Variable Encoding

February 7, 2021

Objectives

Perform preliminary steps for structuring the dataset:

1. The dataset contains 58 variables and over 800k observation points. Remove unwanted variable to reduce the size of the data set. The variables that can be immediately removed were identified in the previous week. Others can be removed along the way as necessary.
2. Provide data encoding for some key variables, such as the survey year, meal name, and time of consumption.
3. Obtain the correct key and the condition for seafood consumption items. There are two potential variables for this.

[1]:
```
import pandas as pd

nhanes = pd.read_csv('../../Data/nhanes.csv')


#The following variables have been deemed irrelevant for this analysis, so they␣
 ↪are dropped.


nhanes = nhanes.drop(['DR1IVARA', 'DR1IVB12', 'DR1ICALC', 'DR1IIRON',␣
 ↪'DR1IZINC', 'DR1ISELE', 'DR1IP205',
            'DR1IP226', 'RIDRETH3', 'DR1I_PF_CUREDMEAT', 'DR1I_PF_ORGAN',␣
 ↪'DR1I_PF_POULT', 'DR1I_PF_MPS_TOTAL',
            'DR1I_PF_EGGS', 'DR1I_PF_NUTSDS', 'DR1I_PF_LEGUMES',␣
 ↪'DR1I_PF_TOTAL', 'DR1I_D_TOTAL',
            'DR1I_D_TOTAL', 'DR1I_D_MILK', 'DR1I_D_YOGURT', 'DR1I_D_CHEESE',␣
 ↪'WTDRD1_6YR'], axis=1)



#Map the survey year data, based on the SDDSRVYR encoding key

#Obtain description and value counts
nhanes['SDDSRVYR'].describe()
nhanes['SDDSRVYR'].value_counts()

#Create Survey Year variable based on lookup, mapping from CDC source
```

```
survey_year_lookup = {4: '2005-2006', 5: '2009-2010', 6: '2011-2012', 7:␣
 ↪'2013-2014', 8:'2015-2016', 9:'2017-2018'}
nhanes['Survey_Year'] = nhanes['SDDSRVYR'].map(survey_year_lookup)

#Check for NAs
print("Survey Year NA count is "+str(nhanes['Survey_Year'].isnull().sum()))


#Map the meal occasion data, based on the DR1.030Z encoding key

#Obtain description and value counts
nhanes['DR1.030Z'].describe()
nhanes['DR1.030Z'].value_counts()

#Create Survey Year variable based on lookup, mapping from CDC source
meal_name_lookup = {1: 'Breakfast', 2: 'Lunch', 3: 'Dinner', 4: 'Supper', 5:
 ↪'Brunch', 6:'Snack',
                    7: 'Drink', 8: 'Infant Feeding', 9: 'Extended consumption',␣
 ↪10: 'Desayano',
                    11: 'Almuerzo', 12: 'Comida', 13: 'Merienda', 14: 'Cena',␣
 ↪15: 'Enter comida',
                    16: 'Botana', 17: 'Bocadillo', 18: 'Tentempie', 19:␣
 ↪'Bebida', 91: 'Other'}


nhanes['Meal_Name'] = nhanes['DR1.030Z'].map(meal_name_lookup)

#Check for NAs
print("Meal Name NA count is "+str(nhanes['Meal_Name'].isnull().sum()))
```

```
Survey Year NA count is 0
Meal Name NA count is 0
```

[2]:
```
#Meal Name Counts - Observation Level
nhanes['Meal_Name'].value_counts()
```

[2]:
```
Dinner                 165082
Lunch                  161393
Breakfast              142660
Snack                  136295
Supper                  42739
Drink                   40487
Extended consumption    25242
Infant Feeding          18184
Cena                    18065
Desayano                16198
Comida                  15428
```

```
Almuerzo                13211
Merienda                 7026
Brunch                   6602
Bebida                   4958
Botana                   3291
Bocadillo                2946
Enter comida             2842
Tentempie                 356
Other                       7
Name: Meal_Name, dtype: int64
```

[3]:
```
#Survey Name Counts - Observation Level
nhanes['Survey_Year'].value_counts()
```

[3]:
```
2011-2012    150991
2005-2006    146940
2009-2010    145703
2015-2016    131394
2013-2014    126503
2017-2018    121481
Name: Survey_Year, dtype: int64
```

Meal Time Variable

The time variable can be used for validity checks on meal name, and data grouping of each subject per name. Acccoring the CDC references, the time was collected in the HHMM format. An initial description shows that the time values are in seconds.

[4]:
```
nhanes['DR1.020'].describe()
```

[4]:
```
count    823012.000000
mean      69462.896823
std       17059.701708
min       18000.000000
25%       55800.000000
50%       68400.000000
75%       84600.000000
max      104340.000000
Name: DR1.020, dtype: float64
```

It seems like the data was collected on a 24 hr cycle starting at 5AM and finishing at 4:59AM the next day.

[5]:
```
#Find time minimum and convert seconds to hours
nhanes['DR1.020'].min()/60/60
```

[5]:
```
5.0
```

[6]:
```
#Find time maximum and convert seconds to hours
nhanes['DR1.020'].max()/60/60
```

[6]:
```
28.983333333333334
```

The code below removes the apparent 5AM time collection bias and creates a time variable in a pandas time format.

[7]:
```
#Create a time column, in a pandas time format

#Remove the 5AM bias from the value in seconds
def remove_time_bias(time_in):
    midnight = 24*60*60
    if (time_in >= midnight):
        time_post = time_in - midnight
    else: time_post = time_in
    return round(time_post)

#Create time variable and convert to time formatefrom DR1.020
nhanes['Time'] = nhanes['DR1.020'].apply(remove_time_bias)
nhanes['Time'] = nhanes['Time'].astype(int)
nhanes['Time'] = nhanes['Time'].round().apply(pd.to_timedelta, unit='s')
```

Determine the key for filtering on seafood meals. The two options are DR1I_PF_SEAFD_TOT, which has the amount of seafood consumed in grams and species, which is populated if the item is seafood. First check if there are any rows where DR1I_PF_SEAFD_TOT is not 0 and species is NA (not seafood). This filters yields rows where all DR1I_PF_SEAFD_TOT are NA.

[10]:
```
#Check if there are rows where DR1I_PF_SEAFD_TOT is not 0 and species is NA
filtered_df1 = nhanes[nhanes['DR1I_PF_SEAFD_TOT'] !=0 ]
filtered_df2 = filtered_df1[filtered_df1['species'].isnull()]
filtered_df2['DR1I_PF_SEAFD_TOT']
```

```
Unnamed: 0          5098
SEQN                5098
WTDRD1              5098
DR1ILINE            5098
DR1FS                  0
DR1IFDCD            5098
DR1IGRMS               0
DR1.020             5098
DR1.030Z            5098
DR1.040Z               0
DR1IKCAL               0
DR1IPROT               0
DR1IPFAT               0
RIAGENDR            5098
RIDAGEYR            5098
```

```
RIDRETH1              5098
DMDEDUC3                 0
DMDEDUC2                 0
DMDHHSIZ              5098
DMDFMSIZ              5098
INDHHIN2              5068
INDFMIN2              5060
INDFMPIR              4741
SDMVPSU               5098
SDMVSTRA              5098
DESCRIPTION           5098
DR1I_PF_SEAFD_HI         0
DR1I_PF_SEAFD_LOW        0
DR1I_PF_MEAT             0
DR1I_PF_SOY              0
SDDSRVYR              5098
DR1I_PF_SEAFD_TOT        0
DR1I_PF_MEAT_TOT         0
species                  0
species_code             0
DR1.030Z_2            5098
Survey_Year           5098
Meal_Name             5098
Time                  5098
dtype: int64
```

[10]:
```
1001     NaN
1002     NaN
1004     NaN
1005     NaN
1008     NaN
          ..
822185   NaN
822186   NaN
822187   NaN
822188   NaN
822189   NaN
Name: DR1I_PF_SEAFD_TOT, Length: 5098, dtype: float64
```

Next check if there are any values where DR1I_PF_SEAFD_TOT is 0, but species is populated with a fish. There are 189 rows that meet this condition, meaning that not all rows that have a species populated have a seafood consumption that is greater than 0 grams. Therefore, it is safest to use DR1I_PF_SEAFD_TOT where it is not 0 or NA as the key for seafood items. That is because this variable is part of the actual survey.

[11]:
```
filtered_df1 = nhanes[nhanes['DR1I_PF_SEAFD_TOT'] == 0 ]
filtered_df2 = filtered_df1[filtered_df1['species'].notnull()]
filtered_df2['DR1I_PF_SEAFD_TOT']
```

[11]:
```
23764     0.0
47269     0.0
52541     0.0
59989     0.0
59996     0.0
          …
815468    0.0
817023    0.0
818396    0.0
818397    0.0
822484    0.0
Name: DR1I_PF_SEAFD_TOT, Length: 189, dtype: float64
```

Apply the identified condition to the key and obtain statistics to ensure that there are no values that seem erroneous. There is a min of 0.01 and a max of 44.12, so no erroneous values are apparent.

[13]:
```
#Apply key and obtain description, to ensure that there are no erroneous values
filtered_df1 = nhanes[nhanes['DR1I_PF_SEAFD_TOT'] != 0 &␣
 ↪nhanes['DR1I_PF_SEAFD_TOT'].notnull()]
filtered_df1['DR1I_PF_SEAFD_TOT'].describe()
```

[13]:
```
count    9789.000000
mean        2.729516
std         3.011448
min         0.010000
25%         0.870000
50%         1.820000
75%         3.490000
max        44.120000
Name: DR1I_PF_SEAFD_TOT, dtype: float64
```

Conclusions

1. Preliminary removal of 23 variables that are identified as not required from previous week. More variables can be removed as the project progresses, to reduce dataset for more complex analysis tasks.
2. Three new variables added as encoders for the following:
   - Survey Year
   - Meal Name
   - Time Conversion: This was done after investigations about time format in the dataset. It seems apparent that the collected time uses a 24H starting at 5AM. So the 5AM time bias is removed and conversion to time format is performed. This operation is a bit time consuming and could be best performed once non-seafood observations are removed.
3. The safest variable to use as a key for filtering out seafood items is DR1I_PF_SEAFD_TOT and the condition is that it is not 0 and not NA.