Assignment2

September 13, 2024

0.1 Assignment 2: Data, Grammar and Engineering

AMS MADE YMS31303 Q1 2024 Instructor: Theodoros Chatzivasileiadis **Instructor:** Hans Hoogenboom **TA:** Ka Yi Chua **Metropolitan Data 1**

0.2 Question 1

We will work with data from the Guardian's version of Wikileaks' Afghanistan war logs. The table is stored on a GoogleDoc on the following address:

https://docs.google.com/spreadsheets/d/1EAx8_ksSCmoWW_SlhFyq2QrRn0FNNhcg1TtDFJzZRgc/edit?hl=en#gid=1

Write a Python code snippet using IPython.display to embed this Google Sheet directly into a Jupyter notebook for easy reference and interaction.

[76]:
```
from IPython.display import IFrame #import IFrame

url = "https://docs.google.com/spreadsheets/d/
↪1EAx8_ksSCmoWW_SlhFyq2QrRn0FNNhcg1TtDFJzZRgc/edit?hl=en#gid=1" #saving url␣
↪to variable

IFrame(url, width=800, height=500) #load screen
```

[76]:
```
<IPython.lib.display.IFrame at 0x79224baa3880>
```

0.3 Question 2

a) Save the csv file to your compluter and Load the data in Tab "TOTAL Casualties".

b) Display the first 5 rows of the dataset using .head().

c) Extract the column names and create a dictionary where each column name is the key, and the first value in that column is the value.

0.3.1 2 Load, display and clean wikileaks table

a) raw load

```
[77]: import pandas as pd

      wikileaks_dataframe=pd.read_csv("wikileaks.csv") #loading original

      wikileaks_dataframe.head() #display first 5 rows
```

```
[77]:   Casualties detailed in the war logs, month by month Unnamed: 1 Unnamed: 2  \
      0                                                       Year      Month    Taliban
      1                                                       2004    January         15
      2                                                       2004   February        NaN
      3                                                       2004      March         19
      4                                                       2004      April          5

        Unnamed: 3     Unnamed: 4                   Unnamed: 5  \
      0  Civilians  Afghan forces  Nato (detailed in spreadsheet)
      1         51             23                          NaN
      2          7              4                            5
      3          2            NaN                            2
      4          3             19                          NaN

                    Unnamed: 6
      0  Nato - official figures
      1                       11
      2                        2
      3                        3
      4                        3
```

b) Change columns, as they weren't the desired names.

```
[78]: wikileaks_dataframe=pd.read_csv("wikileaks.csv", header=1) #adjust column names␣
       ↪to correct
      wikileaks_dataframe.head() #display first 5 rows
```

```
[78]:     Year     Month Taliban Civilians Afghan forces  \
      0  2004.0    January      15        51            23
      1  2004.0   February     NaN         7             4
      2  2004.0      March      19         2           NaN
      3  2004.0      April       5         3            19
      4  2004.0        May      18        29            56

        Nato (detailed in spreadsheet)  Nato - official figures
      0                            NaN                     11.0
      1                              5                      2.0
      2                              2                      3.0
      3                            NaN                      3.0
      4                              6                      9.0
```

c) Table correct columns, make dictionary

```
[79]: dictionary = {} #empty dictionary to load with first row values

      column_names = list(wikileaks_dataframe.columns) #make columns list #https://
       ↪www.geeksforgeeks.org/how-to-get-column-names-in-pandas-dataframe/

      for column in column_names: #go through each column
          dictionary[column] = wikileaks_dataframe[column].iloc[0] #grab first item␣
       ↪from that column #https://www.geeksforgeeks.org/
       ↪how-to-get-first-row-of-pandas-dataframe/

      print(dictionary)
```

```
{'Year': np.float64(2004.0), 'Month': 'January', 'Taliban': '15', 'Civilians':
'51', 'Afghan forces': '23', 'Nato (detailed in spreadsheet)': nan, 'Nato -
official figures': np.float64(11.0)}
```

d) Clean: extract total values, fill NaN values with 0, make all numbers integers as you cannot have 0.5 years or 0.5 people.

```
[80]: wikileaks_dataframe.tail() #check bottom, as error found in totals, "," and in␣
       ↪wrong columns
```

```
[80]:      Year      Month Taliban Civilians Afghan forces  \
      68  2009.0  September     614       197           133
      69  2009.0    October     462       107            86
      70  2009.0   November     410       120            69
      71  2009.0   December     287        87            65
      72     NaN      24498  15,507     4,024         3,820

         Nato (detailed in spreadsheet)  Nato - official figures
      68                             54                     70.0
      69                             76                     74.0
      70                             30                     32.0
      71                             33                     35.0
      72                          1,147                      NaN
```

```
[81]: totals =  wikileaks_dataframe.iloc[-1] #save totals potentially for later

      wikileaks_dataframe = wikileaks_dataframe.drop(index=72)  # removes the last␣
       ↪row, index 72 is the index of this totals row

      column_names.remove("Month") #remove Month as is correctly a string and does␣
       ↪not need to be cleaned

      for column in column_names: #for every column that need to be cleaned
          wikileaks_dataframe[column].fillna(0, inplace=True) #fill NaN with 0
          wikileaks_dataframe[column] = wikileaks_dataframe[column].astype(int)␣
       ↪#change the float or string to integer
```

```python
wikileaks_dataframe.tail() #display last 5 rows to check correctly cleaning and␣
↪that totals has been dropped.
```

```
/tmp/ipykernel_45170/107033045.py:8: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  wikileaks_dataframe[column].fillna(0, inplace=True) #fill NaN with 0
/tmp/ipykernel_45170/107033045.py:8: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  wikileaks_dataframe[column].fillna(0, inplace=True) #fill NaN with 0
/tmp/ipykernel_45170/107033045.py:8: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  wikileaks_dataframe[column].fillna(0, inplace=True) #fill NaN with 0
/tmp/ipykernel_45170/107033045.py:8: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
```

```
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  wikileaks_dataframe[column].fillna(0, inplace=True) #fill NaN with 0
/tmp/ipykernel_45170/107033045.py:8: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  wikileaks_dataframe[column].fillna(0, inplace=True) #fill NaN with 0
/tmp/ipykernel_45170/107033045.py:8: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  wikileaks_dataframe[column].fillna(0, inplace=True) #fill NaN with 0
```

[81]:

```
    Year      Month  Taliban  Civilians  Afghan forces  \
67  2009     August      445        206            190
68  2009  September      614        197            133
69  2009    October      462        107             86
70  2009   November      410        120             69
71  2009   December      287         87             65

    Nato (detailed in spreadsheet)  Nato - official figures
67                              64                       77
68                              54                       70
69                              76                       74
```

```
70                                  30                         32
71                                  33                         35
```

0.4 Question 3

a) Plot a histogram of any numeric column from the dataset.

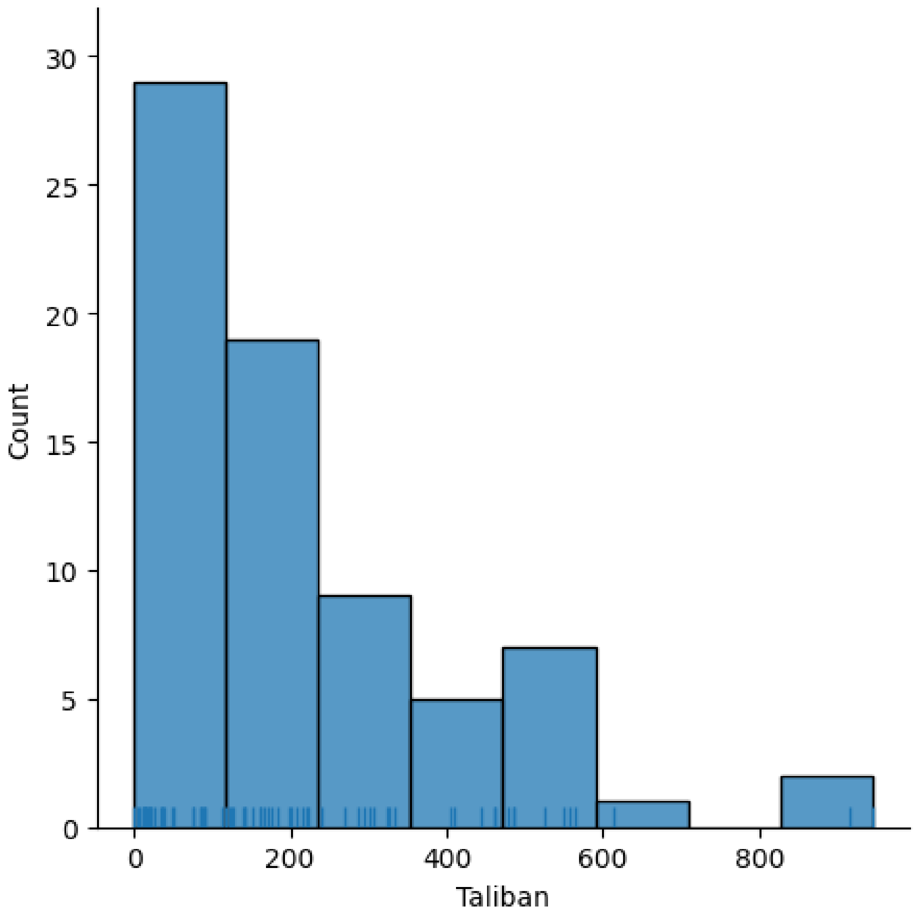
b) Use a kernel density plot (KDE) to visualize the distribution of another numeric column.

c) Create a bar chart of a categorical variable and discuss how the plot changes if you switch to a horizontal bar plot.

a) Histogram

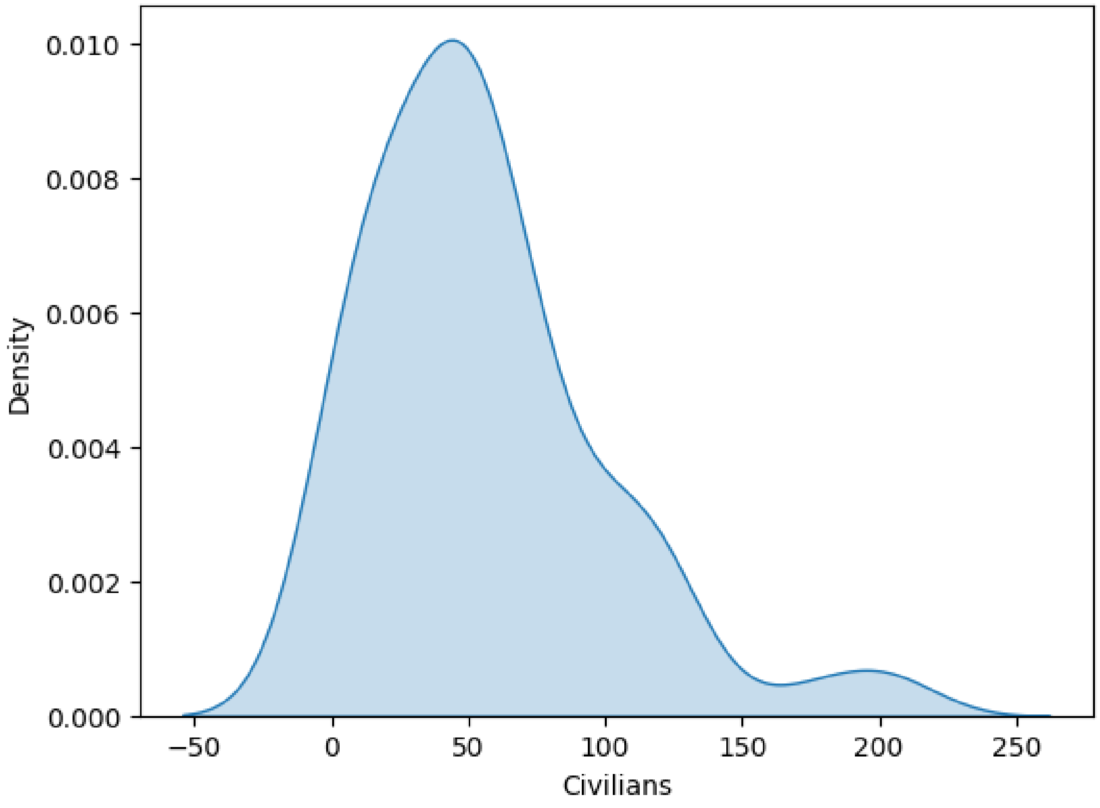
[82]:
```
import seaborn as sns
```

[83]:
```
_ = sns.displot(wikileaks_dataframe['Taliban'], kde=False, rug=True) #plot␣
↪historgram of Taliban casualities (from lab-01-part-01)
```



b) kde

```
[84]: _ = sns.kdeplot(wikileaks_dataframe['Civilians'], fill=True) #plot kernel␣
      ↪density plot of Civilian casualities (from lab-01-part-01)
```



c) Bar and bar-h of categorical variable (casualities per month)

```
[85]: wikileaks_dataframe.head()
```

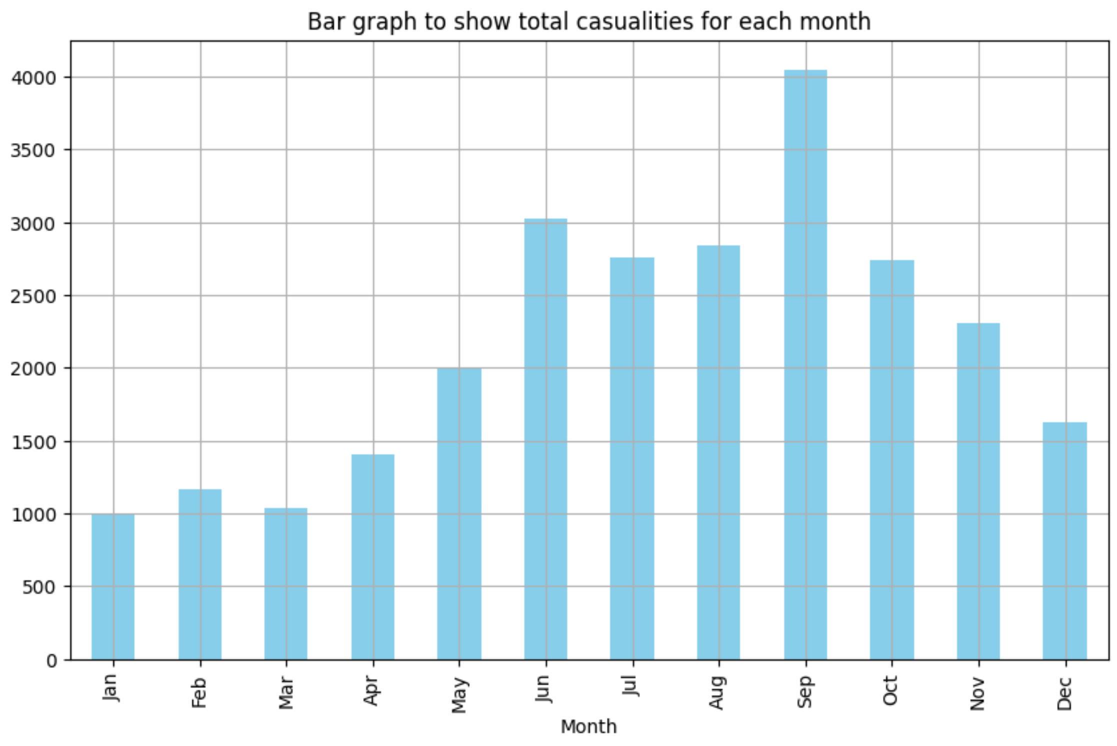
```
[85]:    Year     Month  Taliban  Civilians  Afghan forces  \
      0  2004   January       15         51             23
      1  2004  February        0          7              4
      2  2004     March       19          2              0
      3  2004     April        5          3             19
      4  2004       May       18         29             56

         Nato (detailed in spreadsheet)  Nato - official figures
      0                               0                       11
      1                               5                        2
      2                               2                        3
      3                               0                        3
      4                               6                        9
```

```
[86]: #Number of deaths per month. (Month is a categorical variable)

      months_total = { #create empty dictionary for each month to later load and make␣
       ↪into dataframe
          "Jan": 0,
          "Feb": 0,
          "Mar": 0,
          "Apr": 0,
          "May": 0,
          "Jun": 0,
          "Jul": 0,
          "Aug": 0,
          "Sep": 0,
          "Oct": 0,
          "Nov": 0,
          "Dec": 0
      }

      for index, row in wikileaks_dataframe.iterrows(): #for every row of the␣
       ↪dataframe
          current_month=row["Month"][0:3] #take the first three letters of the month
          #calculate the total casualiatilies by adding up each column of that row␣
       ↪(not year), add it to the right key value in dictionary
          months_total[current_month]+=row["Taliban"] + row["Civilians"] +␣
       ↪row["Afghan forces"] \
              + row["Nato (detailed in spreadsheet)"] + row["Nato - official figures"]


      months_dataframe = pd.DataFrame(list(months_total.items()), columns=['Month',␣
       ↪'Total']) #make the months dictionary into a list
      months_dataframe.head() #display new month dataframe
```

```
[86]:   Month  Total
      0   Jan    999
      1   Feb   1164
      2   Mar   1035
      3   Apr   1403
      4   May   1993
```

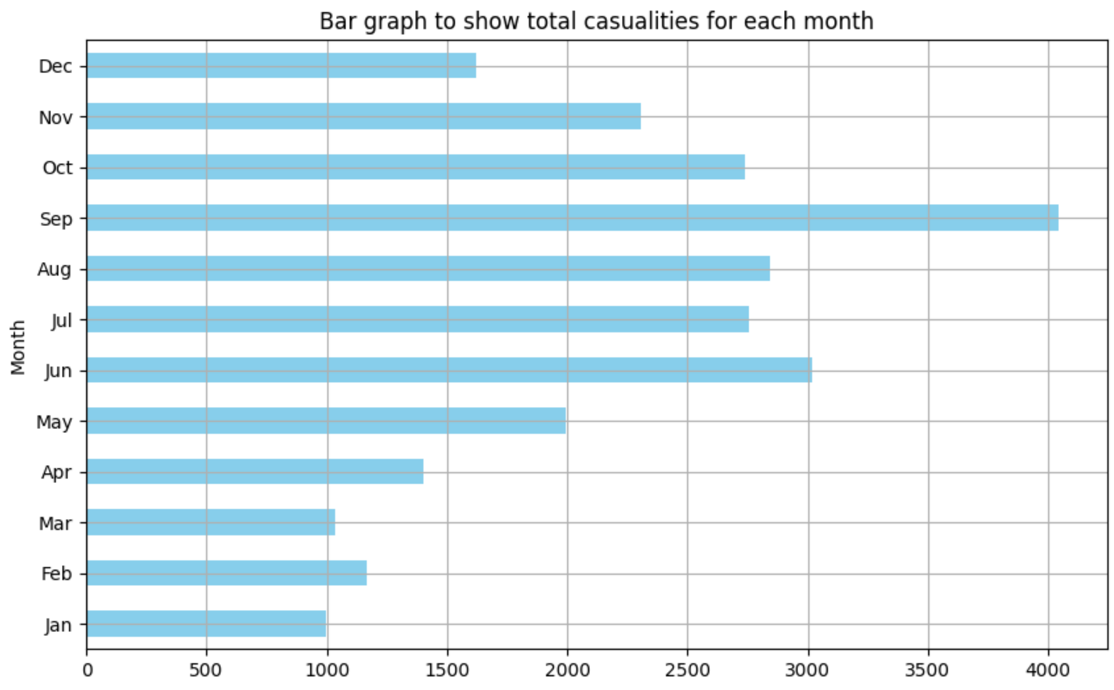
```
[87]: #vertical bar graph of casualities in each month via pandas
      months_dataframe.plot(x='Month', y='Total', kind='bar', color='skyblue',␣
       ↪figsize=(10, 6), grid=True, title='Bar graph to show total casualities for␣
       ↪each month', legend=False)
```

```
[87]: <Axes: title={'center': 'Bar graph to show total casualities for each month'},
      xlabel='Month'>
```



```
[88]: #horiziontal plot
      months_dataframe.plot(x='Month', y='Total', kind='barh', color='skyblue',␣
        ↪figsize=(10, 6), grid=True, title='Bar graph to show total casualities for␣
        ↪each month', legend=False)
```

```
[88]: <Axes: title={'center': 'Bar graph to show total casualities for each month'},
      ylabel='Month'>
```



When the bar graph becomes horizontal, it is much easier to read the month (the categorical variable) as apposed to when the months were displayed at 180 degrees in the vertical bar chart.

0.5 Question 4

a) Write a loop that iterates through each row of a DataFrame and prints the value of one specific column.

b) Modify the loop so that it extracts rows where a numeric column value is greater than a threshold and stores these rows in a new DataFrame.

a) loop prints out one columns in iteration

[89]:
```
def print_each_row(column, dataframe): #define function where you input a␣
↪dataframe and target column
    for row_value in dataframe[column]: #for every row within that column
        print(row_value) #print that row

print_each_row("Afghan forces", wikileaks_dataframe) #test with Afghan column␣
↪and wikileaks dataframe
```

```
23
4
0
19
56
14
```

```
19
10
16
14
36
7
27
0
2
4
25
4
29
10
29
10
24
16
14
106
61
43
50
39
36
42
97
70
14
36
49
35
48
68
125
98
114
89
144
75
67
55
48
35
19
86
65
58
```

```
100
61
57
54
68
52
34
69
94
73
118
133
80
190
133
86
69
65
```

b) function that creates new dataframe with one column of values above threshold

[90]:
```
def extract_row_threshold(column, dataframe, threshold): #define new function␣
 ↪with threshold input
    extracted_values=[] #empty list to extract values into that are above␣
 ↪threshold
    for row_value in dataframe[column]: #for every row in the target column
        if row_value >  threshold: #if that value is larger than threshold
            extracted_values.append(row_value) #add that value to the extract␣
 ↪values list
    new_data = {column:extracted_values} # create a dictionary of one column␣
 ↪with the extracted values above the threshold
    new_dataframe = pd.DataFrame(new_data) #turn this dictionary to a pandas␣
 ↪dataframe
    return new_dataframe #return pandas df

new_dataframe = extract_row_threshold("Afghan forces", wikileaks_dataframe, 10)␣
 ↪#test with threshold 10

new_dataframe.head() #display new table
```

[90]:
```
   Afghan forces
0             23
1             19
2             56
3             14
4             19
```

0.6 Question 5

a) Crate a list containing the names "Civilians" and "Afghan forces"

b) From the imported data keep the the values from these two columns only. Keep in mind that "Year" and "Month" identify each column and sould remain in the dataset.

c) Obtain a monthly total count of casualties for these two groups and create a line and a bar plot of them.

[91]:
```
wikileaks_dataframe.head()
```

[91]:
```
   Year     Month  Taliban  Civilians  Afghan forces  \
0  2004   January       15         51             23
1  2004  February        0          7              4
2  2004     March       19          2              0
3  2004     April        5          3             19
4  2004       May       18         29             56

   Nato (detailed in spreadsheet)  Nato - official figures
0                               0                       11
1                               5                        2
2                               2                        3
3                               0                        3
4                               6                        9
```

[92]:
```
target_list = ["Civilians", "Afghan forces"] #groups we are interested in
columns_to_drop = ["Taliban", "Nato (detailed in spreadsheet)", "Nato -␣
 ↪official figures"]  #columns we are not interested in
q5_wikileaks_dataframe = wikileaks_dataframe.drop(columns=columns_to_drop)␣
 ↪#drop the uninterested columns and create new dataframe for this question
q5_wikileaks_dataframe.head() #display new dataframe with dropped columns
```

[92]:
```
   Year     Month  Civilians  Afghan forces
0  2004   January         51             23
1  2004  February          7              4
2  2004     March          2              0
3  2004     April          3             19
4  2004       May         29             56
```

[93]:
```
q5_wikileaks_dataframe["Total"] = q5_wikileaks_dataframe[target_list[0]] +␣
 ↪q5_wikileaks_dataframe[target_list[1]] #create new column for total Afghan␣
 ↪and civilian casualities
q5_wikileaks_dataframe.head() #display
```

[93]:
```
   Year     Month  Civilians  Afghan forces  Total
0  2004   January         51             23     74
1  2004  February          7              4     11
2  2004     March          2              0      2
```

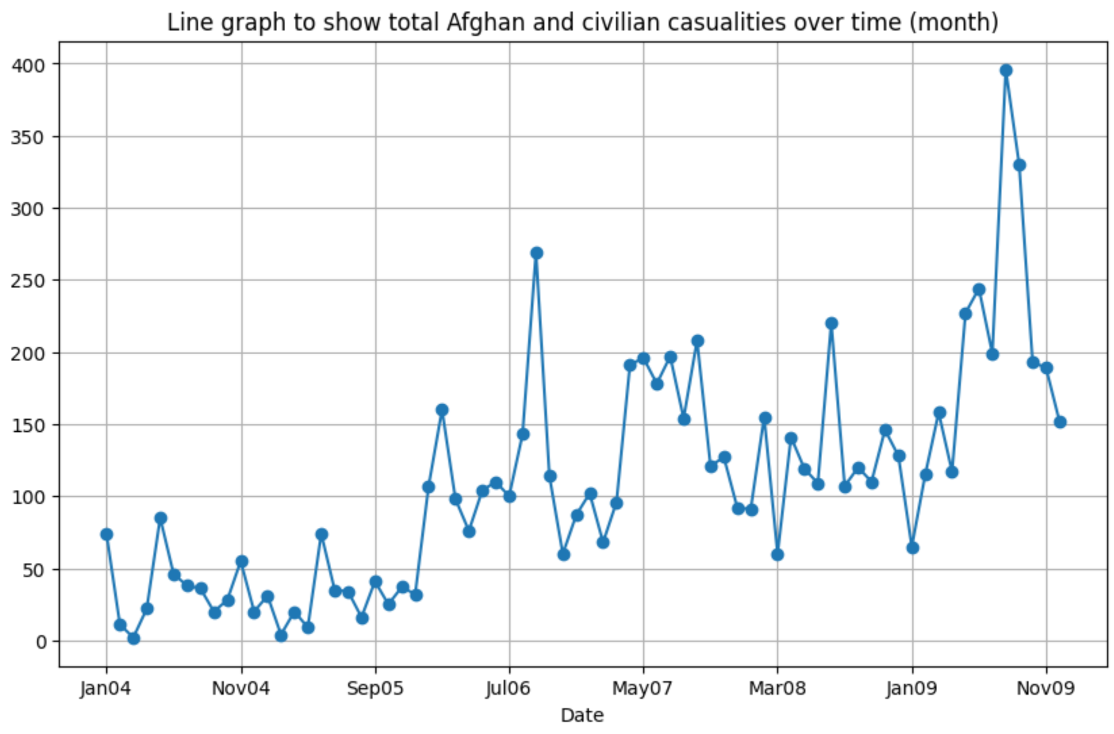
```
3  2004     April          3              19     22
4  2004       May         29              56     85
```

[94]:
```
abbreviated_dates = [] #create empty list for new abbreviate month+year date␣
  ↪for later use in the graph
for index, row in q5_wikileaks_dataframe.iterrows(): #for every row in dataframe
    abbreviated_month=row['Month'][0:3] #take the first three letters of the␣
  ↪month
    abbreviated_year = str(row['Year'])[2:4] #take the last 2 numbers of year
    abbreviated_date=abbreviated_month+abbreviated_year #combine these letters␣
  ↪and numbers
    abbreviated_dates.append(abbreviated_date) #add this total date to all␣
  ↪abbreviated dates list

concise_q5_data = {"Date":abbreviated_dates, "Total":
  ↪q5_wikileaks_dataframe["Total"]} #create new dictionary with abbreviated␣
  ↪dates and total casuality values
concise_q5_dataframe = pd.DataFrame(concise_q5_data) #convert dictionary into␣
  ↪pandas dataframe

concise_q5_dataframe.head() #display
```

[94]:
```
    Date  Total
0  Jan04     74
1  Feb04     11
2  Mar04      2
3  Apr04     22
4  May04     85
```

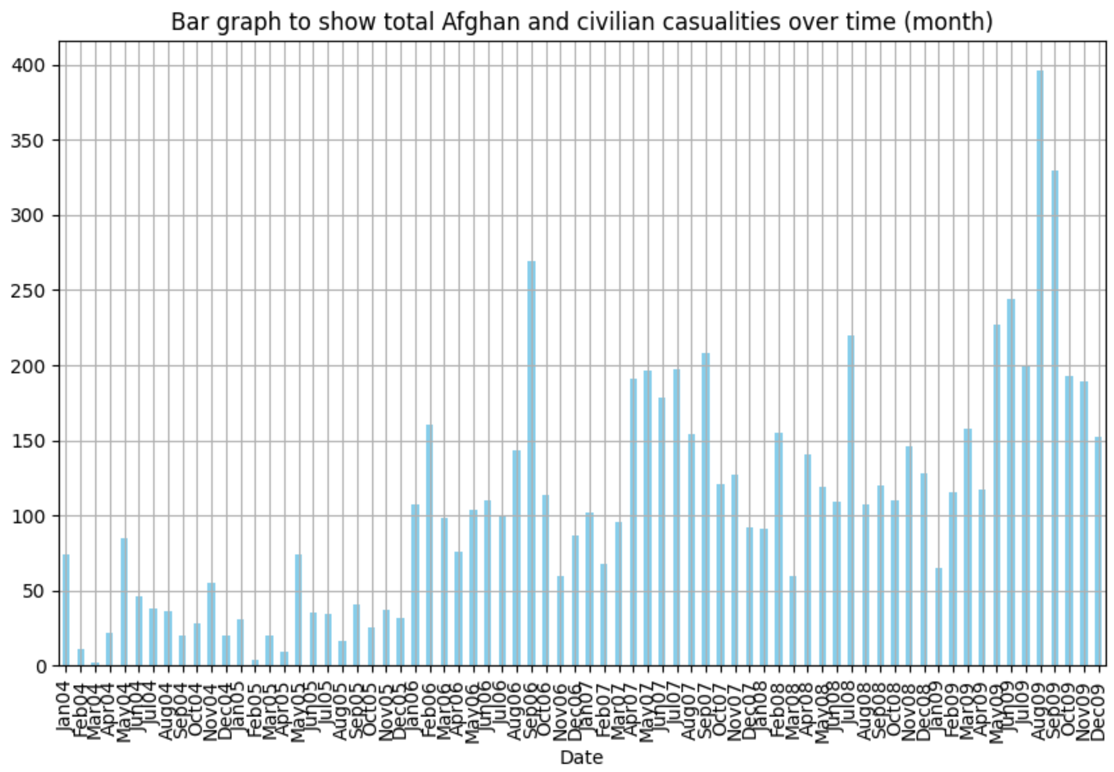
[95]:
```
#line graph using pandas of previous made dataframe
concise_q5_dataframe.plot(x='Date', y='Total', kind='line', marker='o',␣
  ↪figsize=(10, 6), grid=True, title='Line graph to show total Afghan and␣
  ↪civilian casualities over time (month)', legend=False)
```

[95]:
```
<Axes: title={'center': 'Line graph to show total Afghan and civilian
casualities over time (month)'}, xlabel='Date'>
```



[96]:
```
#bar graph using pandas
concise_q5_dataframe.plot(x='Date', y='Total', kind='bar', color='skyblue',␣
  ↪figsize=(10, 6), grid=True, title='Bar graph to show total Afghan and␣
  ↪civilian casualities over time (month)', legend=False)
```

[96]:
```
<Axes: title={'center': 'Bar graph to show total Afghan and civilian casualities
over time (month)'}, xlabel='Date'>
```


Bar graph to show total Afghan and civilian casualities over time (month)
Date
0
50
100
150
200
250
300
350
400