel: AF(Techn* univ* berlin OR TU Berlin)
- Suchergebnisse filtern nach
 - Jahr: 20140101-20141231
 - Dokumententyp: Document Type: Article
- Daten exportieren:
 - Share
 - Add to Folder: Results
 - Folder
 - Select all
 - Export
 - RIS format
 - Save
- RIS umformatieren in *WoS*-Format: RIS-Datei in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateiname `sd20xx.txt`

TEMA

- Suchabfrage: Namensvariante für die eigene Institution (Anführungszeichen nutzen für exakte Suche!) und Jahr
 - Beispiel: "TU Berlin" (Institution), 2014 (Jahr)
- Suchergebnisse filtern nach
 - Dokumententyp: document type = Zeitschrift

3 Eine Analyse durchführen

- Daten exportieren:
 - Titel pro Seite
 - Alle auswählen
 - Auswahl Anzeigen
 - Alle auswählen
 - Auswahl als RIS speichern
- Datenexport wiederholen für alle vorhandenen Datensätze (*TEMA* erlaubt Herunterladen von max. 100 Einträgen pro Durchgang)
- RIS umformatieren in *WoS*-Format: RIS-Dateien in *Citavi* importieren, alle Einträge exportieren in *WoS*-Format
- Speichern: Dateiname `tema20xx.txt`

Web of Science Core Collection

- Suchabfrage: Namensvarianten für die eigene Institution und Jahresangabe
 - Beispiel: `technical university of berlin (organization enhanced) + 2014 (year)`
- Suchergebnisse filtern nach
 - Dokumententyp: `article or review (document type)`
- Daten exportieren:
 - `Save to Other Formats`
 - `Number of Records = Records 1 to 500`
 - `Record Content = Full Record`
 - `File Format = Tab-delimited (Win, UTF-8)`
- Datenexport wiederholen für alle vorhandenen Datensätze (*WoS* erlaubt Herunterladen von max. 500 Einträgen pro Durchgang)
- Ergebnisse in *eine* Datei kopieren (Spaltennamen der Einzeldateien nur einmal übernehmen!)
- Speichern: Dateiname `wos20xx.txt`

DOAJ

Das *DOAJ* stellt seine Daten zur Weiternutzung zur Verfügung, so kann u. a. eine CSV-Datei heruntergeladen werden: <https://doaj.org/faq#metadata>.

Damit die CSV-Datei vom Skript verarbeitet werden kann, muss sie unter dem Namen `doaj.txt` als tab-separierte Textdatei mit der Kodierung UTF-8 im selben Verzeichnis wie das Skript und die anderen einzulesenden Dateien abgespeichert werden.

Anhang

Tabelle 1: Übersicht der einzelnen Skriptdateien

| Skriptdatei | Erläuterung |
|-------------|---|
| cr.py | Abfragen der APIs von CrossRef und DOAJ |
| graphics.py | Visualisierung Ergebnisse |
| main.py | Hauptskript |

Tabelle 2: Übersicht der vom Skript einzulesenden Dateien

| Eingabedatei | Erläuterung |
|-----------------|--|
| docsChecked.txt | Publikationen, für die Affiliation der Erst- bzw. Korrespondenzautorschaft manuell geprüft wurde |
| doaj.txt | DOAJ-Metadaten |
| bsc20xx.txt | Artikeldaten Business Source Complete |
| cab20xx.txt | Artikeldaten CAB Abstracts |
| cinahl20xx.txt | Artikeldaten CINHALL |
| ebsco20xx.txt | Artikeldaten Academic Search Premier (EBSCO) |
| embase20xx.txt | Artikeldaten Embase |
| gf20xx.txt | Artikeldaten GeoRef |
| ieee20xx.txt | Artikeldaten IEEE Xplore |
| inspec20xx.txt | Artikeldaten InSpec |
| lisa20xx.txt | Artikeldaten LISA |
| pq20xx.txt | Artikeldaten ProQuest Social Sciences |
| pubmed20xx.txt | Artikeldaten PubMed |
| scopus20xx.txt | Artikeldaten Scopus |
| sd20xx.txt | Artikeldaten Sport Discus |
| sf20xx.txt | Artikeldaten SciFinder |
| tema20xx.txt | Artikeldaten TEMA |
| wos20xx.txt | Artikeldaten Web of Science |

Tabelle 3: Übersicht der wichtigsten im Skript verwendeten Variablen

| Variable im Skript | Erläuterung |
|--------------------|---|
| checkToDo | Skriptfunktionalität (de-)aktivieren: manuelle Prüfung Erst-/Korrespondenzautorschaft |
| contactCR | Skriptfunktionalität (de-)aktivieren: Abfragen der APIs von CrossRef und DOAJ |
| doAnalysis | Skriptfunktionalität (de-)aktivieren: Datenauswertung (Statistik, Diagramme) |
| doReadIn | Skriptfunktionalität (de-)aktivieren: Einlesen Artikeldaten aus Datenbankrecherchen |
| finalList | Liste aller gefundenen Artikel |
| hybrid | Liste der OA-Artikel in Hybridzeitschriften |
| oaList | Liste der OA-Artikel in OA-Zeitschriften |
| oaWoS | Liste der OA-Artikel, für die Korrespondenzautorschaft bei einer untersuchten Institution liegt |

Tabelle 4: Übersicht der vom Skript ausgegebenen Dateien

| Ausgabedatei | Erläuterung |
|-----------------------------|--|
| allOAPubs.txt | Liste aller gefundenen OA-Artikel |
| allOAPubsWithCorrAuthor.txt | Liste aller OA-Artikel mit Autorin bzw. Autor der analysierten Einrichtung(en) als Erst- bzw. Korrespondenzautor |
| allPubs.txt | Liste aller gefundenen Artikel |
| CRResults | interne Arbeitsdatei der an CrossRef gesendeten Daten und den abgefragten Informationen |
| CRResults.txt | Liste der an CrossRef gesendeten DOIs und den abgefragten Informationen |
| CRResultsDOAJ | interne Arbeitsdatei der an DOAJ gesendeten Daten und den abgefragten Informationen |
| DOAJResults.txt | Liste aller ISSNs, die an die DOAJ-API geschickt wurden und den entsprechenden OA-Status der Zeitschrift |
| docstobechecked.txt | Liste der Publikationen, für die Affiliation der Erst- bzw. Korrespondenzautorschaft manuell zu prüfen ist |
| finalList | interne Arbeitsdatei aller Artikeldaten |
| hybridArticles.txt | Liste der OA-Artikel in Hybridzeitschriften |
| statistics_OA.txt | Statistik der analysierten Artikel |
| statistics_publishers.txt | Häufigkeitsverteilung der OA-Artikel auf Verlage |
| Figure_OAIncrease.png | Diagramm: Zuwachs an (OA-)Artikeln |
| Figure_OANumbers.png | Diagramm: Statistik über analysierte (OA-)Artikel |
| Figure_OAShare.png | Diagramm: Entwicklung des OA-Anteils |