JupyterNotebookDataCaptureTool

June 20, 2022

1 Title: Collecting data using interactive Jupyter widgets

Author details: *Author:* B208593

2 Data

The data are from the NHSRdatasets package: the NHS England accident and emergency (A&E) attendances and admissions (`ae_attendances`) data. A subset of the variables was selected using R for this data capture tool, including period, organisation code, attendances, breaches and performance.

[ ]:
```
#Load the 'pandas' package
import pandas as pd
testData=pd.read_csv("../Data/ae_type1_performance_test_full.csv")
testData
```

[ ]:
```
#Check data types of variables in the data frame
result = testData.dtypes
print("Output:")
print(result)
```

[ ]:
```
#Set up an empty data frame with the selected variables to collect the data␣
 ↪captured by the Juypter widgets
dfTofill = pd.DataFrame({'index': [0],# Integer
                    'period': [pd.Timestamp('20000101')], # Date
                    'org_code': ['NA'], # String
                    'attendances': [0], # Integer
                    'breaches': [0], # Integer
                    'performance': [0.0], # Float
                    'consent': [False]}) # Boolean

dfTofill
```

[ ]:
```
#Save the empty data frame to the 'Data' folder. Only run once otherwise it␣
 ↪will overwrite the data each time we run the code
#dfTofill.to_csv('../Data/CollectedData.csv', index=False)
```

[ ]:
```
#Read in the empty data frame
CollectData=pd.read_csv("../Data/CollectedData.csv")
CollectData
```

[ ]:
```
#We have to use indexing to connect the test data to the original set. We have␣
 ↪to change it for each record
index_number=2826
dfTofill.iloc[0,0]=index_number
dfTofill
```

[ ]:
```
#To use the widgets we have to load the 'ipywidgets' package
import ipywidgets as widgets
```

[ ]:
```
#To display the different objects (widgets) in Jupyter we have to load the␣
 ↪'IPython.display' package
from IPython.display import display
```

3 Consent

We need a widget that displays a boolean value (TRUE or FALSE)

[ ]:
```
#For the consent variable we will use boolean widget (Checkbox widget) that is␣
 ↪designed to display a boolean value (TRUE or FALSE)
a = widgets.Checkbox(
    value=False,
    description='I consent for the data I have provided to be processed and␣
 ↪shared in accordance with data protection regulations with the purpose of␣
 ↪improving care service provision across the UK.',
    disabled=False
)
display(a)
```

[ ]:
```
#Fill in the 6th column (Python indexed) in the empty data frame
dfTofill.iloc[0,6]=a.value
dfTofill
```

4 Period variable

The data type of the period variable is string, character, We need a widget that displays a date format so we can update it for each record.

[ ]:
```
#The data type of the period variable is string, character so we set up a␣
 ↪DatePicker widget to collect the period data
b = widgets.DatePicker(
    description='Period',
    disabled=False
```

```
)
display(b)
```

[ ]:
```
#Fill in the 1st column (Python indexed) in the empty data frame
dfTofill.iloc[0,1]=b.value
dfTofill
```

5 Organisation variable

Data type is string, character. We need a widget that enables us to select the correct organisation code for each record.

[ ]:
```
# We must first use the pandas package unique() function to get the unique␣
↪Organisation data service (ODS) codes in the test data.
org_code=list(testData['org_code'].unique())
org_code
```

[ ]:
```
# To display single selection lists for the org_code variable we set up a␣
↪selection widget
c=widgets.Select(
    options=org_code,
    value='RGT',
    rows=len(org_code),
    description='ODS code:',
    disabled=False
)
display(c)
```

[ ]:
```
#Fill in the 2nd column (Python indexed) in the empty data frame
dfTofill.iloc[0,2]=c.value
dfTofill
```

6 The attendances variable

Data type is numeric, integer so we need a widget that displays numeric data and we can put in the correct number for this variable.

[ ]:
```
e=widgets.IntText(
    value=0,
    description='Attendances:',
    disabled=False)
display(e)
```

[ ]:
```
#Fill in the 3th column (Python indexed) in the empty data frame
dfTofill.iloc[0,3]=e.value
dfTofill
```

7 The breaches variable

Data type is numeric, integer so we need a widget that displays numeric data and we can put in the correct number for this variable.

[ ]:
```
f=widgets.IntText(
    value=0,
    description='Breaches:',
    disabled=False)
display(f)
```

[ ]:
```
#Fill in the 4th column (Python indexed) in the empty data frame
dfTofill.iloc[0,4]=f.value
dfTofill
```

8 The performance variable

Data type is numeric, float so we need a widget that displays numeric, float data and we can put in the correct number for this variable.

[ ]:
```
h=widgets.FloatText(
    value=0.0,
    description='Performance:',
    disabled=False
)
display(h)
```

[ ]:
```
#Fill in the 5th column (Python indexed) in the empty data frame
dfTofill.iloc[0,5]=h.value
dfTofill
```

9 Concatenating the collected data to the CollectData data frame.

To fill in the rows in the empty data frame

[ ]:
```
#We need use the `concat()` function from the Python *pandas* package to append␣
↪the CollectData and dfTofill data frames. The concat() function is used to␣
↪concatenate *pandas* objects.
# CollectData is the first data frame
# dfTofill is the second data frame
CollectData  = pd.concat([CollectData, dfTofill])
display(CollectData)
```

[ ]:
```
#collect only the data that we have consent to do
CollectData=CollectData[CollectData['consent'] == True]
display(CollectData)
```

10 Saving the CollectData data frame

[ ]:
```
#Saving the data collected by your data-capture tool to the working Data folder:
CollectData.to_csv('../Data/CollectedData.csv', index=False)
```

[ ]:
```
#Saving the data collected by your data-capture tool to the working Raw folder:
CollectData.to_csv('../RawData/CollectedDataFinal.csv', index=False)
```