Global Web Index

Junior Data Engineer Task

Matthew Bibby

Introduction

This report aims to convey how the assigned problem was undertaken, as well as the thought processes behind the decisions made, and any problems that were faced. The problem involved a number of steps including; learning about how and why to weight respondents in a population, cleaning a dataset which is going to be worked on, and fixing a bug in a factor weighting function.

Cleaning and Setting up the CSV File

Describing the Data

The first step was to describe the data, this was achieved by using some pandas functions. The .shape function provides some basic information on the data, such as the number of records and columns containted within the data. Most of the data was low integers such as 1 or 0, with some data being slightly higher.

In [1]:

```
import pandas as pd
import numpy as np

def main():

    # Read in input file
    df = pd.read_csv('input_data.csv', index_col=0)

    # Get the shape of the input data
    shape = df.shape

    print("Number of records: " + str(shape[0]))
    print("Number of columns: " + str(shape[1]))

if __name__ == '__main__':
    main()
```

```
Number of records: 100200
Number of columns: 2500
```

Removing Duplicate Indexes

This step involved removing duplicate indexes from the data. The pandas library was used here, with the .duplicated function being used to remove the duplicates. The results below show that there are no longer any duplicated indexes in the data.

In [5]:

```
import pandas as pd
import numpy as np

def main():

    # Read in input file
    df = pd.read_csv('input_data.csv', index_col=0)

    # Remove duplicate indexes, keep the first instance
    dups = df.index.get_level_values(0).get_duplicates()

    print( str(len(dups)) + ' Indexes were found to be duplicates')

if __name__ == '__main__':
    main()
```

```
99 Indexes were found to be duplicates
```

In [1]:

```
import pandas as pd
import numpy as np

def main():

    # Read in the CSV file
    df = pd.read_csv('input_data.csv', index_col=0)

    # Get duplicate indexes
    nodupes = df[~df.index.duplicated(keep='first')]

    nodupestest = nodupes.index.get_level_values(0).get_duplicates()

    print(str(len(nodupestest)) + ' Indexes were found to be duplicates')



if __name__ == '__main__':
    main()
```

```
0 Indexes were found to be duplicates
```

Removing NaNs

This step involved removing NaNs from the data. NaNs could be introduced into the data if the user enters something incorrectly and the input isn't properly checked, or if there is no way for a user to answer a specific question so it is left blank. NaNs could be prevented by checking all input to make sure that is valid, in some cases a default value may also be given if the user doesn't enter something.
In order to remove NaNs the pandas library was used again, the dropna function allows NaNs to be removed from a dataset, in this instance a subset is provided as only columns 'q2' and 'q4' are being worked on in the weighting functions.
The print statements below show the columns that contain NaNs before and after the pandas function has been applied. The data now has no duplicate indexes, and columns 'q2' and 'q4' contain no NaNs.

```
In [2]:
```

```
import pandas as pd
import numpy as np

def main():

    # Read in the CSV file
    df = pd.read_csv('input_data.csv', index_col=0)

    # Get duplicate indexes
    nodupes = df[~df.index.duplicated(keep='first')]

    # Remove NaNs from dataset
    nonulls = nodupes.dropna(subset=['q2', 'q4'])

    print("Without nulls being removed:  " + str(df.columns[df.isnull().any()]))
    print("With nulls being removed:  " + str(nonulls.columns[nonulls.isnull().any()]))

if __name__ == '__main__':
    main()
```

```
Without nulls being removed:  Index(['q2', 'q4'], dtype='object')
With nulls being removed:  Index([], dtype='object')
```

Merging With New Data

The final step involved merging two files together using the respondent number as an index. This was accomplished by performing a left join on the first file with the new file to be added. The left join kept only data that matched between the index from the original data and the index on the new data. In this case, a new column was added to the original data called q3 and every time the new data had an entry that matched an existing respondent number; the q3 data for that index was added.
There are other ways that data can be merged when dealing with two datasets. There are many joins that can be used, for example a right join would perform a similar operation to the left join but only keeping data matching the index on the right. An inner join could also be used to find an intersection between datasets, and an outer join could be used to include all data with a match from either dataset.

In [ ]:

```
import pandas as pd
import numpy as np

def main():

    # Read in input file
    df = pd.read_csv('input_data.csv', index_col=0)

    # Remove duplicate indexes, keep the first instance
    nodupes = df[~df.index.duplicated(keep='first')]

    # Remove NaNs from dataset
    nonulls = nodupes.dropna(subset=['q2', 'q4'])

    # Read in q3 values
    df_q3 = pd.read_csv('q3_column.csv', index_col=0)

    # Merge q3 values with rest of dataframe
    finaldf = nonulls.join(df_q3, how='left')

    # Write out cleaned csv file
    finaldf.to_csv('cleaned.csv')

if __name__ == '__main__':
    main()
```

Fixing Weighting Function Bug

Unfortunately I could not complete this section. After attempting to run the program a number of times I always encountered runtime errors related to a number of indexes. I attempted to search for possible solutions for this but did not want to change much of the code as it may have broken other sections. I'm not sure if this was due to the setup on my personal computer.
Without being able to successfully run the program I can see that the count variable is not used within the weighting function. I would estimate that this could be used in a way to work out a ratio between the actual number of respondents and the universe size, and then assign a weight as used in the example to even them out.