DATA 516 HW2

Win Nawat Suvansinpan

The goal of this assignment is to look at Wikipedia articles on political figures from different countries while focusing on the bias within data. This is done by measuring the coverage and quality of Wikipedia articles on politicians across different countries.[1]
The data sources are:

- Politicians by Country from the English-language Wikipedia
  - Data on most English-language Wikipedia articles within the category "Category:Politicians by nationality" and subcategories, along with the code used to generate that data.
  - https://figshare.com/articles/Untitled_Item/5513449 (https://figshare.com/articles/Untitled_Item/5513449)
- Population data
  - Data on population size of countries in the world as of mid-2018
  - https://www.prb.org/international/indicator/population/table/ (https://www.prb.org/international/indicator/population/table/)

Coverage and quality
A machine learning service called ORES (https://www.mediawiki.org/wiki/ORES) (Objective Revision Evaluation Service) is used to estimate the quality of each article. The tiers of qualities are:

- FA - Featured article
- GA - Good article
- B - B-class article
- C - C-class article
- Start - Start-class article
- Stub - Stub-class article

Coverage is defined by the number of politician articles per country population.

Notes

This cell calls the function that generates the predictions through a for loop of small API calls.
It takes a while to run and can be skipped if `predictions.csv` exists and does not need to be updated.
The cell is commented out.

Part 0: Importing necessary packages

In [331]:
```
import json
import numpy as np
import pandas as pd
import requests
```

Part 1.1: Reading the data

In [332]:
```
page_data_csv = pd.read_csv("page_data.csv")
WPDS_2018_data_csv = pd.read_csv("WPDS_2018_data.csv")
```

Looking at the head of the dataframes.

In [333]:
```
print(page_data_csv.head())
print("df shape is " + str(page_data_csv.shape))
```

```
                                page   country     rev_id
0  Template:ZambiaProvincialMinisters    Zambia  235107991
1                      Bir I of Kanem      Chad  355319463
2   Template:Zimbabwe-politician-stub  Zimbabwe  391862046
3     Template:Uganda-politician-stub    Uganda  391862070
4    Template:Namibia-politician-stub   Namibia  391862409
df shape is (47197, 3)
```

In [334]:
```
print(WPDS_2018_data_csv.head())
print("df shape is " + str(WPDS_2018_data_csv.shape))
```

```
  Geography Population mid-2018 (millions)
0    AFRICA                          1,284
1   Algeria                           42.7
2     Egypt                             97
3     Libya                            6.5
4   Morocco                           35.2
df shape is (207, 2)
```

Part 1.2: Cleaning the data

As mentioned in the instructions [2], both page_data.csv and WPDS_2018_data.csv contain some rows that will need to be filtered out and/or ignore.

- `page_data.csv` Pages that start with 'template' are *not* Wikipedia articles.
- `WPDS_2018_data.csv` Rows with its entire name in upper case under "Geography" are continents, not country.

1.2.1: Working on `page_data.csv`

Extracting rows *without* the string "Template:" under "page".

- `.str.contains(somestring)` returns a boolean. It checks if the string contains *"somestring"*
- The `~` is used to negate a boolean series since we only want FALSE results.
- Calling the new dataframe `page_data`

In [335]:
```
page_data = page_data_csv[~page_data_csv["page"].str.contains("Template:")]
page_data.reset_index(drop=True, inplace=True)

# printing DF head to check results
page_data.head()
```

Out[335]:

	page	country	rev_id
0	Bir I of Kanem	Chad	355319463
1	Information Minister of the Palestinian Nation...	Palestinian Territory	393276188
2	Yos Por	Cambodia	393822005
3	Julius Gregr	Czech Republic	395521877
4	Edvard Gregr	Czech Republic	395526568

1.2.2: Working on `WPDS_2018_data.csv`

Removing rows with all uppercase string under "Geography."

- For all cells with only uppercase letters, the last letter has to be an upper case.
- For all country names with more than 1 letter, the last letter has to be a lower case.
- Therefore, we only have to extract the last letter of the row "Geography" and check its case.
- Extract all results that are TRUE for `.str.islower()`
- Call the new dataframe `WPDS_2018_data`

In [336]:
```
WPDS_2018_data = WPDS_2018_data_csv[WPDS_2018_data_csv["Geography"].str[-1].str.islower()]
WPDS_2018_data.reset_index(drop=True, inplace=True)

# printing DF head to check results
WPDS_2018_data.head()
```

Out[336]:

	Geography	Population mid-2018 (millions)
0	Algeria	42.7
1	Egypt	97
2	Libya	6.5
3	Morocco	35.2
4	Sudan	41.7

In [337]:
```
# sanity check on all .isupper() results. Expect all regional names instead of countries.
# save that as WPDS_2018_data_region
WPDS_2018_data_region = WPDS_2018_data_csv[WPDS_2018_data_csv["Geography"].str[-1].str.isupper()]
WPDS_2018_data_region.reset_index(drop=True, inplace=True)

WPDS_2018_data_region.head(10)
```

Out[337]:

	Geography	Population mid-2018 (millions)
0	AFRICA	1,284
1	NORTHERN AMERICA	365
2	LATIN AMERICA AND THE CARIBBEAN	649
3	ASIA	4,536
4	EUROPE	746
5	OCEANIA	41

Part 2: ORES results on article quality

Steps to obtain the ORES' prediction of article quality is taken from this Github page (https://github.com/Ironholds/data-512-a2/)
Articles are categorized into the following categories:

- A - Featured article
- GA - Good article
- B - B-class article
- C - C-class article
- Start - Start-class article
- Stub - Stub-class article

In [338]:
```
headers = {'User-Agent' : 'https://github.com/winnawat', 'From' : 'nawats@uw.edu'}

def get_ores_data(revision_ids, headers):
    '''
    This function retrieves ORES data given a list of revisin IDs.
    Function taken from https://github.com/Ironholds/data-512-a2/blob/master/hcds-a2-bias_demo.ipynb
    '''

    # Define the endpoint
    endpoint = 'https://ores.wikimedia.org/v3/scores/{project}/?models={model}&revids={revids}'

    # Specify the parameters - smushing all the revision IDs together separated by | marks.
    params = {'project' : 'enwiki',
              'model'   : 'wp10',
              'revids'  : '|'.join(str(x) for x in revision_ids)
              }
    api_call = requests.get(endpoint.format(**params))
    response = api_call.json()
    return response
```

In [339]:
```
def get_prediction_from_ores(revision_ids,ores_response):
    '''
    A function to extract "prediction" from the JSON output given by function get_ores_data
    Returns a dataframe with columns ['rev_id','prediction']
    '''
    predictions = {}
    for revid in revision_ids:
        try:
            predictions[revid] = ores_response['enwiki']['scores'][str(revid)]['wp10']['score']['prediction']
        except KeyError:
            predictions[revid] = "No prediction"

    pred_df = pd.DataFrame.from_dict(predictions, orient='index', columns=["prediction"])
    pred_df.reset_index(inplace=True)
    pred_df.rename(columns={"index":"rev_id"},inplace=True)
    return pred_df
```

The rev_ids are contained in the dataframe `page_data`. Extracting the list of rev_ids to pass to `get_ores_data` and `get_prediction_from_ores`

In [340]:
```
rev_ids = page_data['rev_id'].tolist()

# getting the idex of the last element of the list of rev_ids
last_id = len(rev_ids)

def make_api_calls(last_id, savecsv=True, filename="predictions.csv"):
    '''
    A function which takes in the last index of the rev IDs and make API calls
for 100 IDs at a time.
    Loop executed until it reaches index last_id
    '''
    pred_df = pd.DataFrame(columns=['rev_id', 'prediction'])
    last_index = 0

    # making API calls, 100 rev IDs at a time
    for i in range(100,last_id,100):
        subset_rev_ids = rev_ids[i-100:i]
        ores_json = get_ores_data(subset_rev_ids, headers)
        df1 = get_prediction_from_ores(subset_rev_ids, ores_json)
        pred_df = pred_df.append(df1)
        last_index = i

    # getting the remainder of red_ids that were left out by range()
    lastloop_ids = rev_ids[last_index:last_id]
    ores_json = get_ores_data(lastloop_ids, headers)
    df1 = get_prediction_from_ores(lastloop_ids, ores_json)
    pred_df = pred_df.append(df1)

    pred_df.reset_index(drop=True, inplace=True)

    # either save to csv or return a dataframe
    if (savecsv == True):
        pred_df.to_csv(filename)
    else:
        return pred_df
```

In [341]:
```
# saving this as a csv so the loop does not have to be re-run
# this has to be run once. From now on the ores predictions will be read from
 CSV.

# make_api_calls(last_id)
```

Part 3: Merging the data and generate CSV files

The CSV generated from above is read and merged with the existing data frames.
Any rows with no prediction available is output to `wp_wpds_countries-no_match.csv`
The rest of the data goes into `wp_wpds_politicians_by_country.csv`

In [342]:
```
predictions_csv = pd.read_csv("predictions.csv")
predictions = predictions_csv.drop(columns=['Unnamed: 0'])
page_data_pred = pd.merge(page_data, predictions, on='rev_id', how='outer')
```

In [343]:
```
page_data_pred.tail()
```

Out[343]:

	page	country	rev_id	prediction
46696	Yahya Jammeh	Gambia	807482007	GA
46697	Lucius Fairchild	United States	807483006	C
46698	Fahd of Saudi Arabia	Saudi Arabia	807483153	GA
46699	Francis Fessenden	United States	807483270	C
46700	Ajay Kannoujiya	India	807484325	No prediction

In [344]:
```
WPDS_2018_data.tail()
```

Out[344]:

	Geography	Population mid-2018 (millions)
195	Samoa	0.2
196	Solomon Islands	0.7
197	Tonga	0.1
198	Tuvalu	0.01
199	Vanuatu	0.3

Renaming 'Geography' column in `WPDS_2018_data` to 'country'

In [345]:
```
WPDS_2018_data = WPDS_2018_data.rename(columns={'Geography':'country'})
WPDS_2018_data.tail()
```

Out[345]:

	country	Population mid-2018 (millions)
195	Samoa	0.2
196	Solomon Islands	0.7
197	Tonga	0.1
198	Tuvalu	0.01
199	Vanuatu	0.3

Merging `WPDS_2018_data` to `page_data_pred` by 'country'

In [346]:
```
page_data_pred_country = pd.merge(page_data_pred, WPDS_2018_data, on='country'
, how='outer')
page_data_pred_country.tail()
```

Out[346]:

	page	country	rev_id	prediction	Population mid-2018 (millions)
46716	NaN	French Polynesia	NaN	NaN	0.3
46717	NaN	Guam	NaN	NaN	0.2
46718	NaN	New Caledonia	NaN	NaN	0.3
46719	NaN	Palau	NaN	NaN	0.02
46720	NaN	Samoa	NaN	NaN	0.2

Formatting the column names to the specified schema

Column
country
article_name
revision_id
article_quality
population

In [347]:
```
page_data_pred_country = page_data_pred_country.rename(columns={'page':'articl
e_name',
                                                'rev_id':'revision_id',
                                                'prediction':'article_quality',
                                                'Population mid-2018 (millions)':'popul
ation'})
```

Checking for rows with NaN.

In [348]:
```
np.sum(page_data_pred_country.isna())
```

Out[348]:
```
article_name         20
country               0
revision_id          20
article_quality      20
population         2083
dtype: int64
```

There are 20 rows with page, red_id, and prediction missing and 2083 rows with population missing. Saving these rows in a separate CSV `wp_wpds_countries-no_match.csv` before dropping them.

In [349]:
```
pop_na = page_data_pred_country[page_data_pred_country['population'].isna()]
page_na = page_data_pred_country[page_data_pred_country['article_name'].isna
()]
```

In [350]:
```
all_na = pop_na.append(page_na)
all_na.reset_index(drop=True, inplace=True)
all_na.to_csv('wp_wpds_countries-no_match.csv')
```

Dropping all rows with na and make sure the red_id column is an integer.

In [351]:
```
page_data_pred_country = page_data_pred_country.dropna()
page_data_pred_country.reset_index(drop=True, inplace=True)
page_data_pred_country = page_data_pred_country.astype({'revision_id': 'int'})
page_data_pred_country.tail()
```

Out[351]:

	article_name	country	revision_id	article_quality	population
44613	Rita Sinon	Seychelles	800323154	Stub	0.1
44614	Sylvette Frichot	Seychelles	800323798	Stub	0.1
44615	May De Silva	Seychelles	800969960	Start	0.1
44616	Vincent Meriton	Seychelles	802051093	Stub	0.1
44617	Marie-Louise Potter	Seychelles	804209620	Stub	0.1

Saving the dataframe to CSV

In [352]:
```
page_data_pred_country.to_csv('wp_wpds_politicians_by_country.csv')
```

Part 4.1: Analysis - by country

To estimate the coverage and quality of Wikipedia articles for each country, the following data has to be aggregated by country:

- Number of articles
- Number of high-quality articles
  - High quality articles are those labelled FA and GA

In [353]:
```
# Labeling the high-quality articles
page_data_pred_country['HQA'] = page_data_pred_country['article_quality'].isin
(['FA','GA']).astype(int)
```

In [354]:
```
# Checking the head of the table
page_data_pred_country.head()
```

Out[354]:

	article_name	country	revision_id	article_quality	population	HQA
0	Bir I of Kanem	Chad	355319463	Stub	15.4	0
1	Abdullah II of Kanem	Chad	498683267	Stub	15.4	0
2	Salmama II of Kanem	Chad	565745353	Stub	15.4	0
3	Kuri I of Kanem	Chad	565745365	Stub	15.4	0
4	Mohammed I of Kanem	Chad	565745375	Stub	15.4	0

In [355]:
```
# Checking the high quality articles
page_data_pred_country[page_data_pred_country['HQA'] == 1].head()
```

Out[355]:

	article_name	country	revision_id	article_quality	population	HQA
61	Mahamat Nouri	Chad	792954115	FA	15.4	1
83	Hissène Habré	Chad	803166806	GA	15.4	1
262	Norodom Chakrapong	Cambodia	788905950	GA	16	1
282	Norodom Sihanouk	Cambodia	799302232	FA	16	1
303	Nuon Chea	Cambodia	805876135	GA	16	1

Aggregating the data

In [356]:
```
data_by_country = pd.DataFrame(page_data_pred_country.groupby('country')['revi
sion_id'].count())
data_by_country = data_by_country.rename(columns={'revision_id':'article_coun
t'})
data_by_country['HQA_count'] = pd.DataFrame(page_data_pred_country.groupby('co
untry')['HQA'].sum())
data_by_country = pd.merge(data_by_country, WPDS_2018_data, on='country')
data_by_country = data_by_country.rename(columns={'Population mid-2018 (millio
ns)':'population'})
# removing the commas
data_by_country['population'] = data_by_country['population'].str.replace(',',
'')
data_by_country['population'] = data_by_country['population'].astype(float)

# getting the coverage percentage
data_by_country['coverage_pcnt'] = data_by_country['article_count'] / data_by_
country['population'] /10000

# getting the percentage of high quality articles
data_by_country['quality_pcnt'] = data_by_country['HQA_count'] / data_by_count
ry['article_count'] * 100

data_by_country.head()
```

Out[356]:

	country	article_count	HQA_count	population	coverage_pcnt	quality_pcnt
0	Afghanistan	322	12	36.50	0.000882	3.726708
1	Albania	457	3	2.90	0.015759	0.656455
2	Algeria	116	2	42.70	0.000272	1.724138
3	Andorra	34	0	0.08	0.042500	0.000000
4	Angola	106	0	30.40	0.000349	0.000000

Part 4.2: Analysis - by region

To look at the coverage and quality of articles by region, we have to label the Wikipedia articles by these regions instead of the countries.
The countries within the region come after the region name in the original CSV.

- Extracting the index of the regions

In [357]:
```
WPDS_2018_data_csv[WPDS_2018_data_csv['Geography'].str[-1].str.isupper()]
```

Out[357]:

	Geography	Population mid-2018 (millions)
0	AFRICA	1,284
56	NORTHERN AMERICA	365
59	LATIN AMERICA AND THE CARIBBEAN	649
95	ASIA	4,536
144	EUROPE	746
189	OCEANIA	41

In [358]:
```
region_index = list(WPDS_2018_data_csv[WPDS_2018_data_csv['Geography'].str[-1]
.str.isupper()].index)
region_index
```

Out[358]:
```
[0, 56, 59, 95, 144, 189]
```

Therefore, countries in row 1 to 55 are from Africa region and similar pattern applies to other regions.

- Creating the namelist of countries in the regions

In [359]:
```
reg_africa = list(WPDS_2018_data_csv['Geography'][region_index[0]+1:region_ind
ex[1]])
reg_NA = list(WPDS_2018_data_csv['Geography'][region_index[1]+1:region_index[2
]])
reg_latin = list(WPDS_2018_data_csv['Geography'][region_index[2]+1:region_inde
x[3]])
reg_asia = list(WPDS_2018_data_csv['Geography'][region_index[3]+1:region_index
[4]])
reg_eu = list(WPDS_2018_data_csv['Geography'][region_index[4]+1:region_index[5
]])
reg_oceania = list(WPDS_2018_data_csv['Geography'][region_index[5]+1:])
```

In [360]:
```
# labelling the region of each article
def to_region(country):
    '''
    This function takes in a country name and return the region it belongs to.
    '''
    if country in reg_africa:
        return 'Africa'
    elif country in reg_NA:
        return 'Northern America'
    elif country in reg_latin:
        return 'Latin America and the Caribbean'
    elif country in reg_asia:
        return 'Asia'
    elif country in reg_eu:
        return 'Europe'
    elif country in reg_oceania:
        return 'Oceania'
    else:
        return 'No matching region'
```

In [361]:
```
page_data_pred_country['region'] = page_data_pred_country['country'].apply(to_
region)
```

In [362]:
```
data_by_region = pd.DataFrame(page_data_pred_country.groupby('region')['revisi
on_id'].count())
data_by_region = data_by_region.rename(columns={'revision_id':'article_count'
})
data_by_region['HQA_count'] = pd.DataFrame(page_data_pred_country.groupby('reg
ion')['HQA'].sum())
data_by_region = data_by_region.reset_index()
data_by_region['Geography'] = data_by_region['region'].str.upper()
data_by_region = pd.merge(data_by_region,
                          WPDS_2018_data_csv[WPDS_2018_data_csv['Geography'].s
tr[-1].str.isupper()],
                          on='Geography')
data_by_region = data_by_region.rename(columns={'Population mid-2018 (million
s)':'population'})
data_by_region = data_by_region.drop(columns=['Geography'])

# removing the commas
data_by_region['population'] = data_by_region['population'].str.replace(',',
'')
data_by_region['population'] = data_by_region['population'].astype(float)

# getting the coverage percentage
data_by_region['coverage_pcnt'] = data_by_region['article_count'] / data_by_re
gion['population'] /10000

# getting the percentage of high quality articles
data_by_region['quality_pcnt'] = data_by_region['HQA_count'] / data_by_region[
'article_count'] * 100

data_by_region.head()
```

Out[362]:

	region	article_count	HQA_count	population	coverage_pcnt	quality_pcnt
0	Africa	6861	125	1284.0	0.000534	1.821892
1	Asia	11588	310	4536.0	0.000255	2.675181
2	Europe	15923	322	746.0	0.002134	2.022232
3	Latin America and the Caribbean	5174	69	649.0	0.000797	1.333591
4	Northern America	1940	99	365.0	0.000532	5.103093

Part 5: Results

We are interested in 6 results

- Top 10 countries by coverage
- Bottom 10 countries by coverage
- Top 10 countries by relative quality
- Bottom 10 countries by relative quality
- Geographic regions by coverage
- Geographic regions by quality

Part 5.1: Top 10 countries by coverage

Here, we find the top 10 countries in terms of coverage by sorting(descending) the table obtained above by coverage.

In [363]:
```
top_10_cov = data_by_country.sort_values(by='coverage_pcnt', ascending=False).
head(10)
top_10_cov
```

Out[363]:

	country	article_count	HQA_count	population	coverage_pcnt	quality_pcnt
166	Tuvalu	54	5	0.01	0.540000	9.259259
115	Nauru	52	0	0.01	0.520000	0.000000
135	San Marino	81	0	0.03	0.270000	0.000000
108	Monaco	40	0	0.04	0.100000	0.000000
93	Liechtenstein	28	0	0.04	0.070000	0.000000
161	Tonga	63	0	0.10	0.063000	0.000000
103	Marshall Islands	37	0	0.06	0.061667	0.000000
68	Iceland	202	2	0.40	0.050500	0.990099
3	Andorra	34	0	0.08	0.042500	0.000000
61	Grenada	36	1	0.10	0.036000	2.777778

Part 5.2: Bottom 10 countries by coverage

Here, we find the bottom 10 countries in terms of coverage by sorting(ascending) the table obtained above by coverage.

In [364]:
```
bot_10_cov = data_by_country.sort_values(by='coverage_pcnt', ascending=True).h
ead(10)
bot_10_cov
```

Out[364]:

	country	article_count	HQA_count	population	coverage_pcnt	quality_pcnt
69	India	985	17	1371.3	0.000072	1.725888
70	Indonesia	211	10	265.2	0.000080	4.739336
34	China	1133	41	1393.8	0.000081	3.618711
173	Uzbekistan	28	2	32.9	0.000085	7.142857
51	Ethiopia	101	2	107.5	0.000094	1.980198
82	Korea, North	36	7	25.6	0.000141	19.444444
178	Zambia	25	0	17.7	0.000141	0.000000
159	Thailand	112	3	66.2	0.000169	2.678571
112	Mozambique	58	0	30.5	0.000190	0.000000
13	Bangladesh	321	3	166.4	0.000193	0.934579

Part 5.3: Top 10 countries by quality

Here, we find the top 10 countries in terms of article quality by sorting(descending) the table obtained above by relative quality.

In [365]:
```
top_10_qual = data_by_country.sort_values(by='quality_pcnt', ascending=False).
head(10)
top_10_qual
```

Out[365]:

	country	article_count	HQA_count	population	coverage_pcnt	quality_pcnt
82	Korea, North	36	7	25.60	0.000141	19.444444
137	Saudi Arabia	118	15	33.40	0.000353	12.711864
104	Mauritania	48	6	4.50	0.001067	12.500000
31	Central African Republic	66	8	4.70	0.001404	12.121212
132	Romania	343	39	19.50	0.001759	11.370262
166	Tuvalu	54	5	0.01	0.540000	9.259259
19	Bhutan	33	3	0.80	0.004125	9.090909
44	Dominica	12	1	0.07	0.017143	8.333333
155	Syria	129	10	18.30	0.000705	7.751938
18	Benin	91	7	11.50	0.000791	7.692308

Part 5.4: Bottom 10 countries by quality

Here, we find the bottom 10 countries in terms of quality by sorting(ascending) the table obtained above by relative quality.

In [366]:
```
bot_10_qual = data_by_country.sort_values(by='quality_pcnt', ascending=True).h
ead(10)
bot_10_qual
```

Out[366]:

	country	article_count	HQA_count	population	coverage_pcnt	quality_pcnt
143	Slovakia	116	0	5.40	0.002148	0.0
54	Finland	570	0	5.50	0.010364	0.0
52	Federated States of Micronesia	36	0	0.10	0.036000	0.0
102	Malta	103	0	0.50	0.020600	0.0
50	Estonia	149	0	1.30	0.011462	0.0
49	Eritrea	16	0	6.00	0.000267	0.0
115	Nauru	52	0	0.01	0.520000	0.0
114	Namibia	162	0	2.50	0.006480	0.0
165	Turkmenistan	32	0	5.90	0.000542	0.0
28	Cameroon	104	0	25.60	0.000406	0.0

Part 5.5: Geographic regions by coverage

Since the coverge by region is already been generated in section 4, sorting it in descending order:

In [367]:
```
reg_cov = data_by_region.sort_values(by='coverage_pcnt', ascending=False)
reg_cov
```

Out[367]:

	region	article_count	HQA_count	population	coverage_pcnt	quality_pcnt
5	Oceania	3132	66	41.0	0.007639	2.107280
2	Europe	15923	322	746.0	0.002134	2.022232
3	Latin America and the Caribbean	5174	69	649.0	0.000797	1.333591
0	Africa	6861	125	1284.0	0.000534	1.821892
4	Northern America	1940	99	365.0	0.000532	5.103093
1	Asia	11588	310	4536.0	0.000255	2.675181

Part 5.6: Geographic regions by quality

Since the quality by region is already been generated in section 4, sorting it in descending order:

In [368]:
```
reg_qual = data_by_region.sort_values(by='quality_pcnt', ascending=False)
reg_qual
```

Out[368]:

	region	article_count	HQA_count	population	coverage_pcnt	quality_pcnt
4	Northern America	1940	99	365.0	0.000532	5.103093
1	Asia	11588	310	4536.0	0.000255	2.675181
5	Oceania	3132	66	41.0	0.007639	2.107280
2	Europe	15923	322	746.0	0.002134	2.022232
0	Africa	6861	125	1284.0	0.000534	1.821892
3	Latin America and the Caribbean	5174	69	649.0	0.000797	1.333591

Part 6: Reflections and implications

It is rather difficult to tell a coherent story using just the coverage and quality of Wikipedia articles on politicians for a country. There are many other factors that could influence the number of articles as well as their qualities. Before beginning this analysis, I suspected that countries with extreme values of population, both large and small, could significantly skew the analysis. This suspicion is confirmed by the analysis in part 5. Countries with very small population dominate the top spots for coverage. In contrast, populous countries are found at the bottom of the coverage ranking. This is likely to be due to the fact that the number of politicians, let alone those who are noteworthy enough to have a Wikipedia article, does not scale well with the population. We do not expect India, with population of 1,371 million to have 13,710 times more politician with Wikipedia article about them than Tuvalu whose population is 0.1 million.

Another glaring source of bias is the fact that only data from *English* Wikipedia is used. This leads us to expect that English-speaking countries will have more, and better, articles on Wikipedia than countries where English is not their main language. This is not reflected in the top coverage list since it is mainly dominated by countries with small population. However, we can clearly see this trend in the quality by region list in part 5.6. Despite its lower coverage, North America region tops the list by a large margin when it comes to quality. Also, from part 5.2, the countries with lowest coverage are countries where English is not their main language of communication.

To correct for this, data from Wikipedia articles in other language could be included. If not, the coverage and quality score can be weighted by the percentage of population who can read and write in English. There are many other factors that could affect coverage and quality values. Some of them may actually be worth looking into as a research topic. For instance, internet access might also affect the coverage value of a country. It is highly possible that coutries with low internet poliferation will have lower coverage value. Therefore, we can track how countires' internet access changes through the coverage and quality of their Wikipedia articles. This may even yield a deeper insights into internet *usage* pattern for a country rather than just the degree of internet access since Wikipedia can be loosely translate to individuals useing the internet for education and not entertainment.

In addition, the quality of Wikipedia articles on politicians may reveal some interesting trends of the public's interest towards a country's political relationship with the rest of the world. Notable countries with remarkably high qualty value are North Korea and Saudi Arabia. These are countries whose leaders were taking the spotlight in the international media for a notable amount of time. Public interest may lead to articles on politicians in North Korean and Saudi Arabia being edited and improved, resulting in higher quality. This could be useful for researchers looking to find some sort of measure for international interest in a country's potitics . Still, the coverage and quality measures have quite limited usefulness on their own due to the inherent biases and confounding variables. This dataset should be supplemented with the countries' indicators such as internet poliferation, main language, urban population, political stability and perhaps education.

Appendix

[1]: DATA516 Course wiki, A2 assignment:
https://wiki.communitydata.science/Human_Centered_Data_Science_(Fall_2019)/Assignments#A2:_Bias_in_data
(https://wiki.communitydata.science/Human_Centered_Data_Science_(Fall_2019)/Assignments#A2:_Bias_in_data