Belgium Ancient Book 1601-1625 data analysis and visualization

descriptive statistics

The LIMO database of ancient books contains information about printed publications present at the Library of Leuven which can be dated before 1840. Our specific dataset contains the metadata from ancient Belgian books dating from 1601 to 1625. The metadata is organized into two main categories. First, there is the bibliographical information of the books, for example their (sub)title, author(s), publisher(s), seller(s), printer(s) and so on. Second, there is the holding information of the books, including their provenance, acquisition, library, and physical location.

In [168]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

import nltk
from nltk.corpus import stopwords
from wordcloud import WordCloud
from PIL import Image
from nltk.corpus import stopwords

import folium
from folium import plugins


import gender_guesser.detector

from collections import defaultdict
import re
import networkx as nx
import itertools
from pyvis.network import Network
import community as community_louvain
```

The first part of the exploration is to take a look at the demographic characterists of the authors using the reconciled data

In [308]:

```
## Since the author names in column 700 are separated into two columns 700$0$0$a and 700$1$0$a in ou
# Here I add the reconciled values seprately and join the two dataframe together to get
df_7000a = pd.read_csv('7000a.csv',encoding = 'latin1')
df_7001a = pd.read_csv('7001a.csv',encoding = 'latin1')
df_reconcile=df_7000a.append(df_7001a)
```

```
C:\Users\95327\AppData\Local\Temp\ipykernel_85392\2051073032.py:3: FutureWarning: Th
e frame.append method is deprecated and will be removed from pandas in a future vers
ion. Use pandas.concat instead.
  df_reconcile=df_7000a.append(df_7001a)
```

In [332]:

```
df_reconcile['bool']=None
```

In [336]:

```
# To see whether the authors have been reconciled.
# If both the name and identifier are NaN, it means there are several reconciled value for the same
# If the name is not NaN and the identifier is NaN, it means the name has not been reconciled.
#If they are not reconciled, they will be dropped from the table later.
df_reconcile['bool']=df_reconcile['author name'].isna() ^df_reconcile['identifier'].isna()
```

In [337]:

```
df_reconcile
```

Out[337]:

	author name	identifier	country of citizenship	educated at	field of work	occupation	religion or worldview	sex or gender
0	Albert VII, Archduke of Austria	Q311452	Archduchy of Austria	NaN	NaN	military personnel	Catholic Church	male
1	NaN	NaN	NaN	NaN	NaN	Catholic priest	NaN	NaN
2	Isabella Clara Eugenia	Q158256	Spain	NaN	NaN	politician	Catholic Church	female
3	Alonso de Madrid	Q5670348	Spain	NaN	NaN	literary	Catholicism	male
4	Aegidius Aureaevalensis	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
4289	Diego de Estella	Q325647	Spain	University of Toulouse	NaN	theologian	Catholic Church	male
4290	NaN	NaN	NaN	University of Salamanca	NaN	monk	NaN	NaN
4291	Jan van Blitterswyck	Q27995644	NaN	NaN	NaN	monk	NaN	male
4292	Van Blitterswyck, Maria	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4293	Henricus Sedulius	Q21546722	NaN	NaN	NaN	theologian	NaN	male

4937 rows × 9 columns

In [338]:

```
df_reconcile_clean = df_reconcile.loc[df_reconcile['bool']==False]
```

In [339]:

```
df_reconcile_clean
```

Out[339]:

	author name	identifier	country of citizenship	educated at	field of work	occupation	religion or worldview	sex gend
0	Albert VII, Archduke of Austria	Q311452	Archduchy of Austria	NaN	NaN	military personnel	Catholic Church	ma
1	NaN	NaN	NaN	NaN	NaN	Catholic priest	NaN	Na
2	Isabella Clara Eugenia	Q158256	Spain	NaN	NaN	politician	Catholic Church	fema
3	Alonso de Madrid	Q5670348	Spain	NaN	NaN	literary	Catholicism	ma
6	Bernard of Clairvaux	Q188411	France	NaN	philosophy	theologian	Catholic Church	ma
...	...	...	...	...	...	...	...	
4288	Cornelius Thielmans	Q64031624	NaN	NaN	NaN	writer	NaN	ma
4289	Diego de Estella	Q325647	Spain	University of Toulouse	NaN	theologian	Catholic Church	ma
4290	NaN	NaN	NaN	University of Salamanca	NaN	monk	NaN	Na
4291	Jan van Blitterswyck	Q27995644	NaN	NaN	NaN	monk	NaN	ma
4293	Henricus Sedulius	Q21546722	NaN	NaN	NaN	theologian	NaN	ma

3622 rows × 9 columns

In [314]:

```
df_reconcile_clean['author name'].fillna(method='ffill',inplace=True)
df_reconcile_clean['identifier'].fillna(method='ffill',inplace=True)
```

```
C:\Users\95327\AppData\Local\Temp\ipykernel_85392\3777763541.py:1: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_reconcile_clean['author name'].fillna(method='ffill',inplace=True)
C:\Users\95327\AppData\Local\Temp\ipykernel_85392\3777763541.py:2: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_reconcile_clean['identifier'].fillna(method='ffill',inplace=True)
```

In [315]:

```
df_reconcile_clean
```

Out[315]:

	author name	identifier	country of citizenship	educated at	field of work	occupation	religion or worldview	g
0	Albert VII, Archduke of Austria	Q311452	Archduchy of Austria	NaN	NaN	military personnel	Catholic Church	
1	Albert VII, Archduke of Austria	Q311452	NaN	NaN	NaN	Catholic priest	NaN	
2	Isabella Clara Eugenia	Q158256	Spain	NaN	NaN	politician	Catholic Church	f
3	Alonso de Madrid	Q5670348	Spain	NaN	NaN	literary	Catholicism	
6	Bernard of Clairvaux	Q188411	France	NaN	philosophy	theologian	Catholic Church	
...	...	...	...	...	...	...	...	
4288	Cornelius Thielmans	Q64031624	NaN	NaN	NaN	writer	NaN	
4289	Diego de Estella	Q325647	Spain	University of Toulouse	NaN	theologian	Catholic Church	
4290	Diego de Estella	Q325647	NaN	University of Salamanca	NaN	monk	NaN	
4291	Jan van Blitterswyck	Q27995644	NaN	NaN	NaN	monk	NaN	
4293	Henricus Sedulius	Q21546722	NaN	NaN	NaN	theologian	NaN	

3622 rows × 9 columns

In [316]:

```
df_unique = df_reconcile_clean.drop_duplicates(subset=['identifier'],keep='first')
```

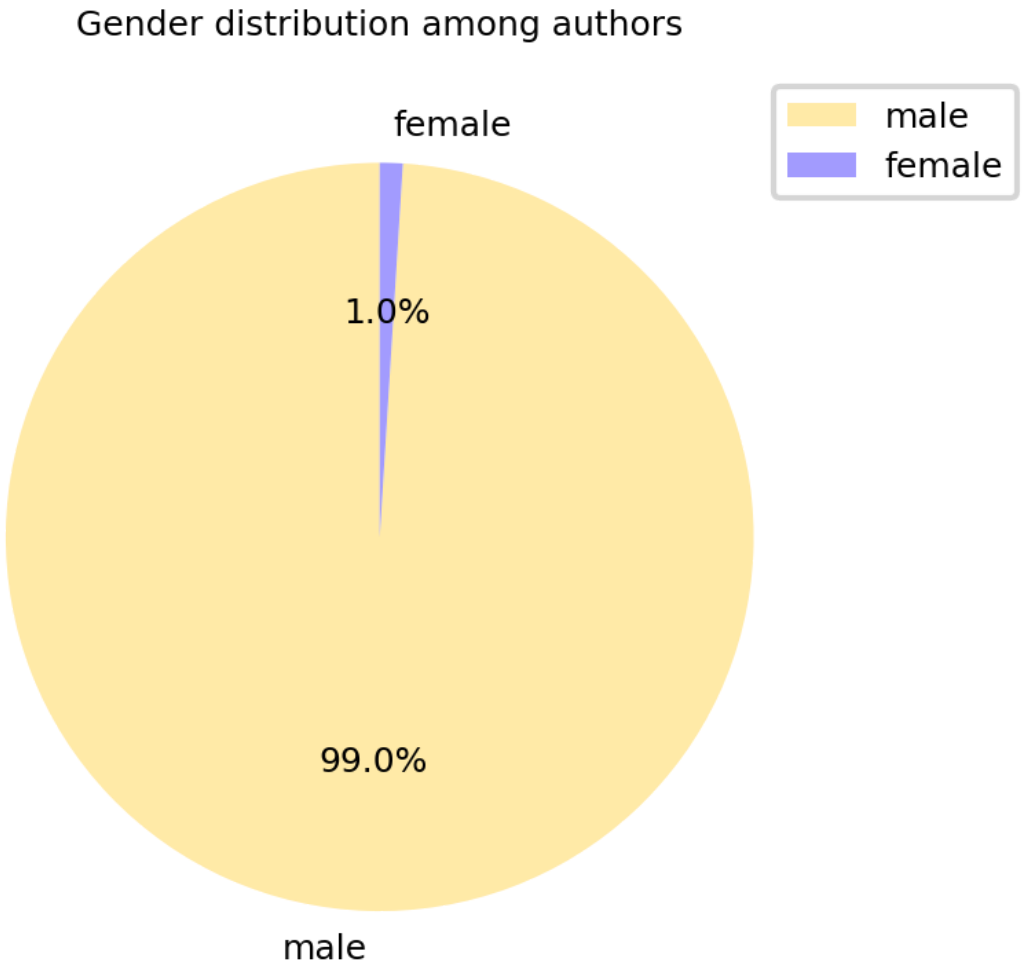
Now we can use the unique dataframe to analysis the authors' gender,religion, citizenship since these values are unique and use cleaned data to analyze the education, occupation and field of work the because they have multiple value

Gender

In [317]:

```
df_gender= df_unique.dropna(subset=['sex or gender'])

sum_male=sum(df_gender['sex or gender']=='male')
sum_female=sum(df_gender['sex or gender']=='female')
category_names=['male','female']
sizes=[sum_male,sum_female]
plt.figure(figsize=(3,3),dpi=300)
custom_colors=['#ffeaa7','#a29bfe']
plt.pie(sizes,labels=category_names,textprops={'fontsize':6},startangle=90,colors=custom_colors,aut
plt.title("Gender distribution among authors",fontsize=6)
plt.legend(bbox_to_anchor=(1.2, 1),fontsize=6)
plt.show()
```



Religion

In [319]:

```
df_religion= df_unique.dropna(subset=['religion or worldview'])
df_religion['religion or worldview'].value_counts()
```

Out[319]:

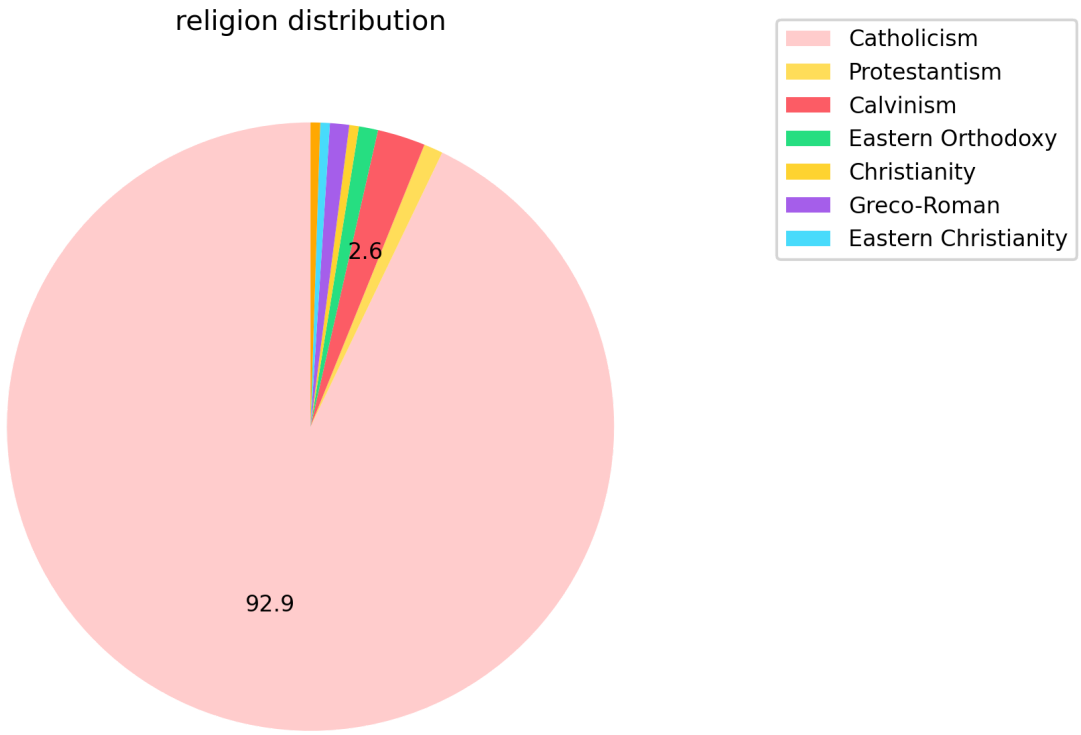
```
Catholic Church            139
Catholicism                 43
Protestantism                5
Eastern Orthodoxy            3
Calvinism                    2
Eastern Christianity         1
Eastern Orthodox Church      1
Christianity                 1
Christian                    1
Greco-Roman religion         1
Name: religion or worldview, dtype: int64
```

In [345]:

```
sum_Catholicism=sum(df_religion['religion or worldview'].isin(['Catholic Church','Catholicism']))
sum_Protestantism=sum(df_religion['religion or worldview']=='Protestantism')
sum_Calvinism=sum(df_religion['religion or worldview']=='Calvinism')
sum_Eastern_Orthodoxy=sum(df_religion['religion or worldview'].isin(['Eastern Orthodoxy ','Eastern O
sum_Christianity =sum(df_religion['religion or worldview'].isin(['Christianity','Christian']))
sum_Greco_Roman=sum(df_religion['religion or worldview']=='Greco-Roman religion')
sum_Eastern_Christianity=sum(df_religion['religion or worldview']=='Eastern Christianity')

category_names=['Catholicism','Protestantism','Calvinism','Eastern Orthodoxy','Christianity','Greco-
sizes=[sum_Catholicism,sum_Christianity,sum_Protestantism,sum_Calvinism,sum_Eastern_Orthodoxy,sum_Ca
plt.figure(figsize=(5,5),dpi=300)
custom_colors=['#ffcccc','#ffdd59','#fc5c65','#26de81','#fed330','#a55eea','#48dbfb','#ffa801']

plt.pie(sizes,textprops={'fontsize':8},startangle=90,colors=custom_colors,autopct = lambda p: forma
plt.legend(bbox_to_anchor=(1.1,1.05),fontsize=8,labels=category_names)
plt.title("religion distribution",fontsize=10)
plt.show()
```

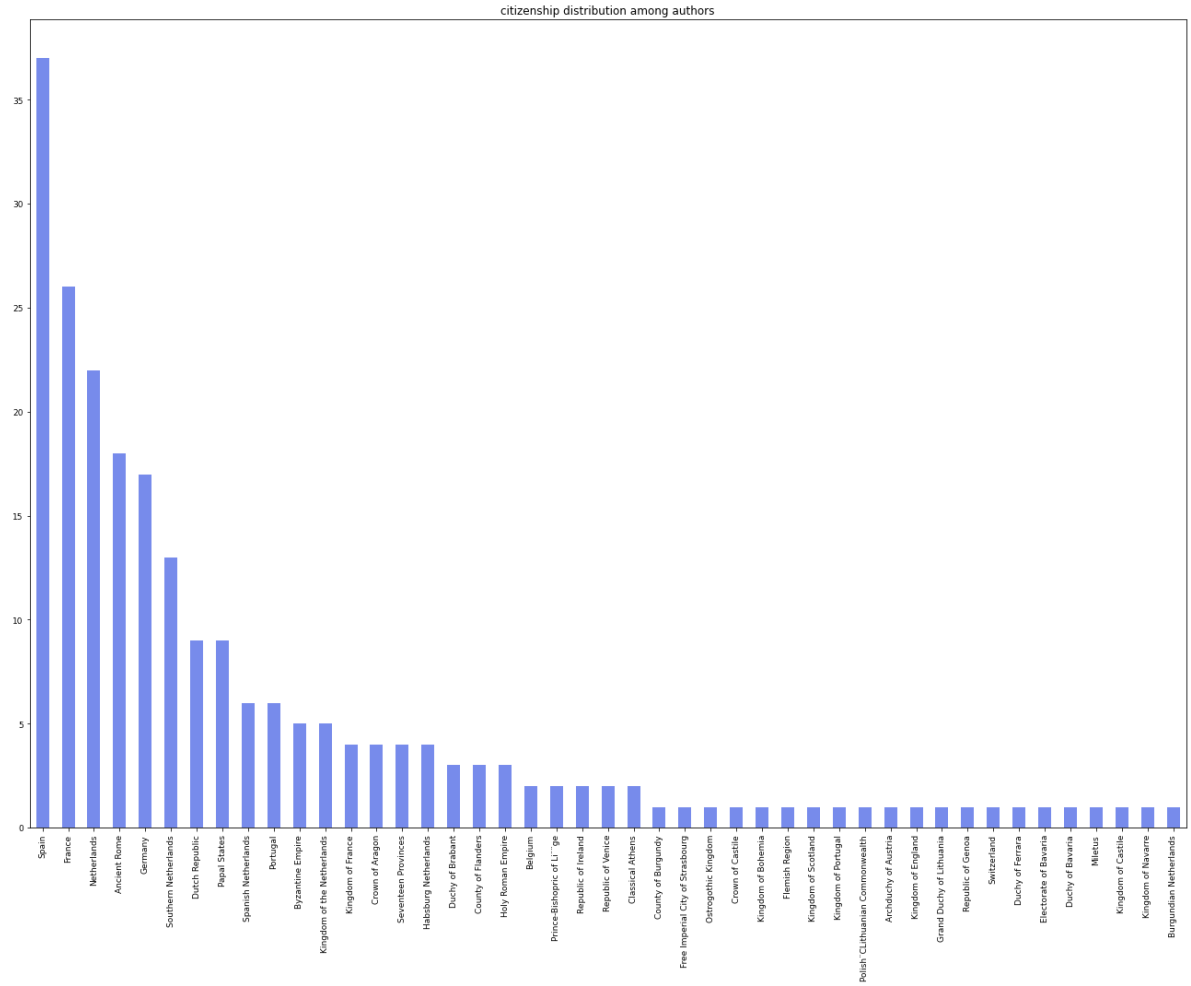


Citizenship

In [326]:

```
df_citizen= df_unique.dropna(subset=['country of citizenship'])
series_citizen =pd.Series(df_citizen['country of citizenship'].value_counts())
plt.figure()

ax = series_citizen.plot(
    kind='bar',
    title ="citizenship distribution among authors",
    figsize=(18, 15),
#     ylim=(2, 9),
    legend=False,
    color='#778beb',
    fontsize=9)
for tick in ax.get_xticklabels():
    tick.set_rotation(90)
plt.tight_layout()
plt.show()
ax.figure.savefig('country distribution.pdf')
```



Field of work

In [340]:

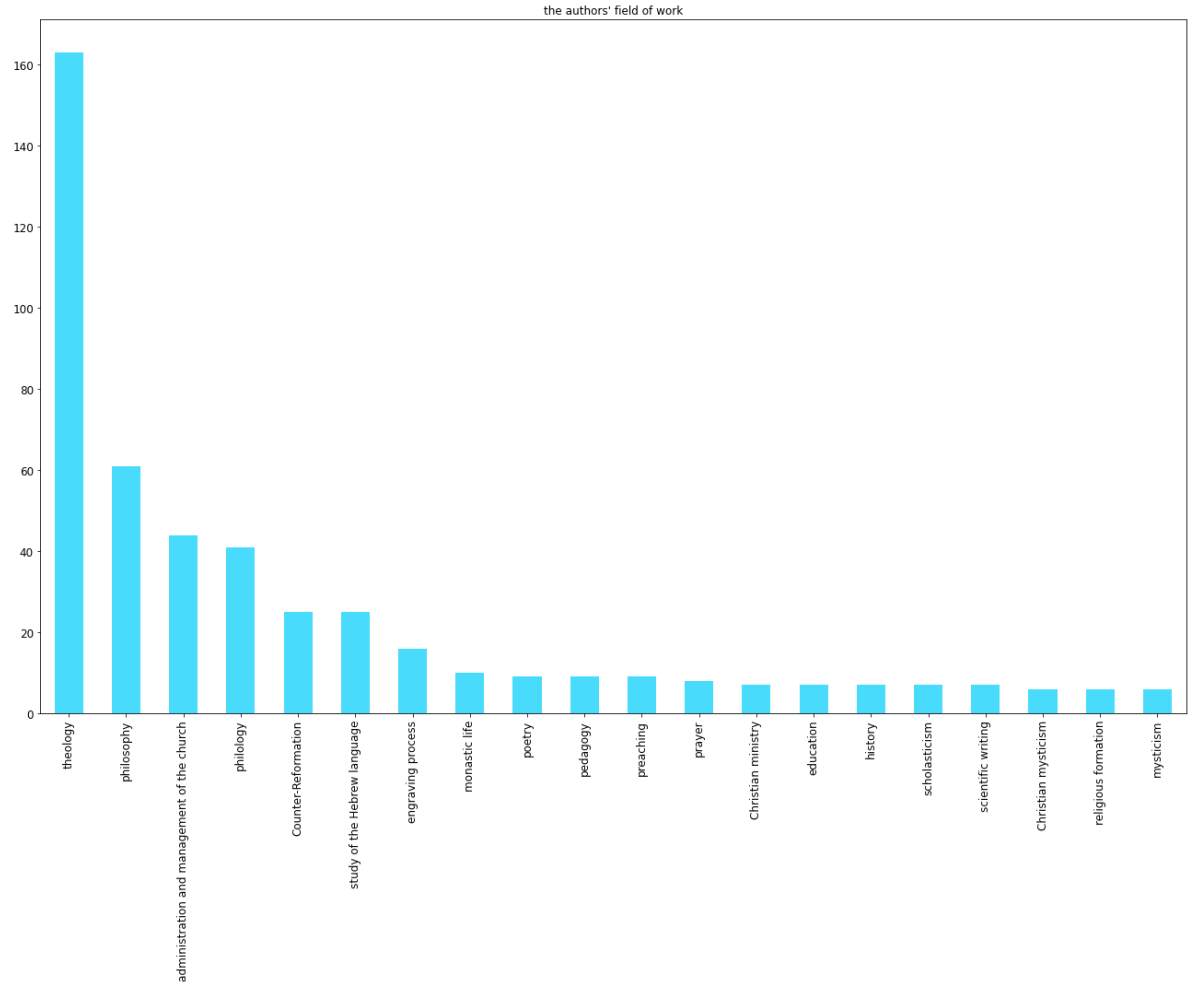
```
df_field = df_reconcile_clean.dropna(subset=['field of work'])
df_field.drop_duplicates(subset=['identifier','field of work'],keep='first')
series_field = df_field['field of work'].value_counts()
```

Out[340]:

```
array(['philosophy', 'theology', 'monastic life',
       'administration and management of the church', 'scholasticism',
       'preaching', 'Christian ministry', 'scientific writing',
       'Christian mysticism', 'education', 'prayer',
       'religious formation', 'mysticism', 'poetry', 'asceticism',
       'religion', 'homiletics', 'politics', 'Reformation',
       'liturgical music', 'philology', 'lintel', 'canon law',
       'engraving process', 'art of painting', 'missionary work',
       'educational system', 'music', 'Counter-Reformation',
       'study of the Hebrew language', 'medicine', 'viticulture',
       'Greek philology', 'Christian Church', 'metal-engraving',
       'architecture', 'Christian philosophy', 'renaissance humanism',
       'religious literature', 'apologetics', 'historiography',
       'classical antiquity', 'linguistics', 'lexicography', 'Dutch',
       'Catholicism', 'pedagogy', 'history', 'logic', 'rhetoric',
       'humanism', 'chronology', 'poetics', 'mathematics', 'astronomy',
       'theology and philosophy', 'fiction', 'literature', 'theatre',
       'military affairs', 'botany', 'alchemy', 'translation',
       'diplomacy', 'Czech studies', 'ecclesiastical hierarchy',
       'art support', 'patronage', 'library science', 'law', 'publisher',
       'prose', 'biography', 'ancient Latin literature', 'court'],
      dtype=object)
```

In [342]:

```
ax_field = series_field[0:20].plot(
    kind='bar',
    title ="the authors' field of work",
    figsize=(18, 15),
#     ylim=(2, 9),
    legend=False,
    color='#48dbfb',
    fontsize=12)
for tick in ax_edu.get_xticklabels():
    tick.set_rotation(90)
plt.tight_layout()
plt.show()
ax_field.figure.savefig('field of work.pdf')
```

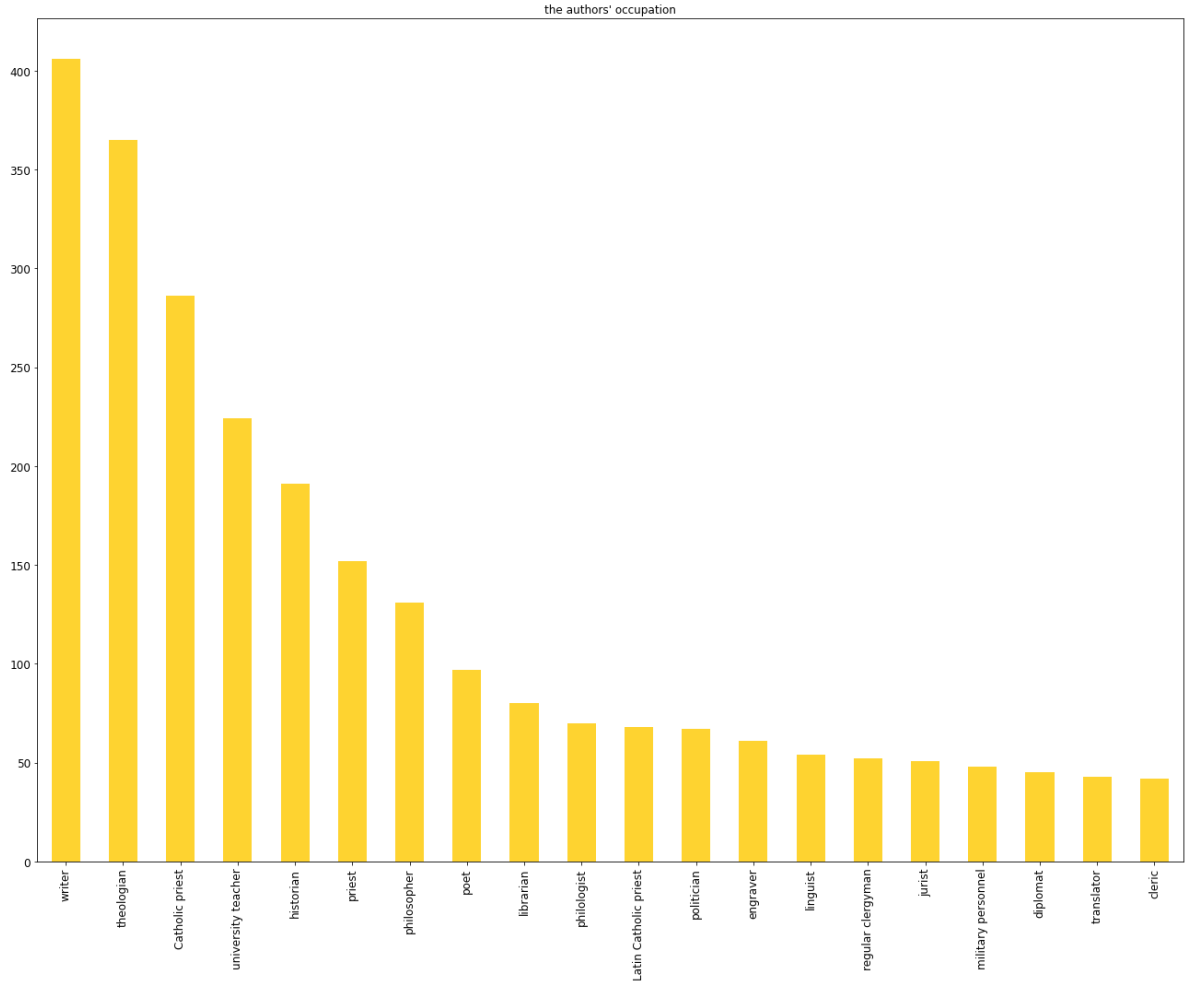


Occupation

In [328]:

```
df_occupation = df_reconcile_clean.dropna(subset=['occupation'])
df_occupation.drop_duplicates(subset=['identifier','occupation'],keep='first')

series_occupation = df_occupation['occupation'].value_counts()
series_occupation
ax_occupation = series_occupation[0:20].plot(
    kind='bar',
    title ="the authors' occupation",
    figsize=(18, 15),
#     ylim=(2, 9),
    legend=False,
    color='#fed330',
    fontsize=12)
for tick in ax_occupation.get_xticklabels():
    tick.set_rotation(90)
plt.tight_layout()
plt.show()
ax_occupation.figure.savefig('occupation.pdf')
```



In [347]:

```
series_occupation[0:20]
```

Out[347]:

```
writer                   406
theologian               365
Catholic priest          286
university teacher       224
historian                191
priest                   152
philosopher              131
poet                      97
librarian                 80
philologist               70
Latin Catholic priest     68
politician                67
engraver                  61
linguist                  54
regular clergyman         52
jurist                    51
military personnel        48
diplomat                  45
translator                43
cleric                    42
Name: occupation, dtype: int64
```

In [ ]:

```

```

Education

In [343]:

```
df_edu = df_reconcile_clean.dropna(subset=['educated at'])
df_edu.drop_duplicates(subset=['identifier','educated at'],keep='first')
series_edu = df_edu['educated at'].value_counts()
series_edu
```

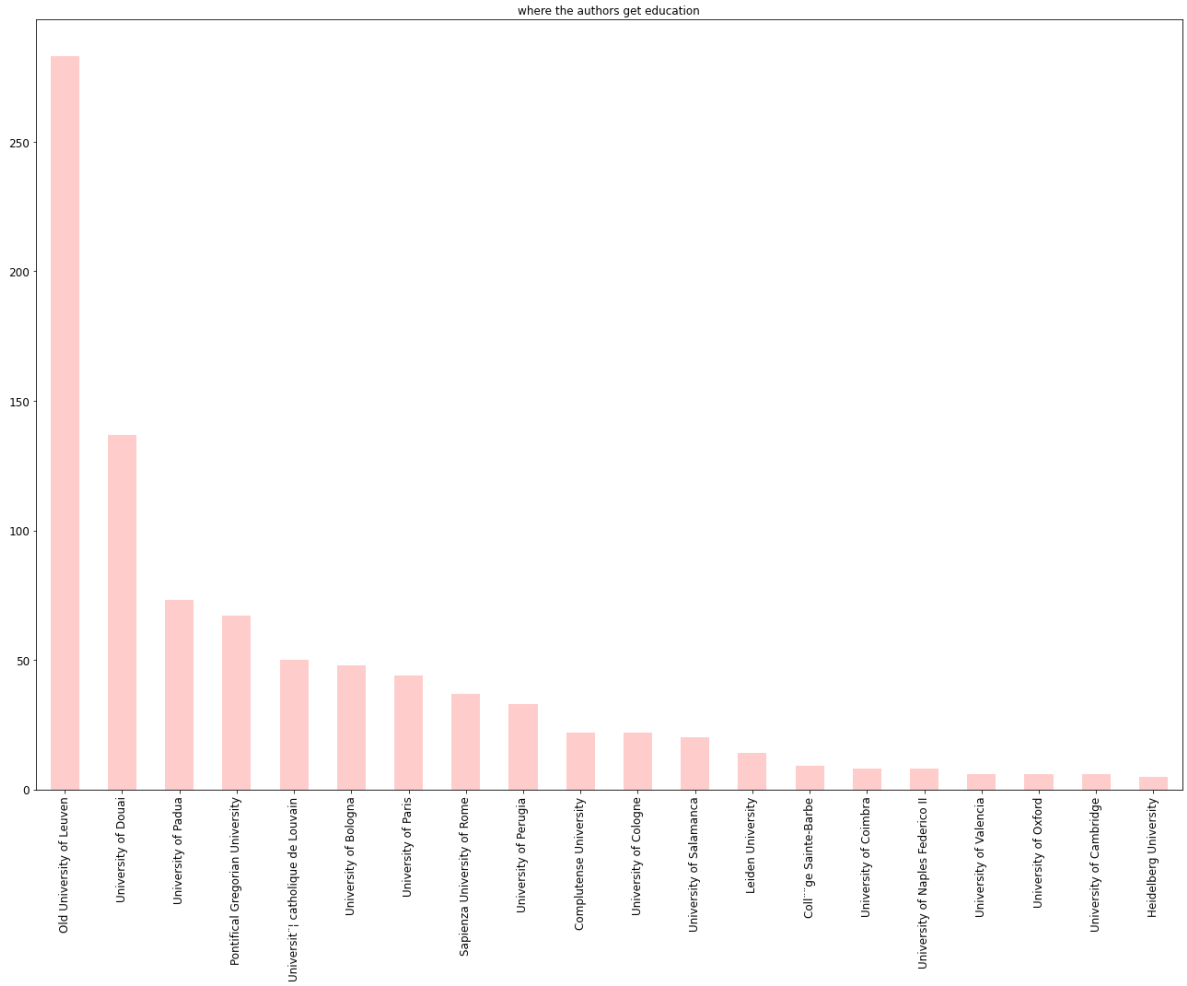
Out[343]:

```
Old University of Leuven             283
University of Douai                  137
University of Padua                   73
Pontifical Gregorian University       67
Universit¨¦ catholique de Louvain     50
                                    ...
Queens' College                        1
University of Turin                    1
Merton College                         1
University of ¨¦vora                   1
University of Helmstedt                1
Name: educated at, Length: 63, dtype: int64
```

In [344]:

```
plt.figure()

ax_edu = series_edu[0:20].plot(
    kind='bar',
    title ="where the authors get education",
    figsize=(18, 15),
#     ylim=(2, 9),
    legend=False,
    color='#ffcccc',
    fontsize=12)
for tick in ax_edu.get_xticklabels():
    tick.set_rotation(90)
plt.tight_layout()
plt.show()
ax_edu.figure.savefig('education distribution.pdf')
```



In [348]:

```
df_edu['educated at'].value_counts()[0:20]
```

Out[348]:

```
Old University of Leuven              283
University of Douai                   137
University of Padua                    73
Pontifical Gregorian University        67
Universit¨¦ catholique de Louvain      50
University of Bologna                  48
University of Paris                    44
Sapienza University of Rome            37
University of Perugia                  33
Complutense University                 22
University of Cologne                  22
University of Salamanca                20
Leiden University                      14
Coll¨¨ge Sainte-Barbe                   9
University of Coimbra                  8
University of Naples Federico II       8
University of Valencia                 6
University of Oxford                   6
University of Cambridge                6
Heidelberg University                  5
Name: educated at, dtype: int64
```

Use the cleaned data

In [ ]:

```
df = pd.read_csv('Belgium-1601-1625_data.csv',encoding = 'latin1')
df.head()
```

Publication Place Analysis

In [5]:

```
# Find out publication location distribution
```

In [7]:

```
publication_place = df['Place_of_publication'].dropna()
publication_place.unique()
```

Out[7]:

```
array(['Antwerp', 'Ghent', 'Leuven', 'Brussels', 'En Brvsselas', 'Mons',
       'A Anvers', 'Ath', 'Li¨¨ge', 'Lvgdvni', 'Avgvst? Ebvronvum',
       'Montibvs', 'Mechelen', 'Tornaci Nerviorvm', 'V?neunt Lovanii',
       'Na de copie Tot Loven', 'S.l.', 'Eerst tot Antwerpen',
       'Pal?opoli Advaticorvm [=Antwerpen]', 'Dvaci', 'Gandavi',
       'A Tournay', 'Middelburgi', 'Parisiis', "Ghedruckt t' Antwerpen",
       'Gheprint Tantwerp?', 'A?Brvxelles', 'Antvverpi?',
       'Palaeopoli Advaticorum', 'Tot Antwerpen', 'Ipris', 'Tournai',
       'Amsterdam', 'Arras', 'Montibus Hannoniae', 'Bruges',
       'Lvgdni Batavorum', 'Lovvain', 'Athi', "t' Ypre",
       'Ypris Flandrorum', 'Paris', 'Louanij', 'LovenI', 'Avtverpi? [!]',
       'Ypre :', 'Ghedruckt tot Loven', '[plaats van uitgave onbekend]',
       'Courtrai', 'Cologne', 'Francofurti', "t'?Ipre", 'In Anverso',
       'Ghedruckt, te Ghendt', 'A Liege', 'Coloni? Agrippin?', 'A Brvges',
       'Tornaci', 'Metelloburgi Mattiacorum [= Antwerp]',
       'drucker tot Ghendt ghesworden', 'Gedruckt tot Lueuen',
       'A Lovvain', 'Na de copije: tot Bruessel', 'Tot Lvyck', 'Louany',
       "gheprint t'Antvverpen", 'Printed, at Antwarp'], dtype=object)
```

In [8]:

```
publication_place=publication_place.replace(['Lvgdvni','Louanij','Na de copie Tot Loven','V?neunt Lo
publication_place=publication_place.replace(['A Anvers','Metelloburgi Mattiacorum [= Antwerp]','Avtv
publication_place=publication_place.replace(['Li¨¨ge','A Liege'],'Liege')
publication_place=publication_place.replace([ 'En Brvsselas','A?Brvxelles', 'Na de copije: tot Brues
publication_place=publication_place.replace([ 'A Brvges','Bruges'],'Bruges')
publication_place=publication_place.replace([ 'Tornaci','Tornaci Nerviorvm','A Tournay'],'Tournai')
publication_place=publication_place.replace(['Ghedruckt, te Ghendt','drucker tot Ghendt ghesworden',
```

In [18]:

```
publication_place
```

Out[18]:

```
1          Antwerp
3            Ghent
5            Ghent
7           Leuven
10        Brussels
            ...
4379       Antwerp
4381       Antwerp
4383       Antwerp
4387       Antwerp
4391       Antwerp
Name: Place_of_publication, Length: 1252, dtype: object
```

In [19]:

```
df_publication = pd.DataFrame(publication_place)
```

In [20]:

```
df_publication
```

Out[20]:

	Place_of_publication
1	Antwerp
3	Ghent
5	Ghent
7	Leuven
10	Brussels
...	...
4379	Antwerp
4381	Antwerp
4383	Antwerp
4387	Antwerp
4391	Antwerp

1252 rows × 1 columns

In [21]:

```
df_publication['Place_of_publication'].unique()
```

Out[21]:

```
array(['Antwerp', 'Ghent', 'Leuven', 'Brussels', 'Mons', 'Ath', 'Liege',
       'Avgvst? Ebvronvum', 'Montibvs', 'Mechelen', 'Tournai', 'S.l.',
       'Dvaci', 'Middelburgi', 'Parisiis', 'Palaeopoli Advaticorum',
       'Ipris', 'Amsterdam', 'Arras', 'Montibus Hannoniae', 'Bruges',
       'Lvgdni Batavorum', 'Athi', "t' Ypre", 'Ypris Flandrorum', 'Paris',
       'Ypre :', '[plaats van uitgave onbekend]', 'Courtrai', 'Cologne',
       'Francofurti', "t'?Ipre", 'Coloni? Agrippin?', 'Tot Lvyck'],
      dtype=object)
```

In [9]:

```
publication_place.value_counts()
```

Out[9]:

```
Antwerp                          863
Leuven                           144
Brussels                         101
Liege                             34
Ghent                             28
Mons                              13
Tournai                            9
Mechelen                           8
S.l.                               6
Ath                                5
Montibvs                           5
t' Ypre                            4
Ipris                              3
Ypris Flandrorum                   3
t'?Ipre                            2
Montibus Hannoniae                 2
Bruges                             2
[plaats van uitgave onbekend]      2
Athi                               2
Dvaci                              2
Francofurti                        1
Cologne                            1
Ypre :                             1
Courtrai                           1
Coloni? Agrippin?                  1
Amsterdam                          1
Paris                              1
Lvgdni Batavorum                   1
Arras                              1
Palaeopoli Advaticorum             1
Parisiis                           1
Middelburgi                        1
Avgvst? Ebvronvum                  1
Tot Lvyck                          1
Name: Place_of_publication, dtype: int64
```

As shown in the List, most of the books were published in Belgium, while a very small portion of books were published in the nearby countries, such as in Paris and Amsterdam

Since the rest cities are either hard to recognize or too small to be include in term of publication number, here I am going to take the top 8 publication cities and map out the location

In [13]:

```
city_list = ['Antwerp','Leuven','Brussels','Liege', 'Ghent', 'Mons','Tournai' , 'Mechelen']
```

In [26]:

```
df_publication=pd.DataFrame(publication_place)
df_publication_top = df_publication.loc[df_publication['Place_of_publication'].isin(city_list)]
df_publication_top.value_counts()
```

Out[26]:

```
Place_of_publication
Antwerp                 863
Leuven                  144
Brussels                101
Liege                    34
Ghent                    28
Mons                     13
Tournai                   9
Mechelen                  8
dtype: int64
```

In [32]:

```
df_publication_top.rename(columns={'Place_of_publication':'city'},inplace=True)
```

...

In [36]:

```
df_publication_top.city
```

Out[36]:

```
1         Antwerp
3           Ghent
5           Ghent
7          Leuven
10       Brussels
           ...
4379      Antwerp
4381      Antwerp
4383      Antwerp
4387      Antwerp
4391      Antwerp
Name: city, Length: 1200, dtype: object
```

In [40]:

```
df_publication_top.index
```

Out[40]:

```
Int64Index([   1,    3,    5,    7,   10,   11,   13,   15,   19,   21,
            ...
            4369, 4371, 4373, 4375, 4377, 4379, 4381, 4383, 4387, 4391],
           dtype='int64', length=1200)
```

In [42]:

```
df_publication_top['lat']=None
df_publication_top['lng']=None
```

...

In [43]:

```
# Search for the latitude and longitude of the 8 cities and add to the dataframe
for i in df_publication_top.index:
    if df_publication_top.city[i] == 'Antwerp':
        df_publication_top['lat'][i]=51.260197
        df_publication_top['lng'][i]=4.402771
    elif df_publication_top.city[i] == 'Leuven':
        df_publication_top['lat'][i]=50.879590
        df_publication_top['lng'][i]=4.700930
    elif df_publication_top.city[i] == 'Brussels':
        df_publication_top['lat'][i]=50.850450
        df_publication_top['lng'][i]=4.348780
    elif df_publication_top.city[i] == 'Liege':
        df_publication_top['lat'][i]=50.633730
        df_publication_top['lng'][i]=5.567490
    elif df_publication_top.city[i] == 'Ghent':
        df_publication_top['lat'][i]=51.050000
        df_publication_top['lng'][i]=3.716670
    elif df_publication_top.city[i] == 'Mons':
        df_publication_top['lat'][i]=50.454130
        df_publication_top['lng'][i]=3.389320
    elif df_publication_top.city[i] == 'Tournai':
        df_publication_top['lat'][i]=50.607150
        df_publication_top['lng'][i]=4.348780
    elif df_publication_top.city[i] == 'Mechelen':
        df_publication_top['lat'][i]=51.025740
        df_publication_top['lng'][i]=4.477620
```

...

In [24]:

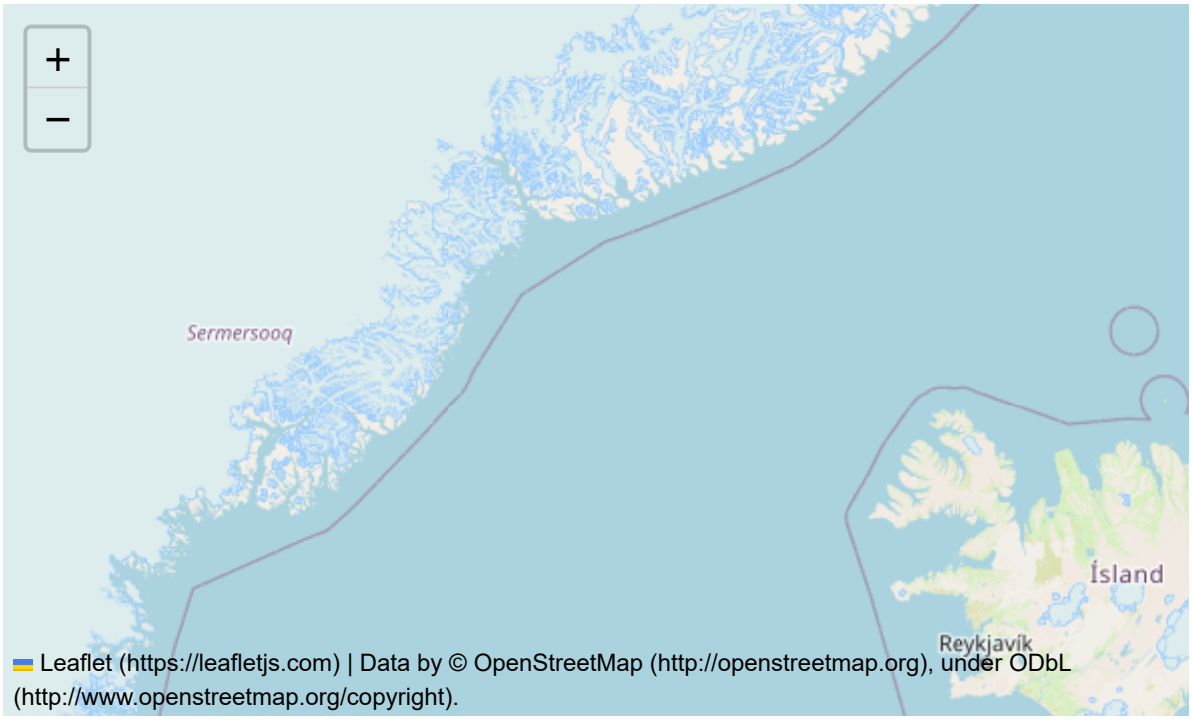
```
latitude= 50.850450
longitude=4.348780

bel_map = folium.Map(location = [latitude,longitude],zoom_start = 8)
```

In [44]:

```
bel_map
```

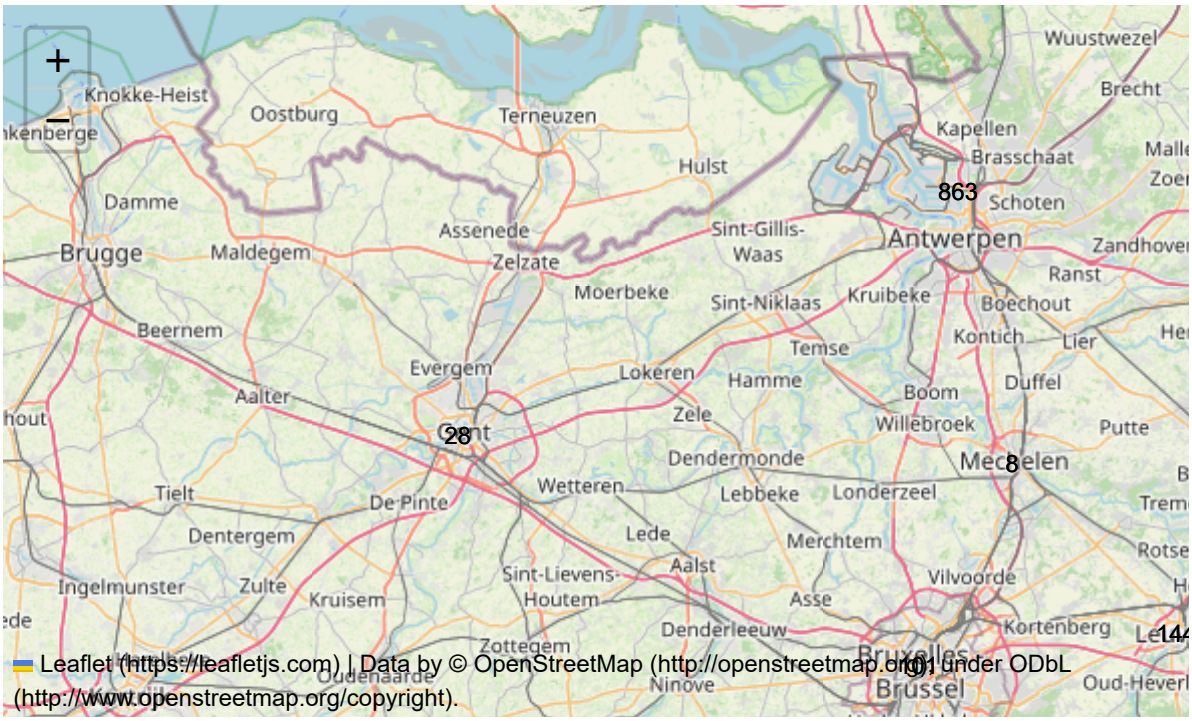
Out[44]:



In [46]:

```
publication = plugins.MarkerCluster().add_to(bel_map)
for lat, lng in zip(df_publication_top.lat,df_publication_top.lng):
    folium.Marker(
    location=[lat,lng],
    icon=None).add_to(publication)
bel_map.add_child(publication)
```

Out[46]:



Book publishers were scattered throughout Belgium, and most of book were published in the centre and north, namely Antwerp, Leuven and Brussels, which became the major cities of Belgium later on.

publication language analysis

In [10]:

```
languages= df['publication_language'].dropna()
languages
```

Out[10]:

```
0       lat
2       dut
4       dut
6       dut
9       dut
       ...
4378    lat
4380    lat
4382    dut
4386    dut
4390    lat
Name: publication_language, Length: 1143, dtype: object
```

In [11]:

```
# Find out what languages are used in the ancient books
languages.unique()
```

Out[11]:

```
array(['lat', 'dut', 'spa', 'fre', 'heb', 'per', 'grc', 'eng'],
      dtype=object)
```

In [12]:

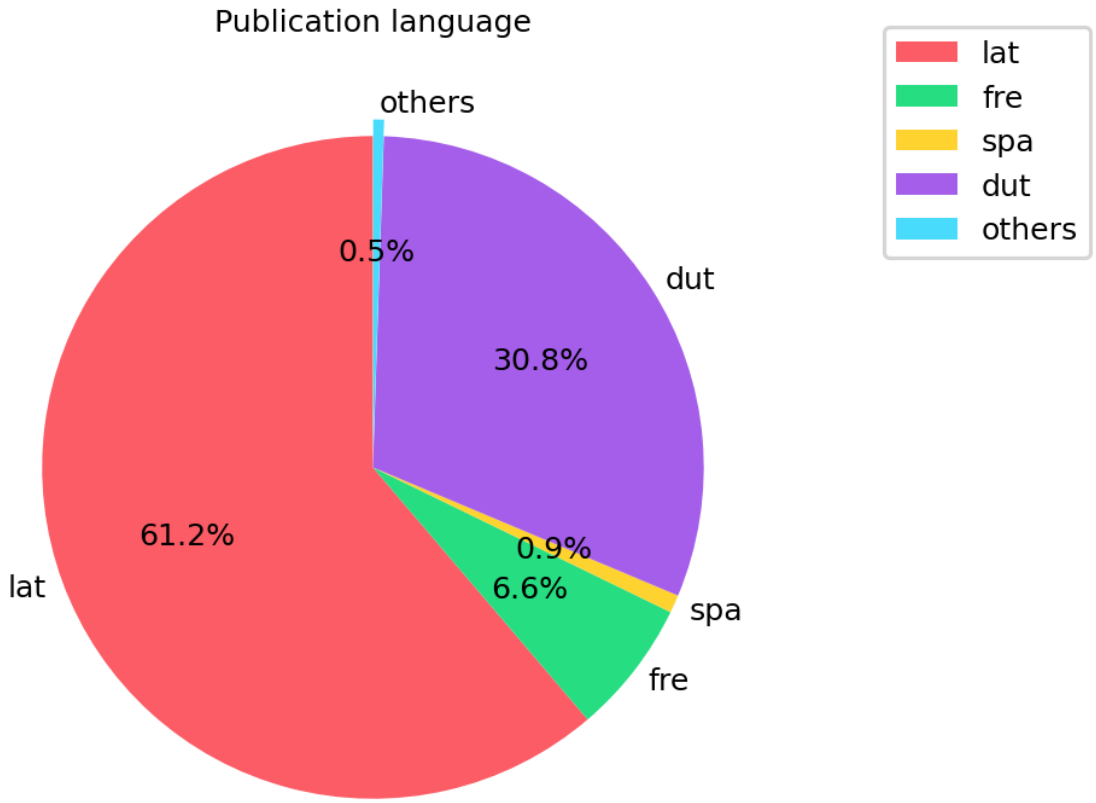
```
# Find out language distribution
languages.value_counts()
```

...

In [18]:

```
labels=['lat', 'fre','spa', 'dut','others']
sizes=[700,75,10,352,6]
plt.figure(figsize=(4,4),dpi=227)
explode = (0,0,0,0,0.05)
custom_colors=['#fc5c65','#26de81','#fed330','#a55eea','#48dbfb']
# ,textprops={'fontsize':6},

plt.pie(sizes,labels=labels,labeldistance=1.05,startangle=90,textprops={'fontsize':8},colors=custom
plt.legend(bbox_to_anchor=(1.1,1.05),fontsize=8,labels=labels)
plt.title('Publication language',fontsize = 8)
plt.show()
```



As shown in the picture, among all the publication languages, Latin makes up the majority, with Dutch coming in second at 30.6%.

linguistic features

latin

In [21]:

```
#In this case I would use the most popular two language, the latin and dutch to find out the linguis
title= df['Main_title'].dropna()
title
```

Out[21]:

```
0       F. Henrici SedvlI ... Pr?scriptiones adversvs ...
2                                 Den crvys-wech Christi
4       Kalengier ende lvst-hof der H. kercke verciert...
6       Den christelycken spieghel, om wel ende deuchd...
9       Edicht ende ordinancie vande eertzhertogen ons...
                              ...
4378    C. Ivlii C?saris Commentariorvm de bello civil...
4380    M. Tvllii Ciceronis Pro lege Manilia ad popvlv...
4382    Soliloqvivm oft Alleenspraecke des H. seraphis...
4386            Van des vvereldts ydelheden te versmaden.
4390    F. Henrici Sedvlii ... Apologeticvs aduersus A...
Name: Main_title, Length: 1357, dtype: object
```

In [24]:

```
df_latin = df.loc[df['publication_language']=='lat']
title_latin = df_latin['Main_title'].dropna()
```

In [29]:

```
latin_stopword='ab, ac, ad, adhic, aliqui, aliquis, an, ante, apud, at, atque, aut, autem, cum, cur,
latin_stopword=latin_stopword.split(', ')
latin_stopword
```

...

In [42]:

```
title_as_string = ' '.join(title_latin)
```

In [43]:

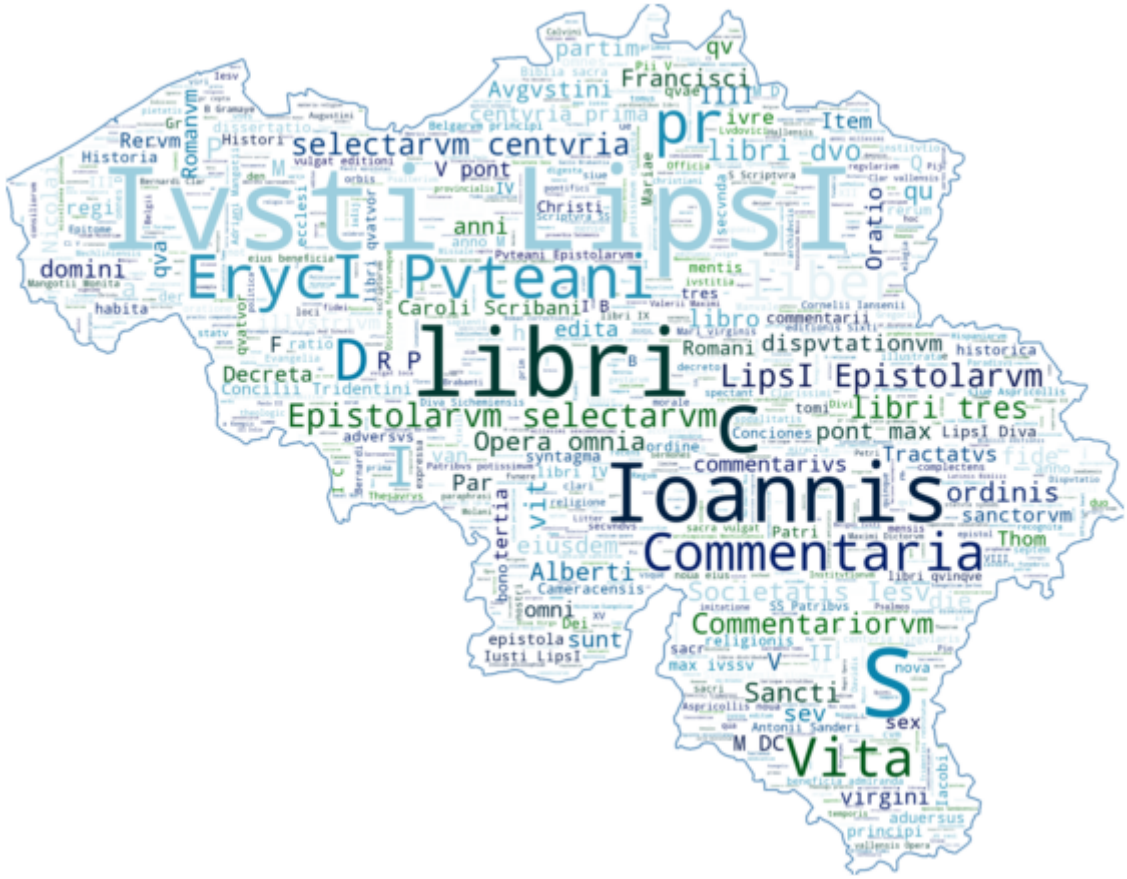
```
belguim_file = 'belgium.png'
icon=Image.open(belguim_file).convert("RGBA")
image_mask=Image.new(mode='RGB',size=icon.size,color=(255,255,255))
image_mask.paste(icon,box=icon)


rgb_array=np.array(image_mask)


word_cloud=WordCloud(mask=rgb_array,background_color='white',stopwords=latin_stopword,max_words=100
word_cloud.generate(title_as_string)

plt.figure(figsize=[16,8])

plt.imshow(word_cloud,interpolation='bilinear')
plt.axis('off')
plt.show()
```



As shown in the picture

```
Ivsti LipsI
ErycI Pvteni
LispI Epistolarvm
Ioannis
Commentaria
```

dutch

In [45]:

```
df_dutch = df.loc[df['publication_language']=='dut']
title_dutch = df_dutch['Main_title'].dropna()
```

In [53]:

```
title_dutch
```

Out[53]:

```
2                                  Den crvys-wech Christi
4       Kalengier ende lvst-hof der H. kercke verciert...
6       Den christelycken spieghel, om wel ende deuchd...
9       Edicht ende ordinancie vande eertzhertogen ons...
14      Een gulden boecxken ghenoemt De conste om Godt...
                             ...
4339                          Beghijnken van Machelen,
4347    Den oorspronck ende cavse vande iaerlicxsche f...
4355    De gheestelycke vryagie, waer Christvs de ziel...
4382    Soliloqvivm oft Alleenspraecke des H. seraphis...
4386            Van des vvereldts ydelheden te versmaden.
Name: Main_title, Length: 352, dtype: object
```

In [50]:

```
title_as_string = ' '.join(title_dutch)
```

In [54]:

```
title_as_string
```

```
...
```

In [55]:

```
dutch_stopword = set(stopwords.words('dutch'))
```

In [56]:

```
word_cloud=WordCloud(mask=rgb_array,background_color='white',stopwords=dutch_stopword,max_words=100
word_cloud.generate(title_as_string)

plt.figure(figsize=[16,8])

plt.imshow(word_cloud,interpolation='bilinear')
plt.axis('off')
plt.show()
```

There are also places for example 'leuven','Antwerp'

french

In [58]:

```
df_french = df.loc[df['publication_language']=='fre']
title_french = df_french['Main_title'].dropna()
title_french
```

Out[58]:

```
175     Practiqve de la perfection et des vertvs chres...
180     La ivstice de S. A. imploree povr la deffense ...
223          Le pelerin moral de F. Bernard dv Verger ...
238     Contemplations tres-pievses svr le crvcifix, e...
243                                Miroir de discipline
                            ...
4237    Histoire de la vie, miracles et translation de...
4288    Histoire des miracles advenvz n'agveres a l'in...
4292    Histoire des miracles advenvz n'agveres a l'in...
4305    Avtres miracles de Nostre Dame av Mont-aigv ad...
4322    Ordonnance et edict perpetvel des archidvcqz ....
Name: Main_title, Length: 75, dtype: object
```

In [61]:

```
french_stopword = set(stopwords.words('french'))
```

In [62]:

```
title_as_string = ' '.join(title_french)
word_cloud=WordCloud(mask=rgb_array,background_color='white',stopwords=french_stopword,max_words=10
word_cloud.generate(title_as_string)

plt.figure(figsize=[16,8])

plt.imshow(word_cloud,interpolation='bilinear')
plt.axis('off')
plt.show()
```

In [44]:

```
# the most common words in different languages
```

Social Network Analysis

In [51]:

```
series_authors = df['700'].dropna()
```

In [52]:

```
series_authors
```

Out[52]:

```
0       ^$1$ $aSedulius, Henricus$ $ $c0.F.M.$ $ $d154...
2                    ^$1$ $ade Coster, Gillis.$ $ $4aut
4                    ^$1$ $ade Coster, Gillis.$ $ $4aut
6       ^$1$ $aAlonso de Madrid$ $ $c0.F.M.$ $ $d1480-...
9       ^$0$ $aAlbrecht$ $ $bVII$ $ $cArchduke of Aust...
                            ...
4378    ^$1$ $aCaesar, Caius Julius$ $ $d100 BC-44 BC$...
4380    ^$1$ $aCicero, Marcus Tullius$ $ $d106 BC-43 B...
4382    ^$1$ $aBonaventura$ $ $c0.F.M.$ $ $d1221-1274$...
4386    ^$1$ $aStella, Didacus$ $ $c0.F.M.$ $ $d1524-1...
4390    ^$1$ $aSedulius, Henricus$ $ $c0.F.M.$ $ $d154...
Name: 700, Length: 1233, dtype: object
```

In [53]:

```
authors = []
roles = []
for i in series_authors:
    patern1 = re.compile(r'\^\$\d\$\s\$\w(.*?)\$')
    patern2 = re.compile(r'\$\s\$\s\$4(\w\w\w)')
    author = patern1.findall(i)
    role = patern2.findall(i)
    authors.append(author)
    roles.append(role)
```

In [141]:

```
# Find out the average number of authors for a book
length_author=[]
for i in authors:
    length_author.append(len(i))
length_author_series = pd.DataFrame(length_author)
length_author_series.mean()
```

Out[141]:

```
0    2.135442
dtype: float64
```

In [154]:

```
(length_author_series==1).sum()
```

Out[154]:

```
0    569
dtype: int64
```

In [155]:

```
length_author_series.max()
```

Out[155]:

```
0    39
dtype: int64
```

There are 569 book who were written by independent author, the average author for a book is 2.13, and the book with the most authors is 'Ivsti Lipsi sapienti? et litterarvm antistitis fama postuma', with 39 authors accomplishing it together.

In [143]:

```
len(length_author_series)
```

Out[143]:

```
1233
```

In [144]:

```
len(length_author_series)
```

Out[144]:

```
1233
```

In [84]:

```
df_authors = pd.DataFrame(authors)
```

...

In [85]:

```
df_roles = pd.DataFrame(roles)
df_roles
```

...

In [90]:

```
author_role=[]
for i in range(0,39):
    for j in range(0,1233):
        author_role.append((df_authors[i][j],df_roles[i][j]))
```

In [91]:

```
author_role
```

```
('Bellarmino, Roberto', 'aut'),
('Colibrant, Rumoldus', 'dte'),
('van Gorcum, Jan', 'aut'),
('Thomas Aquinas', 'aut'),
('Boonen, Jacobus', 'dte'),
('Lucas Brugensis, Franciscus', 'aut'),
('Lipsius, Justus', 'eul'),
('Andreas, Valerius', 'aut'),
('Barradas, Sebastianus', 'aut'),
('Aldobrandini, Pietro', 'dte'),
('Gramaye, Joannes Baptista', 'aut'),
('de Covarrubias y Leyva, Diego', 'aut'),
('Justinianus', 'oth'),
('Lipsius, Justus', 'aut'),
('Lipsius, Justus', 'aut'),
('Cuyckius, Henricus', 'aut'),
('Miraeus, Johannes', 'dte'),
('Hunnaeus, Augustinus', 'aut'),
('Cripius, Guillelmus', 'aut'),
```

In [98]:

```
while (None, None) in author_role:
    author_role.remove((None,None))
```

In [99]:

```
len(author_role)
```

Out[99]:

2638

In [ ]:

```
#Find out people who play multiple roles
```

In [100]:

```
author_role_series = pd.Series(author_role)
author_role_series.value_counts()
```

Out[100]:

```
(Lipsius, Justus, aut)                  68
(Puteanus, Erycius, aut)                34
(Costerus, Franciscus, aut)             34
(Lessius, Leonardus, aut)               27
(Scribani, Carolus, aut)                26
                                        ..
(Moretus, Balthasar, edt)                1
(Divaeus, Petrus, aut)                   1
(Baeckx van Baerlandt, Adrianus, ctb)    1
(Favre, Antoine, clb)                    1
(toe Boecop, Arent, ctb)                 1
Length: 1416, dtype: int64
```

In [106]:

```
s= author_role_series.unique()
```

In [115]:

```
multi_role=[]
for i in range(0,1416):
    for j in range(i+1, 1416):
        if s[i][0]==s[j][0]:
            multi_role.append(s[i])
            multi_role.append(s[j])
```

In [116]:

```
multi_role
```

```
('Oudaert, Nicolaus', 'eul'),
('Bochius, Joannes', 'eul'),
('Bochius, Joannes', 'ctb'),
('Van den Wouwer, Jan', 'apr'),
('Van den Wouwer, Jan', 'ctb'),
('Bulteel, Gislain', 'eul'),
('Bulteel, Gislain', 'ctb'),
('Segeth, Thomas', 'eul'),
('Segeth, Thomas', 'ctb'),
('Lindanus, David', 'eul'),
('Lindanus, David', 'ctb'),
('Bircovius, Simon', 'ctb'),
('Bircovius, Simon', 'eul'),
('Rivius, Gaugericus', 'eul'),
('Rivius, Gaugericus', 'ctb'),
('Bircovius, Fabianus', 'ctb'),
('Bircovius, Fabianus', 'eul'),
('van Hoye, Andr¨¦', 'ctb'),
('van Hoye, Andr¨¦', 'eul')]
```

In [117]:

```
d=defaultdict(list)
```

In [118]:

```
for key,value in multi_role:
    d[key].append(value)
multi_role_list=[]
for i in d.items():
    multi_role_list.append(i)
multi_role_list
```

...

In [120]:

```
df_author_role = pd.DataFrame(multi_role_list)
```

In [121]:

```
df_author_role
```

Out[121]:

	0	1
0	Sedulius, Henricus	[aut, oth, aut, edt, oth, edt]
1	Albrecht	[oth, aut, oth, dte, oth, apb, aut, dte, aut, ...
2	Junius, Balduinus	[aut, com]
3	Florus, Lucius Annaeus.	[aut, ctb]
4	Godefridi, Petrus	[edt, aut]
...	...	...
177	Bochius, Joannes	[eul, ctb]
178	Bulteel, Gislain	[eul, ctb]
179	Segeth, Thomas	[eul, ctb]
180	Bircovius, Simon	[ctb, eul]
181	Bircovius, Fabianus	[ctb, eul]

182 rows × 2 columns

In [134]:

```
df_author_role.rename(columns={0:'name'},inplace = True)
df_author_role.rename(columns={1:'role'},inplace = True)
```

In [137]:

```
df_author_role.set_index('name',inplace=True)
```

In [138]:

```
df_author_role.loc['Albrecht']
```

Out[138]:

```
role    [oth, aut, oth, dte, oth, apb, aut, dte, aut, ...
Name: Albrecht, dtype: object
```

In [156]:

```
authors
```

...

In [307]:

```
# To see how active the author is

#Flatten the nested list
author_list = []
for i in authors:
    for j in i:
        author_list.append(j)

author_count = {}
for i in author_list:
    if i not in author_count:
        author_count[i]=1
    else:
        author_count[i]+=1

# sort active author frequency

sort_author_frequency = [(c,a) for a,c in author_count.items()]
sort_author_frequency.sort(reverse=True)
sort_author_frequency = [(a,c) for c,a in sort_author_frequency]
sort_author_frequency
```

Out[307]:

```
[('Lipsius, Justus', 97),
 ('Puteanus, Erycius', 43),
 ('Albrecht', 37),
 ('Isabella Clara Eugenia', 36),
 ('Costerus, Franciscus', 36),
 ('Lessius, Leonardus', 32),
 ('Scribani, Carolus', 30),
 ('Hovius, Matthias', 30),
 ('Bellarmino, Roberto', 25),
 ('Miraeus, Aubertus', 24),
 ('Spoelbergh, Guilielmus', 20),
 ('Makeblijde, Lodewijk', 20),
 ('Numan, Philippus', 19),
 ('Gramaye, Joannes Baptista', 19),
 ('David, Jan', 18),
 ('a Lapide, Cornelius Cornelii', 16),
 ('Galle, Theodoor', 16),
 ('Beverlinck, Laurentius', 16),
```

Here we see that Lipsius Justus is the most active author

In [167]:

```
unique_author = set(author_list)
unique_author
```

Out[167]:

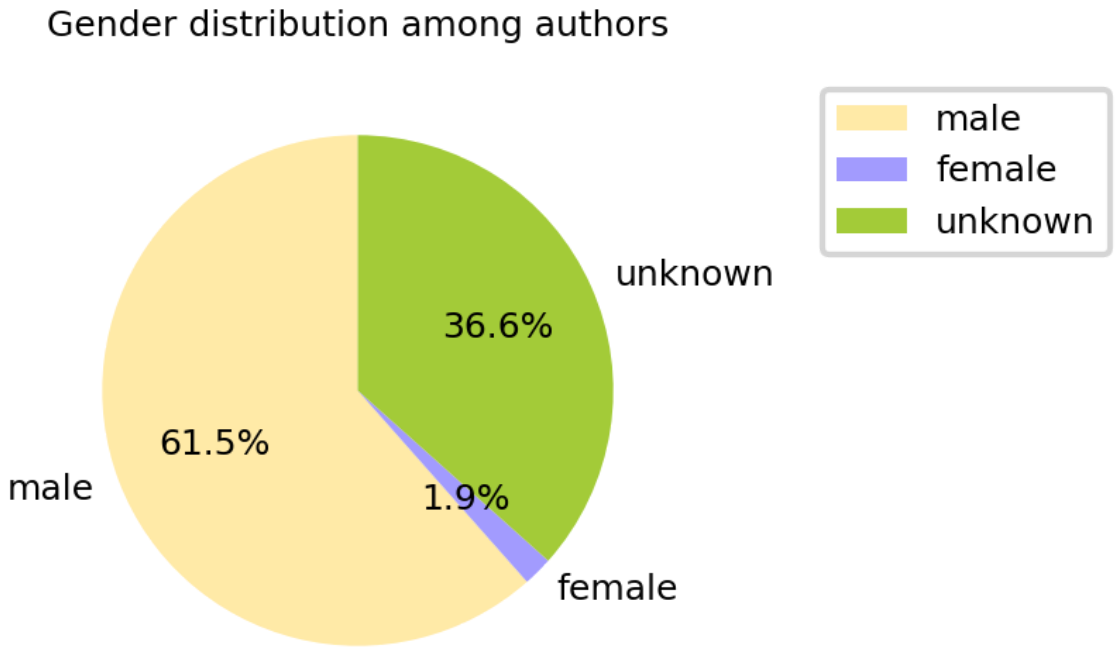
```
{'de Ramires, Maria',
 'Jacobs, Joannes',
 'Brouwer, Christophorus',
 'von Bayern, Ferdinand',
 'De Balinghem, Antonius',
 'Paludanus, Johannes',
 'Herman, Hugo',
 'Clenardus, Nicolaus',
 'de Petter, Hieronymus',
 'de Wannemaker, Philippus',
 'Divaeus, Petrus',
 'Joannes',
 'Junius, Cornelius',
 'Corenus, Jacobus',
 'de Casta?iza, Juan',
 'Bartholomaeus a Salutio.',
 'Vorsterman, Lucas Emil',
 'van de Casteele, Peeter',
```

In [198]:

```
#Find out gender distribution among all the author
firstname=[]
for i in unique_author:
    name = i.split(', ')
    firstname.append(name[-1])

gender = []
gender_detector= gender_guesser.detector.Detector()
for name in firstname:
    guess = gender_detector.get_gender(name)
    gender.append(guess)

gender_series = pd.Series(gender)
sum_male=sum(gender_series=='male')
sum_female=sum(gender_series=='female')
sum_unknown=sum(gender_series=='unknown')
category_names=['male','female','unknown']
sizes=[sum_male,sum_female,sum_unknown]
plt.figure(figsize=(2,2),dpi=300)
custom_colors=['#ffeaa7','#a29bfe','#A3CB38']
plt.pie(sizes,labels=category_names,textprops={'fontsize':6},startangle=90,colors=custom_colors,aut
plt.title("Gender distribution among authors",fontsize=6)
plt.legend(bbox_to_anchor=(1.2, 1),fontsize=6)
plt.show()
```



In [ ]:

```
# To find out the authors' social network
```

In [157]:

```
network_list = []
for i in authors:
    if len(i)>1:
        network_list.append(i)
network_list
```

...

In [159]:

```
relations=[]
for i in network_list:
    relations.append(list(itertools.combinations(i,2)))
relations
```

Out[159]:

```
[[('Alonso de Madrid', 'vanden Broecke, Franciscus')],
 [('Albrecht', 'Isabella Clara Eugenia')],
 [('Alonso de Madrid', 'Farzyn, Jacobus'),
  ('Alonso de Madrid', 'Henten, Johannes'),
  ('Farzyn, Jacobus', 'Henten, Johannes')],
 [('Gerlach Peters', 'van Gorcum, Jan'),
  ('Gerlach Peters', 'van Heese, Nicasius'),
  ('van Gorcum, Jan', 'van Heese, Nicasius')],
 [('Arnoldus ab Isca', 'vanden Calster, Anna.')],
 [('Albrecht', 'Isabella Clara Eugenia')],
 [('Bijns, Anna.', 'Pippinck, B. Henrick.')],
 [('Jordan, Raymondus', 'van Alen, Jan')],
 [('Godefridi, Petrus', 'Vervoort, Frans')],
 [('Florianus, Joannes.', 'Ovidius Naso, Publius')],
 [('Campi de Salutio, Bartholomeus', 'Van Blitterswyck, Jan')],
 [('Smising, Theodorus', 'von Hohenzollern-Sigmaringen, Eitel Friedrich')],
 [('Bonaventura', 'Spoelbergh, Guilielmus')],
 [('Bonaventura', 'Spoelbergh, Guilielmus')],
```

In [160]:

```
# Flatten the nested list
relations_list = []
for i in relations:
    for j in i:
        relations_list.append(j)
```

In [161]:

```
relations_count ={}
for i in relations_list:
    if i not in relations_count:
        relations_count[i]=1
    else:
        relations_count[i]+=1
relations_count
```

...

In [183]:

```
# sort relationship frequency
sort_relations_frequency = [(c,a) for a,c in relations_count.items()]
sort_relations_frequency.sort(reverse=True)
```

In [173]:

```
sort_relations_frequency
```

...

Here we can see that Albrecht and Isabella Clara Eugenia had a close connection in co-authoring a book

In [185]:

```
relation_network = [(a,c) for c,a in sort_relations_frequency if c>=3]
relation_network
```

```
 (('Gudelinus, Petrus Paulus', 'Gudelinus, Philippus'), 3),
 (('Gudelinus, Petrus', 'Gudelinus, Philippus'), 3),
 (('Gudelinus, Petrus', 'Gudelinus, Petrus Paulus'), 3),
 (('Gregorius', 'Sixtus'), 3),
 (('Gregorius', 'Rocca, Angelo'), 3),
 (('Gregorius', 'Ridolfi, Petrus'), 3),
 (('Gregorius', 'Joannes Diaconus Hymonides'), 3),
 (('Gregorius', 'Innocentius'), 3),
 (('Gregorius', 'Basa, Dominicus'), 3),
 (('Galle, Cornelius', 'Lipsius, Justus'), 3),
 (('De Haen, Willem', 'Jansenius, Cornelius'), 3),
 (('Clemens', 'Pius'), 3),
 (('Boonen, Jacobus', 'Zachmoorter, Michiel'), 3),
 (('Bonaventura', 'Thielmans, Cornelius'), 3),
 (('Bolswert, Bo?tius Adamsz.', 'Sucquet, Antonius'), 3),
 (('Bellarmino, Roberto', 'Sforza, Francesco'), 3),
 (('Baronius, Caesar', 'Spondanus, Henricus'), 3),
 (('Baronius, Caesar', 'Rubens, Peter Paul'), 3),
 (('Albrecht', 'Miraeus, Aubertus'), 3),
 (('Albrecht', 'Lessius, Leonardus'), 3)]
```

In [186]:

```
column_from = [i[0][0] for i in relation_network]
column_to = [i[0][1] for i in relation_network]
column_value = [i[1] for i in relation_network]
```

In [187]:

```
data={'FROM':column_from,
      'TO':column_to,
      'VALUE':column_value
    }
relation_df=pd.DataFrame(data)
```

In [188]:

```
G = nx.from_pandas_edgelist(relation_df,
                            source = 'FROM',
                            target = 'TO',
                            edge_attr = 'VALUE',
                            create_using=nx.Graph())
plt.figure(figsize=(10,10))
pos = nx.kamada_kawai_layout(G)
nx.draw(G, with_labels = True, node_color = 'skyblue',edge_cmap=plt.cm.Blues, pos=pos)
plt.show()
```

...

In [189]:

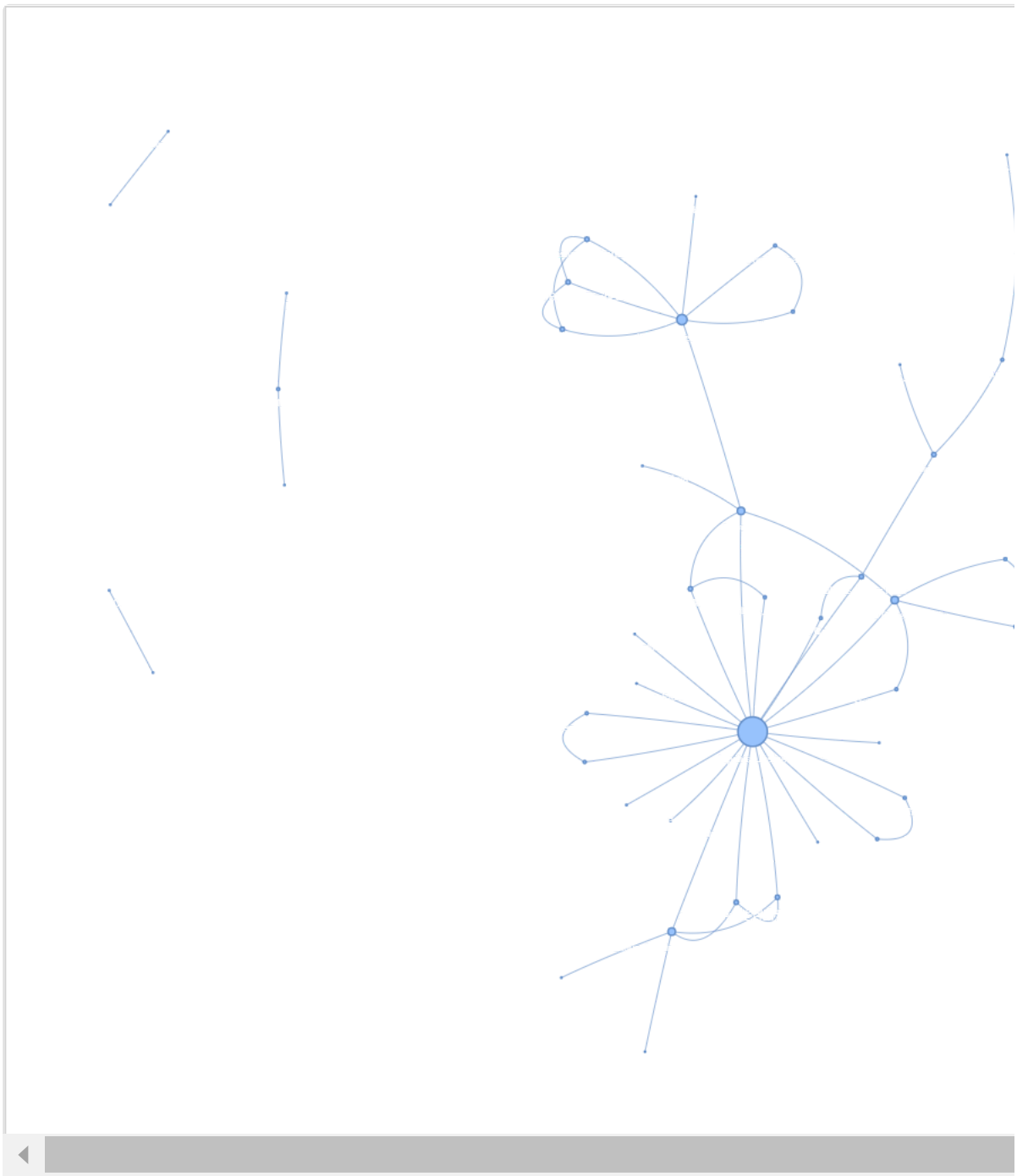
```
net = Network(notebook = True, width = '1000px',height = '700px',bgcolor='#222222',font_color='whit

node_degree = dict(G.degree)
nx.set_node_attributes(G,node_degree,'size')


net.from_nx(G)
net.show('author.html')
```

Local cdn resources have problems on chrome/safari when used in jupyter-notebook.

Out[189]:



```
Since
```

In [371]:

```
# the most important person
# degree centrality

degree_dict = nx.degree_centrality(G)
degree_dict
```

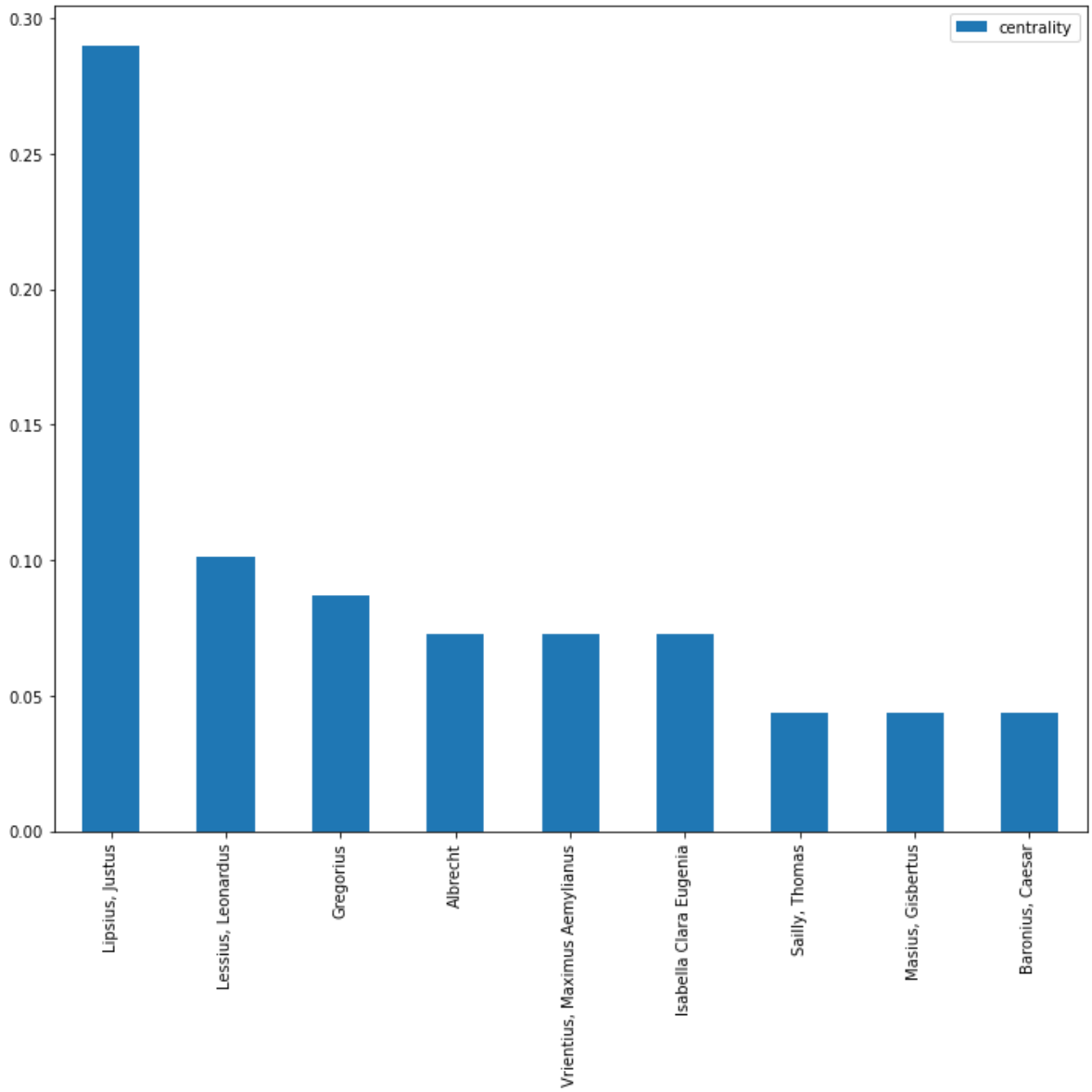
Out[371]:

{'Albrecht': 0.07246376811594203,
 'Isabella Clara Eugenia': 0.07246376811594203,
 'Lipsius, Justus': 0.2898550724637681,
 'Sweerts, Pierre Fran?ois': 0.043478260869565216,
 'Seneca, Lucius Annaeus': 0.014492753623188406,
 'Puteanus, Erycius': 0.028985507246376812,
 'David, Jan': 0.028985507246376812,
 'Galle, Theodoor': 0.028985507246376812,
 'Costerus, Franciscus': 0.014492753623188406,
 'Hovius, Matthias': 0.028985507246376812,
 'Oudaert, Nicolaus': 0.028985507246376812,
 'de Cro?, Charles': 0.014492753623188406,
 'Vrientius, Maximus Aemylianus': 0.07246376811594203,
 'Galle, Cornelius': 0.028985507246376812,
 'Rubens, Peter Paul': 0.043478260869565216,
 'Baronius, Caesar': 0.043478260869565216,
 'Rosweydus, Heribertus': 0.028985507246376812,
 'de Gouda, Joannes': 0.028985507246376812,
 'Lessius, Leonardus': 0.10144927536231885,
 'Thomas a Kempis': 0.028985507246376812,
 'Sommalius, Henricus': 0.014492753623188406,
 'Van den Wouwer, Jan': 0.028985507246376812,
 'Pighius, Stephanus': 0.028985507246376812,
 'Miraeus, Aubertus': 0.043478260869565216,
 'Fabricius, Gulielmus': 0.014492753623188406,
 'Beyerlinck, Laurentius': 0.028985507246376812,
 'Sailly, Thomas': 0.043478260869565216,
 'Masius, Gisbertus': 0.043478260869565216,
 'Hugo, Herman': 0.014492753623188406,
 'David, Joannes': 0.014492753623188406,
 'de Pretere, Guillaume': 0.014492753623188406,
 'Bonfrerius, Jacobus': 0.014492753623188406,
 'Hogius, Michael': 0.028985507246376812,
 'a Lapide, Cornelius Cornelii': 0.014492753623188406,
 'Zoes, Gerardus': 0.028985507246376812,
 'Zoes, Nicolaus': 0.028985507246376812,
 'Sucquet, Antonius': 0.043478260869565216,
 'zu dem Berch, Elisabeth': 0.043478260869565216,
 'Rodoan, Carolus Philippus': 0.014492753623188406,
 'Valerius Maximus.': 0.028985507246376812,
 'van Ravelingen, Frans': 0.014492753623188406,
 'Pr¨®chnicki, Jan Andrzej': 0.014492753623188406,
 'Moretus, Balthasar': 0.014492753623188406,
 'Lernutius, Janus': 0.043478260869565216,
 'Ignatius de Loyola': 0.028985507246376812,
 'La¨ªnez, Diego': 0.014492753623188406,
 'Aquaviva, Claudius': 0.014492753623188406,
 'Heinsius, Daniel': 0.014492753623188406,
 'Gudelinus, Petrus Paulus': 0.028985507246376812,
 'Gudelinus, Philippus': 0.028985507246376812,
 'Gudelinus, Petrus': 0.028985507246376812,

'Gregorius': 0.08695652173913043,
'Sixtus': 0.014492753623188406,
'Rocca, Angelo': 0.014492753623188406,
'Ridolfi, Petrus': 0.014492753623188406,
'Joannes Diaconus Hymonides': 0.014492753623188406,
'Innocentius': 0.014492753623188406,
'Basa, Dominicus': 0.014492753623188406,
'De Haen, Willem': 0.014492753623188406,
'Jansenius, Cornelius': 0.014492753623188406,
'Clemens': 0.014492753623188406,
'Pius': 0.014492753623188406,
'Boonen, Jacobus': 0.014492753623188406,
'Zachmoorter, Michiel': 0.014492753623188406,
'Bonaventura': 0.014492753623188406,
'Thielmans, Cornelius': 0.014492753623188406,
'Bolswert, Bo?tius Adamsz.': 0.014492753623188406,
'Bellarmino, Roberto': 0.014492753623188406,
'Sforza, Francesco': 0.014492753623188406,
'Spondanus, Henricus': 0.014492753623188406}

In [372]:

```
degree_df = pd.DataFrame.from_dict(degree_dict,orient = 'index',columns = ['centrality'])
degree_df.sort_values('centrality',ascending = False)[0:9].plot(kind = 'bar',figsize=(10, 10))
plt.tight_layout()
plt.savefig('degree.pdf',dpi = 300)
```

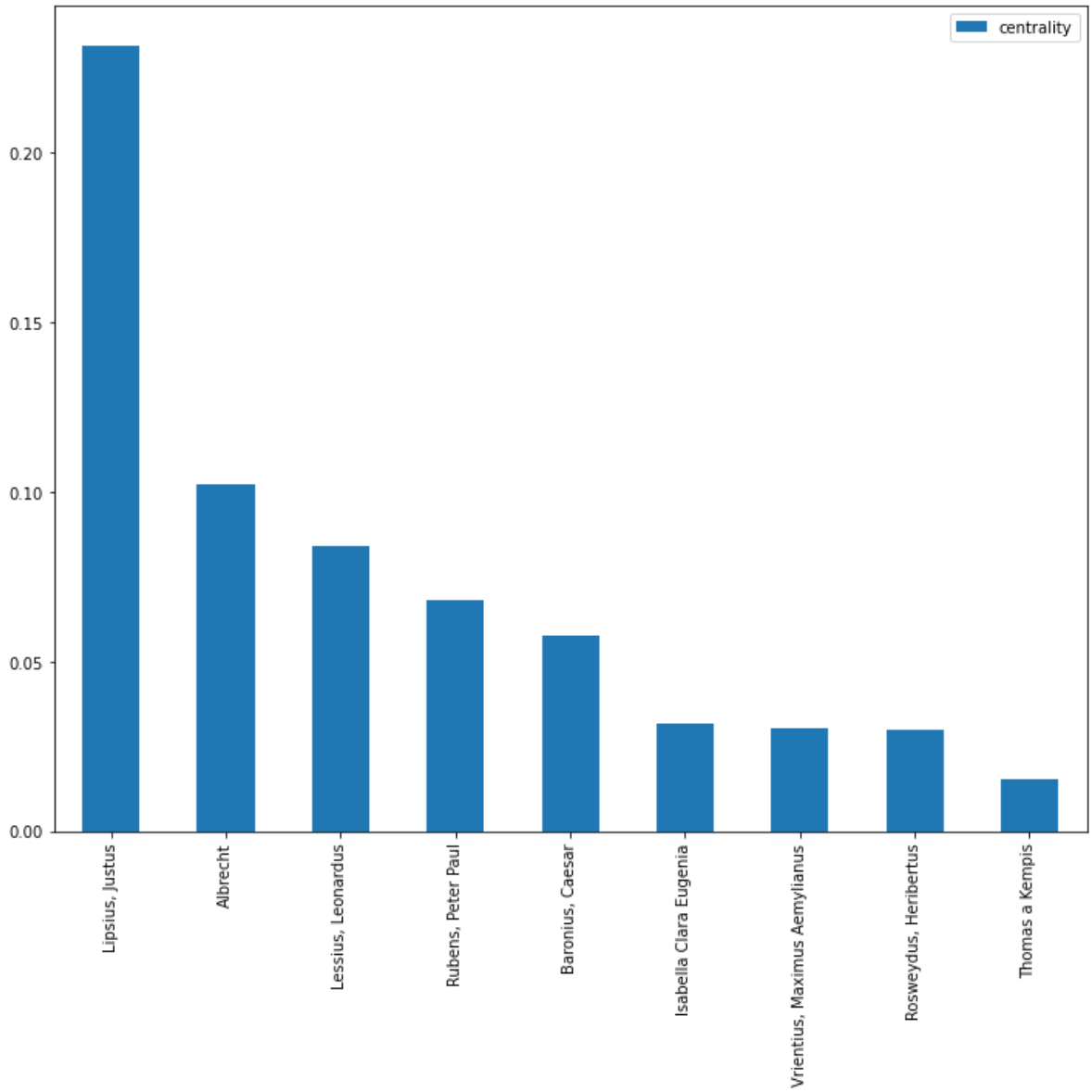


In [370]:

```
# Betweeness centrality
betweenness_dict = nx.betweenness_centrality(G)
betweenness_df = pd.DataFrame.from_dict(betweenness_dict,orient = 'index',columns = ['centrality'])

#Plot top 10 nodes
betweenness_df.sort_values('centrality',ascending = False)[0:9].plot(kind='bar',figsize=(10, 10))
plt.tight_layout()
plt.savefig('betweeness.pdf',dpi = 300)

## pretty much the same result
```

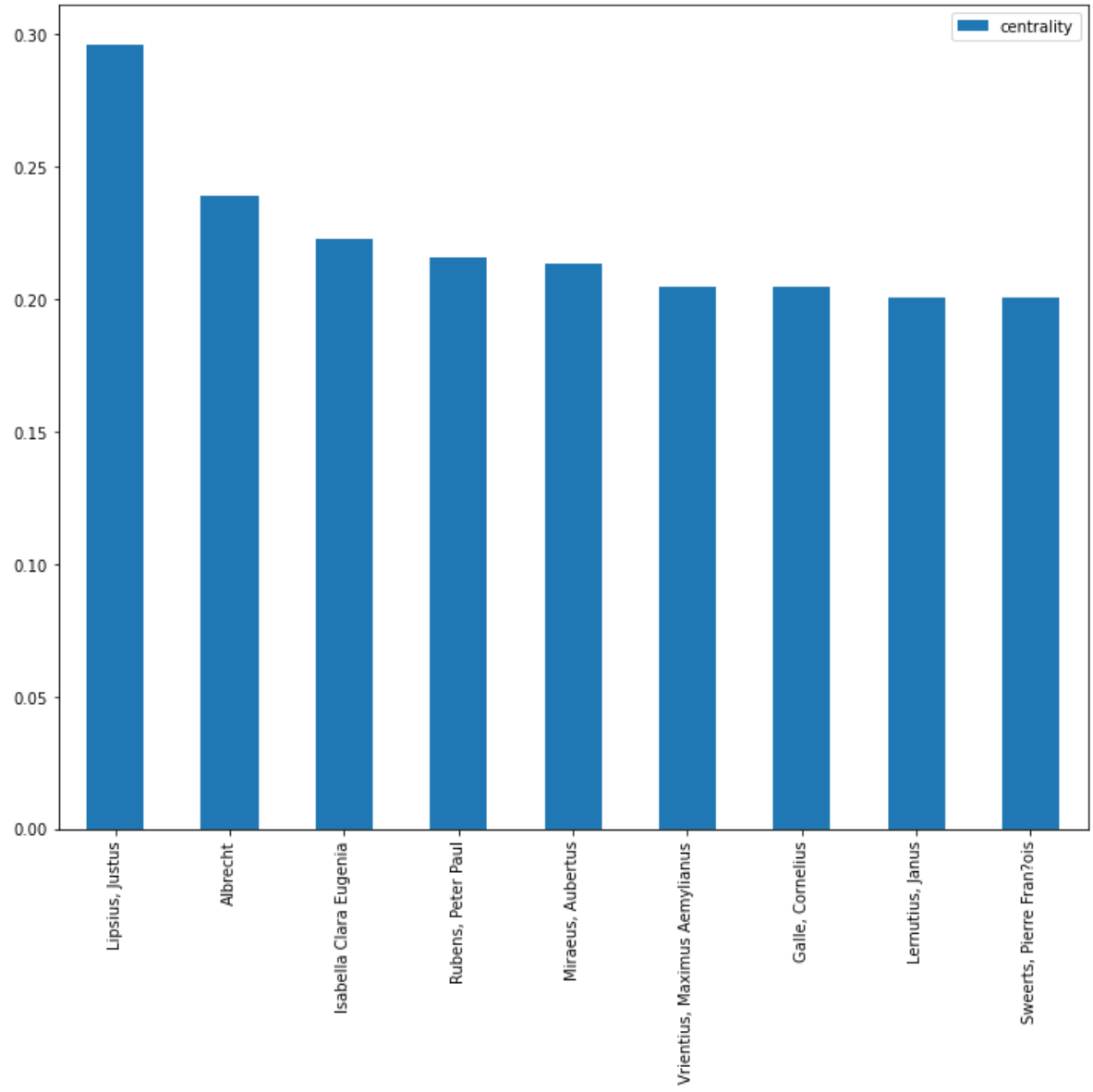


In [367]:

```
# Closeness centrality
closeness_dict = nx.closeness_centrality(G)
closeness_df = pd.DataFrame.from_dict(closeness_dict,orient = 'index',columns = ['centrality'])

#Plot top 10 nodes
closeness_df.sort_values('centrality',ascending = False)[0:9].plot(kind='bar',figsize=(10, 10))
plt.tight_layout()
plt.savefig('closeness.pdf',dpi = 300)


## pretty much the same result,but the gaps are smaller. this is a dense network
```



In [194]:

```
# Save centrality measures
nx.set_node_attributes(G, degree_dict,'degree_centrality')
nx.set_node_attributes(G, betweenness_dict,'betweenness_centrality')
nx.set_node_attributes(G, closeness_dict,'closeness_centrality')
```

In [195]:

```
communities = community_louvain.best_partition(G)
```

In [196]:

```
nx.set_node_attributes(G,communities,'group')
```

In [197]:

```
com_net =Network(notebook=True, width = '1000px',height = '700px',bgcolor='#222222',font_color='whi
com_net.from_nx(G)
com_net.show('author_communities.html')
```

Local cdn resources have problems on chrome/safari when used in jupyter-notebook.

Out[197]: