Mapping the spatial references in Pliny the Elder's Natural History through distant reading

Dawn, Lizao Zhuang (r0914937)

this is an abstract

0.1 Introduction

The *Natural History* stands as a monumental and comprehensive work compiled by the Roman writer, Pliny the Elder, during the first century AD. This vast compendium encompasses a wide range of subjects, including natural phenomena, geography, anthropology, zoology, botany, and much more. Pliny's extensive undertaking aimed to compile and organize the wealth of knowledge available at the time, showcasing the interconnectedness of the natural world and humanity's place within it (Gibson and Morello 2011).

At the heart of this remarkable work lies the presence of natural and human spaces, woven into the fabric of Pliny's discourse. These not only denote specific geographic locations but also encapsulate the cultural, historical, and societal contexts in which they exist. They serve as signposts within the vast expanse of *Natural History*, guiding readers through diverse lands and illuminating the diverse experiences and knowledge associated with each place.

On the other hand, the portrayal and depiction of the places, particularly those located far beyond the borders of the Roman Empire, can provide valuable insights into the ways in which Pliny the Elder and his contemporaries perceived and conceptualized the world around them.

In this regards, this analysis aims to map the physical locations mentioned in the books, also contributing to define their role and position in the network of interconnections within Pliny's work.

The exploration is conducted combining two different methodologies of Natural Language Processing, in order to have an overview of the discourse about "India" in *Natural History* of Pliny the Elder.

0.1.1 Description of the text (data)

original in Latin but a translation in English is used, extension of the text (n of books, n of tokens), contents, general scope: repository of knowledge; thematic structure by book.

The original text of *Natural History* was in Latin, a translation in English available in digital version provided by TOPOSText project with annotations of people's names, places' name and coordinated is used in the exploration.

There are 36 books of the encyclopedia in total, containing

Texts mentioning regions related to "India" in the context were extracted with a range of coordinates referring to Barrington Atlas of the Greek and Roman World by R. J.A. Talbert.

0.1.2 State-of-art

0.2 Research Question

0.2.1 Spatial perspective

0.2.2 Focus on India

0.2.3 Discourse about India in Natural History

What information is provided about India? What Indian places are described? How is India described? How is this information structured?

0.3 Methodology

Description of the workflow

0.4 Data preparation

0.4.1 Scrape text with geographical annotation

The text of the whole book has been digitized and annotated with people's name, places' name and coordinates by TOPOSText project since 2012. This invaluable resource allows for the creation of a dataset that includes both the textual contents and geographical annotations, which can be utilized to investigate the distribution of place names in the entire text and examine the frequencies and patterns of geographically-related content.

```
# link for digitized text of Natural History_book1-11
url1 = "https://topostext.org/work/148"

# link for digitized text of Natural History_book12-37
url2 = "https://topostext.org/work/153"
```

The geographical annotations can be parsed with functions available in Beautiful Soup library, and the first five returned annotations are shown as follows:

```
# check the place names with annotation in the first part of the digitized book

response = requests.get(url1)
soup = BeautifulSoup(response.content, features="lxml")

links = soup.find_all("a", {"class": "place"})

for link in links[:5]:
    print(link)
```

```
<a about="https://topostext.org/place/380237SAca" class="place" lat="37.992" long="23.707">A
<a about="https://topostext.org/place/419125LPal" class="place" lat="41.8896" long="12.4884">
<a about="https://topostext.org/place/419125LEsq" class="place" lat="41.895" long="12.496">Es
<a about="https://topostext.org/place/419125SCap" class="place" lat="41.8933" long="12.483">C
<a about="https://topostext.org/place/419125PRom" class="place" lat="41.891" long="12.486">Ro
```

With defining a function, all the texts with the geographical annotations can be parsed and stored as a dataframe, containing information in 8 columns as:

1. Unique ID assigned
2. ToposText_ID (which identifies the distinct location)
3. Place name
4. Reference (indicate where the place name occurs in the book)
5. Latitude
6. Longitude
7. Book number the place mentioned in
8. Chapter number the place mentioned in
9. Paragraph number the place mentioned in
10. Plain text of the paragraph where the place is mentioned

```
# function for text with geographical name annotation scraping
```

```
def toposgeotext(url):
    response = requests.get(url)
    if response.status_code != 200:
        raise FileNotFoundError("Failed to retrieve HTML content: " + url)

    data = []
    soup = BeautifulSoup(response.content, features="lxml")
    links = soup.find_all("a", {"class": "place"})

    for link in links:
        Place_Name = link.contents[0]  # Place name
        ToposText_ID = link.get('about')  # ToposText ID
        Lat = link.get('lat')
        Long = link.get('long')
        Parent = link.find_parent("p")
        Text = Parent.text  # Extract related text
        Reference = Parent.get("id")  # Indicate book, chapter, paragraph

        # Separate the information in Text using the regular expression pattern
        match = re.search(r'§\s+(\d+\.\d+\.\d*)\s+(.*)$', Text)
        if match:
            Chapternparagraph = match.group(1)  # Extract the reference from the pattern
            Text = match.group(2)  # Extract the remaining text from the pattern

            # Extract the book, chapter, and paragraph components
            book, chapter, paragraph = Chapternparagraph.split('.')

            UUID4 = uuid.uuid4()  # Create a unique ID

            data.append({
                'UUID4': UUID4,
                'ToposText_ID': ToposText_ID,
                'Place_Name': Place_Name,
                'Reference': Reference,
                'Lat': Lat,
                'Long': Long,
                'Book': book,  # Add the book component to the DataFrame
                'Chapter': chapter,  # Add the chapter component to the DataFrame
                'Paragraph': paragraph,  # Add the paragraph component to the DataFrame
                'Text': Text
            })
```

```
    df = pd.DataFrame(data)
    # Convert data types to numeric
    df['Book'] = pd.to_numeric(df['Book'], errors='coerce')
    df['Chapter'] = pd.to_numeric(df['Chapter'], errors='coerce')
    df['Paragraph'] = pd.to_numeric(df['Paragraph'], errors='coerce')
    return df

# construct the dataframe for two parts of the digitized text with geographical annotation
geodf1 = toposgeotext(url1)
geodf2 = toposgeotext(url2)

geotext_whole = pd.concat([geodf1, geodf2], ignore_index=True)

geotext_whole.to_csv('geotext_whole.csv')

geotext_whole.head()
```

	UUID4	ToposText_ID	Place_Name	Refer
0	f8bbe55f-283c-49a2-ac89-35e0b24b46fe	https://topostext.org/place/380237SAca	Academy	urn:c
1	a804496a-f5b4-4359-abfe-a19f43ca58ba	https://topostext.org/place/419125LPal	Palatine	urn:c
2	2d69bd88-f0f7-411c-9261-989e4575cdcc	https://topostext.org/place/419125LEsq	Esquiline	urn:c
3	3e44e671-8506-46b5-9d1f-6bb9179427bd	https://topostext.org/place/419125SCap	Capitol	urn:c
4	18d4d160-548b-4f2e-a2ba-481cbe7b29c3	https://topostext.org/place/419125PRom	Rome	urn:c

There are 5595 locations mentioned in book 1-11 and 3281 locations mentioned in book 12-37. The combined dataframe for the whole book, has the shape of (8876, 10). And the output has been stored as .csv for record.

0.4.2 Scrape text of the entire book

The text of the entire book is also scraped as a reference.

```
# function for scraping entire text from ToposText with given html

def topostext(url):
    response = requests.get(url)
    if response.status_code != 200:
        raise FileNotFoundError("Failed to retrieve HTML content: " + url)
```

```
    data = []
    soup = BeautifulSoup(response.content, features="lxml")
    links = soup.find_all("p")  # Find all <p> tags instead of filtering by class

    for link in links:
        match = re.search(r'§\s+(\d+\.\d+\.\d*)\s+(.*)$', link.text)
        if match:
            Chapternparagraph = match.group(1)  # Extract the reference from the pattern
            # Extract the book, chapter, and paragraph components
            book, chapter, paragraph = Chapternparagraph.split('.')
            Text = match.group(2)  # Extract the text from the pattern
            Reference = link.get("id")  # Indicate book, chapter, paragraph
            UUID4 = uuid.uuid4()  # Create a unique ID

            data.append({
                'UUID4': UUID4,
                'Reference': Reference,
                'Book': book,  # Add the book component to the DataFrame
                'Chapter': chapter,  # Add the chapter component to the DataFrame
                'Paragraph': paragraph,  # Add the paragraph component to the DataFrame
                'Text': Text
            })

    df = pd.DataFrame(data)
    # Convert data types to numeric
    df['Book'] = pd.to_numeric(df['Book'], errors='coerce')
    df['Chapter'] = pd.to_numeric(df['Chapter'], errors='coerce')
    df['Paragraph'] = pd.to_numeric(df['Paragraph'], errors='coerce')
    return df

# construct the dataframe for two parts of the digitized text with the topostext function
df1 = topostext(url1)
df2 = topostext(url2)

# combine the two parts of scraped text
wholebook = pd.concat([df1, df2], ignore_index=True)

# store the sparsed text into csv file
wholebook.to_csv('wholebooktext.csv')

wholebook.head()
```

	UUID4	Reference	Book	Chapter	Paragrap
0	9dc3ebec-2788-41d5-aae7-c5822472d873	urn:cts:latinLit:phi0978.phi001:1.1.1	1	1	1.0
1	5a3b400b-e2fe-42e0-b8e8-425b69257ae0	urn:cts:latinLit:phi0978.phi001:1.2.1	1	2	1.0
2	3e6bdc07-568b-43ae-8ebd-537355f33d4f	urn:cts:latinLit:phi0978.phi001:1.3.1	1	3	1.0
3	88f9960b-bf90-40da-b140-48344acaf23e	urn:cts:latinLit:phi0978.phi001:1.4.1	1	4	1.0
4	95522a01-f6dd-43ea-94b0-82eacc2b3a8f	urn:cts:latinLit:phi0978.phi001:1.5.1	1	5	1.0

0.5 Data Analysis

0.5.1 Word frequency

0.5.2 Network analysis for Named Entity

0.5.3 Topic modelling

The combined dataframe for texts in the whole book has the shape of (3493, 6). And the output has been stored as .csv for record.

0.6 Conclusions

0.7 Old structure

0.7.1 Overview of geographical related texts

What topics popped up from the context of place names?

0.7.1.1 Distribution of place names in the entire book

The normalized frequency of place name references in *Natural History* was calculated as the ratio of counts of the occurrences of place names in each book to the word lengths of the book (Table 3). As depicted in Figure 1, the findings indicate that books 3-6 prominently feature a higher frequency of place name references. This observation is consistent with content structure of *Natural History*, that books 3-6 centered around the themes of **"Geography and ethnography"**, is expected to contain a great number of location references.

```
# Group the items by "Book" and calculate the total word length

book_word_lengths = wholebook.groupby('Book')['Text'].apply(lambda x: x.str.split().str.le
book_word_lengths.columns = ['Book', 'Total_length']
```

```
# Set the "Book" column as the index
book_word_lengths.set_index('Book', inplace=True)

place_counts = geotext_whole['Book'].value_counts().reset_index()
place_counts.columns = ['Book', 'Place_count']
place_counts.set_index('Book', inplace=True)

place_distribution = book_word_lengths.merge(place_counts, on='Book')
place_distribution['Place_freq'] = place_distribution['Place_count']/place_distribution['T

place_distribution
```

Table 3: Distribution of place names in Natural History

Book	Total_length	Place_count	Place_freq
1	2778	1	0.000360
2	30570	406	0.013281
3	18037	1007	0.055830
4	15434	1309	0.084813
5	18872	1112	0.058923
6	27890	1012	0.036285
7	21204	225	0.010611
8	24176	185	0.007652
9	19197	140	0.007293
10	20816	121	0.005813
11	27345	77	0.002816
12	13906	188	0.013519
13	13243	164	0.012384
14	15277	189	0.012372
15	14552	135	0.009277
16	25442	180	0.007075
17	29387	82	0.002790
18	35850	222	0.006192
19	18822	146	0.007757
20	22743	21	0.000923
21	17896	95	0.005308
22	16491	24	0.001455
23	15764	17	0.001078
24	17491	56	0.003202
25	16734	85	0.005079

Book	Total_length	Place_count	Place_freq
26	15448	35	0.002266
27	12444	40	0.003214
28	26476	28	0.001058
29	13976	31	0.002218
30	14395	23	0.001598
31	12204	222	0.018191
32	14635	76	0.005193
33	17946	113	0.006297
34	18972	193	0.010173
35	21282	277	0.013016
36	21295	357	0.016764
37	22255	282	0.012671

```
fig, ax = plt.subplots(figsize=(10, 3))  # Adjust the figsize to increase the plot size

place_distribution['Place_freq'].plot.bar(ax=ax)
plt.xlabel('Book')
plt.ylabel('Frequency')

# Adjust the plot alignment
plt.tight_layout(rect=[0, 0, 0.8, 1])  # Change the rect values to adjust the alignment

plt.show()
```

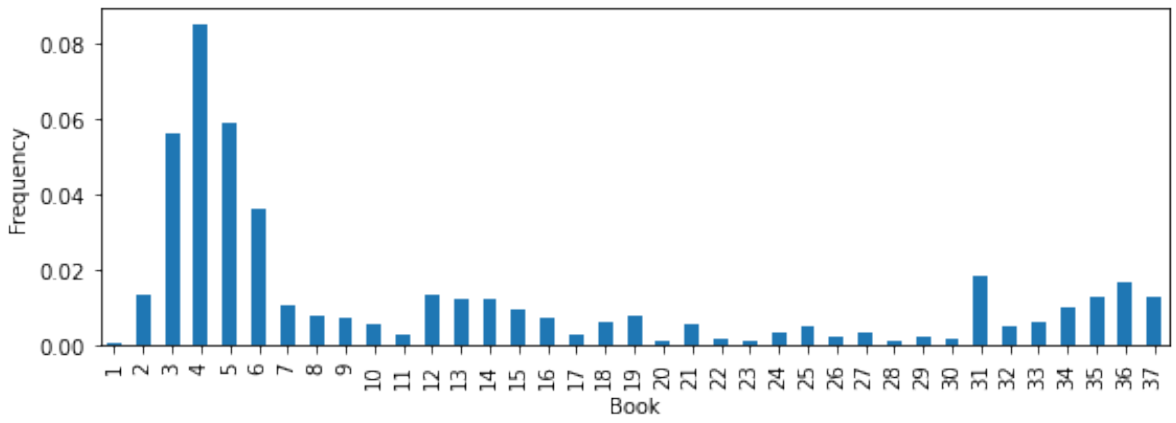

Figure 1: Place name distribution in Natural History

0.7.1.2 Topic modelling on geographical location related text

Genism library is used for semantic vectorization and implemetion of Latent Dirichlet Allocation (LDA) model for the topic modelling in the captioned text.

And the library of pyLDAvis is applied for an interactive visualization.

```
from nltk.tokenize import sent_tokenize, word_tokenize
import string
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

def tokenize_paragraphs(texts):
    ## apply word tokenization function of NLTK module to each paragraph
    words = [word_tokenize(text, language='English') for text in texts]


    ## set list of stop words to be excluded from the tokenlist with stopwords function in
    stop_words = set(stopwords.words('English'))

    lemmatizer = WordNetLemmatizer()

    ## append words of text into a tokenlist
    tokenlistpara = []
    for paragraph in words:
        paragraph_tokens = []
        for token in paragraph:
            if token.lower() and token.lower() not in stop_words and len(token.lower()) >
                ## lemmatize the tokens
                lemmatized_token = lemmatizer.lemmatize(token.lower())
                paragraph_tokens.append(lemmatized_token)
        tokenlistpara.append(paragraph_tokens)
    return tokenlistpara


para = tokenize_paragraphs(geotext_whole['Text'].drop_duplicates())

## import the necessary modules

from gensim import corpora, models
import gensim

# turn our tokenized documents into a id <-> term dictionary
```

```
paradictionary = corpora.Dictionary(para)

# convert tokenized documents into a document-term matrix
paracorpus = [paradictionary.doc2bow(text) for text in para]

# generate LDA model with tokenlist in paragraphs
paraldamodel = gensim.models.ldamodel.LdaModel(paracorpus, num_topics=5, id2word = paradic

## display the assigned 5 topics and 30 words in each topic
from pprint import pprint
pprint(paraldamodel.print_topics(num_topics=5, num_words=30))
```

```
[(0,
  '0.010*"also" + 0.004*"picture" + 0.003*"painted" + 0.003*"milk" + '
  '0.003*"sponge" + 0.003*"first" + 0.003*"dung" + 0.003*"bird" + 0.002*"egg" '
  '+ 0.002*"made" + 0.002*"time" + 0.002*"horse" + 0.002*"year" + '
  '0.002*"caesar" + 0.002*"painting" + 0.002*"one" + 0.002*"goat" + '
  '0.002*"said" + 0.002*"two" + 0.002*"called" + 0.002*"give" + 0.002*"boy" + '
  '0.002*"onion" + 0.002*"among" + 0.002*"day" + 0.002*"great" + 0.002*"sheep" '
  '+ 0.002*"make" + 0.002*"famous" + 0.001*"wine"'),
 (1,
  '0.018*"also" + 0.011*"kind" + 0.007*"called" + 0.007*"stone" + 0.007*"like" '
  '+ 0.006*"wine" + 0.006*"colour" + 0.006*"leaf" + 0.006*"one" + '
  '0.005*"plant" + 0.005*"tree" + 0.005*"used" + 0.005*"water" + 0.005*"found" '
  '+ 0.005*"white" + 0.005*"root" + 0.004*"taken" + 0.004*"made" + '
  '0.004*"variety" + 0.004*"oil" + 0.004*"name" + 0.004*"seed" + 0.004*"black" '
  '+ 0.003*"make" + 0.003*"grows" + 0.003*"even" + 0.003*"honey" + '
  '0.003*"juice" + 0.003*"said" + 0.003*"two"'),
 (2,
  '0.007*"called" + 0.007*"also" + 0.007*"island" + 0.006*"day" + 0.006*"sea" '
  '+ 0.006*"one" + 0.005*"name" + 0.005*"place" + 0.005*"river" + 0.004*"wind" '
  '+ 0.004*"mile" + 0.004*"city" + 0.004*"soil" + 0.004*"time" + 0.003*"year" '
  '+ 0.003*"district" + 0.003*"earth" + 0.003*"land" + 0.003*"people" + '
  '0.003*"sun" + 0.003*"country" + 0.003*"part" + 0.003*"great" + 0.003*"two" '
  '+ 0.003*"italy" + 0.003*"nation" + 0.003*"region" + 0.002*"spring" + '
  '0.002*"first" + 0.002*"distance"'),
 (3,
  '0.016*"river" + 0.016*"mile" + 0.013*"town" + 0.011*"called" + 0.009*"sea" '
  '+ 0.009*"name" + 0.007*"distance" + 0.006*"water" + 0.006*"also" + '
  '0.006*"city" + 0.005*"island" + 0.005*"come" + 0.005*"two" + '
  '0.005*"hundred" + 0.004*"gulf" + 0.004*"upon" + 0.004*"place" + '
  '0.004*"formerly" + 0.004*"promontory" + 0.004*"people" + 0.004*"coast" + '
```

```
  '0.004*"one" + 0.003*"part" + 0.003*"nation" + 0.003*"mountain" + '
  '0.003*"side" + 0.003*"lie" + 0.003*"length" + 0.003*"distant" + '
  '0.003*"mouth"'),
 (4,
  '0.010*"also" + 0.008*"even" + 0.007*"one" + 0.005*"made" + 0.004*"first" + '
  '0.004*"year" + 0.004*"time" + 0.003*"rome" + 0.003*"man" + 0.003*"king" + '
  '0.003*"people" + 0.003*"work" + 0.003*"statue" + 0.003*"day" + 0.003*"tree" '
  '+ 0.003*"place" + 0.003*"used" + 0.003*"name" + 0.003*"great" + '
  '0.003*"gold" + 0.003*"temple" + 0.002*"among" + 0.002*"men" + 0.002*"case" '
  '+ 0.002*"animal" + 0.002*"many" + 0.002*"two" + 0.002*"called" + '
  '0.002*"although" + 0.002*"life"')]
```

The text data undergoes tokenization and lemmatization using functions from the NLTK package. This preprocessing step aims to obtain meaningful words that facilitate the inference of potential topics based on grouped keywords. To ensure the modeling results consist of words with descriptive meaning, stop words in English are excluded, along with tokens having a length less than 2, when preparing the corpus for input into the LDA module.

After several tryouts, the number of topics is set to 5, and the passes is set to 20, in order to generate distinct and non-overlapping topic clusters.

The following visualization presents the top 30 keywords for each topic, along with their respective weights, which rank their contributions to the topic.

```
# label: fig-geotexttm
# fig-cap: Topic modelling of place names related text

import pyLDAvis
import pyLDAvis.gensim_models

# Convert the LdaModel object to a pyLDAvis-compatible format
vis_datapara = pyLDAvis.gensim_models.prepare(paraldamodel, paracorpus, paradictionary, R=

# Enable the Jupyter notebook inline display of visualizations
pyLDAvis.enable_notebook()

# Display the pyLDAvis visualization with adjusted size
pyLDAvis.display(vis_datapara)
```

```
<IPython.core.display.HTML object>
```

In the left panel of the above interactive chart, each bubble represents a topic, and the size of the bulbble indicates the percentage of the texts in the corpus contributing to the topic. The

distance between the bubbles implies the extent of difference between them. And a good topic model is expected to have big and non-overlapping bubbles scattered throughout the chart (Tran 2022).

And in the right panel, the blue bars represent the overall frequency of each word in the corpus. If no topic is selected, the blue bars of the most frequently used words will be displayed. When hovering on the bubbles in the left panel, there will be red bars in the right panel giving the estimated number of times a given term was generated by a given topic. The word with the longest red bar is estimated to be used the most in the texts belonging to that topic.

An intriguing observation about the overall result of the topic modelling is that the word "also" comprises a large protion in the given text, and appears in all assigned topics. Taking the encyclopedia scope of Natural History into consideration, it may imply that the place names are prone to be mentioned in a context of enumerason and comparison. In the literary studies by Pollard (2009) and Murphy (2003), Pliny gave a critical description of the geographical surroundings and their exotic counterparts (e.g., Po River and Nile River), which may confirms it worthwhile getting a deeper exploration in the usage and reference of the place names in Natual History in order to map the scope and vision he attempted to display in the encyclopedia by Pliny the Elder.

More specifically, a rough generalization can be drown for each topic with the dominant words in it as follows, which may help to conclude the themes and keywords for geography related context in *Natural History*.

Topic 1: **Artistic Elements and Objects** - The presence of paintings, milk, sponges, and other objects adds to the artistic and visual aspects of the context.

Topic 2: **Botanical and Natural Elements** - Various plants, trees, colors, and natural materials contribute to the botanical richness depicted in the book.

Topic 3: **Geographic Features and Places** - Islands, rivers, cities, and other geographical features play a significant role in the narrative, highlighting the diverse landscapes explored in the text.

Topic 4: **Distance and Proximity** - Distances, towns, rivers, and seas provide insights into the spatial relationships and navigational aspects within the book.

Topic 5: **Historical and Cultural References** - Roman history, statues, temples, and notable figures showcase the historical and cultural context prevalent in the book.

In addition, as shown in the visualization chart, the Topic 5: **Historical and Cultural References** and Topic 2: **Botanical and Natural Elements** seem to be the most prominent topics about geographical location related text in *Natural History*.

In conclusion, the general exploratory analysis about geographical location related text in *Natural History* shows that in the books about geography and ethnography, and mining and mineralogy, place names are most frequently referred. And the potential topics about geograpical location related contents are "Artistic Elements and Objects", "Geographic Features and

Places", "Distance and Proximity", "Historical and Cultural References" and "Botanical and Natural Elements", with the latter two as the most prominent topics in the context.

Considering the comprehensive scope of *Natural History*, the presence of concrete place names provides a valuable opportunity to delve deeper into Pliny the Elder's perception and imagination of landscapes. Therefore, it is worthwhile to embark on a more detailed examination of the distribution, significance, and contextualization of place names in *Natural History* to gain insights into how Pliny the Elder crafted the narrative and conveyed his understanding of the world.

0.7.2 Prominent location mentioned in Natural History

What place stands out in the narrative? And how does it align with the scope and underlying concept of *Natural History*?

0.7.2.1 Place name distribution

By grouping the "ToposText_ID" (as indicator for distinct geographical loactions in the text) in the earlier constructed dataframe, there are 2052 unique places mentioned in *Natural History*.

The top 20 most frequent place names mentioned (as 1% of total) in *Natural History* is shown in Table 4.

```
## sort the top 1% places referred in descending order

place = geotext_whole.groupby('ToposText_ID')[['Place_Name','Lat','Long']].value_counts().
sorted_place = place.sort_values(by='Count', ascending=False)
top_count = sorted_place.iloc[19]['Count']
topplace = sorted_place[sorted_place['Count'] >= top_count]
topplace
```

Table 4: Top 20 mentioned place names in Natural History

	ToposText_ID	Place_Name	Lat	Long	Count
1687	https://topostext.org/place/406163RIta	Italy	40.6	16.3	292
2034	https://topostext.org/place/419125PRom	Rome	41.891	12.486	269
52	https://topostext.org/place/271307REgy	Egypt	27.1	30.7	261
82	https://topostext.org/place/300740RInd	India	30	74	167
57	https://topostext.org/place/280400RAra	Arabia	28	40	123
320	https://topostext.org/place/355390RSyr	Syria	35.5	39	109
255	https://topostext.org/place/350330RCyp	Cyprus	35	33	85

	ToposText_ID	Place_Name	Lat	Long	Count
109	https://topostext.org/place/312301WNil	Nile	30.0918	31.2313	85
2282	https://topostext.org/place/441073LAlp	Alps	44.142	7.343	82
766	https://topostext.org/place/376145RSic	Sicily	37.6	14.5	71
275	https://topostext.org/place/352252IKre	Crete	35.2052	25.1836	64
7	https://topostext.org/place/130350REth	Ethiopia	13.01	35.01	58
417	https://topostext.org/place/364282IRho	Rhodes	36.4408	28.2244	56
966	https://topostext.org/place/380237PAth	Athens	37.9718	23.72793	56
2043	https://topostext.org/place/419125SCap	Capitol	41.8933	12.483	52
298	https://topostext.org/place/353403WEup	Euphrates	35.2791	40.2708	47
2241	https://topostext.org/place/435335WPon	Pontus	43.5	33.5	47
1839	https://topostext.org/place/411146RCam	Campania	41.1	14.6	46
1480	https://topostext.org/place/397443RArm	Armenia	39.702	44.298	45
17	https://topostext.org/place/195390WEry	Red Sea	19.5	39	42
545	https://topostext.org/place/369103PCar	Carthage	36.85	10.32	42
602	https://topostext.org/place/370340RCil	Cilicia	37.01	34.01	42

The place names referenced in Natural History are geographically mapped, with each location marked on the map using its corresponding coordinates. A dot is assigned to represent each place, with the size and color of the dot reflecting the frequency of its mention in the book. The larger and darker the dot, the more frequently the place is referenced within the context of *Natural History*.

An intriguing observation from the output, as depicted in Figure 2, is the prominence of India—a region outside the Mediterranean—despite its high frequency of mentions.

```
import folium
from branca.colormap import LinearColormap

# Create a Folium map centered on a specific location
map = folium.Map(location=[32, 53], zoom_start=3)

# Define the minimum and maximum count values for the color scale
min_count = place['Count'].min()
max_count = place['Count'].max()

# Create a color map based on the count values
color_map = LinearColormap(colors=['#FFCC00', '#FF6600'],
                           vmin=min_count,
                           vmax=max_count)
```

```
# Iterate over each row in the dataframe
for index, row in place.iterrows():
    place_name = row['Place_Name']
    count = row['Count']
    latitude = row['Lat']
    longitude = row['Long']

    # Get the color based on the count using the color map
    color = color_map(count)

    # Create a circle marker with a size based on the count and color based on the count
    folium.CircleMarker(
        location=[latitude, longitude],
        radius=count/10,  # Adjust the size of the marker based on the count
        weight=0,
        fill=True,
        fill_color=color,
        fill_opacity=1,
        tooltip=f'{place_name}: {count} counts'
    ).add_to(map)

# Add the color scale to the map
color_map.caption = 'Count'
color_map.add_to(map)

# Display the map
map
```

```
<folium.folium.Map at 0x17c69eb4490>
```

Figure 2: Place name distribution map

0.7.2.2 Zooming into "India"

As highlighted in the research conducted by Nappo (2017), the era of Pliny the Elder's writing of *Natural History* witnessed a thriving Indo-Roman trade relationship. The prominence of the term "India" within the text suggests that this trade connection holds considerable significance in the narrative of *Natural History*.

To provide more comprehensive contextual analysis, the focus is extended beyond solely "India" to the regions that encompass the empires of the Indian subcontinent. The approximate range

of coordinates defining the target region is as follows:[1]

Latitude: Northernmost point: Approximately 37.6 degrees North (located in the region of Jammu and Kashmir in India) Southernmost point: Approximately 5.5 degrees North (located in the region of Dondra Head in Sri Lanka)

Longitude: Westernmost point: Approximately 60.9 degrees East (located in the region of Gwadar in Pakistan) Easternmost point: Approximately 97.4 degrees East (located in the region of Kibithu in India)

And a dataframe for Indian subcontinent related texts can be filetered with the captioned coordinates range.

```
# convert the 'Lat' and 'Long' information in the original dataframe to numeric format
geotext_whole['Lat'] = geotext_whole['Lat'].astype(float)
geotext_whole['Long'] = geotext_whole['Long'].astype(float)

# define the filter range of latitude and longitude ranges for Indian subcontinent
lat_range = (5.5, 37.6)
long_range = (60.9, 97.4)

# create a boolean mask for filtering
mask = (geotext_whole['Lat'].between(*lat_range)) & (geotext_whole['Long'].between(*long_r

# apply the mask to filter the dataframe
geotext_india = geotext_whole[mask]
geotext_india.to_csv('geotext_indianregion.csv')

geotext_india.head()
```

	UUID4	ToposText_ID	Place_Name	Refe
85	30629c68-c5a2-45ec-9c6e-ae2e0ceaecab	https://topostext.org/place/300740RInd	India	urn:
92	958f3c3d-bbfc-4488-8009-0185494ace05	https://topostext.org/place/300740RInd	India	urn:
93	11517672-3e13-4fcc-ac4a-f7ce58f814a7	https://topostext.org/place/300740RInd	India	urn:
218	617dbd2d-e1cf-4858-91b4-ed6a959c9cee	https://topostext.org/place/254683WInd	Indus	urn:
326	c2dad95b-6687-4cca-9482-4c449fafdd88	https://topostext.org/place/340670RBac	Bactria	urn:

The shape of the filtered dateframe for texts and place coordinates related to Indian subcontinent is (241, 10). And the dataframe is also saved as .csv for further reference.

[1]Given the challenges in determining the precise coordinates of the Empires in the Indian region during the 1st century AD, an approximate range of coordinates for the current Indian subcontinent is used as a rough estimation.

And the places refered in the captioned region in the data frame are: [‘India’ ‘Indus’ ‘Bactria’ ‘Ganges’ ‘Acesinus’ ‘Oxus’ ‘Hydaspes’ ‘Taprobane’ ‘Arachosia’ ‘Muziris’ ‘Baragaza’ ‘Aria’ ‘Ceylon’].

The comparison between the total number of place names and the place names specifically related to the Indian subcontinent mentioned in each book, is depicted in Figure 3. The difference in numbers between the two categories is significant, as indicated by the large disparity.

To facilitate a more effective comparison of the referencing trends across different books, Figure 4 presents subplots with varying y-axis scales. This approach allows for a clearer visualization of the trends and patterns in place name references throughout the various books.

```
import matplotlib.pyplot as plt

# Compute the grouped count of place names in the current DataFrame
allplacecount = geotext_whole.groupby('Book')['Chapter'].count()

# Compute the grouped count of place names in the other DataFrame
indianplacecount = geotext_india.groupby('Book')['Place_Name'].count()

# Combine the counts into a single DataFrame
combined_counts = pd.concat([allplacecount, indianplacecount], axis=1)
combined_counts.columns = ['All Place Names', 'Place Names of Indian subcontinent']
combined_counts.fillna(0, inplace=True)

fig, ax = plt.subplots(figsize=(10, 8))

# Plot the combined counts as a bar chart with different colors
combined_counts.plot.bar(color=['blue', 'orange'], ax=ax)


# Customize the plot
ax.set_xlabel('Book')
ax.set_ylabel('Count')

# Show the plot
plt.show()
```

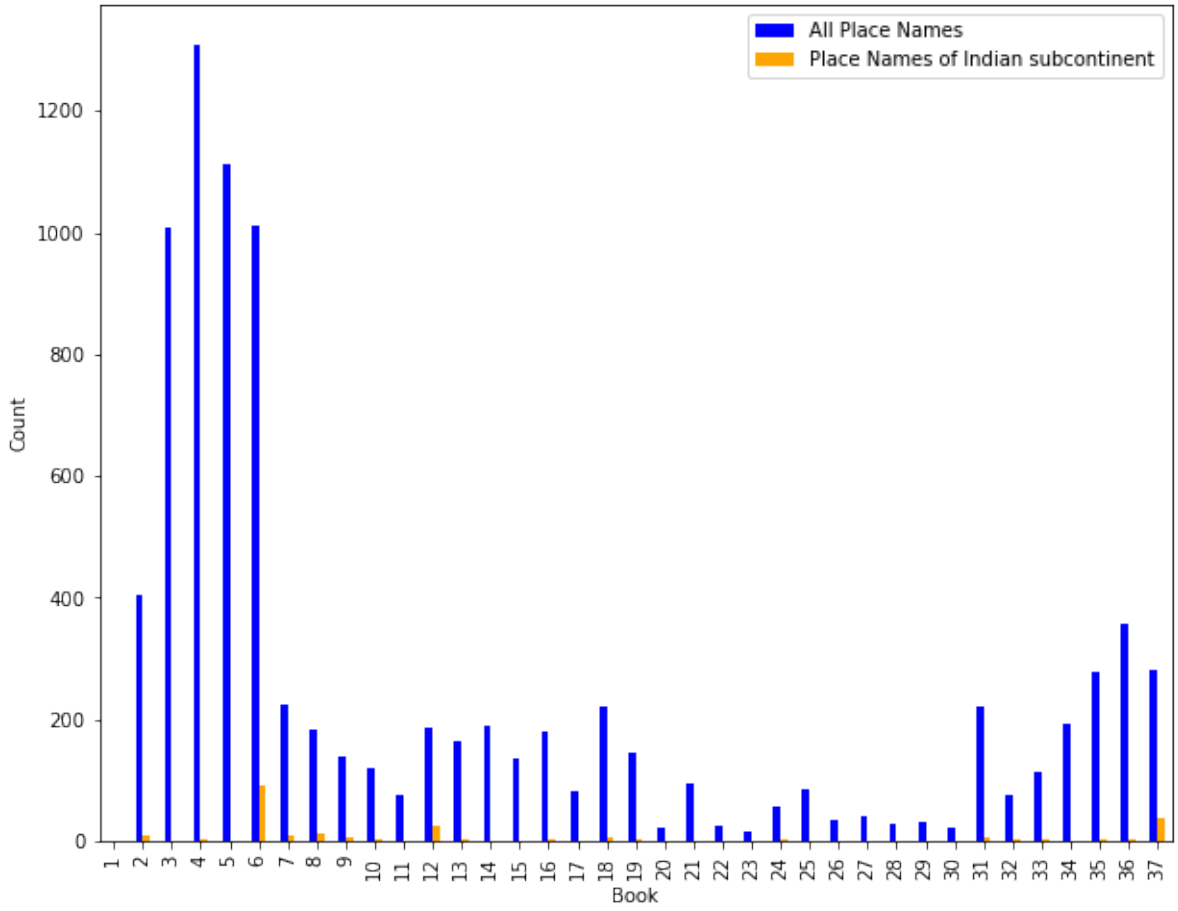

Figure 3: Occurence count for all place names and place names of Indian subcontinent in each book

```
# Set up the figure and axes for the subplots
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(18, 6))

# Plot the bar chart for 'All Place Names' on the first subplot
combined_counts['All Place Names'].plot(kind='bar', ax=ax1, color='blue', alpha=0.7)
ax1.set_xlabel('Book')
ax1.set_ylabel('Count')
ax1.set_title('Count Trend - All Place Names')

# Plot the bar chart for 'Other Place Names' on the second subplot
combined_counts['Place Names of Indian subcontinent'].plot(kind='bar', ax=ax2, color='oran
ax2.set_xlabel('Book')
```

```
ax2.set_ylabel('Count')
ax2.set_title('Count Trend - Place Names of Indian subcontinent')

# Adjust the spacing between subplots
plt.subplots_adjust(wspace=0.4)

# Show the plot
plt.show()
```

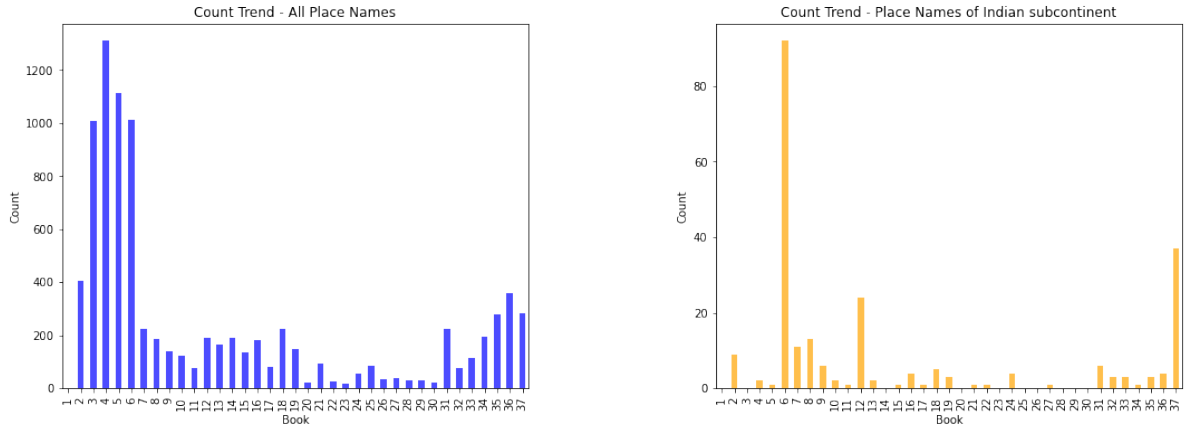

Figure 4: Occurence count for all place names and place names of Indian subcontinent in each book_different y-axis scales

The figures reveal a distinct difference between the occurrence trends of place names related to the Indian subcontinent and all place names collectively. Specifically, the referencing of the Indian subcontinent is highly concentrated in books 6, 12, and 37 of Pliny's narrative. This discrepancy indicates that the mentioning of place names from the Indian subcontinent is closely tied to specific themes and topics within Pliny's work.

In this regard, three methodologies have been employed to analyze the texts pertaining to the Indian subcontinent in *Natural History*, including collocation analysis, topic modeling, and network analysis. The objective of these analyses is to delve deeper into the textual content, unraveling the intricate relationships and uncovering the underlying themes and connections associated with the place names of the Indian subcontinent.

Through collocation analysis, the aim is to identify significant word combinations and phrases that co-occur with the place names of the Indian subcontinent. This analysis provides insights into the specific linguistic patterns and contextual associations surrounding these locations, shedding light on their cultural, historical, and geographical significance.

Topic modeling allows for a broader exploration of the thematic landscape within which the

Indian subcontinent place names are embedded. By clustering related words and identifying prevalent topics, this methodology helps to discern the major themes and subject matters that emerge from Pliny's narrative, providing a comprehensive understanding of the broader context in which these place names are referenced.

Furthermore, network analysis offers a visual representation of the interconnections among the place names of the Indian subcontinent and other entities in Pliny's work. By examining the relationships between different locations and named entities, this analysis uncovers the geographical and conceptual networks that exist within the text, revealing how the Indian subcontinent place names contribute to the overall structure and narrative flow of *Natural History*.

Together, these methodologies aim to provide a nuanced and comprehensive exploration of the texts related to the Indian subcontinent in *Natural History*. By delving into the linguistic, thematic, and network aspects of these place names, a deeper understanding of their significance and their role in shaping Pliny's narrative can be achieved.

0.7.2.2.1 Frequency list and collocations in Indian subcontinent related texts

Through the utilization of measures available in the NLTK package, a word frequency list and a list of collocating bi-grams of the texts pertaining to the Indian subcontinent are generated to investigate potential keywords and themes of interest.

To enhance the relevance and descriptive nature of the frequency list, particular attention has been given to exclude two commonly encountered but less informative words, namely "india" and "also", from the token list.

```
indiatokenlist = [word for paragraph in tokenize_paragraphs(geotext_india['Text'].drop_dup
indiatokenlist = [token for token in indiatokenlist if token != 'india' and token != 'also

# apply the freqency distribution function to the token list
freq_dist = nltk.FreqDist(indiatokenlist)

# Sort the words by frequency in descending order
sorted_words = freq_dist.most_common()

# Get the frequency of the 20th word
top_count = sorted_words[188][1]

# Get all words with frequencies greater than or equal to the frequency of the 20th word
top_words = [(word, count) for word, count in sorted_words if count >= top_count]

corpus_size = len(indiatokenlist)
```

```
top_freq = len(top_words)
top_freq_ratio = round(top_freq/corpus_size*100)
top_freq_ratio

display(Markdown("""Among {corpus_size} tokens of the whole corpus for Indian subcontinent
{top_freq} (the top {top_freq_ratio}%) frequent words is filterd out and shown in @fig-fre
```

Among 18775 tokens of the whole corpus for Indian subcontinent related text, 197 (the top 1%) frequent words is filterd out and shown in Figure 5 and Figure 6.

```
import squarify
import matplotlib.cm as cm

# top_words contains the list of (word, count) tuples
labels = [word for word, _ in top_words]
sizes = [count for _, count in top_words]

# Normalize the sizes to range between 0 and 1
normalized_sizes = [(size - min(sizes)) / (max(sizes) - min(sizes)) for size in sizes]

# Define the color scale
color_scale = cm.Blues

plt.figure(figsize=(18, 10))
squarify.plot(sizes=sizes, label=labels, alpha=0.7, color=color_scale(normalized_sizes))

# Add labels and title
plt.axis('off')

# Show the treemap
plt.show()
```

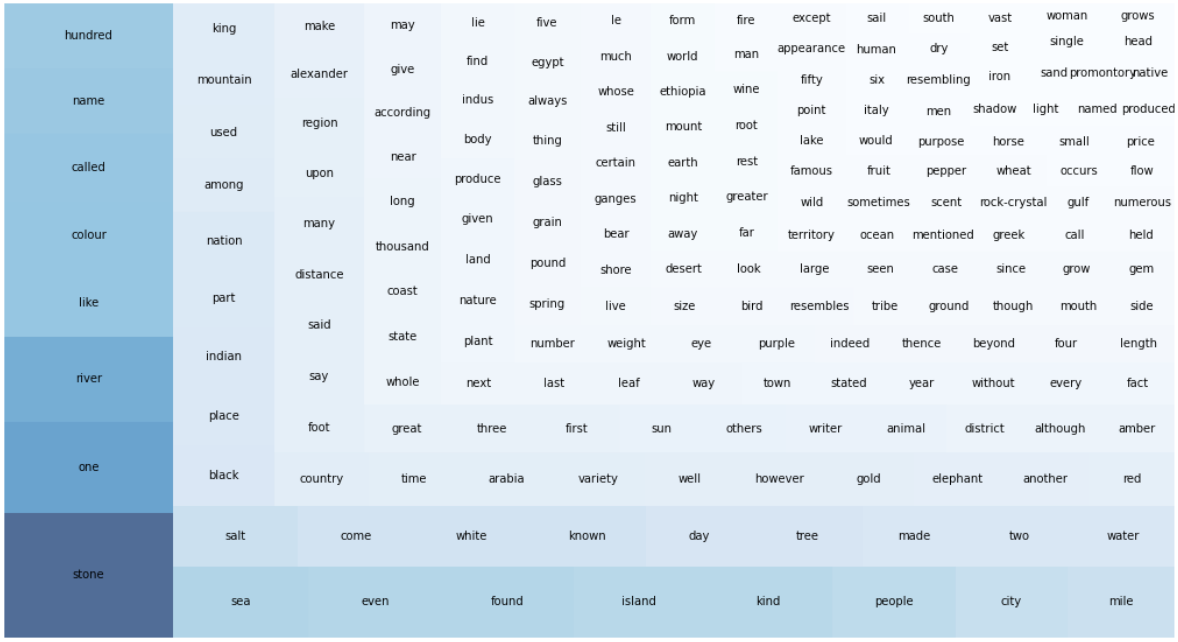

Figure 5: Top 1% frequent words in Indian subcontinent related text as tree map

```
from wordcloud import WordCloud
# top_words contains the list of (word, count) tuples
wordcloud_data = {word: count for word, count in top_words}

# Create a WordCloud object
wordcloud = WordCloud(width=800, height=400, background_color='white')

# Generate the word cloud from the word frequency data
wordcloud.generate_from_frequencies(wordcloud_data)

# Display the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')

# Show the plot
plt.show()
```

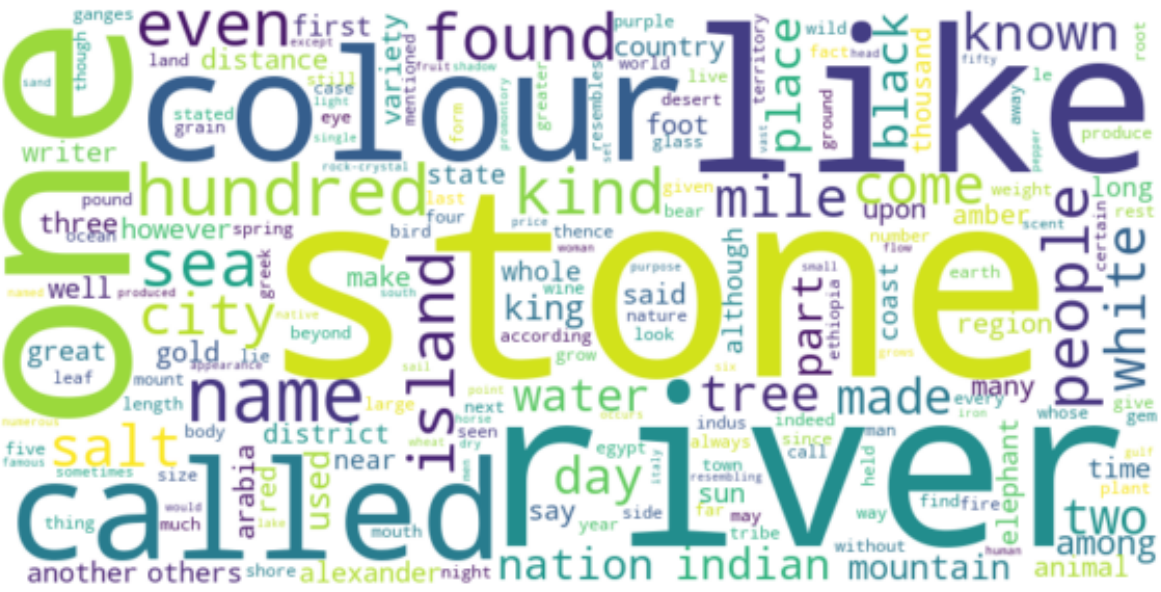

Figure 6: Top 1% frequent words in Indian subcontinent related text as word cloud

As depicted in the visualizations, the words “stone,” “river,” and “color” notably stand out, suggesting their prominence in the narrative pertaining to the regions of the Indian subcontinent. This observation is indicative of the significant references to precious stones and the origins and transportation routes associated with the trade of such valuable commodities.

The collocating bi-grams associated with place names of the Indian subcontinent region are extracted based on the top 20 highest scores in the likelihood ration measurement. A higher likelihood ratio score indicates a stronger association or collocation between the words, suggesting that they are more likely to appear together in the given text.

The extracted collocations undergo a filtering process that specifically includes those involving keywords of place names within the regions of the Indian subcontinent, whihc enables a focused analysis of collocations directly relevant to the geographic context.

```
keywords = ['India', 'Indus', 'Bactria', 'Ganges', 'Acesinus', 'Oxus', 'Hydaspes', 'Taprob

finder = BigramCollocationFinder.from_documents(para)

for keyword in keywords:
    finder.apply_word_filter(lambda w: w == keyword.lower())

measures = BigramAssocMeasures()
```

```
finder.nbest(measures.likelihood_ratio, 20)
```

```
[('already', 'mentioned'),
 ('present', 'day'),
 ('alexander', 'great'),
 ('father', 'liber'),
 ('taken', 'drink'),
 ('formerly', 'called'),
 ('majesty', 'augustus'),
 ('fifty', 'mile'),
 ('late', 'majesty'),
 ('next', 'come'),
 ('roman', 'citizen'),
 ('mile', 'circumference'),
 ('human', 'being'),
 ('greek', 'name'),
 ('late', 'lamented'),
 ('marcus', 'varro'),
 ('one', 'hundred'),
 ('hundred', 'fifty'),
 ('rising', 'dog-star'),
 ('emperor', 'nero')]
```

Interestingly, in the flitered bi-grams, 20% of them are referring to human names or names of gods in myths (e.g. Alexander III, the Great (king of Macedon); Octavius Caesar Augustus (Roman Emperor); Nero (Roman emperor); Marcus Varro (ancient Latin scholar), Father Liber (referring to Dionysus, Greek god of winemaking and wine)).

As shown in the quotation of Book 16, Chapter 62, Paragraph 1, the word "India" was mentioned in the context of an introduction of a plant, as a counterpart in the plant origin, and as a conquered land intertwining with the historical story about how the plant was brought to Rome by Alexander the Great.

> 16.62.1 It is said that ivy now grows in Asia Minor. Theophrastus about 314 BC. had stated that it did not grow there, nor yet in **India** except on Mount Meros, and indeed that Harpalus had used every effort to grow it in Media without success, while **Alexander** had come back victorious from **India** with his army wearing wreaths of ivy, because of its rarity, in imitation of **Father Liber**; and it is even now used at solemn festivals among the peoples of Thrace to decorate the wands of that god, and also the worshippers' helmets and shields, although it is injurious to all trees and plants and destructive to tombs and walls, and very agreeable to chilly snakes, so that it is surprising that any honour has been paid to it.

##(More detailed analysis and illustration will be further conducted for the pattern of interactions between Indian subcontinent place names and human names in the book.)

0.7.2.3 Topic modelling about Indian subcontinent region related texts

Since the corpus size for text pertaining Indian subcontient region is rather small, with certain tryouts, the the number of topics is set as 3 and the passes is set as 40 to get the most noon-overlapping topic clusters.

The word "India" is excluded from the corpus in order to get more descriptive keywords which may contribe to a more concrete topic summary.

The top 30 keywords for each topic, along with their respective weights, which rank their contributions to the topic is shown and visualized as follows.

```
## topic modelling about texts related to "India"

indiapara = tokenize_paragraphs(geotext_india['Text'].drop_duplicates())
indiapara = [[word for word in paragraph if word != 'india'] for paragraph in indiapara]

indiadictionary = corpora.Dictionary(indiapara)
indiacorpus = [indiadictionary.doc2bow(text) for text in indiapara]

indialdamodel = gensim.models.ldamodel.LdaModel(indiacorpus, num_topics=3, id2word = india

pprint(indialdamodel.print_topics(num_topics=3, num_words=30))
```

```
[(0,
  '0.025*"stone" + 0.007*"also" + 0.007*"river" + 0.007*"found" + '
  '0.007*"colour" + 0.006*"like" + 0.005*"one" + 0.005*"name" + 0.005*"island" '
  '+ 0.005*"white" + 0.004*"hundred" + 0.004*"mile" + 0.004*"gold" + '
  '0.004*"variety" + 0.003*"come" + 0.003*"glass" + 0.003*"city" + '
  '0.003*"known" + 0.003*"many" + 0.003*"sea" + 0.003*"gem" + 0.003*"black" + '
  '0.003*"even" + 0.003*"nation" + 0.003*"thence" + 0.003*"place" + '
  '0.002*"according" + 0.002*"distance" + 0.002*"kind" + 0.002*"alexander"'),
 (1,
  '0.010*"also" + 0.006*"called" + 0.006*"one" + 0.005*"hundred" + '
  '0.005*"people" + 0.004*"name" + 0.004*"tree" + 0.004*"kind" + 0.004*"river" '
  '+ 0.004*"colour" + 0.004*"like" + 0.004*"city" + 0.003*"even" + '
  '0.003*"mile" + 0.003*"two" + 0.003*"black" + 0.003*"known" + 0.003*"island" '
  '+ 0.003*"used" + 0.003*"part" + 0.003*"amber" + 0.003*"indian" + '
  '0.003*"made" + 0.003*"foot" + 0.003*"come" + 0.003*"mountain" + 0.003*"sea" '
  '+ 0.002*"make" + 0.002*"arabia" + 0.002*"king"'),
```

```
(2,
 '0.010*"salt" + 0.007*"also" + 0.006*"sea" + 0.006*"one" + 0.005*"even" + '
 '0.004*"day" + 0.004*"river" + 0.004*"water" + 0.004*"time" + 0.003*"among" '
 '+ 0.003*"great" + 0.003*"spring" + 0.003*"kind" + 0.003*"elephant" + '
 '0.003*"found" + 0.003*"island" + 0.002*"animal" + 0.002*"alexander" + '
 '0.002*"called" + 0.002*"made" + 0.002*"near" + 0.002*"night" + '
 '0.002*"country" + 0.002*"king" + 0.002*"people" + 0.002*"well" + '
 '0.002*"land" + 0.002*"two" + 0.002*"name" + 0.002*"place"')]
```

```
# Convert the LdaModel object to a pyLDAvis-compatible format
vis_dataindia = pyLDAvis.gensim_models.prepare(indialdamodel, indiacorpus, indiadictionary

# Enable the Jupyter notebook inline display of visualizations
pyLDAvis.enable_notebook()

# Display the pyLDAvis visualization
pyLDAvis.display(vis_dataindia)
```

```
<IPython.core.display.HTML object>
```

The three generated topics for the Indian subcontinent related texts can be summarized based on the dominant words as follows:

Topic 1: **Stones, Rivers, and Islands** - various elements related to stones, rivers, and islands. It also touches upon the notion of distance and the mention of gold and gems.

Topic 2: **Cities, Trees, and Natural Features** - cities, trees, and natural features. It also mentions amber, mountains, and the connection to Arabia.

Topic 3: **Salt, Sea, and Water** - salt, the sea, and water-related concepts. It also touches upon topics such as animals, Alexander the Great, and the notion of a country.

And Topic 1: **Stones, Rivers, and Islands** takes the forefront among the other topics.

Consistent with the findings in the frequency list of the corpus, it is evident that "stones" and "rivers" hold a significant presence in the narrative concerning the Indian subcontinent.

0.7.2.3.1 Network analysis about Indian subcontinent region related texts

Two separate network analyses were conducted. The first analysis focused on exploring the relationships between place names mentioned throughout the entire book. The second analysis specifically examined the name entities of people and place names associated with the Indian subcontinent regions. Nodes and edges were generated for both analyses and imported into

Gephi for visualization. By studying the clustering patterns of place names and people in the resulting network graphs, valuable insights can be gained into both the overall context of the book and the specific context of the Indian subcontinent within *Natural History*.

```
# define a function for generating the edge data for co-occurence of two places in a same
def place_name_edge (inputdf):

    co_occurrence = pd.DataFrame(columns=['source', 'target'])

    for i in range(len(inputdf)):
        for j in range(i+1, len(inputdf)):
            place1 = inputdf['Place_Name'].iloc[i]
            place2 = inputdf['Place_Name'].iloc[j]

            if place1 == place2:
                continue

            if geotext_whole['Book'].iloc[i] == geotext_whole['Book'].iloc[j]:
                if geotext_whole['Chapter'].iloc[i] == geotext_whole['Chapter'].iloc[j]:
                    if geotext_whole['Paragraph'].iloc[i] == geotext_whole['Paragraph'].il
                        co_occurrence = pd.concat([co_occurrence, pd.DataFrame({'source':
    return co_occurrence

# store the edges and nodes generated for network analysis about place names in the entire

geonameco = place_name_edge(geotext_whole)

geonameco.rename(columns={'Place1': 'source', 'Place2': 'target'}, inplace=True)
geonameco.to_csv('geonameco.csv', index=False)

unique_places = pd.DataFrame(geotext_whole['Place_Name'].drop_duplicates()).rename(columns
unique_places['label'] = unique_places['id']
unique_places.to_csv('place_nodes.csv', index=False)
```

In the network analysis for place names throughout the entire book, unique place names are seen as nodes, and once two place names co-occur in the same paragraph, it will be counted as one edge. There are total 2255 nodes and 52602 edges in the prepared data.

As shown in Figure 7, the size of the node represents the betweeness centrality a place name mentioned in the book, and the weight of edge between two nodes represents the time the two place names appeared in the same paragraph (as seen in the same context). Gone through a Force Atlas 2 layout algorithm, the graph also demonstrates the rough cluster of place names which tend to be mentioned together.

In the case of “India”, it is observed mostly incooperates with “Egypt”, “Arabia” and “Nille”, which tend to be appearing in the description of trading route.

```
from IPython.display import Image
Image(filename='geonamenw.png')
```

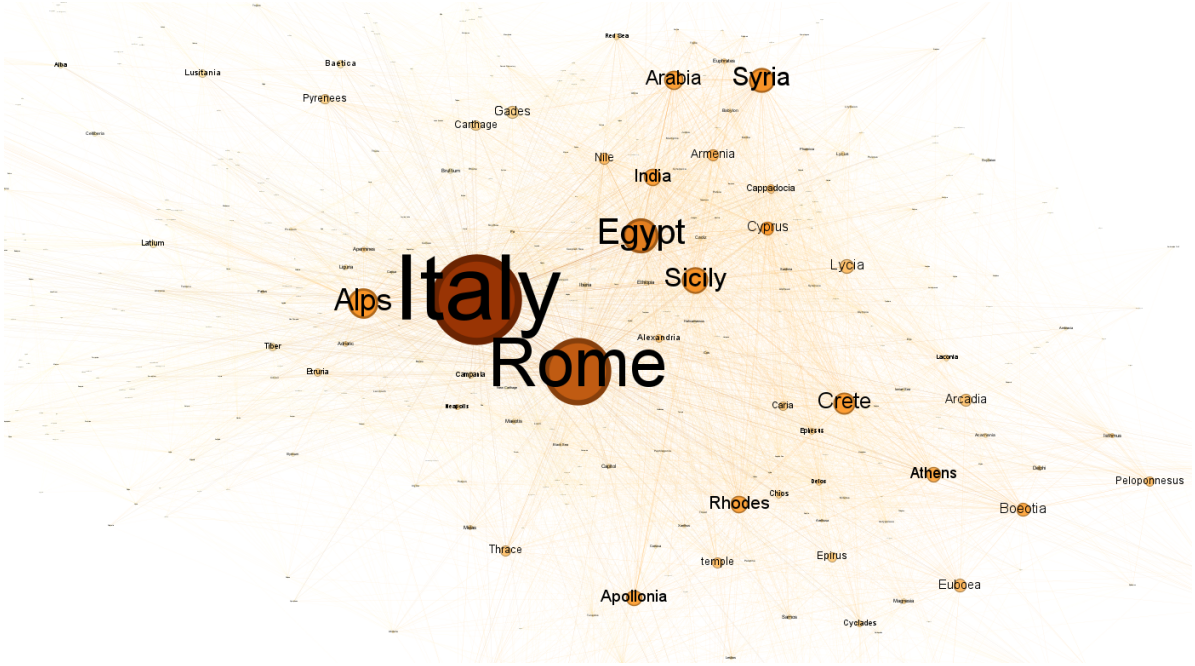

Figure 7: Network graph for place namesmentioned in Natural History

To gain a more detailed cluster of narrative contents about Indian subcontinent in *Natural History*, the idea is to generate a network for book number, place names and person names in the target corpus. The person name nodes are retrieved from the tagging of text given by the pretrained multilingual Name Entity Recognition model WikiNEuRal (Tedeschi et al. 2021).

##(will further compare with scraping person name annotations from ToposText, to see which way gets more accurate information.)

The tags for name entity groups retrieved from WikiNEuRal model is appended as a new column in the corpus dataframe. And the tags as “PER”, which means “person name” are further extracted as another column.

```
# apply the NER model to the Indian subcontinent related texts and store the retrieved tag
geotext_india['Text_ner'] =  geotext_india['Text'].progress_apply(ner)
```

```
# append the words with tag as "PER" i.e. person name into the dataframe
geotext_india['PER_names'] = geotext_india['Text_ner'].apply(lambda x: [entity['word'] for
geotext_india[['Place_Name','Book','Chapter','Paragraph','Text_ner','PER_names']].head()
```

	Place_Name	Book	Chapter	Paragraph	Text_ner	PER_na
85	India	2	75	1.0	[{'entity_group': 'LOC', 'score': 0.99636984, ...	[Onesicr
92	India	2	75	1.0	[{'entity_group': 'LOC', 'score': 0.99636984, ...	[Onesicr
93	India	2	75	1.0	[{'entity_group': 'LOC', 'score': 0.99636984, ...	[Onesicr
218	Indus	2	98	1.0	[{'entity_group': 'LOC', 'score': 0.999539, 'w...	[]
326	Bactria	2	110	1.0	[{'entity_group': 'LOC', 'score': 0.87196684, ...	[Ctesias,

The rows containing no person name were dropped and those with multiple person name records were exploded to separate rows.

```
geotext_india_exploded = geotext_india.explode('PER_names')
geotext_india_exploded.dropna(subset=['PER_names'], inplace=True)
geotext_india_exploded[['Place_Name', 'Book','PER_names']]
```

	Place_Name	Book	PER_names
85	India	2	Onesicritus
85	India	2	Alexander
85	India	2	Alexander
85	India	2	Onesicritus
92	India	2	Onesicritus
...	...	...	...
8842	India	37	Jupiter
8847	India	37	Xenocrates
8866	Indus	37	Democritus
8873	India	37	Nature
8873	India	37	Nature

Within the Indian subcontinent context, the nodes consist of three types, namely **place name**, **person name** and **book number**.

And there are four types of edges being recorded and combined, including the co-occurence of:

1. **place name** and **person name** in the same paragraph
2. **person name** and **book number**

3. **place name** and **book number**
4. **place name** and **place name** in the same paragraph

```
edge1 = geotext_india_exploded[['Place_Name', 'PER_names']]
edge1.rename(columns={'Place_Name': 'source', 'PER_names': 'target'}, inplace=True)

edge2 = geotext_india_exploded[['PER_names','Book']]
edge2.rename(columns={'PER_names': 'source', 'Book': 'target'}, inplace=True)

edge3 = geotext_india[['Place_Name', 'Book']]
edge3.rename(columns={'Place_Name': 'source', 'Book': 'target'}, inplace=True)

edge4 = place_name_edge(geotext_india)

# combine the four types of edges and save to .csv
combined_edge = pd.concat([edge1, edge2, edge3, edge4], axis=0)
combined_edge = combined_edge.reset_index(drop=True)
combined_edge.to_csv('indiatext_edge.csv', index=False)


node1 = pd.DataFrame(geotext_india['Place_Name'].drop_duplicates()).rename(columns={'Place
node1['label'] = node1['id']
node1['type'] = 'Place_Name'

node2 = pd.DataFrame(geotext_india['Book'].drop_duplicates()).rename(columns={'Book': 'id'
node2['label'] = 'Book ' + node2['id'].astype(str)
node2['type'] = 'Book'

node3 = pd.DataFrame(geotext_india_exploded['PER_names'].drop_duplicates()).rename(columns
node3['label'] = node3['id']
node3['type'] = 'Person_Name'

# combine the three types of nodes and save to .csv
combined_node = pd.concat([node1, node2, node3], axis=0)
combined_node = combined_node.reset_index(drop=True)
combined_node.to_csv('indiatext_node.csv', index=False)

node_count2 = len(combined_node)
edge_count2 = len(combined_edge)

display(Markdown("""In the network analysis for place names and person names within the In
and {edge_count} edges in the prepared data.
```

```
""".format(node_count = node_count2, edge_count = edge_count2)))
```

In the network analysis for place names and person names within the Indian subcontinent context, there are total 164 nodes and 1458 edges in the prepared data.

As manifested in **?@fig-indiantext_clustering**, there is obvious clustering of person names occuring in Indian subcontinent related texts. In other words, groups of person names are tend to be referenced in some specific topics.

##(more detailed illustration will be further conducted.)

Gibson, Roy, and Ruth Morello, eds. 2011. *Pliny the Elder: Themes and Contexts*. Brill. https://brill.com/edcollbook/title/14893.

Murphy, Trevor. 2003. "11. Pliny's Naturalis Historia: The Prodigal Text." In, 301–22. BRILL. https://doi.org/10.1163/9789004217157_012.

Nappo, Dario. 2017. *Money and Flows of Coinage in the Red Sea Trade*. Vol. 1. Oxford University Press. https://doi.org/10.1093/oso/9780198790662.003.0017.

Pollard, Elizabeth Ann. 2009. "Pliny's Natural History and the Flavian Templum Pacis: Botanical Imperialism in First-Century <Small Class="caps" Xmlns:m="http://Www.w3.org/1998/Math/MathML" Xmlns:mml="http://Www.w3.org/1998/Math/Mat Xmlns:xlink="http://Www.w3.org/1999/Xlink">c.e</Small>. Rome." *Journal of World History* 20 (3): 309–38. https://doi.org/10.1353/jwh.0.0074.

Tedeschi, Simone, Valentino Maiorca, Niccolò Campolungo, Francesco Cecconi, and Roberto Navigli. 2021. "WikiNEuRal: Combined Neural and Knowledge-Based Silver Data Creation for Multilingual NER." In, 25212533. Punta Cana, Dominican Republic: Association for Computational Linguistics. https://aclanthology.org/2021.findings-emnlp.215.

Tran, Khuyen. 2022. "pyLDAvis: Topic Modelling Exploration Tool That Every NLP Data Scientist Should Know." https://neptune.ai/blog/pyldavis-topic-modelling-exploration-tool-that-every-nlp-data-scientist-should-know.