Introduction

The proposed assessment involves analyzing a typical annotation file used in computational biology, specifically focusing on protein-coding genes. The main tasks include retrieving the annotation file, parsing it to extract relevant information, converting the data into appropriate formats, writing the parsed content to a JSON file, counting the number of protein-coding genes per chromosome, and visualizing these counts. My solution involves using Python and its libraries to automate these tasks efficiently. I approached the problem by breaking down each requirement into manageable steps, ensuring clarity and accuracy at each stage.

Assignment Instructions

1. Retrieve the file from the provided URL **[1]**
2. Parse the file into a suitable in-memory data structure
3. Convert each column to an appropriate data type.
4. Write the parsed contents to a JSON file
5. Produce a count of the number of protein-coding genes on each chromosome
6. Create a visualization of the per-chromosome protein-coding gene counts

Breakdown of Problem

1. **Retrieving the File:** Use Python's *'request'* library to download the file.
2. **Parsing the File:** Read the file into a Pandas DataFrame for easy manipulation
3. **Data Type Conversion:** Convert Columns to appropriate data types to ensure accuracy
4. **JSON Output:** Write the parsed data into a JSON file for easy access and readability.
5. **Counting Genes:** Filter the DataFrame to count the number of protein-coding genes per chromosome
6. **Visualization:** Use '*matplotib'* and *'seaborn'* to create bar plot visualizing the gene counts per chromosome.

Solution

1. **Retrieving the File :**
   - Use the '*request'* library to download the annotation file from the provided URL
2. **Parsing the File :**
   - Read the downloaded file into Pandas DataFrame using '*pd.read_csv'*.
3. **Data Type Conversion :**
   - Ensure columns such as *'#tax_id'* and *'GeneID'* are converted to integers.
4. **JSON Output:**
   - Use the *'to_json'* method of the DataFrame to write the parsed data to a JSON file.
5. **Counting Genes:**
   - Filter the DataFrame to include only protein-coding genes and use *'value_counts'* to count occurrences per chromosome.
6. **Visualization :**
   - Create a bar plot using *'matplotlib'* and *'seaborn'* to display the counts,

Psuedo-Code

```
Pseudocode
Step 1. Download and Unzip File:

Define a function download_and_unzip with parameters url and output_filename.
Send a GET request to the URL and save the response as a .gz file.
Open the .gz file and extract its contents to a .txt file.

Step 2. Parse the File:

Define a function parse_file with parameter filename.
Read the .txt file into a DataFrame with tab-separated values.
Convert the #tax_id column to integer type.
Return the DataFrame.
Step 3. Write to JSON:

Define a function write_to_json with parameters df and output_filename.
Convert the DataFrame to JSON format and save it.
Step 4. Count Protein-Coding Genes per Chromosome:

Define a function count_protein_coding_genes with parameter df.
Filter the DataFrame for protein-coding genes.
Count the occurrences of each chromosome.
Return the counts.
Step 5. Create Visualization:

Define a function create_visualization with parameter chromosome_counts.
Create a bar plot with chromosomes on the x-axis and gene counts on the y-axis.
Set labels, title, and rotate x-axis labels.
Display the plot.
```

1. **File Retrieval and Extraction:** It downloads a compressed gene information file from the NCBI FTP server, uncompresses it, and saves it as a text file.
2. **Data Parsing:** It reads the uncompressed file into a Pandas DataFrame, specifying that the `#tax_id` column should be treated as integers for accurate processing.
3. **JSON Conversion:** The DataFrame is converted into a JSON format and saved, which makes it easier to work with in different applications or share with others.
4. **Gene Counting:** The script filters the DataFrame to count protein-coding genes for each chromosome, which provides insight into gene distribution across chromosomes.
5. **Visualization:** Finally, it generates a bar plot to visually represent the count of protein-coding genes per chromosome, helping to quickly understand the distribution of these genes.

Python-Code
Step 1

```python
# Step 1: Retrieve and unzip the file
def download_and_unzip(url, output_filename):
    response = requests.get(url, stream=True)
    with open(output_filename + '.gz', 'wb') as out_file:
        shutil.copyfileobj(response.raw, out_file)
    with gzip.open(output_filename + '.gz', 'rb') as f_in:
        with open(output_filename, 'wb') as f_out:
            shutil.copyfileobj(f_in, f_out)

download_and_unzip("https://ftp.ncbi.nlm.nih.gov/gene/DATA/GENE_INFO/Mammalia/Homo_sapiens.gene_info.gz", "Homo_sapiens.gene_info.txt")
```

Explanation:

1. **Imports:**
    - `requests`: Library for making HTTP requests.
    - `gzip`: Module for working with gzip-compressed files.
    - `shutil`: Module for high-level file operations.
2. **Function `download_and_unzip(url, output_filename)`:**
    - **Downloading the file:**
        - `requests.get(url, stream=True)`: Sends a GET request to the specified URL to stream the file.
        - `with open(output_filename + '.gz', 'wb') as out_file`: Opens a file in write-binary mode to save the downloaded gzip file.
        - `shutil.copyfileobj(response.raw, out_file)`: Copies the streamed response content to the output file.
    - **Unzipping the file:**
        - `with gzip.open(output_filename + '.gz', 'rb') as f_in`: Opens the downloaded gzip file in read-binary mode.
        - `with open(output_filename, 'wb') as f_out`: Opens the output file in write-binary mode.
        - `shutil.copyfileobj(f_in, f_out)`: Copies the content from the gzip file to the output file.

3.**Usage:**

- Downloads and extracts the file from the provided URL, saving it as `Homo_sapiens.gene_info.txt`.

```python
# Step 2: Parse the file
def parse_file(filename):
    df = pd.read_csv(filename, sep="\t", dtype=str)
    df['#tax_id'] = df['#tax_id'].astype(int)
    return df

df = parse_file("Homo_sapiens.gene_info.txt")  # Humans gene information
```

Explanation:

1. **Imports:**
    - `pandas as pd`: Library for data manipulation and analysis.

2. **Function `parse_file(filename)`:**
    - **Reading the file:**
        - `pd.read_csv(filename, sep="\t", dtype=str)`: Reads the tab-separated file into a DataFrame, treating all columns as strings.
    - **Data type conversion:**
        - `df['#tax_id'] = df['#tax_id'].astype(int)`: Converts the `#tax_id` column to integers for correct data representation.
    - **Return:**
        - Returns the parsed DataFrame.
3. **Usage:**
    - Parses the `Homo_sapiens.gene_info.txt` file into a DataFrame `df`.

```
# Step 3: Write to JSON
def write_to_json(df, output_filename):
    df.to_json(output_filename, orient="records", lines=True)

write_to_json(df, "Homo_sapiens_gene_info.json")
```

Explanation:

1. **Imports:**
    - `json`: Library for working with JSON data.
2. **Function `write_to_json(df, output_filename)`:**
    - **Writing DataFrame to JSON:**
        - `df.to_json(output_filename, orient="records", lines=True)`: Converts the DataFrame to a JSON file with each record as a separate line.
3. **Usage:**
    - Writes the parsed DataFrame to a JSON file named `Homo_sapiens_gene_info.json`.

```
# Step 4: Count Protein-coding genes per chromosome
def count_protein_coding_genes(df):
    protein_coding_genes = df[df['type_of_gene'] == 'protein-coding']
    chromosome_counts = protein_coding_genes['chromosome'].value_counts()
    return chromosome_counts

chromosome_counts = count_protein_coding_genes(df)
print(chromosome_counts)
```

Explanation:

1. **Function `count_protein_coding_genes(df)`:**
    - **Filtering protein-coding genes:**
        - `df[df['type_of_gene'] == 'protein-coding']`: Filters the DataFrame to include only rows where the `type_of_gene` is 'protein-coding'.
    - **Counting genes per chromosome:**

        - `protein_coding_genes['chromosome'].value_counts()`: Counts the occurrences of each chromosome in the filtered DataFrame.
    - **Return:**
        - Returns the counts of protein-coding genes per chromosome as a Series.
2. **Usage:**
    - Counts and prints the number of protein-coding genes per chromosome.

```
# Step 5: Create Visualization
def create_visualization(chromosome_counts):
    plt.figure(figsize=(12, 6))
    sns.barplot(x=chromosome_counts.index, y=chromosome_counts.values)
    plt.xlabel('Chromosome')
    plt.ylabel('Protein-Coding Gene Count')
    plt.title('Protein-Coding Gene Count Per Chromosome')
    plt.xticks(rotation=90)
    plt.show()

create_visualization(chromosome_counts)
```

Explanation:

1. **Imports:**
    - `seaborn as sns`: Library for statistical data visualization.
    - `matplotlib.pyplot as plt`: Library for creating static, animated, and interactive visualizations.
2. **Function `create_visualization(chromosome_counts)`:**
    - **Plot setup:**
        - `plt.figure(figsize=(12, 6))`: Sets the figure size.
    - **Creating the bar plot:**
        - `sns.barplot(x=chromosome_counts.index, y=chromosome_counts.values)`: Creates a bar plot with chromosomes on the x-axis and gene counts on the y-axis.
    - **Labeling the plot:**
        - `plt.xlabel('Chromosome')`: Sets the x-axis label.
        - `plt.ylabel('Protein-Coding Gene Count')`: Sets the y-axis label.
        - `plt.title('Protein-Coding Gene Count Per Chromosome')`: Sets the plot title.
        - `plt.xticks(rotation=90)`: Rotates x-axis labels for better readability.
    - **Displaying the plot:**
        - `plt.show()`: Displays the plot.
3. **Usage:**
    - Creates and displays a bar plot visualizing the count of protein-coding genes per chromosome.

Testing/Debugging

Results

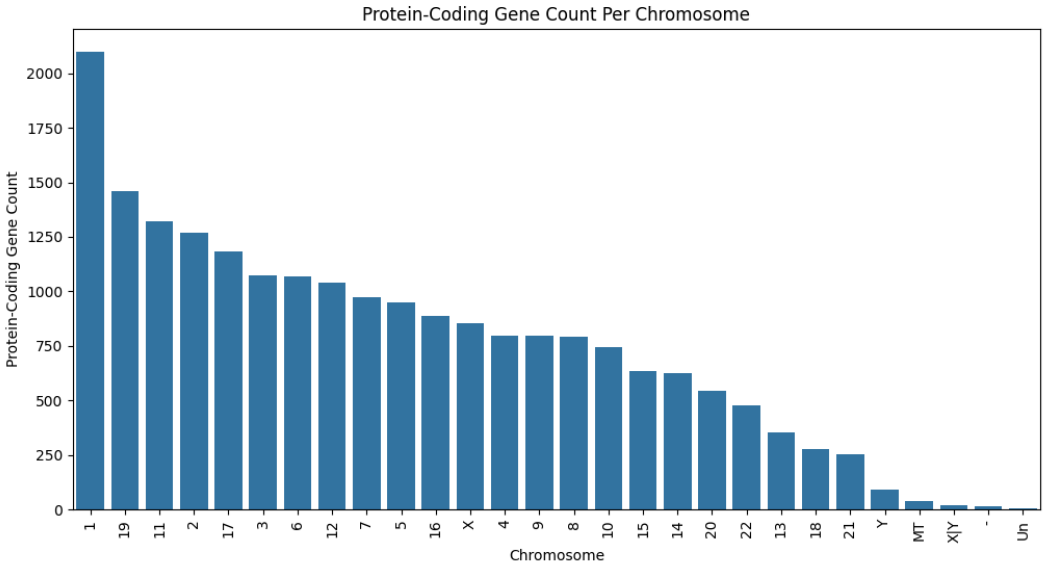
The expected results include the JSON file containing the parsed annotation data and a bar plot visualizating the number of protein-coding genes per chromosome. The bar plot is expected to show chromosome 1

with the highest number of protein-coding genes, followed by the other chromosomes in descending order of gene count.

Terminal Results

```
chromosome
1       2099
19      1459
11      1322
2       1269
17      1181
3       1072
6       1069
12      1038
7        975
5        948
16       886
X        852
4        797
9        796
8        792
10       742
15       634
14       627
20       546
22       477
13       355
18       277
21       254
Y         91
MT        39
X|Y       20
-         13
Un         5
Name: count, dtype: int64
```

Bar Plot Results



Resources

1. **Libraries used**
   - https://pandas.pydata.org/docs/
   - https://requests.readthedocs.io/en/latest/
   - https://matplotlib.org/stable/users/index.html
   - https://seaborn.pydata.org/
2. **Python Functions used**
   - https://www.w3schools.com/python/python_ref_string.asp
   - https://docs.python.org/3/library/stdtypes.html
   - https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
   - https://matplotlib.org/stable/api/pyplot_summary.html
3. **Information on Protein-Coding Genes in Chromosomes**
   - https://www.ncbi.nlm.nih.gov/gene
   - https://en.wikipedia.org/wiki/Human_genome#Protein-coding_genes
   - https://www.genome.gov/human-genome-project