Working with data types and structures in Python and R: Assessment R Markdown Document

B208593

22 June, 2022

Link to the GitHub repository

I. Loading NHSR datasets

The data I used for this assessment are from the NHSRdatasets package: the NHS England accident and emergency attendances and admissions (*ae_attendances*) data set.

1. Load packages

First we need to load the packages we will use in the script.

```
library(NHSRdatasets)
```

```
library(tidyverse)
```

```
library(here)
```

```
library(knitr)
```

```
library(scales)
```

```
library(lubridate)
```

```
library(caret)
```

2. Load the NHS England accident and emergency attendances and admissions (*ae_attendances*) dataset

```
data(ae_attendances)
```

3. Viewing the *ae_attendances* data

```
data(ae_attendances)
ae<-ae_attendances
class(ae)
```

Missing data

We need to check for missing data in the *ae_attendances* data as missing data can significantly impact any attempt to gain meaningful insight from data.

```
# Calculate how many NAs there are in each variable
ae %>%
  map(is.na) %>%
map(sum)
```

Adding index column to raw data

To set up a data capture tool in Python, the raw data are needed to be split into training and testing sets. We need to add an index column to the raw data so we can link the partitioned data sets to the raw data.

```
ae <- rowid_to_column(ae, "index")
```

Save the raw *ae_attendances* data to your 'RawData' folder

```
write_csv(ae, here("RawData", "ae_attendances.csv"))
```

Creating a new variable

We need to calculate monthly four hour waiting time target performance for the exploratory analysis

```
ae<- ae %>%
mutate(performance = round(1- breaches / attendances,2))
```

4. Selecting variables for the data capture tool

The variables I have selected for the data capture tool are: index, period, org_ code, attendance, breaches and performance

Filter type 1 A&E departments and select variables for the data capture tool

```
ae2<-ae %>%
  filter(type=="1") %>%
  select(-type, -admissions)
```

Tabulate the raw data

We tabulate the raw data to make the date and numbers more readable (date will appear in month-year format and numbers will have a coma at 1000s).

```
ae2 %>%
  mutate_at(vars(period), format, "%b-%y") %>%
  mutate_at(vars(attendances, breaches), comma) %>%
  head(10) %>%
  kable()
```

Save csv file to 'RawData' folder

```
write_csv(ae2, here("RawData", "ae_type1_performance.full.csv"))
```

5. Split the raw data into test and training data

We have to work out the proportion of the raw data to assign to the training data (it should be 10-15).

```
prop<-(1-(15/nrow(ae2)))
print(prop)
```

Random number generator

The 'set.seed()' function is a random number generator which will make sure that every time we run this script the raw data will be split into the same test and training data.

```
set.seed(333)
```

Split dataset

We use the 'createDataPartition()' function to split our raw data into test and training data sets. The 'trainIndex' will contain the training data.

```
trainIndex <- createDataPartition(ae2$index, p = prop,
                                  list = FALSE,
                                  times = 1)
```

```
ae2Train <- ae2[ trainIndex,]
```

Save *ae_type1_performance* training data to the 'Data' folder

```
write_csv(ae2Train, here("Data", "ae_type1_performance_train_full.csv"))
```

Extract the *ae_type1_performance* test data records not in the 'trainIndex' so we will left with the test data.

```
ae2Test  <- ae2[-trainIndex,]
```

Marker test data

We need to set aside the first record from the ae_type1_performance test data for the markers to test and evaluate the data-capture tool.

```
ae2TestMarker  <- ae2Test[1,]
```

Save *ae_type1_performance* marker test data to the 'Data' folder

```
write_csv(ae2TestMarker, here("Data", "ae_type1_performance_test_marker_full.csv"))
```

Set aside the remaining records to test the data-capture tool.

```
ae2Test  <- ae2Test[2:nrow(ae2Test),]
```

Finally save *ae_type1_performance* test data to the 'Data' folder

```
write_csv(ae2Test, here("Data", "ae_type1_performance_test_full.csv"))
```

II. Data capture tool in Jupyter notebook

1. Data upload

We need to upload the previously selected subset of the variables (*ae_type1_performance_test_full*) for the data capture tool.

Load the pandas package

To import the data, we need to load the pandas package that is used for data manipulation and analysis.

Data type

We need to check the data type of the variables in the test data frame in order to know what type of widget we need to use.

2. Setting up an empty data frame

First we need to set up an empty data frame in the working data folder to collect the data captured by the Juypter widgets containing only the variables we want to capture Next we save the empty data frame to the Data folder. Finally we read in the empty data frame to collect the data from the Jupyter-widgets.

3. Indexing in Python

The first variable contains the index number, that allows us to connect the test data to the original data set "../RawData/ae_type1_performance_test_full". We need to change this for every record in the test dataset.

4. Widgets for the variables

We need to select widgets (eg. button, dropdown or textbox) to collect the test data in Python. To use the widget framework, we need to import the *ipywidgets* Python package.

Consent variable

Before we collect the data , we need to get consent from the end-user to process and share the data we will collect with the data capture tool. The consent data type has 2 options, TRUE or FALSE (logical data type) so we need to use a boolean widget (Checkbox widget) that is designed to display a boolean value.

Period variable

Data type is string, character. DatePicker widget was selected to capture/collect period variable as date.

Org_code variable

Data type is string, character. Selection widget was selected to capture this variable with displaying a single selection list.

Attendances, breaches variables

Data type for these variables is numeric, integer. Numeric widget was selected (*IntText*) for displaying integers.

Performance variable

Data type is numeric, float. Numeric widget (*FloatText*) was selected to collected this data.

5. Concatenating the collected data to the CollectData data frame.

Every row needs to be filled in the empty data frame by using 'concat()' function to concatenate pandas objects (11 rows of data).

Consent check

Before we save our data to file, we must make sure we have consent to do so (we have consent filled in as TRUE).

Saving the CollectData data frame

We need to save the data collected by the data-capture tool to the Data and Raw folders.

III. Data dictionary

1. Load packages

```
library(dataMeta)
library (tidyverse)
library(here)
```

2. Data used

Test data was collected by the Jupyter Notebook "./ipynbScripts/JupyterNotebookDataCaptureTool.ipynb".

Read the collected data from the 'RawData' folder.

```
CollectedData=read_csv(here("RawData", "CollectedDataFinal.csv"))
```

Check type of the 'period' variable

```
class(CollectedData$period)
```

It's identified as *date* but we need the 'period' to be *character* for the Data dictionary.

Change 'period' variable from *date* to *character*

```
CollectedData$period<-as.character(CollectedData$period)
```

Check type of 'period' variable

```
class(CollectedData$period)
```

Now it's a *character* variable.

3. Building a data dictionary for the data captured by the data capture tool

Building a linker data frame

First a linker data frame needs to be created: for the different variable descriptions and for the variable types two string vectors will be created.

Variable descriptions

Setting up a string vector for the different variable descriptions.

```
variable_description <- c("The index column that allows us to link the data collected to the original a
"The month that this activity relates to, stored as a date (1st of each month).",
"The Organisation data service (ODS) code for the organisation.",
"The number of attendances for type 1 A&E department at this organisation for this month.",
"The number of attendances that breached the four-hour target.",
"The performance ([1 - breaches]/attendances) calculated for type 1 A&E departments.",
"The consent from the end-user to process and share the data collected with the data capture tool.")
print(variable_description)
```

Variable types

Setting up a string vector with values 0 or 1 for the different variable types. For variables with quantitative values we will use 0 (measured values) and 1 for for variables with fixed values (e.g. codes).

To view the variable types in the 'CollectedData' data frame.

```
glimpse(CollectedData)
```

We have four variables with quantitative values (*index,attendances, breaches, performance*) and three variables with fixed values (*date, org_code, consent*)

```
variable_type <- c(0, 1, 1, 0, 0, 0, 1)
print(variable_type)
```

4. Linker

We need to build an intermediary (linker) data frame between the 'CollectedData' and the data dictionary so the `build_linker()` function is used from the *dataMeta* package.

```
linker<-build_linker(CollectedData, variable_description, variable_type)
print(linker)
```

5. Data dictionary

To construct a data dictionary for a 'CollectedData' data frame with the help of the linker data frame between we use the `build_dict()` function .

```
dictionary <- build_dict(my.data = CollectedData, linker = linker)
glimpse(dictionary)
```

Fill in the organisation codes

```
dictionary[5,4]<-" NHS Trust - CAMBRIDGE UNIVERSITY HOSPITALS NHS FOUNDATION TRUST (RGT)."
dictionary[6,4]<-" NHS Trust - NORFOLK AND NORWICH UNIVERSITY HOSPITALS NHS FOUNDATION TRUST (RM1)."
dictionary[7,4]<-" NHS Trust - UNIVERSITY HOSPITALS BIRMINGHAM NHS FOUNDATION TRUST (RRK)."
dictionary[8,4]<-" NHS Trust - EAST CHESHIRE NHS TRUST (RJN)."
dictionary[9,4]<-" NHS Trust - SALISBURY NHS FOUNDATION TRUST (RNZ)."
dictionary[10,4]<-" NHS Trust - BIRMINGHAM WOMEN'S AND CHILDREN'S NHS FOUNDATION TRUST (RQ3)."
dictionary[11,4]<-" NHS Trust - WIRRAL UNIVERSITY TEACHING HOSPITAL NHS FOUNDATION TRUST (RBL)."
dictionary[12,4]<-" NHS Trust - LONDON NORTH WEST UNIVERSITY HEALTHCARE NHS TRUST (R1K)."
dictionary[13,4]<-" NHS Trust - MEDWAY NHS FOUNDATION TRUST (RPA)."
dictionary[14,4]<-" NHS Trust - WESTON AREA HEALTH NHS TRUST (RA3)."
dictionary[15,4]<-" NHS Trust - EAST SUFFOLK AND NORTH ESSEX NHS FOUNDATION TRUST (RDE)."
```

Save the data dictionary for CollectedData to the 'RawData' folder

```
write_csv(dictionary, here("RawData", "CollectedData_DataDictionary.csv"))
```

6. Append data dictionary to the 'CollectedData'

Attributes need to be built in as metadata to the 'CollectedData' using the 'incorporate_attr()' function. To run this function, the 'CollectedData' and dictionary and main_string are required to be inputs.

Create main_string for attributes

Main_string is a character string describing the 'CollectedData' data frame.

```
main_string <- "This data describes the NHS England type 1 accident and emergency (A&E) attendances and
main_string
```

Incorporate attributes as metada

The 'incorporate_attr()' function is used to return an R dataset containing metadata stored in its attributes.

```
complete_CollectedData <- incorporate_attr(my.data = CollectedData, data.dictionary = dictionary,
main_string = main_string)
attributes(complete_CollectedData)$author[1]<-"B208593"
complete_CollectedData
attributes(complete_CollectedData)
```

Save the CollectedData with attributes

```
save_it(complete_CollectedData, here("RawData", "complete_CollectedData"))
```