

Analyse von EAD-XML Daten des Kalliope Verbundkatalogs

ZBIW Data Librarian 2022/2023, Modul 6, Projektarbeit, Katja Jana



<https://kalliope-verbund.info/>: Nachweise von Archivalien aus Nachlassverzeichnissen und Nachlässen verschiedener kulturellen Institutionen und Gedächtnisorganisationen.



Merkliste

DE

EN

Über Kalliope

Teilnahme

Community

Standards

Support

Kalliope Verbundkatalog

Suchanfrage eingeben



Teilnahme

Ob Bibliothek, Archiv, Museum oder verwandte Einrichtung, ... Nachlässe online erschließen.

[lesen Sie hier mehr]

Tour d'Horizon

Erfahren Sie mehr über die Möglichkeiten dieses Katalogs, seine Suchfunktionen, ...

[lesen Sie hier mehr]

Erschließungsregeln

Regeln und Standards für die Erschließung von Nachlässen und Autographensammlungen ...

[lesen Sie hier mehr]

<https://kalliope-verbund.info/>: Nachweise von Archivalien aus Nachlassverzeichnissen und Nachlässen verschiedener kulturellen Institutionen und Gedächtnisorganisationen.

- Retrokonversion Zentralkartei der Autographen zwischen 2001 und 2006: Bestände von nahezu 540 Institutionen
- Erschließungsplattform: momentan weisen circa 1000 Gedächtnisinstitutionen ihre Nachlässe und Autographen in Kalliope nach.
- Normdaten: Personen- und Organisationsdaten sind, wenn vorhanden, mit GND-Normdaten verknüpft.

Kalliope EAD-XML-Format

- aktueller Gesamtabzug der Kalliope-Datenbank: 11,0 GiB, 29.130 Dateien
(von 1,6 KiB bis 178,2 MiB)
- EAD-XML-Format: EAD-XML Standard legt die Kodierung von archivarischen Findbüchern fest
- Dokumentation:
<https://kalliope-verbund.info/files/480b012f45a0f002ec3095b6819ef8aea33078de.pdf>
<https://www.loc.gov/ead/>

Exemplarische Betrachtung von DE-1a_4273.xml mit Python library BeautifulSoup

```
from bs4 import BeautifulSoup
import pandas as pd
import re

with open("ead_DE-1a_4273.xml", 'r') as testead:
    soup = BeautifulSoup(testead, "xml")
len(soup.find_all('ead'))
```

1

Exemplarische Betrachtung von DE-1a_4273.xml mit Python library BeautifulSoup

```
In [45]: # repositories = soup.find_all('repository')
# institution = soup.find_all('corpname')
GND_authors = soup.find_all(role="Verfasser", authfilename=True)
```

```
In [46]: GND_authors
```

```
<persname authfilename="118526200" encodinganalog="DE-611-PS-15457" normal="Döblin, Alfred" role="Verfasser" source="GND">Döblin, Alfred (1878-1957)</persname>,
<persname authfilename="118526200" encodinganalog="DE-611-PS-15457" normal="Döblin, Alfred" role="Verfasser" source="GND">Döblin, Alfred (1878-1957)</persname>,
<persname authfilename="118526200" encodinganalog="DE-611-PS-15457" normal="Döblin, Alfred" role="Verfasser" source="GND">Döblin, Alfred (1878-1957)</persname>,
<persname authfilename="118526200" encodinganalog="DE-611-PS-15457" normal="Döblin, Alfred" role="Verfasser" source="GND">Döblin, Alfred (1878-1957)</persname>,
<persname authfilename="118526200" encodinganalog="DE-611-PS-15457" normal="Döblin, Alfred" role="Verfasser" source="GND">Döblin, Alfred (1878-1957)</persname>,
<persname authfilename="116152745" encodinganalog="DE-611-PS-15467" normal="Döblin, Erna" role="Verfasser" source="GND">Döblin, Erna (1888-1957)</persname>,
<persname authfilename="116152745" encodinganalog="DE-611-PS-15467" normal="Döblin, Erna" role="Verfasser" source="GND">Döblin, Erna (1888-1957)</persname>,
<persname authfilename="118969846" encodinganalog="DE-611-PS-36047" normal="Eeden, Frederik van" role="Verfasser" source="GND">Eeden, Frederik van (1860-1932)</persname>,
<persname authfilename="118688227" encodinganalog="DE-611-PS-38758" normal="Ehrenstein, Albert" role="Verfasser" source="GND">Ehrenstein, Albert (1886-1950)</persname>,
<persname authfilename="118688227" encodinganalog="DE-611-PS-38758" normal="Ehrenstein, Albert" role="Verfasser" source="GND">Ehrenstein, Albert (1886-1950)</persname>,
```

```
In [48]: len(soup.find_all(role="Verfasser", authfilename=True))
```

```
Out[48]: 3774
```

Exemplarische Betrachtung von DE-1a_4273.xml mit Python library BeautifulSoup

```
In [16]: tag3 = soup.corpname  
type(tag3)
```

```
Out[16]: bs4.element.Tag
```

```
In [24]: tag3.name
```

```
Out[24]: 'corpname'
```

```
In [45]: tag3.attrs
```

```
Out[45]: {'role': 'Aufbewahrungsort',  
          'encodinganalog': 'DE-611-KS-32',  
          'source': 'ISIL',  
          'authfilename': 'DE-1a'}
```

```
In [46]: tag3.string
```

```
Out[46]: 'Staatsbibliothek zu Berlin. Handschriftenabteilung'
```

Exemplarische Betrachtung von DE-1a_4273.xml mit Python library BeautifulSoup

```
In [40]: GND_Pers = soup.find_all(authfilenumber=re.compile(".*"), role = "Verfasser")
extract_attr = [x["authfilenumber"] for x in GND_Pers]
print(extract_attr)
```

Exemplarische Betrachtung von DE-1a_4273.xml mit Python library BeautifulSoup

```
In [41]: df = pd.DataFrame(extract_attr, columns = ["GND-Nummer"])
```

```
In [42]: df
```

```
Out[42]:
```

GND-Nummer	
0	118846124
1	118846124
2	118846124
3	118846124
4	118846124
...	...
3769	118636278
3770	118636278
3771	118636278
3772	118636278
3773	118637479

3774 rows × 1 columns

Einlesen des Gesamtabzugs von Kalliope mit Python library BeautifulSoup

```
In [2]: import os
from bs4 import BeautifulSoup
import pandas as pd
import re

In [9]: path = '/home_ext/PK/b-kj102/Dokumente/ZBIW_datalib_Projekt_Kalliope/testset_ead'
list_ead = []
for filename in os.listdir(path):
    if filename.endswith(".xml"):
        fullpath = os.path.join(path, filename)
        #Liste einfügen

        # Get Page, Make Soup
        soup = BeautifulSoup(open(fullpath), 'xml')
        # soup = BeautifulSoup((list_ead.append(soup)), 'xml')

        list_ead.append(soup)
        #list_ead.append(soup)

        # .append einfügen damit alle Daten ausgelesen werden
        print(soup.title)
        print('=====corpname')
        print(len(soup.find_all('corpname')))
        print('=====ead')
        print(len(soup.find_all('ead')))
        print('=====')

# soup = BeautifulSoup(list_ead, 'xml')
#soup = BeautifulSoup((list_ead.append(soup)), 'xml')
```

```
None
=====
corpname
2
=====
ead
1
=====
None
=====
corpname
1
=====
ead
```

Data Toolkit Catmandu

Catmandu

a data toolkit

This handbook is contains the aggregated content of [Catmandu documentation wiki](#). Feel free to improve the documentation there!

Table of Contents

- [1 Introduction](#) ↗
- [2 Installation](#) ↗
 - [2.0.1 Debian](#)
 - [2.0.2 Ubuntu Server 12.04.4 LTS](#)
 - [2.0.3 CentOS 6.4](#)
 - [2.0.4 CentOS 7](#)
 - [2.0.5 openSUSE](#)

Data Toolkit Catmandu – Skript zum Parsen der Attribute in >persname<

- shellscript catmandu loop:

```
for file in $(find . -type f | grep xml); do
    catmandu convert XML --path '//persname' to CSV \
    --fix my.fix \
    --fields "GNDnr,Name,ID-Bestand,Name_normiert,Rolle,Source" \
    --header 0 < "$file" >> loop_allfields.csv
```

Data Toolkit Catmandu – Skript zum Parsen der Attribute in >persname<

- my.fix:

```
set_field(GNDnr, "NaN");
set_field(ID-Bestand, "NaN");
set_field(Rolle, "NaN");
set_field(Name_normiert, "NaN");
set_field(Sourc, "NaN");

cp(persname.authfilenumber,GNDnr)
cp(persname.content,Name)
cp(persname.encodinganalog,ID-Bestand)
cp(persname.normal,Name_normiert)
cp(persname.role,Rolle)
cp(persname.source,Source)
```

Kommandozeile: Mittels uniq und grep die Daten bereinigen

```
$ sort loop_allfields.csv | uniq > uniqloop_allfields.csv
```

```
$ grep -E "^[0-9],{8,11}" uniqloop_allfields.csv > uniqloop_allfields_cleangnd.csv
```

Kommandozeile: Mittels uniq und grep die Daten bereinigen

- In aDIS/BMS, auf dem die Kalliope Datenbank beruht, sind manche Normdaten fehlerhaft zugewiesen werden, z.B. stehen an manchen Stellen Orcid- statt GND-Ids,
- Bereinigte Daten: Anzahl der Einträge von 8538251 auf 1743865 (sort | uniq) auf 817629 (grep). Dabei wurden auch fehlerhaft gelieferte Daten ausgefiltert, wo GND-Ids als Links eingetragen waren. Bei Gelegenheit müsste hier nochmal feiner gefiltert werden. Daher erklärt sich die hohe Anzahl von circa 200.000 entfernten Einträgen.

>persname< in Pandas DataFrame (vor grep)

```
In [3]: dfuniq = pd.read_csv("uniqueloop_allfields.csv", low_memory=False)
dfuniq
```

Out[3]:

	persname.authfilenumber	persname.content	persname.encodinganalog	persname.normal	persname.role	persname.source
0	0000-0001-5535-5894	Mastrocinque, Attilio (1952-)	NaN	Mastrocinque, Attilio	Verfasser	GND
1	0000-0001-6364-7723	Baxmann, Inge	NaN	Baxmann, Inge	Korrespondenzpartner	GND
2	0000-0001-6364-7723	Baxmann, Inge	NaN	Baxmann, Inge	NaN	GND
3	0000-0001-6364-7723	Hassauer, Friederike (1951-)	NaN	Hassauer, Friederike	Verfasser	GND
4	0000-0001-6364-7723	Hegenbarth-Rösgen, Annelie	NaN	Hegenbarth-Rösgen, Annelie	Verfasser	GND
...
1743859	Z0117759	Schreber, Daniel Gottlob Moritz (Erwähnte Person)	NaN	Schreber, Daniel Gottlob Moritz (Erwähnte Person)	Erwähnt	DE-MUS-853418- Objektdatenbank
1743860	Z0117759	Unbekannt	NaN	Unbekannt	Verfasser	DE-MUS-853418- Objektdatenbank
1743861	Z0118268	Siemering, Leopold Rudolf	NaN	Siemering, Leopold Rudolf	Verfasser	DE-MUS-853418- Objektdatenbank
1743862	Z0118274	Siemering, Leopold Rudolf	NaN	Siemering, Leopold Rudolf	Verfasser	DE-MUS-853418- Objektdatenbank
1743863	Z0118298	Grod, ...	NaN	Grod, ...	Genannte Person	DE-MUS-853418- Objektdatenbank

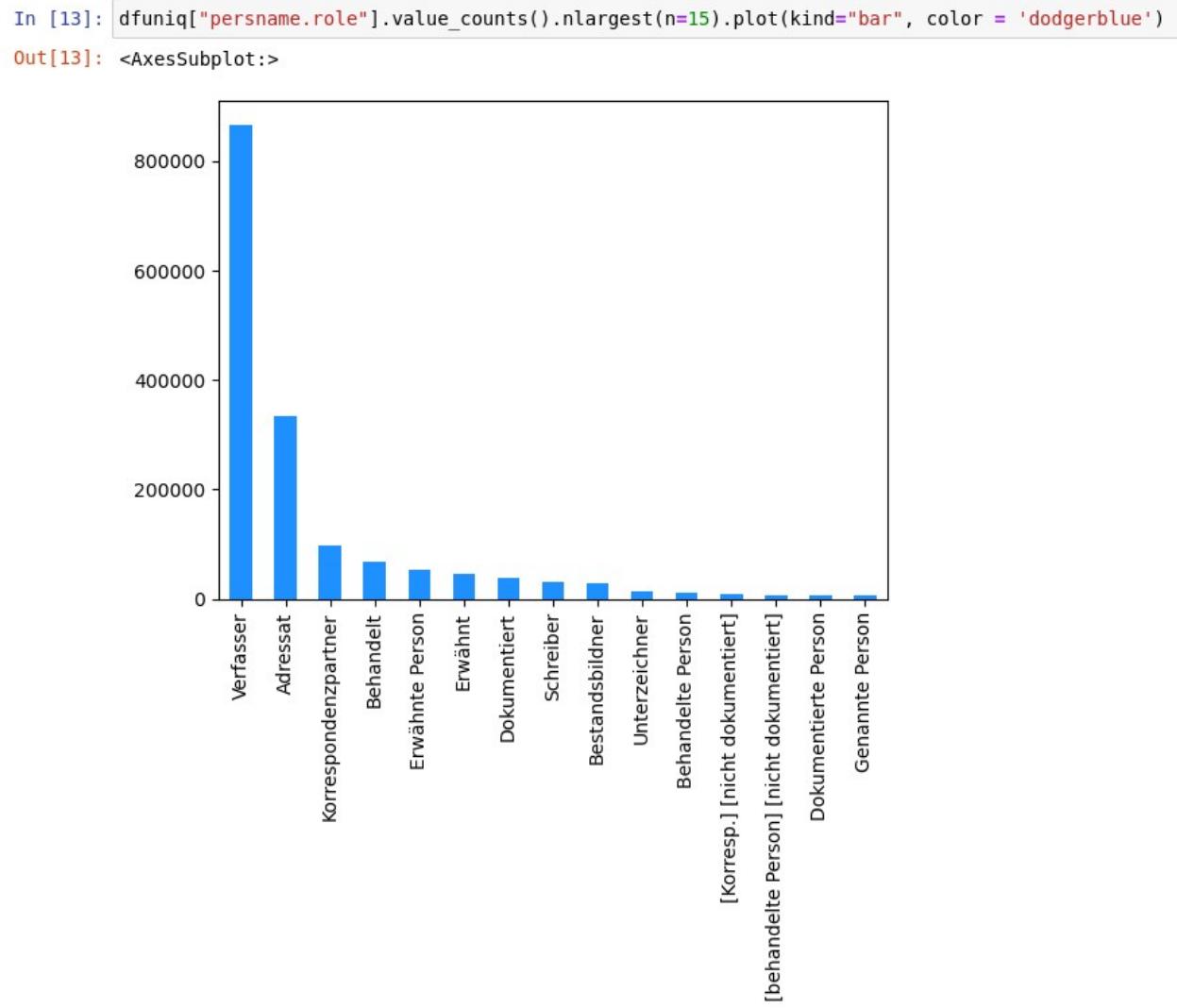
1743864 rows × 6 columns

>persname< in Pandas DataFrame (vor grep): sources Normdaten

```
In [5]: dfuniq["persname.source"].value_counts()
```

```
Out[5]: GND                  794826  
KPE                  355417  
DE-2498                257147  
HelveticArchives        93090  
DE-MUS-853418-Objektdatenbank 34816  
SLA                  7831  
DE-MUS-853418-Personendatenbank 7137  
Name: persname.source, dtype: int64
```

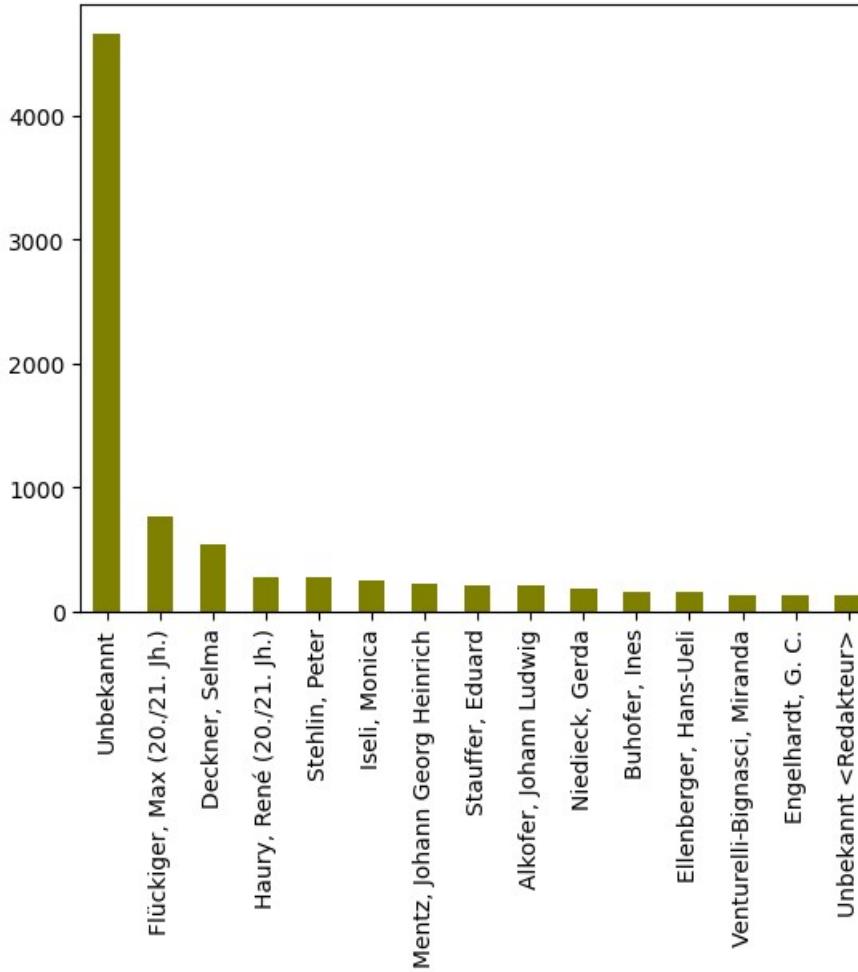
>persname< in Pandas DataFrame (vor grep): roles



>persname< in Pandas DataFrame (vor grep): Namen

```
In [14]: dfuniq["persname.normal"].value_counts().nlargest(n=15).plot(kind="bar", color='olive')
```

```
Out[14]: <AxesSubplot:>
```



>persname< in Pandas DataFrame (nach grep, bereinigte Ids)

```
In [23]: df_cleangnd = pd.read_csv("cleangnd_uniqloopallfields.csv", low_memory=False)
df_cleangnd
```

Out[23]:

	persname.authfilename	persname.content	persname.encodinganalog	persname.normal	persname.role	persname.source
0	000188328	Universität Hamburg / Germanisches Seminar	DE-611-PS-30181216	Universität Hamburg / Germanisches Seminar	Adressat	GND
1	0011647811X	Endrulat, Bernhard	DE-611-PS-30160604	Endrulat, Bernhard	Adressat	GND
2	0011647811X	Endrulat, Bernhard	DE-611-PS-30160604	Endrulat, Bernhard	Verfasser	GND
3	00328524	Schönemann, Toni (1872-1941)	DE-611-PS-328524	Schönemann, Toni	Verfasser	GND
4	00350491	Kurnoth, Waldemar (1880-1962)	DE-611-PS-350491	Kurnoth, Waldemar	Verfasser	GND
...
817623	98149482X	Vorderegger, Roger	DE-611-PS-30147384	Vorderegger, Roger	Verfasser	GND
817624	991305973	Johann Friedrich Gleditsch <Leipzig>	NaN	Johann Friedrich Gleditsch <Leipzig>	Genannte Körperschaft	GND
817625	996017402	Hinz & Kunst (Hamburg)	DE-611-PS-30173633	Hinz & Kunst (Hamburg)	Adressat	GND
817626	999235982	Arnoldische Buchhandlung <Dresden>	NaN	Arnoldische Buchhandlung <Dresden>	Adressat	GND
817627	999235982	Arnoldische Buchhandlung <Dresden>	NaN	Arnoldische Buchhandlung <Dresden>	Genannte Körperschaft	GND

817628 rows × 6 columns

```
In [24]: df_cleangnd["persname.source"].value_counts()
```

Out[24]:

GND	768298
KPE	49330
Name: persname.source, dtype: int64	

```
In [25]: df_cleangnd = df_cleangnd.loc[df_cleangnd['persname.source'] == 'GND']
```

>persname< in Pandas DataFrame (nach grep, bereinigte Ids):

Erstellen eines neuen DataFrames für die vier häufigst vergebenen persname.role

- Zunächst vier einzelne DataFrames für persname.role:
Verfasser, Adressat,
Korrespondenzpartner, Behandelt
- Filtern der jeweils alle vielfach vorkommenden GND-Nummern
- Mit .concat konnten diese vier DataFrames zusammengeführt

```
In [30]: df_cleangnd["persname.role"].value_counts().nlargest(n=50)
```

Out[30]:	Verfasser	318164
	Adressat	128576
	Korrespondenzpartner	46017
	Behandelt	45596
	Erwähnt	34461
	Dokumentiert	30032
	Erwähnte Person	23307
	Bestandsbildner	23165
	Schreiber	12519
	Behandelte Person	8639
	Unterzeichner	8029
	[Korresp.] [nicht dokumentiert]	6959
	Künstler	3757
	[behandelte Person] [nicht dokumentiert]	3703
	[Portrait] [nicht dokumentiert]	3613
	Werktitel Person	3453
	Komponist	2905
	Mitunterzeichner	2843
	Dokumentierte Person	2708
	...	---

>persname< in Pandas DataFrame (nach grep, bereinigte Ids):

Erstellen eines neuen DataFrames für die vier häufigst vergebenen persname.role
Beispiel Adressat

```
In [32]: df_Adressat = df_cleangnd.loc[df_cleangnd['persname.role'] == 'Adressat']
```

```
#df_Qalamos_refined_english = df_Qalamos_refined.loc[df_Qalamos_refined['Oberfläche'] == 'englisch']  
df_Adressat
```

```
Out[32]:
```

	persname.authfilenumber	persname.content	persname.encodinganalog	persname.normal	persname.role	persname.source
0	000188328	Universität Hamburg / Germanisches Seminar	DE-611-PS-30181216	Universität Hamburg / Germanisches Seminar	Adressat	GND
1	0011647811X	Endrulat, Bernhard	DE-611-PS-30160604	Endrulat, Bernhard	Adressat	GND
5	004801458	Staats- und Universitätsbibliothek (Hamburg)	DE-611-PS-30167471	Staats- und Universitätsbibliothek (Hamburg)	Adressat	GND
6	00671366	Mitscherlich, Alexander	DE-611-PS-30145709	Mitscherlich, Alexander	Adressat	GND
13	040425320	Reineck, Hans (-1538)	DE-611-PS-909235	Reineck, Hans	Adressat	GND
...
817616	965660435	-	NaN	Urner, Klaus	Adressat	GND

>persname< in Pandas DataFrame (nach grep, bereinigte Ids):

Erstellen eines neuen DataFrames für die vier häufigst vergebenen persname.role
Beispiel Adressat

In [36]:

```
u_df_Adressat = df_Adressat.drop_duplicates(subset = "persname.authfilenumber", keep = "first")
u_df_Adressat
```

Out[36]:

	persname.authfilenumber	persname.content	persname.encodinganalog	persname.normal	persname.role	persname.source
0	000188328	Universität Hamburg / Germanisches Seminar	DE-611-PS-30181216	Universität Hamburg / Germanisches Seminar	Adressat	GND
1	0011647811X	Endrulat, Bernhard	DE-611-PS-30160604	Endrulat, Bernhard	Adressat	GND
5	004801458	Staats- und Universitätsbibliothek (Hamburg)	DE-611-PS-30167471	Staats- und Universitätsbibliothek (Hamburg)	Adressat	GND
6	00671366	Mitscherlich, Alexander	DE-611-PS-30145709	Mitscherlich, Alexander	Adressat	GND
13	040425320	Reineck, Hans (-1538)	DE-611-PS-909235	Reineck, Hans	Adressat	GND
...
817616	965660435	-	NaN	Urner, Klaus	Adressat	GND
817618	967306337	Moosbrugger, Pius	DE-611-PS-30145232	Moosbrugger, Pius	Adressat	GND
817622	98149482X	Vorderegger, Roger	DE-611-PS-30147384	Vorderegger, Roger	Adressat	GND
817625	996017402	Hinz & Kunst (Hamburg)	DE-611-PS-30173633	Hinz & Kunst (Hamburg)	Adressat	GND
817626	999235982	Arnoldische Buchhandlung <Dresden>	NaN	Arnoldische Buchhandlung <Dresden>	Adressat	GND

101623 rows × 6 columns

>persname< in Pandas DataFrame (nach grep, bereinigte Ids):

Erstellen eines neuen DataFrames für die vier häufigst vergebenen persname.role

```
frames = [u_df_Verfasser, u_df_Adressat, u_df_Korrespondenzpartner, u_df_Behandelt]  
u_df_all = pd.concat(frames)  
u_df_all
```

Out[51]:

	persname.authfilenumber	persname.content	persname.encodinganalog	persname.normal	persname.role	persname.source
2	0011647811X	Endrulat, Bernhard	DE-611-PS-30160604	Endrulat, Bernhard	Verfasser	GND
3	00328524	Schönemann, Toni (1872-1941)	DE-611-PS-328524	Schönemann, Toni	Verfasser	GND
4	00350491	Kurnoth, Waldemar (1880-1962)	DE-611-PS-350491	Kurnoth, Waldemar	Verfasser	GND
7	00671366	Mitscherlich, Alexander	DE-611-PS-30145709	Mitscherlich, Alexander	Verfasser	GND
9	015098230	Igler, André	NaN	Igler, André	Verfasser	GND
...
817578	400141299	Maximilian (Römisch-Deutsches Reich, Kaiser, I.)	DE-611-PS-909797	Maximilian (Römisch-Deutsches Reich, Kaiser, I.)	Behandelt	GND
817596	571680178	Aellen, Hermann (1887-1939)	NaN	Aellen, Hermann	Behandelt	GND
817604	577643304	Cassou, Jean (1897-1986)	NaN	Cassou, Jean	Behandelt	GND
817606	67820551	Porret, Eugene (-1987)	DE-611-PS-30154032	Porret, Eugene	Behandelt	GND
817611	900638699	Magli, Ida (1925-2016)	NaN	Magli, Ida	Behandelt	GND

435210 rows × 6 columns

>persname< in Pandas DataFrame (nach grep, bereinigte Ids):

Erstellen eines neuen DataFrames für die vier häufigst vergebenen persname.role

```
In [53]: u_df_all["persname.role"].value_counts()
```

```
Out[53]: Verfasser          248540  
Adressat            101623  
Korrespondenzpartner    43094  
Behandelt           41953  
Name: persname.role, dtype: int64
```

```
In [54]: u_df_all["persname.normal"].value_counts()
```

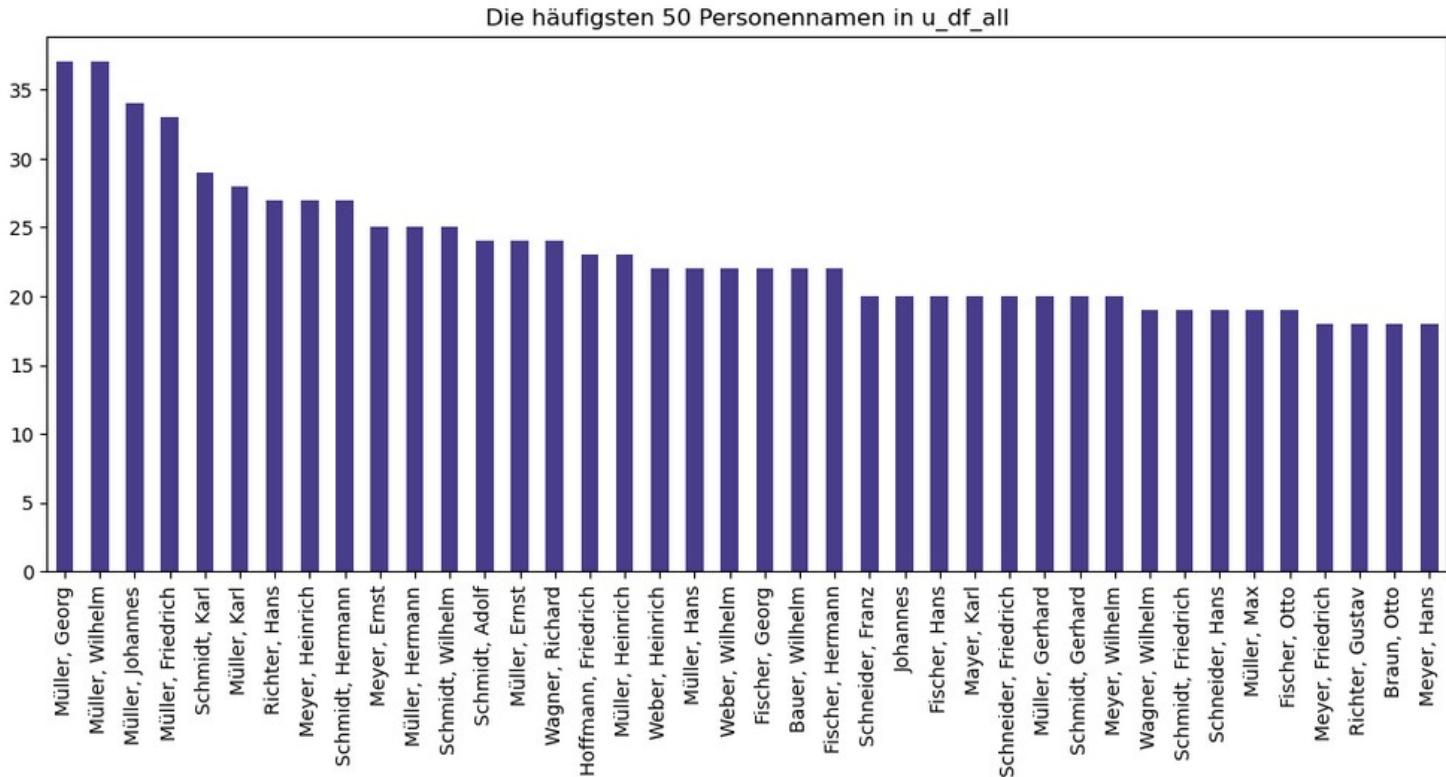
```
Out[54]: Müller, Georg          37  
Müller, Wilhelm          37  
Müller, Johannes         34  
Müller, Friedrich        33  
Schmidt, Karl             29  
..  
Stiebitz, Richard          1  
Stiebitz, Johann Friedrich 1  
Stieda, Ernst              1  
Stein-Landesmann, Alice    1  
Porret, Eugene              1  
Name: persname.normal, Length: 273298, dtype: int64
```

>persname< in Pandas DataFrame (nach grep, bereinigte Ids):

Erstellen eines neuen DataFrames für die vier häufigst vergebenen persname.role

```
In [35]: u_df_all["persname.normal"].value_counts().nlargest(n=40).plot(kind="bar", figsize=(13,5),
                                                               color = 'darkslateblue', title = "Die häufigsten 50 Personennamen in u_df_all")
```

```
Out[35]: <AxesSubplot:title={'center':'Die häufigsten 50 Personennamen in u_df_all'}>
```



>persname< in Pandas DataFrame (nach grep, bereinigte Ids):

Erstellen eines neuen DataFrames für die vier häufigst vergebenen persname.role

```
In [36]: u_df_all["persname.normal"].unique()
```

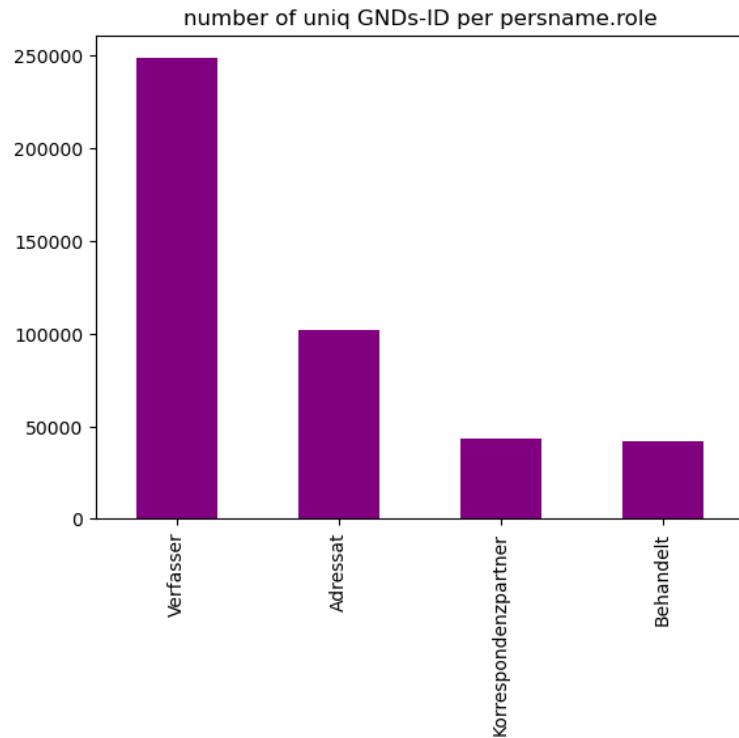
```
Out[36]: array(['Endrulat, Bernhard', 'Schönemann, Toni', 'Kurnoth, Waldemar', ...,
   'Geminiani, Francesco', 'Knorr von Rosenroth, Christian',
   'Porret, Eugene'], dtype=object)
```

```
In [70]: u_df_all["persname.normal"].nunique()
```

```
Out[70]: 273298
```

*>persname< in Pandas DataFrame (nach grep, bereinigte IDs):
Erstellen eines neuen DataFrames für die vier häufigst vergebenen persname.role*

```
In [56]: #df_Qalamos_refined_english["Precision"].value_counts().nlargest(n=10).plot(kind="bar",
#color="purple", title = "precision englische Oberfläche")
u_df_all["persname.role"].value_counts().plot(kind='bar', color='purple', title='number of uniq GNDs-ID per persname.role')
Out[56]: <AxesSubplot:title={'center':'number of uniq GNDs-ID per persname.role'}>
```

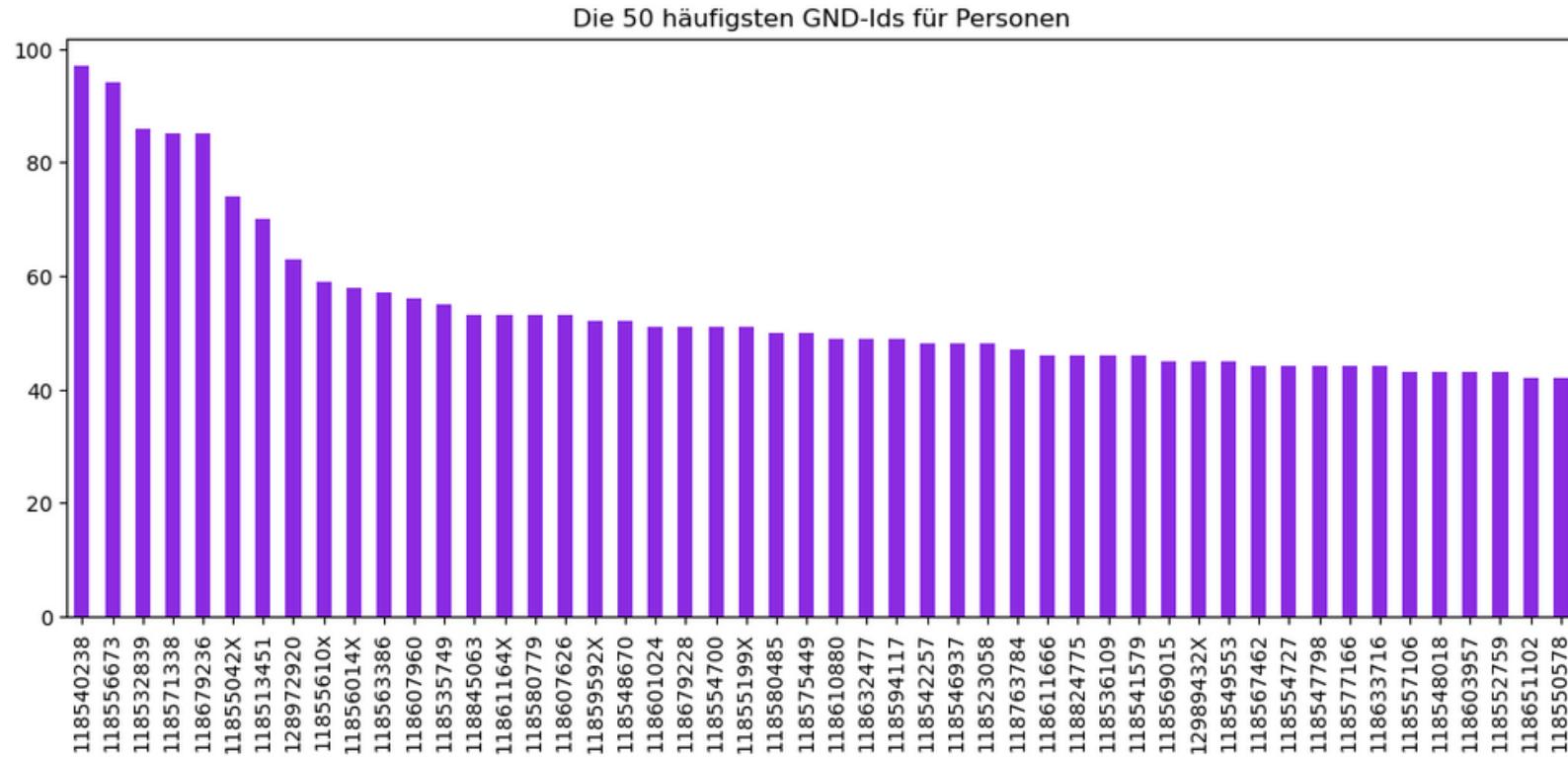


>persname< in Pandas DataFrame (nach grep, bereinigte Ids):

Erstellen eines neuen DataFrames für die vier häufigst vergebenen persname.role

```
In [60]: df_cleangnd["persname.authfilenumber"].value_counts().nlargest(n=50).plot(kind="bar", figsize=(13,5),  
color = 'blueviolet', title = "Die 50 häufigsten GND-Ids für Personen")
```

```
Out[60]: <AxesSubplot:title={'center':'Die 50 häufigsten GND-Ids für Personen'}>
```

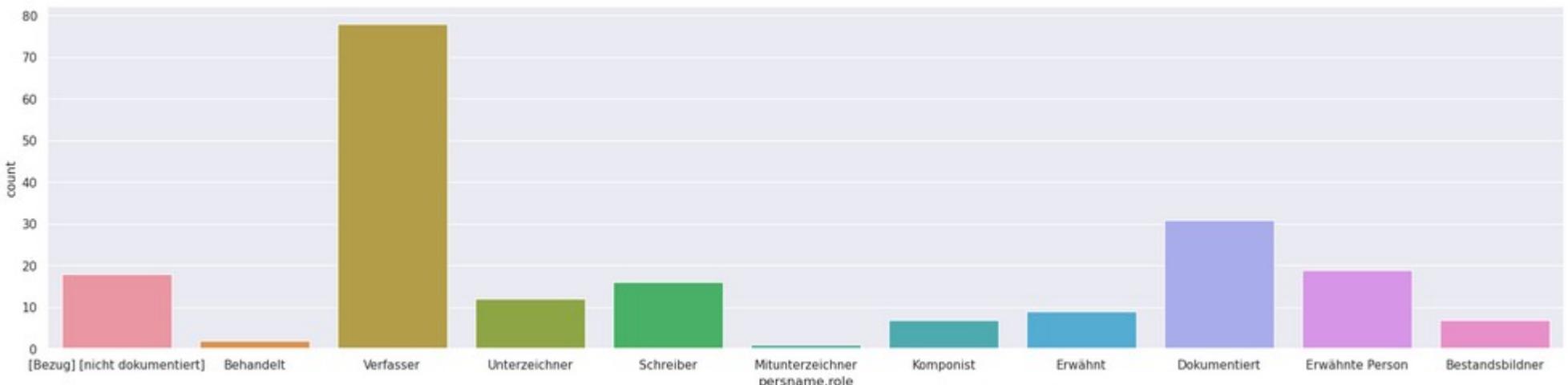


>persname< in Pandas DataFrame (nach grep, bereinigte Ids):

Erstellen eines neuen DataFrames für die vier häufigst vergebenen persname.role

```
sns.catplot(x=more_data_gnd, data=datax, kind='count', aspect = 4) #data =  
data_gnd,
```

ValueError: The truth value of a Series is ambiguous. Use a.empty, a.bool(), a.item(), a.any() or a.all().

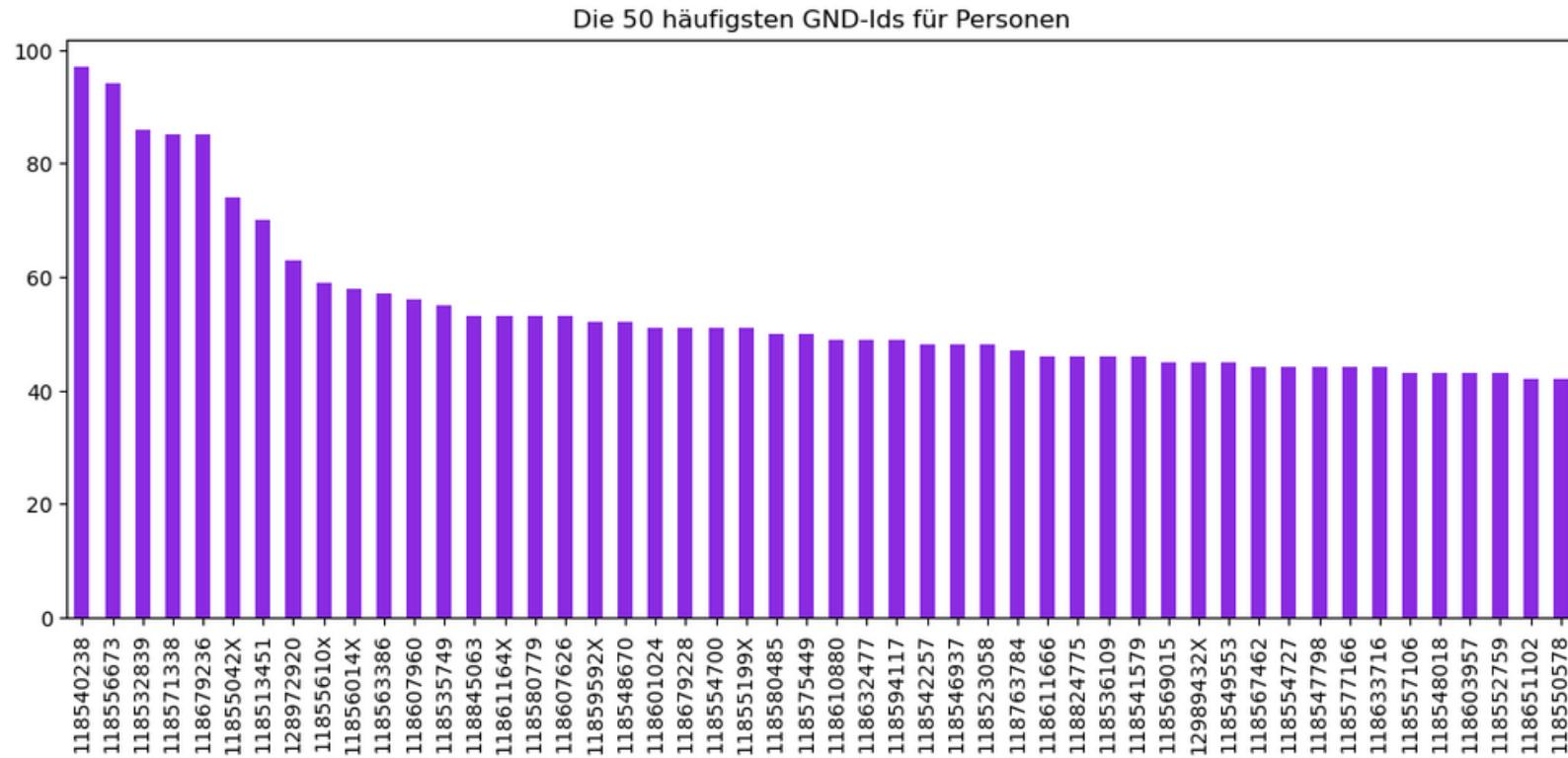


>persname< in Pandas DataFrame (nach grep, bereinigte Ids):

Erstellen eines neuen DataFrames für die vier häufigst vergebenen persname.role

```
In [60]: df_cleangnd["persname.authfilenumber"].value_counts().nlargest(n=50).plot(kind="bar", figsize=(13,5),  
color = 'blueviolet', title = "Die 50 häufigsten GND-Ids für Personen")
```

```
Out[60]: <AxesSubplot:title={'center':'Die 50 häufigsten GND-Ids für Personen'}>
```



Vieelen Dank!

ZBIW Data Librarian 2022/2023, Modul 6, Projektarbeit, Katja Jana,

