Evaluating Text Complexity Readability Measures for English and Chinese CBS Research Seminar, PolyU

Leo Lei Lei
Shanghai International Studies University

2025-03-10

Table of contents

1 Readability of English texts: Relationship between readability and citation count 3
1.1 Installing the package . . . 3
1.1.1 Homepage . . . 3
1.1.2 Installation . . . 3
1.2 Loading the packages . . . 3
1.3 Testing the package . . . 4
1.4 Experiment on *Nature* abstracts . . . 12
1.4.1 Loading the dataset . . . 12
1.4.2 Calculating the readability measures . . . 14
1.4.3 Calculating the normed citation count . . . 16
1.4.4 Correlation between readability and citation count . . . 20

References 22

1 Readability of English texts: Relationship between readability and citation count

In this section, we will showcase how to calculate readability of English texts. We will also explore the relationship between readability and citation count with an experiment on *Nature* abstracts.

The packages we will use in the seminar are:

1. `readability`
2. `pandas`
3. `spacy`
4. `numpy`
5. `matplotlib`
6. `seaborn`

1.1 Installing the package

1.1.1 Homepage

https://pypi.org/project/readability/

1.1.2 Installation

```
!pip install readability -i https://pypi.tuna.tsinghua.edu.cn/simple

!pip install https://github.com/andreasvc/readability/tarball/master
```

1.2 Loading the packages

```
import readability
import pandas as pd
```

1.3 Testing the package

The pacakge includes nine indices of "readability grades", i.e.,

1. Kincaid
2. ARI
3. Coleman-Liau
4. FleschReadingEase
5. GunningFogIndex
6. LIX
7. SMOGIndex
8. RIX,
9. DaleChallIndex

and more than ten indices of "sentence info", i.e.,

1. characters_per_word
2. syll_per_word
3. words_per_sentence
4. sentences_per_paragraphtype_token_ratio
5. characters
6. syllables
7. words
8. wordtypes
9. sentences
10. paragraphs
11. long_words
12. complex_words
13. complex_words_dc

```
mytext = 'The United Nations is an international organization founded in
 ↪  1945. Currently made up of 193 Member States, the UN and its work are
 ↪  guided by the purposes and principles contained in its founding Charter.
 ↪  The UN has evolved over the years to keep pace with a rapidly changing
 ↪  world.'

results1 = readability.getmeasures(mytext, lang='en')
```

```
4
5 results1
```

```
OrderedDict([('readability grades',
              OrderedDict([('Kincaid', 20.13632653061224),
                           ('ARI', 25.658775510204087),
                           ('Coleman-Liau', 11.794659367346942),
                           ('FleschReadingEase', 37.96938775510207),
                           ('GunningFogIndex', 25.314285714285717),
                           ('LIX', 77.57142857142857),
                           ('SMOGIndex', 17.491376746189438),
                           ('RIX', 14.0),
                           ('DaleChallIndex', 6.619583673469387)])),
             ('sentence info',
              OrderedDict([('characters_per_word', 4.795918367346939),
                           ('syll_per_word', 1.4081632653061225),
                           ('words_per_sentence', 49.0),
                           ('sentences_per_paragraph', 1.0),
                           ('type_token_ratio', 0.8571428571428571),
                           ('directspeech_ratio', 0.0),
                           ('characters', 235),
                           ('syllables', 69),
                           ('words', 49),
                           ('wordtypes', 42),
                           ('sentences', 1),
                           ('paragraphs', 1),
                           ('long_words', 14),
                           ('complex_words', 7),
                           ('complex_words_dc', 13)])),
             ('word usage',
              OrderedDict([('tobeverb', 2),
                           ('auxverb', 0),
                           ('conjunction', 2),
                           ('pronoun', 2),
                           ('preposition', 8),
                           ('nominalization', 1)])),
             ('sentence beginnings',
              OrderedDict([('pronoun', 0),
                           ('interrogative', 0),
                           ('article', 1),
                           ('subordination', 0),
```

```
                            ('conjunction', 0),
                            ('preposition', 0)]))])
```

```
results1['readability grades']
```

```
OrderedDict([('Kincaid', 20.13632653061224),
             ('ARI', 25.658775510204087),
             ('Coleman-Liau', 11.794659367346942),
             ('FleschReadingEase', 37.96938775510207),
             ('GunningFogIndex', 25.314285714285717),
             ('LIX', 77.57142857142857),
             ('SMOGIndex', 17.491376746189438),
             ('RIX', 14.0),
             ('DaleChallIndex', 6.619583673469387)])
```

```
results1['readability grades']['FleschReadingEase']
```

```
37.96938775510207
```

Note that the `readability` package will calculate a string as ONE SENTENCE. That is, although the string contain many sentences, it will calculate the number of sentences as 1, which may affect the results.

The following code shows the incorrect results.

```
results1['sentence info']

# Note: incorrect results in the line below:
# ('sentences', 1),
```

```
OrderedDict([('characters_per_word', 4.795918367346939),
             ('syll_per_word', 1.4081632653061225),
             ('words_per_sentence', 49.0),
             ('sentences_per_paragraph', 1.0),
             ('type_token_ratio', 0.8571428571428571),
             ('directspeech_ratio', 0.0),
             ('characters', 235),
             ('syllables', 69),
             ('words', 49),
             ('wordtypes', 42),
```

```
('sentences', 1),
('paragraphs', 1),
('long_words', 14),
('complex_words', 7),
('complex_words_dc', 13)])
```

To address this issue, we will use the `spacy` package to split the text into sentences. In the following code, we will define a function to split the text into sentences.

```
# Install spaCy
!pip install -U pip setuptools wheel
!pip install spacy

# Download best-matching version of specific model for your spaCy
 ↪  installation
python -m spacy download en_core_web_sm
```

```
import spacy

# Load the spaCy model
nlp = spacy.load('en_core_web_sm')

# Define a function to split the paragraph into sentences
# and return a list of splitted sentences
def split_paragraph_into_sentences(paragraph):

    # Parse the paragraph
    doc = nlp(paragraph)

    # Split the parsed paragraph into sentences
    sentences = [sent.text for sent in doc.sents]

    return sentences
```

Now, we test the function `split_paragraph_into_sentences` we just defined.

```
# split the paragraph into sentences
# return
my_sents = split_paragraph_into_sentences(mytext)
my_sents
```

```
['The United Nations is an international organization founded in 1945.',
 'Currently made up of 193 Member States, the UN and its work are guided by the purposes and
 'The UN has evolved over the years to keep pace with a rapidly changing world.']
```

We will also define a function to calculate the readability measures of a text. The function will return a tuple, which contains two elements: 1) a list of readability measure names, and 2) a list of corresponding readability measure values.

```
# define a function to get all measures

def get_readability_measures(str):
    """
    Args:
        str: the input string

    Returns:

        A tuple: a tuple contains 1) readability measure names, and 2) corresponding readability measure values
    """
    results = readability.getmeasures(str, lang='en')
    # print(results['readability grades']) # return a data type: OrderedDict

    ordered_dict1 = results['readability grades']
    ordered_dict2 = results["sentence info"]

    # add ordered_dict2 to order_dict1
    ordered_dict1.update(ordered_dict2)

    return (list(ordered_dict1.keys()), list(ordered_dict1.values()))
```

Now, we test the function `get_readability_measures` we just defined.

```
get_readability_measures(my_sents[0])
```

```
(['Kincaid',
  'ARI',
  'Coleman-Liau',
  'FleschReadingEase',
  'GunningFogIndex',
  'LIX',
```

```
 'SMOGIndex',
 'RIX',
 'DaleChallIndex',
 'characters_per_word',
 'syll_per_word',
 'words_per_sentence',
 'sentences_per_paragraph',
 'type_token_ratio',
 'directspeech_ratio',
 'characters',
 'syllables',
 'words',
 'wordtypes',
 'sentences',
 'paragraphs',
 'long_words',
 'complex_words',
 'complex_words_dc'],
[10.73,
 11.359000000000002,
 15.931588900000001,
 35.94500000000003,
 12.0,
 50.0,
 10.745966692414834,
 4.0,
 6.812000000000001,
 5.9,
 1.9,
 10.0,
 1.0,
 1.0,
 0.0,
 59,
 19,
 10,
 10,
 1,
 1,
 4,
 2,
 4])
```

Let's use the function `get_readability_measures` to calculate the readability measures of a paragraph, and save the results in a dataframe.

The logic here is that:

1. Split the paragraph into sentences
2. Calculate the readability measures of each sentence
3. Calculate the average readability measures of the sentences
4. Save the results in a dataframe, with the result of each sentence as a row.

```
import numpy as np

list_temp = []

readability_measures = None

for i in range(len(my_sents)):

    mystr = my_sents[i]

    results = get_readability_measures(mystr)
    readability_measures = results[0]
    list_temp.append(results[1]) # return the values of readability measures

    # print(list(results))

# Calculate the average readability measures of each abstract
avg_measures = np.mean(list_temp, axis=0) # axis=0 means calculate the
 ↪  average of each column
avg_measures = avg_measures.tolist() # convert the numpy array to list

mydf_out = pd.DataFrame(columns=readability_measures)

# add the average readability measures to the dataframe as the first row of
 ↪  the dataframe
mydf_out.loc[0] = avg_measures

mydf_out
```

	Kincaid	ARI	Coleman-Liau	FleschReadingEase	GunningFogIndex	LIX	SMOGIndex
0	8.217778	9.98575	11.168122	65.236667	12.644444	46.055556	11.326255

Now we will define a function to calculate the average readability measures of a paragraph, and save the results in a list based on the code above.

```
def get_mean_readability_measures(list_of_sents):
  '''
  Args:
      my_sents: a list of sentences
  Returns:
      A tuple: a tuple contains
      1) readability measure names, and
      2) corresponding readability measure values
  '''
  list_temp = []

  readability_measures = None

  for i in range(len(list_of_sents)):

      mystr = list_of_sents[i]

      results = get_readability_measures(mystr)
      readability_measures = results[0]
      list_temp.append(results[1]) # return the values of readability measures

      # print(list(results))

  # Calculate the average readability measures of each abstract
  avg_measures = np.mean(list_temp, axis=0) # axis=0 means calculate the average of each column
  avg_measures = avg_measures.tolist() # convert the numpy array to list

  return (readability_measures, avg_measures)
```

Now we will test the function `get_mean_readability_measures` we just defined.

```
my_measures = get_mean_readability_measures(my_sents)

print(my_measures[0])
print(my_measures[1])
```

```
['Kincaid', 'ARI', 'Coleman-Liau', 'FleschReadingEase', 'GunningFogIndex', 'LIX', 'SMOGIndex
```

```
[8.217777777777778, 9.98575, 11.168121630555556, 65.2366666666667, 12.644444444444446, 46.055
```

1.4 Experiment on *Nature* abstracts

In this section, we will use the functions we defined to calculate the readability measures of *Nature* abstracts, and save the results in a dataframe. The logic here is that:

1. Load the dataset
2. Split the abstract into sentences
3. Calculate the average readability measures of each abstract with the function `get_mean_readability_measures`
4. Save the results in a dataframe, with the result of each abstract as a row.
5. Calculate the correlation between the readability measures and the normed citation count.

1.4.1 Loading the dataset

The dataset we will use is the dataset of *Nature* abstracts, which was download from Scopus on March 9, 2025. The dataset contains abstracts *Nature* articles published from 2011 to 2020. The dataset is saved in the file `nature_abstracts_2011_2020_scopus.csv`.

Now, we load the dataset.

```
mydf = pd.read_csv('nature_abstracts_2011_2020_scopus.csv')

mydf.head()
```

	Authors	Author full names
0	Senzaki M.; Barber J.R.; Phillips J.N.; Carter...	Senzaki, Masayuki (56380383600); Barber, Jesse...
1	Salahudeen A.A.; Choi S.S.; Rustagi A.; Zhu J....	Salahudeen, Ameen A. (57522403600); Choi, Shan...
2	Young A.W.; Eckner W.J.; Milner W.R.; Kedar D....	Young, Aaron W. (57205574645); Eckner, William...
3	Lee J.; Robinson M.E.; Ma N.; Artadji D.; Ahme...	Lee, Jaewoong (56539403200); Robinson, Mark E....
4	Scheitl C.P.M.; Ghaem Maghami M.; Lenz A.-K.; ...	Scheitl, Carolin P. M. (57212219230); Ghaem Ma...

Check out the information of the dataset with the following code. From the insults, we can see that the dataset contains 19 columns, and 9478 rows.

The column `Abstract` contains the abstracts of the *Nature* articles. The column `Cited by` contains the number of citations of each article. The column `Year` contains the year of publication of each article. The column `EID` is the Scopus ID of each article.

```
mydf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9478 entries, 0 to 9477
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Authors            9408 non-null   object
 1   Author full names  9408 non-null   object
 2   Author(s) ID       9408 non-null   object
 3   Title              9478 non-null   object
 4   Year               9478 non-null   int64
 5   Source title       9478 non-null   object
 6   Volume             9477 non-null   float64
 7   Issue              9423 non-null   object
 8   Art. No.           10 non-null     float64
 9   Page start         9467 non-null   object
 10  Page end           9248 non-null   object
 11  Page count         9247 non-null   float64
 12  Cited by           9478 non-null   int64
 13  DOI                9455 non-null   object
 14  Link               9478 non-null   object
 15  Abstract           9478 non-null   object
 16  Document Type      9478 non-null   object
 17  Source             9478 non-null   object
 18  EID                9478 non-null   object
dtypes: float64(3), int64(2), object(14)
memory usage: 1.4+ MB
```

We will now develop a new dataframe to contain the columns of data that we need for the experiment:

1. `EID`: the Scopus ID of each article
2. `Year`: the year of publication of each article
3. `Cited by`: the number of citations of each article
4. `Abstract`: the abstracts of the *Nature* articles

```
mydf2 = mydf[['EID', 'Year', 'Cited by', 'Abstract']]
mydf2.head()
```

	EID	Year	Cited by	Abstract
0	2-s2.0-85095816319	2020	116	Expansion of anthropogenic noise and night lig...
1	2-s2.0-85096616590	2020	259	The distal lung contains terminal bronchioles ...
2	2-s2.0-85097599586	2020	143	The preparation of large, low-entropy, highly ...
3	2-s2.0-85094971082	2020	62	Interferon-induced transmembrane protein 3 (IF...
4	2-s2.0-85094212584	2020	69	Nearly all classes of coding and non-coding RN...

1.4.2 Calculating the readability measures

Then, we will use the funtions we defined to calculate the readability measures of *Nature* abstracts, and save the results in a dataframe. The logic here is that: 1. Split the abstract into sentences 2. Calculate the readability measures of each abstract with the function `get_mean_readability_measures` 3. Save the results in a dataframe, with the result of each sentence as a row.

```
import numpy as np

mydf_final = []

# use mydf2.sample(n = 100, random_state=2025) to test the code

for index, row in mydf2.sample(n = 60, random_state=2025).iterrows():
# for index, row in mydf2.iterrows():
    eid = row['EID']
    year = row['Year']
    citation_count = row['Cited by']
    abstract = row['Abstract']

    my_sents = split_paragraph_into_sentences(abstract)

    try:
        avg_measures = get_mean_readability_measures(my_sents)[1]
        row_data = [eid, year, citation_count] + avg_measures
        # Append the row data to the final dataframe
        mydf_final.append(row_data)
    except:
        print("Error: ", eid)
        next

# Create the final dataframe
mydf_final = pd.DataFrame(mydf_final, columns=['EID', 'year',
 ↪  'citation_count'] + readability_measures)
```

```

mydf_final.head()
```

	EID	year	citation_count	Kincaid	ARI	Coleman-Liau	FleschReadingEase
0	2-s2.0-85017189811	2017	157	15.913886	18.746947	19.146654	20.152393
1	2-s2.0-84930658418	2015	101	15.437418	18.870844	18.476814	21.119409
2	2-s2.0-79955482843	2011	358	11.259332	14.052478	14.735255	46.101922
3	2-s2.0-79959219765	2011	65	13.575762	18.345035	14.157913	46.266335
4	2-s2.0-85046344158	2018	316	11.940534	16.225532	17.787451	42.405451

We check out the information of the dataset with the following code. From the insults, we can see that the dataset contains 27 columns, and 9478 rows.

```
mydf_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 27 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   EID                      60 non-null     object
 1   year                     60 non-null     int64
 2   citation_count           60 non-null     int64
 3   Kincaid                  60 non-null     float64
 4   ARI                      60 non-null     float64
 5   Coleman-Liau             60 non-null     float64
 6   FleschReadingEase        60 non-null     float64
 7   GunningFogIndex          60 non-null     float64
 8   LIX                      60 non-null     float64
 9   SMOGIndex                60 non-null     float64
 10  RIX                      60 non-null     float64
 11  DaleChallIndex           60 non-null     float64
 12  characters_per_word      60 non-null     float64
 13  syll_per_word            60 non-null     float64
 14  words_per_sentence       60 non-null     float64
 15  sentences_per_paragraph  60 non-null     float64
 16  type_token_ratio         60 non-null     float64
 17  directspeech_ratio       60 non-null     float64
 18  characters               60 non-null     float64
```

```
 19  syllables                60 non-null     float64
 20  words                    60 non-null     float64
 21  wordtypes                60 non-null     float64
 22  sentences                60 non-null     float64
 23  paragraphs               60 non-null     float64
 24  long_words               60 non-null     float64
 25  complex_words            60 non-null     float64
 26  complex_words_dc         60 non-null     float64
dtypes: float64(24), int64(2), object(1)
memory usage: 12.8+ KB
```

1.4.3 Calculating the normed citation count

Now, we will calculate the normed citation count of each article. The normed citation count is the number of citations of each article divided by the mean number of citations of all the articles published in the same year of the target article.

For example, if the number of citations of the target article is 100, and the mean number of citations of all the articles published in the same year of the target article is 20, then the normed citation count of the target article is 5.

Now, we first calculate the mean number of citations for each year.

```
# print(mydf_final['year'].unique())

mean_citation_count_per_year = mydf2.groupby('Year')['Cited by'].mean()

# convert the series to dataframe
mydf_mean_citation_count =
 ↪  mean_citation_count_per_year.to_frame().reset_index()

mydf_mean_citation_count.columns = ['year', 'mean_citation_count_per_year']
mydf_mean_citation_count
```

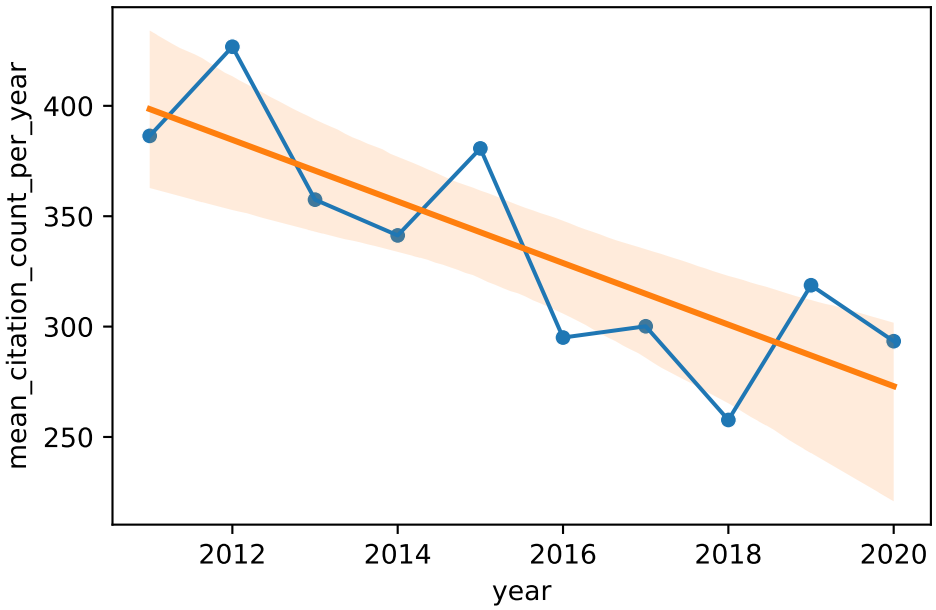
	year	mean_citation_count_per_year
0	2011	386.381215
1	2012	426.767612
2	2013	357.498162
3	2014	341.303644
4	2015	380.713568
5	2016	295.033088

	year	mean_citation_count_per_year
6	2017	300.140859
7	2018	257.698039
8	2019	318.727941
9	2020	293.383023

We draw a line and scatter plot to show the relationship between the normed citation count and year. See the code below.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Draw a line and scatter plot
sns.scatterplot(x='year', y='mean_citation_count_per_year',
 ↪  data=mydf_mean_citation_count)

sns.lineplot(x='year', y='mean_citation_count_per_year',
 ↪  data=mydf_mean_citation_count)

# Add a regression line
sns.regplot(x='year', y='mean_citation_count_per_year',
 ↪  data=mydf_mean_citation_count, scatter=False)
```

```
/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Convert inf value

/opt/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119: FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version. Convert inf value
```



Now, we merge the dataframe `mydf_final` and `mydf_mean_citation_count` on the column `year`. See the code below.

```
mydf_final2 = pd.merge(mydf_final, mydf_mean_citation_count, on='year')

mydf_final2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 28 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   EID                     60 non-null     object
 1   year                    60 non-null     int64
 2   citation_count          60 non-null     int64
 3   Kincaid                 60 non-null     float64
 4   ARI                     60 non-null     float64
 5   Coleman-Liau            60 non-null     float64
 6   FleschReadingEase       60 non-null     float64
 7   GunningFogIndex         60 non-null     float64
 8   LIX                     60 non-null     float64
 9   SMOGIndex               60 non-null     float64
 10  RIX                     60 non-null     float64
```

```
 11  DaleChallIndex                60 non-null     float64
 12  characters_per_word           60 non-null     float64
 13  syll_per_word                 60 non-null     float64
 14  words_per_sentence            60 non-null     float64
 15  sentences_per_paragraph       60 non-null     float64
 16  type_token_ratio              60 non-null     float64
 17  directspeech_ratio            60 non-null     float64
 18  characters                    60 non-null     float64
 19  syllables                     60 non-null     float64
 20  words                         60 non-null     float64
 21  wordtypes                     60 non-null     float64
 22  sentences                     60 non-null     float64
 23  paragraphs                    60 non-null     float64
 24  long_words                    60 non-null     float64
 25  complex_words                 60 non-null     float64
 26  complex_words_dc              60 non-null     float64
 27  mean_citation_count_per_year  60 non-null     float64
dtypes: float64(25), int64(2), object(1)
memory usage: 13.3+ KB
```

Last, we calculate the normed citation count of each article. See the code below.

```
mydf_final2['normed_citation_count'] = mydf_final2['citation_count'] /
 ↪  mydf_final2['mean_citation_count_per_year']

mydf_final2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 29 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   EID                           60 non-null     object
 1   year                          60 non-null     int64
 2   citation_count                60 non-null     int64
 3   Kincaid                       60 non-null     float64
 4   ARI                           60 non-null     float64
 5   Coleman-Liau                  60 non-null     float64
 6   FleschReadingEase             60 non-null     float64
 7   GunningFogIndex               60 non-null     float64
 8   LIX                           60 non-null     float64
```

```
 9   SMOGIndex                   60 non-null     float64
 10  RIX                         60 non-null     float64
 11  DaleChallIndex              60 non-null     float64
 12  characters_per_word         60 non-null     float64
 13  syll_per_word               60 non-null     float64
 14  words_per_sentence          60 non-null     float64
 15  sentences_per_paragraph     60 non-null     float64
 16  type_token_ratio            60 non-null     float64
 17  directspeech_ratio          60 non-null     float64
 18  characters                  60 non-null     float64
 19  syllables                   60 non-null     float64
 20  words                       60 non-null     float64
 21  wordtypes                   60 non-null     float64
 22  sentences                   60 non-null     float64
 23  paragraphs                  60 non-null     float64
 24  long_words                  60 non-null     float64
 25  complex_words               60 non-null     float64
 26  complex_words_dc            60 non-null     float64
 27  mean_citation_count_per_year  60 non-null     float64
 28  normed_citation_count       60 non-null     float64
dtypes: float64(26), int64(2), object(1)
memory usage: 13.7+ KB
```

```
mydf_final3 = mydf_final2.iloc[:, np.r_[3:12, 28]]

mydf_final3.head()
```

	Kincaid	ARI	Coleman-Liau	FleschReadingEase	GunningFogIndex	LIX	SMOGIndex
0	15.913886	18.746947	19.146654	20.152393	20.914015	66.845118	16.813959
1	12.993721	14.741115	15.189140	35.097516	18.677735	55.106028	14.493676
2	14.938345	17.213750	17.636954	22.515664	17.984714	65.450384	14.211004
3	13.113333	13.040000	15.495726	6.390000	14.533333	69.666667	8.477226
4	10.926670	14.474781	14.706661	50.564893	14.448903	52.812045	11.834212

1.4.4 Correlation between readability and citation count

Now, we will calculate the correlation between the readability measures and the normed citation count. See the code below.

```
mydf_final3.corr()['normed_citation_count']
```

```
Kincaid                -0.081146
ARI                     0.030425
Coleman-Liau            0.120490
FleschReadingEase       0.170999
GunningFogIndex         0.084369
LIX                    -0.204352
SMOGIndex               0.147503
RIX                     0.105969
DaleChallIndex         -0.215368
normed_citation_count   1.000000
Name: normed_citation_count, dtype: float64
```

Refer to the articles (Lei & Yan, 2016; Wen & Lei, 2022) for a discussion of readability and correlation.

References

References

Lei, L., & Yan, S. (2016). Readability and citations in information science: Evidence from abstracts and articles of four journals (2003-2012). *Scientometrics*, *108*(3), 1155–1169. https://doi.org/10.1007/s11192-016-2036-9

Wen, J., & Lei, L. (2022). Adjectives and adverbs in life sciences across 50 years: Implications for emotions and readability in academic texts. *Scientometrics*. https://doi.org/10.1007/s11192-022-04453-z