James Fisher ANA500 5-12-2024

Micro-Project 1: Finding and Preparing Data

Data Source: 2022 BRFSS

Target Variable: Difficulty Concentrating or Remembering (CIMEMLOS)

In [2]:
```
## Setup workspace

#import libraries
import os
import pandas as pd
import numpy as np

#set working directory
# os.chdir('/Users/jimmyfisher/Desktop/ANA500')

#import raw BRFSS data from XPT file
data = pd.read_sas('brfss2022.XPT')
```

Create df with variables of interest selected from data dictionary at https://www.cdc.gov/brfss/annual_data/2022/zip/codebook22_llcp-v2-508.zip using early childhood trauma variables and the following publication as a guide:

Casanova, R., Saldana, S., Lutz, M. W., Plassman, B. L., Kuchibhatla, M., & Hayden, K. M. (2020). Investigating predictors of cognitive decline using machine learning. The Journals of Gerontology: Series B, 75(4), 733-742.

In [114…
```
#create df with target variable and subset df with features of interest
df = data[['CIMEMLOS', '_AGEG5YR', '_RACE1','_SEX', '_PHYS14D', '_MENT14D', '_BMI5', 'EXERANY2', 'SLEPTIM1', 'ADDEPEV3',
           'MARITAL', 'EDUCA', 'VETERAN3', 'INCOME3', 'CVDSTRK3', 'CVDINFR4', 'CVDCRHD4', 'ACEPRISN', 'ACEDIVRC',
           'ACEPUNCH', 'ACEHURT1', 'ACESWEAR', 'ACETOUCH', 'ACETTHEM', 'ACEHVSEX', 'SDHISOLT']]

#filter out survey respondents with no record of having been asked survey question about
#memory loss or confusion (variable CIMEMLOS), including any adult < 45 years old.
df = df[df['_AGEG5YR'] >= 6]
df = df.dropna(subset=['CIMEMLOS'])
df = df.reset_index(drop=True)

#print columns of df
print("Columns:\n",df.columns)

#show first 5 rows of DF
print("Head:\n",df.head(5))
```

```
Columns:
 Index(['CIMEMLOS', '_AGEG5YR', '_RACE1', '_SEX', '_PHYS14D', '_MENT14D',
       '_BMI5', 'EXERANY2', 'SLEPTIM1', 'ADDEPEV3', 'MARITAL', 'EDUCA',
       'VETERAN3', 'INCOME3', 'CVDSTRK3', 'CVDINFR4', 'CVDCRHD4', 'ACEPRISN',
       'ACEDIVRC', 'ACEPUNCH', 'ACEHURT1', 'ACESWEAR', 'ACETOUCH', 'ACETTHEM',
       'ACEHVSEX', 'SDHISOLT'],
      dtype='object')
Head:
    CIMEMLOS  _AGEG5YR  _RACE1  _SEX  _PHYS14D  _MENT14D   _BMI5  EXERANY2  \
0       1.0      12.0     1.0   1.0       1.0       1.0  1953.0       1.0
1       2.0       9.0     1.0   2.0       1.0       1.0  3109.0       1.0
2       1.0      12.0     1.0   1.0       2.0       9.0     NaN       1.0
3       2.0      13.0     1.0   1.0       1.0       1.0  2439.0       1.0
4       2.0      10.0     1.0   1.0       2.0       2.0  3228.0       1.0

   SLEPTIM1  ADDEPEV3  ...  CVDCRHD4  ACEPRISN  ACEDIVRC  ACEPUNCH  ACEHURT1  \
0       8.0       2.0  ...       2.0       2.0       2.0       1.0       1.0
1       6.0       2.0  ...       2.0       2.0       1.0       3.0       3.0
2       7.0       1.0  ...       2.0       2.0       2.0       1.0       1.0
3       8.0       2.0  ...       2.0       2.0       2.0       3.0       3.0
4       7.0       2.0  ...       1.0       2.0       2.0       1.0       1.0

   ACESWEAR  ACETOUCH  ACETTHEM  ACEHVSEX  SDHISOLT
0       1.0       1.0       1.0       1.0       5.0
1       3.0       3.0       1.0       1.0       4.0
2       3.0       1.0       1.0       1.0       4.0
3       1.0       1.0       1.0       1.0       5.0
4       1.0       1.0       1.0       1.0       4.0

[5 rows x 26 columns]
```

In [115…
```
print(df.shape)

#create correction matrix to identify potential predictors
corr = df.corr()
print(corr)
```

```
(64675, 26)
          CIMEMLOS  _AGEG5YR    _RACE1      _SEX  _PHYS14D  _MENT14D  \
CIMEMLOS  1.000000  0.007883  0.013060  0.003213 -0.017363 -0.020786
_AGEG5YR  0.007883  1.000000 -0.053368  0.025969  0.046075 -0.054490
_RACE1    0.013060 -0.053368  1.000000 -0.028929  0.052490  0.051368
_SEX      0.003213  0.025969 -0.028929  1.000000  0.034899  0.060957
_PHYS14D -0.017363  0.046075  0.052490  0.034899  1.000000  0.293215
_MENT14D -0.020786 -0.054490  0.051368  0.060957  0.293215  1.000000
_BMI5    -0.017894 -0.152350  0.023943 -0.032839  0.071891  0.049047
EXERANY2 -0.005840  0.065315  0.039646  0.038601  0.157656  0.094186
SLEPTIM1  0.020578  0.077440  0.035156  0.020167  0.085983  0.073679
ADDEPEV3  0.098926  0.091871  0.032017 -0.086584 -0.058684 -0.115179
MARITAL   0.011457  0.074056  0.069319  0.061890  0.066808  0.076759
EDUCA     0.037271 -0.021780 -0.083191 -0.002798 -0.106974 -0.069131
VETERAN3  0.064167 -0.094075  0.055027  0.264753  0.012021  0.026527
INCOME3   0.055850  0.151768  0.052860  0.058521  0.028012  0.016463
CVDSTRK3  0.059293 -0.034569  0.012062  0.010628 -0.025342  0.004572
CVDINFR4  0.041195 -0.028144  0.028039  0.045397  0.010590  0.021474
CVDCRHD4  0.052242 -0.013864  0.019841  0.029410  0.012530  0.030040
ACEPRISN  0.212419  0.042944  0.026153  0.001337  0.013395  0.022217
ACEDIVRC  0.133612  0.070973  0.054797 -0.006326  0.029155  0.022079
ACEPUNCH  0.129863 -0.049040  0.074532 -0.006836  0.061845  0.090888
ACEHURT1  0.122213 -0.028288  0.081091 -0.038328  0.065901  0.088921
ACESWEAR  0.110284 -0.077249  0.033610 -0.015228  0.061373  0.110923
ACETOUCH  0.143269 -0.044980  0.062290  0.087132  0.065348  0.087647
ACETTHEM  0.151383 -0.042826  0.063704  0.053256  0.056816  0.081226
ACEHVSEX  0.159388 -0.026860  0.060531  0.038050  0.067941  0.083279
SDHISOLT  0.124709  0.085213 -0.012379 -0.054146 -0.103339 -0.188912

             _BMI5  EXERANY2  SLEPTIM1  ADDEPEV3  ...  CVDCRHD4  ACEPRISN  \
CIMEMLOS -0.017894 -0.005840  0.020578  0.098926  ...  0.052242  0.212419
_AGEG5YR -0.152350  0.065315  0.077440  0.091871  ... -0.013864  0.042944
_RACE1    0.023943  0.039646  0.035156  0.032017  ...  0.019841  0.026153
_SEX     -0.032839  0.038601  0.020167 -0.086584  ...  0.029410  0.001337
_PHYS14D  0.071891  0.157656  0.085983 -0.058684  ...  0.012530  0.013395
_MENT14D  0.049047  0.094186  0.073679 -0.115179  ...  0.030040  0.022217
_BMI5     1.000000  0.133998 -0.005865 -0.081450  ... -0.012083 -0.010304
EXERANY2  0.133998  1.000000  0.069204 -0.033092  ...  0.024509  0.044747
SLEPTIM1 -0.005865  0.069204  1.000000  0.027042  ...  0.037037  0.036937
ADDEPEV3 -0.081450 -0.033092  0.027042  1.000000  ...  0.087697  0.067535
MARITAL   0.022230  0.078779  0.046337 -0.006304  ...  0.034745  0.027976
EDUCA    -0.085944 -0.164710 -0.050038  0.028364  ... -0.014481  0.014103
VETERAN3 -0.001241  0.005260  0.025392  0.036392  ...  0.057783  0.055102
INCOME3  -0.068375  0.017164  0.058212  0.068891  ...  0.030532  0.081857
CVDSTRK3 -0.011885 -0.000017  0.013074  0.113015  ...  0.121505  0.041575
CVDINFR4 -0.022212  0.010741  0.041202  0.085872  ...  0.189952  0.025981
CVDCRHD4 -0.012083  0.024509  0.037037  0.087697  ...  1.000000  0.021616
ACEPRISN -0.010304  0.044747  0.036937  0.067535  ...  0.021616  1.000000
ACEDIVRC  0.003935  0.032044  0.031312  0.065206  ...  0.027098  0.539673
ACEPUNCH  0.042456  0.053854  0.031895  0.010389  ...  0.027919  0.474620
ACEHURT1  0.038524  0.055359  0.033168 -0.000907  ...  0.041082  0.472701
ACESWEAR  0.049865  0.029029  0.020982 -0.034358  ...  0.046417  0.424262
ACETOUCH  0.042651  0.054298  0.020886 -0.028632  ...  0.029985  0.499283
ACETTHEM  0.038007  0.056903  0.026754 -0.008003  ...  0.035296  0.531927
ACEHVSEX  0.033098  0.054026  0.026629  0.007239  ...  0.023812  0.550748
SDHISOLT -0.050825 -0.062109  0.010307  0.186371  ...  0.001354  0.071901

          ACEDIVRC  ACEPUNCH  ACEHURT1  ACESWEAR  ACETOUCH  ACETTHEM  \
CIMEMLOS  0.133612  0.129863  0.122213  0.110284  0.143269  0.151383
_AGEG5YR  0.070973 -0.049040 -0.028288 -0.077249 -0.044980 -0.042826
_RACE1    0.054797  0.074532  0.081091  0.033610  0.062290  0.063704
_SEX     -0.006326 -0.006836 -0.038328 -0.015228  0.087132  0.053256
_PHYS14D  0.029155  0.061845  0.065901  0.061373  0.065348  0.056816
_MENT14D  0.022079  0.090888  0.088921  0.110923  0.087647  0.081226
_BMI5     0.003935  0.042456  0.038524  0.049865  0.042651  0.038007
EXERANY2  0.032044  0.053854  0.055359  0.029029  0.054298  0.056903
SLEPTIM1  0.031312  0.031895  0.033168  0.020982  0.020886  0.026754
ADDEPEV3  0.065206  0.010389 -0.000907 -0.034358 -0.028632 -0.008003
MARITAL   0.038013  0.042994  0.040342  0.027999  0.046363  0.046436
EDUCA     0.003775 -0.065141 -0.044770  0.007866 -0.015781 -0.024544
VETERAN3  0.038200  0.034009  0.014653  0.029667  0.069257  0.064091
INCOME3   0.060385  0.042896  0.039209  0.033262  0.058865  0.057281
CVDSTRK3  0.028982  0.038535  0.039666  0.042106  0.034217  0.039288
CVDINFR4  0.033712  0.028966  0.012830  0.029450  0.023925  0.027397
CVDCRHD4  0.027098  0.027919  0.041082  0.046417  0.029985  0.035296
ACEPRISN  0.539673  0.474620  0.472701  0.424262  0.499283  0.531927
ACEDIVRC  1.000000  0.307565  0.317764  0.264680  0.339229  0.364513
ACEPUNCH  0.307565  1.000000  0.555916  0.496932  0.454457  0.468544
ACEHURT1  0.317764  0.555916  1.000000  0.571121  0.497465  0.498244
ACESWEAR  0.264680  0.496932  0.571121  1.000000  0.471480  0.465704
ACETOUCH  0.339229  0.454457  0.497465  0.471480  1.000000  0.837053
ACETTHEM  0.364513  0.468544  0.498244  0.465704  0.837053  1.000000
ACEHVSEX  0.369465  0.466984  0.493491  0.464977  0.772554  0.838613
SDHISOLT  0.040705 -0.046523 -0.095074 -0.133007 -0.072902 -0.058436

          ACEHVSEX  SDHISOLT
CIMEMLOS  0.159388  0.124709
_AGEG5YR -0.026860  0.085213
_RACE1    0.060531 -0.012379
_SEX      0.038050 -0.054146
_PHYS14D  0.067941 -0.103339
_MENT14D  0.083279 -0.188912
_BMI5     0.033098 -0.050825
EXERANY2  0.054026 -0.062109
SLEPTIM1  0.026629  0.010307
ADDEPEV3  0.007239  0.186371
MARITAL   0.049978 -0.089842
EDUCA    -0.032085  0.031156
```

```
VETERAN3  0.066207 -0.002244
INCOME3   0.065942  0.054449
CVDSTRK3  0.038950  0.039513
CVDINFR4  0.026192  0.019134
CVDCRHD4  0.023812  0.001354
ACEPRISN  0.550748  0.071901
ACEDIVRC  0.369465  0.040705
ACEPUNCH  0.466984 -0.046523
ACEHURT1  0.493491 -0.095074
ACESWEAR  0.464977 -0.133007
ACETOUCH  0.772554 -0.072902
ACETTHEM  0.838613 -0.058436
ACEHVSEX  1.000000 -0.048974
SDHISOLT -0.048974  1.000000

[26 rows x 26 columns]
```

Beginning with the "confusion & memory loss" target variable (CIMEMLOS), multi-level categorical variables were then grouped where appropriate to simplify. Missing values (NaN; respondent refused to answer; respondent reported not knowing) were replaced using utilizing mode imputation for categorical values and mean imputation for continuous variables.

In [116…

```
### Variable: CIMEMLOS (worsening confusion or memory loss)

#show pre-processing freqency distribution of variable values
print("pre-processing CIMEMLOS frequency:\n",df['CIMEMLOS'].value_counts())
print("\nNumber of NaN values in column: ",df['CIMEMLOS'].isna().sum(), "\n")

#convert categorical variable to a binary variable change '2' (no) to 0
df['CIMEMLOS'] = df['CIMEMLOS'].replace(2,0)

#replace 7s (don't know / not sure) and 9s (refused to answer) using mode imputation
df['CIMEMLOS'] = df['CIMEMLOS'].replace(7,df['CIMEMLOS'].mode()[0])
df['CIMEMLOS'] = df['CIMEMLOS'].replace(9,df['CIMEMLOS'].mode()[0])

#show post-processing frequency distribution of variable values
print("post-processing CIMEMLOS frequency:\n",df['CIMEMLOS'].value_counts())
```

```
pre-processing CIMEMLOS frequency:
 CIMEMLOS
2.0    56945
1.0     7003
7.0      474
9.0      253
Name: count, dtype: int64

Number of NaN values in column:  0

post-processing CIMEMLOS frequency:
 CIMEMLOS
0.0    57672
1.0     7003
Name: count, dtype: int64
```

In [117…

```
### Variable: _AGEG5YR (age categories)

#show pre-processing freqency distribution of variable values
print("pre-processing frequency:\n",df['_AGEG5YR'].value_counts())
print("\nNumber of NaN values in _AGEG5YR column: ",df['_AGEG5YR'].isna().sum(), "\n")

#recode 5-year age increments to begin at 1 (instead of 6, Age 45 to 49)
df['_AGEG5YR'] = df['_AGEG5YR'].replace(6,1)
df['_AGEG5YR'] = df['_AGEG5YR'].replace(7,2)
df['_AGEG5YR'] = df['_AGEG5YR'].replace(8,3)
df['_AGEG5YR'] = df['_AGEG5YR'].replace(9,4)
df['_AGEG5YR'] = df['_AGEG5YR'].replace(10,5)
df['_AGEG5YR'] = df['_AGEG5YR'].replace(11,6)
df['_AGEG5YR'] = df['_AGEG5YR'].replace(12,7)
df['_AGEG5YR'] = df['_AGEG5YR'].replace(13,8)

#replace 14s (don't know / refused / missing) using mode imputation
df['_AGEG5YR'] = df['_AGEG5YR'].replace(14,df['_AGEG5YR'].mode()[0])

#show post-processing frequency distribution of variable values
print("post-processing _AGEG5YR frequency:\n",df['_AGEG5YR'].value_counts())
```

```
pre-processing frequency:
 _AGEG5YR
10.0    9952
11.0    9543
9.0     9279
13.0    7920
8.0     7429
12.0    7424
7.0     6682
6.0     5199
14.0    1247
Name: count, dtype: int64

Number of NaN values in _AGEG5YR column:  0

post-processing _AGEG5YR frequency:
 _AGEG5YR
5.0    11199
6.0     9543
4.0     9279
8.0     7920
3.0     7429
7.0     7424
2.0     6682
1.0     5199
Name: count, dtype: int64
```

In [118…

```
### Variable: _RACE1 (race/ethnicity categories)

#show pre-processing freqency distribution of variable values
print("pre-processing frequency:\n",df['_RACE1'].value_counts())
print("\nNumber of NaN values in _RACE1 column: ",df['_RACE1'].isna().sum(), "\n")

#replace 9s (don't know / not sure / refused) using mode imputation
df['_RACE1'] = df['_RACE1'].replace(9,df['_RACE1'].mode()[0])

#show post-processing frequency distribution of variable values
print("post-processing _RACE1 frequency:\n",df['_RACE1'].value_counts())
```

```
pre-processing frequency:
 _RACE1
1.0    54291
2.0     4213
8.0     2335
9.0     1954
7.0      871
3.0      496
4.0      448
5.0       67
Name: count, dtype: int64

Number of NaN values in _RACE1 column:  0

post-processing _RACE1 frequency:
 _RACE1
1.0    56245
2.0     4213
8.0     2335
7.0      871
3.0      496
4.0      448
5.0       67
Name: count, dtype: int64
```

In [119…

```
### Variable: _SEX

#show pre-processing freqency distribution of variable values
print("pre-processing frequency:\n",df['_SEX'].value_counts())
print("\nNumber of NaN values in _SEX column: ",df['_SEX'].isna().sum(), "\n")

    ## no processing required
```

```
pre-processing frequency:
 _SEX
2.0    35810
1.0    28865
Name: count, dtype: int64

Number of NaN values in _SEX column:  0
```

In [120…

```
### Variable: _PHYS14D (Three-level not good physical health status)

#show pre-processing freqency distribution of variable values
print("pre-processing frequency:\n",df['_PHYS14D'].value_counts())
print("\nNumber of NaN values in _PHYS14D column: ",df['_PHYS14D'].isna().sum(), "\n")

#replace 9s (don't know / refused / missing) using mode imputation
df['_PHYS14D'] = df['_PHYS14D'].replace(9,df['_PHYS14D'].mode()[0])

#show post-processing frequency distribution of variable values
print("post-processing _PHYS14D frequency:\n",df['_PHYS14D'].value_counts())
```

```
pre-processing frequency:
 _PHYS14D
1.0    38344
2.0    14802
3.0     9755
9.0     1774
Name: count, dtype: int64

Number of NaN values in _PHYS14D column:  0

post-processing _PHYS14D frequency:
 _PHYS14D
1.0    40118
2.0    14802
3.0     9755
Name: count, dtype: int64
```

In [121…

```
### Variable: _MENT14D (Three-level not good mental health status)

#show pre-processing freqency distribution of variable values
print("pre-processing frequency:\n",df['_MENT14D'].value_counts())
print("\nNumber of NaN values in _MENT14D column: ",df['_MENT14D'].isna().sum(), "\n")

#replace 9s (don't know / refused / missing) using mode imputation
df['_MENT14D'] = df['_MENT14D'].replace(9,df['_MENT14D'].mode()[0])

#show post-processing frequency distribution of variable values
print("post-processing _MENT14D frequency:\n",df['_MENT14D'].value_counts())
```

```
pre-processing frequency:
 _MENT14D
1.0    42980
2.0    13477
3.0     6897
9.0     1321
Name: count, dtype: int64

Number of NaN values in _MENT14D column:  0

post-processing _MENT14D frequency:
 _MENT14D
1.0    44301
2.0    13477
3.0     6897
Name: count, dtype: int64
```

In [122…

```
### Variable: _BMI5 (body mass index)

#show pre-processing variable distribution values
print("pre-processing data:\n")
print("Mean: ",df['_BMI5'].mean())
print("Median: ",df['_BMI5'].median())
print("Q1: ",df['_BMI5'].quantile(0.25))
print("Q3: ",df['_BMI5'].quantile(0.75))
print("\nNumber of NaN values in _BMI5 column: ",df['_BMI5'].isna().sum(), "\n")

#replace NaN rows with mean BMI values
df['_BMI5'] = df['_BMI5'].fillna(df['_BMI5'].mean())

#incorporate the 2 "implied decimal places" indicated by data dictionary and limit to 2 decimals
df['_BMI5'] = df['_BMI5']/100
df['_BMI5'] = df['_BMI5'].round(2)

#show post-processing variable distribution values
print("post-processing data:\n")
print("Mean: ",df['_BMI5'].mean())
print("Median: ",df['_BMI5'].median())
print("Q1: ",df['_BMI5'].quantile(0.25))
print("Q3: ",df['_BMI5'].quantile(0.75))
print("\nNumber of NaN values in _BMI5 column: ",df['_BMI5'].isna().sum(), "\n")
```

```
pre-processing data:

Mean:  2863.673091571452
Median:  2746.0
Q1:  2437.0
Q3:  3174.0

Number of NaN values in _BMI5 column:  4285

post-processing data:

Mean:  28.636947506764592
Median:  28.13
Q1:  24.56
Q3:  31.32

Number of NaN values in _BMI5 column:  0
```

In [123…

```
### Variable: EXERANY2 (past-month exercise)

#show pre-processing freqency distribution of variable values
print("pre-processing EXERANY2 frequency:\n",df['EXERANY2'].value_counts())
print("\nNumber of NaN values in column: ",df['EXERANY2'].isna().sum(), "\n")

#convert categorical variable to a binary variable change '2' (no) to 0
df['EXERANY2'] = df['EXERANY2'].replace(2,0)
```

```
#replace 7s (don't know / not sure) and 9s (refused to answer) using mode imputation
df['EXERANY2'] = df['EXERANY2'].replace(7,df['EXERANY2'].mode()[0])
df['EXERANY2'] = df['EXERANY2'].replace(9,df['EXERANY2'].mode()[0])

#show post-processing frequency distribution of variable values
print("post-processing EXERANY2 frequency:\n",df['EXERANY2'].value_counts())
```

```
pre-processing EXERANY2 frequency:
 EXERANY2
1.0    48021
2.0    16446
7.0      151
9.0       57
Name: count, dtype: int64

Number of NaN values in column:  0

post-processing EXERANY2 frequency:
 EXERANY2
1.0    48229
0.0    16446
Name: count, dtype: int64
```

In [124…

```
### Variable: SLEPTIM1 (average reported sleep per night)

#show pre-processing variable distribution values
print("pre-processing data:\n")
print("Mean: ",df['SLEPTIM1'].mean())
print("Median: ",df['SLEPTIM1'].median())
print("Q1: ",df['SLEPTIM1'].quantile(0.25))
print("Q3: ",df['SLEPTIM1'].quantile(0.75))
print("\nNumber of NaN values in SLEPTIM1 column: ",df['SLEPTIM1'].isna().sum(), "\n")
print("pre-processing SLEPTIM1 frequency:\n",df['SLEPTIM1'].value_counts())

#replace 77s (don't know / not sure) and 99s (refused to answer) using mean imputation (excluding 77s and 99s)
sleepMean = df['SLEPTIM1'][df['SLEPTIM1'] <77].mean()
df['SLEPTIM1'] = df['SLEPTIM1'].replace(77,sleepMean)
df['SLEPTIM1'] = df['SLEPTIM1'].replace(99,sleepMean)

#show post-processing variable distribution values
print("post-processing data:\n")
print("Mean: ",df['SLEPTIM1'].mean())
print("Median: ",df['SLEPTIM1'].median())
print("Q1: ",df['SLEPTIM1'].quantile(0.25))
print("Q3: ",df['SLEPTIM1'].quantile(0.75))
print("\nNumber of NaN values in SLEPTIM1 column: ",df['SLEPTIM1'].isna().sum(), "\n")
```

```
pre-processing data:

Mean:  8.027398531117123
Median:  7.0
Q1:  6.0
Q3:  8.0

Number of NaN values in SLEPTIM1 column:  0

pre-processing SLEPTIM1 frequency:
 SLEPTIM1
8.0     19346
7.0     19291
6.0     12881
5.0      3778
9.0      3527
10.0     1782
4.0      1670
77.0      738
12.0      459
3.0       434
2.0       214
1.0       124
11.0      105
99.0       77
16.0       57
14.0       52
15.0       49
13.0       29
18.0       25
20.0       20
24.0        9
19.0        3
23.0        2
22.0        2
17.0        1
Name: count, dtype: int64
post-processing data:

Mean:  7.120623238333855
Median:  7.0
Q1:  6.0
Q3:  8.0

Number of NaN values in SLEPTIM1 column:  0
```

In [125…

```
### Variable: ADDEPEV3 (told had depressive disorder)

#show pre-processing freqency distribution of variable values
```

```
print("pre-processing ADDEPEV3 frequency:\n",df['ADDEPEV3'].value_counts())
print("\nNumber of NaN values in column: ",df['ADDEPEV3'].isna().sum(), "\n")

#convert categorical variable to a binary variable change '2' (no) to 0
df['ADDEPEV3'] = df['ADDEPEV3'].replace(2,0)

#replace 7s (don't know / not sure) and 9s (refused to answer) using mode imputation
df['ADDEPEV3'] = df['ADDEPEV3'].replace(7,df['ADDEPEV3'].mode()[0])
df['ADDEPEV3'] = df['ADDEPEV3'].replace(9,df['ADDEPEV3'].mode()[0])

#show post-processing frequency distribution of variable values
print("post-processing ADDEPEV3 frequency:\n",df['ADDEPEV3'].value_counts())
```

```
pre-processing ADDEPEV3 frequency:
 ADDEPEV3
2.0    51806
1.0    12531
7.0      245
9.0       93
Name: count, dtype: int64

Number of NaN values in column:  0

post-processing ADDEPEV3 frequency:
 ADDEPEV3
0.0    52144
1.0    12531
Name: count, dtype: int64
```

In [126…

```
### Variable: MARITAL (marital status)

#show pre-processing freqency distribution of variable values
print("pre-processing MARITAL frequency:\n",df['MARITAL'].value_counts())
print("\nNumber of NaN values in column: ",df['MARITAL'].isna().sum(), "\n")

#data recoded into 3-level variable: married(1), divorced/widowed(2), or other(3)
df['MARITAL'] = df['MARITAL'].replace(3,2)
df['MARITAL'] = df['MARITAL'].replace(4,3)
df['MARITAL'] = df['MARITAL'].replace(5,3)
df['MARITAL'] = df['MARITAL'].replace(6,3)
df['MARITAL'] = df['MARITAL'].replace(9,3)

#show post-processing freqency distribution of variable values
print("post-processing MARITAL frequency:\n",df['MARITAL'].value_counts())
```

```
pre-processing MARITAL frequency:
 MARITAL
1.0    36519
3.0    10259
2.0    10200
5.0     4836
6.0     1376
4.0      958
9.0      527
Name: count, dtype: int64

Number of NaN values in column:  0

post-processing MARITAL frequency:
 MARITAL
1.0    36519
2.0    20459
3.0     7697
Name: count, dtype: int64
```

In [127…

```
### Variable: EDUCA (educational attainment)

#show pre-processing freqency distribution of variable values
print("pre-processing EDUCA frequency:\n",df['EDUCA'].value_counts())
print("\nNumber of NaN values in column: ",df['EDUCA'].isna().sum(), "\n")

#data recoded into 3-level variable: no high school diploma(0), only high school diploma/GED (1),
#  and at least one 4-year college degree(2)
df['EDUCA'] = df['EDUCA'].replace(1,0)
df['EDUCA'] = df['EDUCA'].replace(2,0)
df['EDUCA'] = df['EDUCA'].replace(3,0)
df['EDUCA'] = df['EDUCA'].replace(4,1)
df['EDUCA'] = df['EDUCA'].replace(5,1)
df['EDUCA'] = df['EDUCA'].replace(6,2)

#replaced 9s (refused) using mode imputation
df['EDUCA'] = df['EDUCA'].replace(9,df['EDUCA'].mode()[0])

#show post-processing freqency distribution of variable values
print("post-processing EDUCA frequency:\n",df['EDUCA'].value_counts())
print("\nNumber of NaN values in column: ",df['EDUCA'].isna().sum(), "\n")
```

```
pre-processing EDUCA frequency:
 EDUCA
6.0    28066
5.0    17917
4.0    15299
3.0     2143
2.0      889
9.0      297
1.0       64
Name: count, dtype: int64

Number of NaN values in column:  0

post-processing EDUCA frequency:
 EDUCA
1.0    33513
2.0    28066
0.0     3096
Name: count, dtype: int64

Number of NaN values in column:  0
```

In [128…

```
### Variable: VETERAN3 (served on active duty in the armed forces)

#show pre-processing freqency distribution of variable values
print("pre-processing VETERAN3 frequency:\n",df['VETERAN3'].value_counts())
print("\nNumber of NaN values in column: ",df['VETERAN3'].isna().sum(), "\n")

#convert categorical variable to a binary variable change '2' (no) to 0
df['VETERAN3'] = df['VETERAN3'].replace(2,0)

#replace 7s (don't know / not sure) and 9s (refused to answer) using mode imputation
df['VETERAN3'] = df['VETERAN3'].replace(7,df['VETERAN3'].mode()[0])
df['VETERAN3'] = df['VETERAN3'].replace(9,df['VETERAN3'].mode()[0])

#show post-processing frequency distribution of variable values
print("post-processing VETERAN3 frequency:\n",df['VETERAN3'].value_counts())
```

```
pre-processing VETERAN3 frequency:
 VETERAN3
2.0    54442
1.0    10065
9.0      134
7.0       34
Name: count, dtype: int64

Number of NaN values in column:  0

post-processing VETERAN3 frequency:
 VETERAN3
0.0    54610
1.0    10065
Name: count, dtype: int64
```

In [129…

```
### Variable: INCOME3 (annual household income)

#show pre-processing freqency distribution of variable values
print("pre-processing INCOME3 frequency:\n",df['INCOME3'].value_counts())
print("\nNumber of NaN values in column: ",df['INCOME3'].isna().sum(), "\n")

#left 77s (don't know / not sure) and 99s (refused) alone to assess later due to possible predictive power (using dummy variables)

#replaced single missing record with mode imputation
df['INCOME3'] = df['INCOME3'].fillna(7)

#show post-processing freqency distribution of variable values
print("post-processing INCOME3 frequency:\n",df['INCOME3'].value_counts())
```

```
pre-processing INCOME3 frequency:
 INCOME3
7.0     9199
99.0    8225
8.0     7436
6.0     7299
9.0     7219
5.0     6556
77.0    4697
4.0     3084
10.0    3019
11.0    2884
3.0     2224
2.0     1730
1.0     1102
Name: count, dtype: int64

Number of NaN values in column:  1

post-processing INCOME3 frequency:
 INCOME3
7.0     9200
99.0    8225
8.0     7436
6.0     7299
9.0     7219
5.0     6556
77.0    4697
4.0     3084
10.0    3019
11.0    2884
3.0     2224
2.0     1730
1.0     1102
Name: count, dtype: int64
```

In [130…

```python
### Variable: CVDSTRK3 (had stroke)

#show pre-processing freqency distribution of variable values
print("pre-processing CVDSTRK3 frequency:\n",df['CVDSTRK3'].value_counts())
print("\nNumber of NaN values in column: ",df['CVDSTRK3'].isna().sum(), "\n")

#convert categorical variable to a binary variable change '2' (no) to 0
df['CVDSTRK3'] = df['CVDSTRK3'].replace(2,0)

#replace 7s (don't know / not sure) and 9s (refused to answer) using mode imputation
df['CVDSTRK3'] = df['CVDSTRK3'].replace(7,df['CVDSTRK3'].mode()[0])
df['CVDSTRK3'] = df['CVDSTRK3'].replace(9,df['CVDSTRK3'].mode()[0])

#show post-processing freqency distribution of variable values
print("post-processing CVDSTRK3 frequency:\n",df['CVDSTRK3'].value_counts())
```

```
pre-processing CVDSTRK3 frequency:
 CVDSTRK3
2.0    60575
1.0     3863
7.0      205
9.0       32
Name: count, dtype: int64

Number of NaN values in column:  0

post-processing CVDSTRK3 frequency:
 CVDSTRK3
0.0    60812
1.0     3863
Name: count, dtype: int64
```

In [131…

```python
### Variable: CVDINFR4 (had heart attack)

#show pre-processing freqency distribution of variable values
print("pre-processing CVDINFR4 frequency:\n",df['CVDINFR4'].value_counts())
print("\nNumber of NaN values in column: ",df['CVDINFR4'].isna().sum(), "\n")

#convert categorical variable to a binary variable change '2' (no) to 0
df['CVDINFR4'] = df['CVDINFR4'].replace(2,0)

#replace 7s (don't know / not sure) and 9s (refused to answer) using mode imputation
df['CVDINFR4'] = df['CVDINFR4'].replace(7,df['CVDINFR4'].mode()[0])
df['CVDINFR4'] = df['CVDINFR4'].replace(9,df['CVDINFR4'].mode()[0])

#show post-processing freqency distribution of variable values
print("post-processing CVDINFR4 frequency:\n",df['CVDINFR4'].value_counts())
```

```
pre-processing CVDINFR4 frequency:
 CVDINFR4
2.0    59103
1.0     5107
7.0      420
9.0       45
Name: count, dtype: int64

Number of NaN values in column:  0

post-processing CVDINFR4 frequency:
 CVDINFR4
0.0    59568
1.0     5107
Name: count, dtype: int64
```

In [132…

```
### Variable: CVDCRHD4 (had angina or coronary heart disease)

#show pre-processing freqency distribution of variable values
print("pre-processing CVDCRHD4 frequency:\n",df['CVDCRHD4'].value_counts())
print("\nNumber of NaN values in column: ",df['CVDCRHD4'].isna().sum(), "\n")

#convert categorical variable to a binary variable change '2' (no) to 0
df['CVDCRHD4'] = df['CVDCRHD4'].replace(2,0)

#replace 7s (don't know / not sure) and 9s (refused to answer) using mode imputation
df['CVDCRHD4'] = df['CVDCRHD4'].replace(7,df['CVDCRHD4'].mode()[0])
df['CVDCRHD4'] = df['CVDCRHD4'].replace(9,df['CVDCRHD4'].mode()[0])

#show post-processing freqency distribution of variable values
print("post-processing CVDCRHD4 frequency:\n",df['CVDCRHD4'].value_counts())
```

```
pre-processing CVDCRHD4 frequency:
 CVDCRHD4
2.0    58326
1.0     5547
7.0      751
9.0       51
Name: count, dtype: int64

Number of NaN values in column:  0

post-processing CVDCRHD4 frequency:
 CVDCRHD4
0.0    59128
1.0     5547
Name: count, dtype: int64
```

In [133…

```
### Variable: ACEPRISN (lived as child with some who served prison/jail time)

#show pre-processing freqency distribution of variable values
print("pre-processing ACEPRISN frequency:\n",df['ACEPRISN'].value_counts())
print("\nNumber of NaN values in column: ",df['ACEPRISN'].isna().sum(), "\n")

#convert categorical variable to a binary variable change '2' (no) to 0
df['ACEPRISN'] = df['ACEPRISN'].replace(2,0)

#since most records lack responses to this question, variable was recoded to support a second
#analysis on the subset of records for which these data are available. As such, 7s (don't know /
#not sure) and 9s (refused) were merely recoded for now as a third "refused/unknown" category (2).
df['ACEPRISN'] = df['ACEPRISN'].replace(7,2)
df['ACEPRISN'] = df['ACEPRISN'].replace(9,2)

#NaN values will be removed prior to this subcohort analysis.

#show post-processing freqency distribution of variable values
print("post-processing ACEPRISN frequency:\n",df['ACEPRISN'].value_counts())
```

```
pre-processing ACEPRISN frequency:
 ACEPRISN
2.0    17770
1.0      915
9.0      215
7.0       56
Name: count, dtype: int64

Number of NaN values in column:  45719

post-processing ACEPRISN frequency:
 ACEPRISN
0.0    17770
1.0      915
2.0      271
Name: count, dtype: int64
```

In [134…

```
### Variable: ACEDIVRC (parents divorced or separated when respondent was a child)

#show pre-processing freqency distribution of variable values
print("pre-processing ACEDIVRC frequency:\n",df['ACEDIVRC'].value_counts())
print("\nNumber of NaN values in column: ",df['ACEDIVRC'].isna().sum(), "\n")

#convert categorical variable to a binary variable change '2' (no) to 0
df['ACEDIVRC'] = df['ACEDIVRC'].replace(2,0)

#since most records lack responses to this question, variable was recoded to support a second
#analysis on the subset of records for which these data are available. As such, 7s (don't know /
#not sure), 8s (parents not married) and 9s (refused) were merely recoded for now as a third
#"refused/unknown/other" category (2).
```

```
df['ACEDIVRC'] = df['ACEDIVRC'].replace(7,2)
df['ACEDIVRC'] = df['ACEDIVRC'].replace(8,2)
df['ACEDIVRC'] = df['ACEDIVRC'].replace(9,2)

#NaN values will be removed prior to this subcohort analysis.

#show post-processing freqency distribution of variable values
print("post-processing ACEDIVRC frequency:\n",df['ACEDIVRC'].value_counts())
```

```
pre-processing ACEDIVRC frequency:
 ACEDIVRC
2.0    13937
1.0     4467
9.0      236
8.0      184
7.0      121
Name: count, dtype: int64

Number of NaN values in column:  45730

post-processing ACEDIVRC frequency:
 ACEDIVRC
0.0    13937
1.0     4467
2.0      541
Name: count, dtype: int64
```

In [135…

```
### Variable: ACEPUNCH (parents/adults in home beat each other when respondent was a child)

#show pre-processing freqency distribution of variable values
print("pre-processing ACEPUNCH frequency:\n",df['ACEPUNCH'].value_counts())
print("\nNumber of NaN values in column: ",df['ACEPUNCH'].isna().sum(), "\n")

#recode never(1) to no(0)
df['ACEPUNCH'] = df['ACEPUNCH'].replace(1,0)

#recode once(2) and "more than once"(3) to yes(1)
df['ACEPUNCH'] = df['ACEPUNCH'].replace(2,1)
df['ACEPUNCH'] = df['ACEPUNCH'].replace(3,1)

#since most records lack responses to this question, variable was recoded to support a second
#analysis on the subset of records for which these data are available. As such, 7s (don't know /
#not sure), and 9s (refused) were merely recoded for now as a third
#"refused/unknown" category (2).
df['ACEPUNCH'] = df['ACEPUNCH'].replace(7,2)
df['ACEPUNCH'] = df['ACEPUNCH'].replace(9,2)

#NaN values will be removed prior to this subcohort analysis.

#show post-processing freqency distribution of variable values
print("post-processing ACEPUNCH frequency:\n",df['ACEPUNCH'].value_counts())
```

```
pre-processing ACEPUNCH frequency:
 ACEPUNCH
1.0    15432
3.0     2213
2.0      675
9.0      315
7.0      292
Name: count, dtype: int64

Number of NaN values in column:  45748

post-processing ACEPUNCH frequency:
 ACEPUNCH
0.0    15432
1.0     2888
2.0      607
Name: count, dtype: int64
```

In [136…

```
### Variable: ACEHURT1 (parents/adults in home beat respondent as a child)

#show pre-processing freqency distribution of variable values
print("pre-processing ACEHURT1 frequency:\n",df['ACEHURT1'].value_counts())
print("\nNumber of NaN values in column: ",df['ACEHURT1'].isna().sum(), "\n")

#recode never(1) to no(0)
df['ACEHURT1'] = df['ACEHURT1'].replace(1,0)

#recode once(2) and "more than once"(3) to yes(1)
df['ACEHURT1'] = df['ACEHURT1'].replace(2,1)
df['ACEHURT1'] = df['ACEHURT1'].replace(3,1)

#since most records lack responses to this question, variable was recoded to support a second
#analysis on the subset of records for which these data are available. As such, 7s (don't know /
#not sure), and 9s (refused) were merely recoded for now as a third
#"refused/unknown" category (2).
df['ACEHURT1'] = df['ACEHURT1'].replace(7,2)
df['ACEHURT1'] = df['ACEHURT1'].replace(9,2)

#NaN values will be removed prior to this subcohort analysis.

#show post-processing freqency distribution of variable values
print("post-processing ACEHURT1 frequency:\n",df['ACEHURT1'].value_counts())
```

```
pre-processing ACEHURT1 frequency:
 ACEHURT1
1.0    13727
3.0     3546
2.0     1149
9.0      336
7.0      143
Name: count, dtype: int64

Number of NaN values in column:  45774

post-processing ACEHURT1 frequency:
 ACEHURT1
0.0    13727
1.0     4695
2.0      479
Name: count, dtype: int64
```

In [137…

```python
### Variable: ACESWEAR (parents/adults in home swore at and insulted respondent as a child)

#show pre-processing freqency distribution of variable values
print("pre-processing ACESWEAR frequency:\n",df['ACESWEAR'].value_counts())
print("\nNumber of NaN values in column: ",df['ACESWEAR'].isna().sum(), "\n")

#recode never(1) to no(0)
df['ACESWEAR'] = df['ACESWEAR'].replace(1,0)

#recode once(2) and "more than once"(3) to yes(1)
df['ACESWEAR'] = df['ACESWEAR'].replace(2,1)
df['ACESWEAR'] = df['ACESWEAR'].replace(3,1)

#since most records lack responses to this question, variable was recoded to support a second
#analysis on the subset of records for which these data are available. As such, 7s (don't know /
#not sure), and 9s (refused) were merely recoded for now as a third
#"refused/unknown" category (2).
df['ACESWEAR'] = df['ACESWEAR'].replace(7,2)
df['ACESWEAR'] = df['ACESWEAR'].replace(9,2)

#NaN values will be removed prior to this subcohort analysis.

#show post-processing freqency distribution of variable values
print("post-processing ACESWEAR frequency:\n",df['ACESWEAR'].value_counts())
```

```
pre-processing ACESWEAR frequency:
 ACESWEAR
1.0    12475
3.0     5021
2.0      792
9.0      337
7.0      257
Name: count, dtype: int64

Number of NaN values in column:  45793

post-processing ACESWEAR frequency:
 ACESWEAR
0.0    12475
1.0     5813
2.0      594
Name: count, dtype: int64
```

In [138…

```python
### Variable: ACETOUCH (respondent molested by a person 5+ years older or an adult while a child)

#show pre-processing freqency distribution of variable values
print("pre-processing ACETOUCH frequency:\n",df['ACETOUCH'].value_counts())
print("\nNumber of NaN values in column: ",df['ACETOUCH'].isna().sum(), "\n")

#recode never(1) to no(0)
df['ACETOUCH'] = df['ACETOUCH'].replace(1,0)

#recode once(2) and "more than once"(3) to yes(1)
df['ACETOUCH'] = df['ACETOUCH'].replace(2,1)
df['ACETOUCH'] = df['ACETOUCH'].replace(3,1)

#since most records lack responses to this question, variable was recoded to support a second
#analysis on the subset of records for which these data are available. As such, 7s (don't know /
#not sure), and 9s (refused) were merely recoded for now as a third
#"refused/unknown" category (2).
df['ACETOUCH'] = df['ACETOUCH'].replace(7,2)
df['ACETOUCH'] = df['ACETOUCH'].replace(9,2)

#NaN values will be removed prior to this subcohort analysis.

#show post-processing freqency distribution of variable values
print("post-processing ACETOUCH frequency:\n",df['ACETOUCH'].value_counts())
```

```
pre-processing ACETOUCH frequency:
 ACETOUCH
1.0    16068
3.0     1513
2.0      808
9.0      388
7.0       75
Name: count, dtype: int64

Number of NaN values in column:  45823

post-processing ACETOUCH frequency:
 ACETOUCH
0.0    16068
1.0     2321
2.0      463
Name: count, dtype: int64
```

In [139…

```
### Variable: ACETTHEM (person 5+ years older or an adult tried to make respondent touch them sexually while a child)

#show pre-processing freqency distribution of variable values
print("pre-processing ACETTHEM frequency:\n",df['ACETTHEM'].value_counts())
print("\nNumber of NaN values in column: ",df['ACETTHEM'].isna().sum(), "\n")

#recode never(1) to no(0)
df['ACETTHEM'] = df['ACETTHEM'].replace(1,0)

#recode once(2) and "more than once"(3) to yes(1)
df['ACETTHEM'] = df['ACETTHEM'].replace(2,1)
df['ACETTHEM'] = df['ACETTHEM'].replace(3,1)

#since most records lack responses to this question, variable was recoded to support a second
#analysis on the subset of records for which these data are available. As such, 7s (don't know /
#not sure), and 9s (refused) were merely recoded for now as a third
#"refused/unknown" category (2).
df['ACETTHEM'] = df['ACETTHEM'].replace(7,2)
df['ACETTHEM'] = df['ACETTHEM'].replace(9,2)

#NaN values will be removed prior to this subcohort analysis.

#show post-processing freqency distribution of variable values
print("post-processing ACETTHEM frequency:\n",df['ACETTHEM'].value_counts())
```

```
pre-processing ACETTHEM frequency:
 ACETTHEM
1.0    16732
3.0     1066
2.0      583
9.0      363
7.0       86
Name: count, dtype: int64

Number of NaN values in column:  45845

post-processing ACETTHEM frequency:
 ACETTHEM
0.0    16732
1.0     1649
2.0      449
Name: count, dtype: int64
```

In [140…

```
### Variable: ACEHVSEX (person 5+ years older or an adult tried to force respondent to have sex while a child)

#show pre-processing freqency distribution of variable values
print("pre-processing ACEHVSEX frequency:\n",df['ACEHVSEX'].value_counts())
print("\nNumber of NaN values in column: ",df['ACEHVSEX'].isna().sum(), "\n")

#recode never(1) to no(0)
df['ACEHVSEX'] = df['ACEHVSEX'].replace(1,0)

#recode once(2) and "more than once"(3) to yes(1)
df['ACEHVSEX'] = df['ACEHVSEX'].replace(2,1)
df['ACEHVSEX'] = df['ACEHVSEX'].replace(3,1)

#since most records lack responses to this question, variable was recoded to support a second
#analysis on the subset of records for which these data are available. As such, 7s (don't know /
#not sure), and 9s (refused) were merely recoded for now as a third
#"refused/unknown" category (2).
df['ACEHVSEX'] = df['ACEHVSEX'].replace(7,2)
df['ACEHVSEX'] = df['ACEHVSEX'].replace(9,2)

#NaN values will be removed prior to this subcohort analysis.

#show post-processing freqency distribution of variable values
print("post-processing ACEHVSEX frequency:\n",df['ACEHVSEX'].value_counts())
```

```
pre-processing ACEHVSEX frequency:
 ACEHVSEX
1.0    17400
3.0      637
9.0      368
2.0      302
7.0       86
Name: count, dtype: int64

Number of NaN values in column:  45882

post-processing ACEHVSEX frequency:
 ACEHVSEX
0.0    17400
1.0      939
2.0      454
Name: count, dtype: int64
```

In [141…

```
### Variable: SDHISOLT (feelings of social isolation)

#show pre-processing freqency distribution of variable values
print("pre-processing SDHISOLT frequency:\n",df['SDHISOLT'].value_counts())
print("\nNumber of NaN values in column: ",df['SDHISOLT'].isna().sum(), "\n")

#replace NaN rows with mode of distribution
df['SDHISOLT'] = df['SDHISOLT'].fillna(df['SDHISOLT'].mode())

#replace NaN's, 7s (don't know / not sure) and 9s (refused to answer) using mode imputation
df['SDHISOLT'] = df['SDHISOLT'].replace(7,df['SDHISOLT'].mode()[0])
df['SDHISOLT'] = df['SDHISOLT'].replace(9,df['SDHISOLT'].mode()[0])

#show post-processing freqency distribution of variable values
print("post-processing SDHISOLT frequency:\n",df['SDHISOLT'].value_counts())
```

```
pre-processing SDHISOLT frequency:
 SDHISOLT
5.0    25264
4.0    14122
3.0     9907
2.0     2063
1.0     1439
7.0      390
9.0      191
Name: count, dtype: int64

Number of NaN values in column:  11299

post-processing SDHISOLT frequency:
 SDHISOLT
5.0    25845
4.0    14122
3.0     9907
2.0     2063
1.0     1439
Name: count, dtype: int64
```

In [2]:

```
print('Booyah!')
```

```
Booyah!
```

In [ ]: