Loading NHSRDatasets for Exploratory Analysis and Data Capture Tool

B199291

27 June, 2022

My GitHub Repo : Click Here

#Loading NHSRdatasets and required packages.

```
library(NHSRdatasets)
library(tidyverse)
library(here)
library(knitr)
library(scales)
library(lubridate)
library(caret)
library(ggplot2)
```

#Loading ae_attendances Dataset

```
data(ae_attendances)
```

Let's have a look at the ae_attendances data

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

I looked the ae_attedances data review and its class using class function and glimpse function from tidyverse. The data frame has 12,765 rows of data and six columns of different variables with different classes. I can see period (date variable), org_code and type (factor variable), attendances, breaches and admissions as numerice(double precision)variables. As per my intention, I will use type(factor) variable to subset into new data set.

Missing data checking

```
ae %>%
  map(is.na) %>%
map(sum)
```

Just to make sure that there is no missing data in the table and the data is complete.

Let's add an index link column to ae_attendances data

For DCT development and training and testing dataset separation, i will add index ref column.

```
ae <- rowid_to_column(ae, "Index")
glimpse(ae)
write_csv(ae, here("RefData", "ae_attendances.csv"))
```

Let's subset my raw data first into type 1 org only data

```
ae_type1 <- subset(ae,type=='1')
unique(ae_type1$type)
unique(ae_type1$org_code)
```

Let's tablulate Type1 hospital data and save for my upcoming works

```
ae_type1 %>%
  mutate_at(vars(period), format, "%b-%y") %>%
  mutate_at(vars(attendances, breaches, admissions), comma) %>%
  head(10) %>%
  kable()
write_csv(ae_type1, here("WorkingData", "ae_type1.csv"))
```

Getting Total Attendances of all Type 1 hopsitals

```
Type1_attendances <- ae_type1 %>%
  group_by(org_code) %>%
  summarise_at(vars(attendances, breaches), sum)
glimpse(Type1_attendances)
```

Getting Type1 hospitals with attendances in descending order and saving for record

```
Type1_att_descending <-Type1_attendances[order(-Type1_attendances$attendances),]
glimpse(Type1_att_descending)
write_csv(Type1_att_descending, here("WorkingData", "Type1_attendances_descending.csv"))
```

Subsetting new df with top 5 hospitals

```
Top5_attendance <- head(Type1_att_descending,5)
write_csv(Top5_attendance, here("WorkingData", "Top5_attendance.csv"))
```

Tabulating Top 5 most visited hospitals

```
Top5_attendance %>%
  mutate_at(vars(attendances, breaches), comma) %>%
  kable()
```

Creating new dataset of top 5 hospitals for creating visual

```
Top5_visualready <- filter(ae_type1, org_code %in% Top5_attendance$org_code)
Top5_visualready$performance <- 1-(Top5_visualready$breaches/Top5_visualready$attendances)
```

```
glimpse(Top5_visualready)
write_csv(Top5_visualready, here("WorkingData", "Top5_visualready.csv"))
```

Visualising Top 5 hospital breach percentage by time

```
ggplot(Top5_visualready, aes(x=period, y=performance, group=org_code, colour=org_code)) +
  geom_line (size=0.75) + scale_y_continuous(labels = percent) +
  scale_x_date(date_labels = "%b-%y", date_breaks = "11 month")+
  labs(x = "Month of attendance",
       y = "% of A&E attendances that met 4 hour standard in Top 5 visited
       hospitals",
       title = "NHS England accident and emergency (A&E) four hour performance in Top 5
       visited hospitals",
       caption = "Source: NHSRdatasets")+
  theme(plot.title = element_text(hjust = 0.5))
```

Separating provisionalae_type1 data into training and testing sets

Indexing to create train and test data

```
nrow(ae_type1) #to confirm number of rows again
prop<-(1-(15/nrow(ae_type1)))
print(prop)
set.seed(333)
#Partitioning the raw data into the test and training data.
trainIndex <- createDataPartition(ae_type1$Index, p = prop,
                                  list = FALSE,
                                  times = 1)
head(trainIndex)
# All records that are in the trainIndex are assigned to the training data.
ae_type1Train <- ae_type1[ trainIndex,]
nrow(ae_type1Train)
```

There are 12,753 records in your training data. That is a large dataset!

Creating Training Dataset

```
ae_type1Train %>%
  mutate_at(vars(period), format, "%b-%y") %>%
  mutate_at(vars(attendances, breaches), comma) %>%
  head(10) %>%
  kable()
write_csv(ae_type1Train, here("WorkingData", "ae_type1_train.csv"))
```

Creating Testing Dataset

```
ae_type1_test  <- ae_type1[-trainIndex,]
nrow(ae_type1_test)
```

```
ae_type1TestMarker  <- ae_type1_test[1,]
ae_type1TestMarker  %>%
  mutate_at(vars(period), format, "%b-%y") %>%
  mutate_at(vars(attendances, breaches), comma) %>%
  head(10) %>%
  kable()
write_csv(ae_type1TestMarker, here("WorkingData", "ae_type1_testmarker.csv"))
```

Tabulating and saving TestMarker

Saving remaining test records

```
ae_type1_test  <- ae_type1_test[2:nrow(ae_type1_test),]
ae_type1_test  %>%
  mutate_at(vars(period), format, "%b-%y") %>%
  mutate_at(vars(attendances, breaches), comma) %>%
  head(10) %>%
  kable()
write_csv(ae_type1_test, here("WorkingData", "ae_type1_test.csv"))
```

Data Dictionary for Type 1 Hospitals Data Capture Tool

B199291

27 June, 2022

Load packages for dat dictionary creation

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

Read csv funtion to look collected dataset.

```
CollectedData=read_csv(here("RefData", "CollectedData_final.csv"))
glimpse(CollectedData)
```

##Creating Variable Description

```
variable_description <- c("The index column that allows us to link the data
collected to the original ae_attendances data in the 'RefData' folder.",
"The month that this activity relates to, stored as a date (1st of each month).",
"The Organisation data service (ODS) code for the organisation. If you want to
know the organisation associated with a particular ODS code, you can look it up
from the following address: https://odsportal.digital.nhs.uk/Organisation/Search.",
"The number of attendances for this department type at this organisation for
this month.","The number of attendances that breached the four-hour target.",
"The consent from the end-user to process and share the data collected with
the data capture tool.")
print(variable_description)
```

Creating Variable types

```
glimpse(CollectedData)
```

We have three quantitative values (measured values) variables and three fixed values (allowable values or codes) variables.

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

###**Building Data Linker for connection between Collected Data and Data Dictionary

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

Data dictionary

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

```
dictionary[5,4]<-"NHS Trust - CAMBRIDGE UNIVERSITY HOSPITALS NHS FOUNDATION TRUST"
dictionary[6,4]<-"NHS Trust - NORFOLK AND NORWICH UNIVERSITY HOSPITALS NHS FOUNDATION TRUST"
dictionary[7,4]<-"NHS Trust - UNIVERSITY HOSPITALS BIRMINGHAM NHS FOUNDATION TRUST "
dictionary[8,4]<-"NHS Trust - EAST CHESHIRE NHS TRUST "
dictionary[9,4]<-"NHS Trust - SALISBURY NHS FOUNDATION TRUST "
dictionary[10,4]<-"NHS Trust - BIRMINGHAM WOMEN'S AND CHILDREN'S NHS FOUNDATION TRUST "
dictionary[11,4]<-"NHS Trust - WIRRAL UNIVERSITY TEACHING HOSPITAL NHS FOUNDATION TRUST "
dictionary[12,4]<-"NHS Trust - LONDON NORTH WEST UNIVERSITY HEALTHCARE NHS TRUST "
dictionary[13,4]<-"NHS Trust - MEDWAY NHS FOUNDATION TRUST "
dictionary[14,4]<-"NHS Trust - WESTON AREA HEALTH NHS TRUST"
dictionary[15,4]<-"NHS Trust - EAST SUFFOLK AND NORTH ESSEX NHS FOUNDATION TRUST "
```

Let's save the data dictionary for CollectedData to the 'RefData' folder

```
glimpse(dictionary)
write_csv(dictionary, here("RefData", "CollectedData_Datadictionary.csv"))
```

Append data dictionary to the CollectedData

Create main_string for attributes

```
main_string <- "This data describes the NHS England accident and emergency
(A&E) attendances and breaches of four-hour wait time target data of Type 1
Hospitals from the *NHSRdatasets* package collected by the data capture tool."
main_string
```

Incorporate attributes as metada

```
complete_CollectedData <- incorporate_attr(my.data = CollectedData,
data.dictionary = dictionary,main_string = main_string)
attributes(complete_CollectedData)$author[1]<-"B199291"
complete_CollectedData
attributes(complete_CollectedData)
```

Save the CollectedData with attributes

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

CollectingDataUsingInteractiveJupyterWidgetsWithEndUserReactiveForm_B

June 27, 2022

1 Title: Data Capture tool for Collecting Attendance and Breach data of Type 1 Hospitals

Author details: B199291. *Contact details:* .
Notebook and data info: This Notebook provides a web form to collect the NHS England accident and emergency attendances and admissions (ae_attendances) in the Type 1 A&E Departments and save it to my 'RefData' folder. **Data:** Data consists of date, numerical data and character data.

2 Data

The data I will be managing on this project are from the NHSRdatasets package.The dataset I have chosen to manage from the NHSRdatasets package is the NHS accident and emergency (A&E) attendances and admissions (`ae_attendances`) data. The `ae_attendances` data includes reported attendances, four-hour breaches and admissions for all A&E departments in UK for 2016/17 through 2018/19 (Apr-Mar). For the analysis of NHS UK to be able to monitor the benchmark of type 1 hospitals,I will use a subset of the variables which are organization codes of type 1 emergency departments,recorded month, attendances and breaches, and subsetted the data into test and training data.

[1]:
```
#Load the 'pandas' package to import data
import pandas as pd
testData=pd.read_csv("../WorkingData/ae_type1_test.csv")
testData
```

[1]:

	Index	period	org_code	type	attendances	breaches	admissions
0	2826	2016-07-01	RGT	1	10081	2288	3123
1	2865	2016-07-01	RM1	1	10303	1080	2891
2	3941	2016-04-01	RRK	1	9044	1195	2744
3	5431	2017-12-01	RJN	1	4238	1208	971
4	5940	2017-11-01	RNZ	1	3940	209	1131
5	6727	2017-08-01	RQ3	1	3801	86	754
6	7952	2017-05-01	RBL	1	8270	2120	2491
7	8427	2019-03-01	R1K	1	12753	3178	6522
8	10489	2018-10-01	RPA	1	7710	1750	2065

```
9   12390  2018-05-01      RA3     1         4390       348        1090
10  12463  2018-04-01      RDE     1         7942       523        2398
```

[2]:
```
#checking the data type of the reference data frame so that we can collect␣
↪right data type when we build data capture tool
result = testData.dtypes
print(result)
```

```
Index           int64
period         object
org_code       object
type            int64
attendances     int64
breaches        int64
admissions      int64
dtype: object
```

[3]:
```
#looking columan names to map data collection tool
testData.head(n=1)
```

[3]:
```
   Index      period org_code  type  attendances  breaches  admissions
0   2826  2016-07-01      RGT     1        10081      2288        3123
```

We need to set up an empty data frame including consent information in the working data folder to collect the data captured by the Juypter widgets.

[4]:
```
dfTofill = pd.DataFrame({'index': [0],# Integer
                         'period': [pd.Timestamp('20000101')], # Date
                         'org_code': ['NA'], # String
                         'attendances': [0], # Integer
                         'breaches': [0], # Integer
                         'consent': [False]}) # Boolean

dfTofill
```

[4]:
```
   index     period org_code  attendances  breaches  consent
0      0 2000-01-01       NA            0         0    False
```

Save the empty data frame to my 'WorkingData' folder and will comment out to prevent duplication

[5]:
```
#dfTofill.to_csv('../WorkingData/CollectedData.csv', index=False)
```

[6]:
```
CollectData=pd.read_csv("../WorkingData/CollectedData.csv")
CollectData
```

[6]:
```
   index      period  org_code  attendances  breaches  consent
0      0  2000-01-01       NaN            0         0    False
```

[8]:
```
testData.head(n=11)
```

[8]:

	Index	period	org_code	type	attendances	breaches	admissions
0	2826	2016-07-01	RGT	1	10081	2288	3123
1	2865	2016-07-01	RM1	1	10303	1080	2891
2	3941	2016-04-01	RRK	1	9044	1195	2744
3	5431	2017-12-01	RJN	1	4238	1208	971
4	5940	2017-11-01	RNZ	1	3940	209	1131
5	6727	2017-08-01	RQ3	1	3801	86	754
6	7952	2017-05-01	RBL	1	8270	2120	2491
7	8427	2019-03-01	R1K	1	12753	3178	6522
8	10489	2018-10-01	RPA	1	7710	1750	2065
9	12390	2018-05-01	RA3	1	4390	348	1090
10	12463	2018-04-01	RDE	1	7942	523	2398

3 Indexing to connect with original ae_type1.csv data

[120]:
```
index_number=12463
dfTofill.iloc[0,0]=index_number
dfTofill
```

[120]:

	index	period	org_code	attendances	breaches	consent
0	12463	2018-05-01	RA3	4390	348	True

[10]:
```
import ipywidgets as widgets
from IPython.display import display
```

3.1 Consent Form using Checkbox widget

[11]:
```
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
)
```

[12]:
```
display(a)
```

```
Checkbox(value=False, description='I consent for the data I have provided to be processed and s
```

[121]:
```
#linking with dftofill dataset
dfTofill.iloc[0,5]=a.value
dfTofill
```

[121]:

	index	period	org_code	attendances	breaches	consent
0	12463	2018-05-01	RA3	4390	348	True

4 Creating widget to collect Date

[20]:
```
print(result[1])
#checking datatypes of period and it is string
```

```
object
```

[85]:
```
testData.head(n=11)
```

[85]:

	Index	period	org_code	type	attendances	breaches	admissions
0	2826	2016-07-01	RGT	1	10081	2288	3123
1	2865	2016-07-01	RM1	1	10303	1080	2891
2	3941	2016-04-01	RRK	1	9044	1195	2744
3	5431	2017-12-01	RJN	1	4238	1208	971
4	5940	2017-11-01	RNZ	1	3940	209	1131
5	6727	2017-08-01	RQ3	1	3801	86	754
6	7952	2017-05-01	RBL	1	8270	2120	2491
7	8427	2019-03-01	R1K	1	12753	3178	6522
8	10489	2018-10-01	RPA	1	7710	1750	2065
9	12390	2018-05-01	RA3	1	4390	348	1090
10	12463	2018-04-01	RDE	1	7942	523	2398

[14]:
```
b = widgets.DatePicker(
    description='Period',
    disabled=False
)
display(b)
```

```
DatePicker(value=None, description='Period')
```

[122]:
```
dfTofill.iloc[0,1]=b.value
dfTofill
```

[122]:

	index	period	org_code	attendances	breaches	consent
0	12463	2018-04-01	RA3	4390	348	True

4.1 Creating widget to collect org_code of Type 1 Hospitals

[27]:
```
print(result[2])
#checking data type and it is string
```

```
object
```

[16]:
```
import numpy as np
testData.describe(include='all')
```

[16]:

	Index	period	org_code	type	attendances	breaches	admissions
count	11.000000	11	11	11.0	11.000000	11.000000	11.000000
unique	NaN	10	11	NaN	NaN	NaN	NaN
top	NaN	2016-07-01	RGT	NaN	NaN	NaN	NaN
freq	NaN	2	1	NaN	NaN	NaN	NaN
mean	7222.818182	NaN	NaN	1.0	7497.454545	1271.363636	2380.000000
std	3478.562456	NaN	NaN	0.0	3033.109835	980.352311	1614.526494
min	2826.000000	NaN	NaN	1.0	3801.000000	86.000000	754.000000
25%	4686.000000	NaN	NaN	1.0	4314.000000	435.500000	1110.500000
50%	6727.000000	NaN	NaN	1.0	7942.000000	1195.000000	2398.000000
75%	9458.000000	NaN	NaN	1.0	9562.500000	1935.000000	2817.500000
max	12463.000000	NaN	NaN	1.0	12753.000000	3178.000000	6522.000000

[18]:
```
org_code=list(testData['org_code'].unique())
org_code
```

[18]:
```
['RGT', 'RM1', 'RRK', 'RJN', 'RNZ', 'RQ3', 'RBL', 'R1K', 'RPA', 'RA3', 'RDE']
```

[88]:
```
testData.head(n=11)
```

[88]:

	Index	period	org_code	type	attendances	breaches	admissions
0	2826	2016-07-01	RGT	1	10081	2288	3123
1	2865	2016-07-01	RM1	1	10303	1080	2891
2	3941	2016-04-01	RRK	1	9044	1195	2744
3	5431	2017-12-01	RJN	1	4238	1208	971
4	5940	2017-11-01	RNZ	1	3940	209	1131
5	6727	2017-08-01	RQ3	1	3801	86	754
6	7952	2017-05-01	RBL	1	8270	2120	2491
7	8427	2019-03-01	R1K	1	12753	3178	6522
8	10489	2018-10-01	RPA	1	7710	1750	2065
9	12390	2018-05-01	RA3	1	4390	348	1090

10	12463	2018-04-01	RDE	1	7942	523	2398

[19]:
```
c=widgets.Select(
    options=org_code,
    value='RGT',
    rows=len(org_code),
    description='ODS code:',
    disabled=False
)
display(c)
```

```
Select(description='ODS code:', options=('RGT', 'RM1', 'RRK', 'RJN', 'RNZ', 'RQ3', 'RBL', 'R1K
```

[123]:
```
dfTofill.iloc[0,2]=c.value
dfTofill
```

[123]:

	index	period	org_code	attendances	breaches	consent
0	12463	2018-04-01	RDE	4390	348	True

5 Creating widget to collect attendance variable

[34]:
```
print(result[4])
```

```
int64
```

[90]:
```
testData.head(n=11)
```

[90]:

	Index	period	org_code	type	attendances	breaches	admissions
0	2826	2016-07-01	RGT	1	10081	2288	3123
1	2865	2016-07-01	RM1	1	10303	1080	2891
2	3941	2016-04-01	RRK	1	9044	1195	2744
3	5431	2017-12-01	RJN	1	4238	1208	971
4	5940	2017-11-01	RNZ	1	3940	209	1131
5	6727	2017-08-01	RQ3	1	3801	86	754
6	7952	2017-05-01	RBL	1	8270	2120	2491
7	8427	2019-03-01	R1K	1	12753	3178	6522
8	10489	2018-10-01	RPA	1	7710	1750	2065
9	12390	2018-05-01	RA3	1	4390	348	1090
10	12463	2018-04-01	RDE	1	7942	523	2398

[22]:
```
d=widgets.IntText(
    value=0,
    description='Attendances:',
    disabled=False)
display(d)
```

```
IntText(value=0, description='Attendances:')
```

[124]:
```
dfTofill.iloc[0,3]=d.value
dfTofill
```

[124]:

	index	period	org_code	attendances	breaches	consent
0	12463	2018-04-01	RDE	7942	348	True

6 Creating widget to collect breaches variable

[38]:
```
print(result[5])
```

```
int64
```

[92]:
```
testData.head(11)
```

[92]:

	Index	period	org_code	type	attendances	breaches	admissions
0	2826	2016-07-01	RGT	1	10081	2288	3123
1	2865	2016-07-01	RM1	1	10303	1080	2891
2	3941	2016-04-01	RRK	1	9044	1195	2744
3	5431	2017-12-01	RJN	1	4238	1208	971
4	5940	2017-11-01	RNZ	1	3940	209	1131
5	6727	2017-08-01	RQ3	1	3801	86	754
6	7952	2017-05-01	RBL	1	8270	2120	2491
7	8427	2019-03-01	R1K	1	12753	3178	6522
8	10489	2018-10-01	RPA	1	7710	1750	2065
9	12390	2018-05-01	RA3	1	4390	348	1090
10	12463	2018-04-01	RDE	1	7942	523	2398

[25]:
```
e=widgets.IntText(
    value=0,
    description='Breaches:',
    disabled=False)
display(e)
```

```
IntText(value=0, description='Breaches:')
```

[125]:
```
dfTofill.iloc[0,4]=e.value
dfTofill
```

[125]:

	index	period	org_code	attendances	breaches	consent
0	12463	2018-04-01	RDE	7942	523	True

7 Concatenating the collected data to the CollectData data frame.

Let us 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.

[126]:
```
CollectData  = pd.concat([CollectData, dfTofill])
display(CollectData)
```

	index	period	org_code	attendances	breaches	consent
0	2826	2016-07-01	RGT	10081	2288	True
0	2865	2016-07-01	RM1	10303	1080	True
0	3941	2016-04-01	RRK	9044	1195	True
0	5431	2017-12-01	RJN	4238	1208	True
0	5940	2017-11-01	RNZ	3940	209	True
0	6727	2017-08-01	RQ3	3801	86	True
0	7952	2017-05-01	RBL	8270	2120	True
0	8427	2019-03-01	R1K	12753	3178	True
0	10489	2018-10-01	RPA	7710	1750	True
0	12390	2018-05-01	RA3	4390	348	True
0	12463	2018-04-01	RDE	7942	523	True

7.1 Making sure consent is given before saving the data

[127]:
```
CollectData=CollectData[CollectData['consent'] == True]
display(CollectData)
```

	index	period	org_code	attendances	breaches	consent
0	2826	2016-07-01	RGT	10081	2288	True
0	2865	2016-07-01	RM1	10303	1080	True
0	3941	2016-04-01	RRK	9044	1195	True
0	5431	2017-12-01	RJN	4238	1208	True
0	5940	2017-11-01	RNZ	3940	209	True
0	6727	2017-08-01	RQ3	3801	86	True
0	7952	2017-05-01	RBL	8270	2120	True
0	8427	2019-03-01	R1K	12753	3178	True
0	10489	2018-10-01	RPA	7710	1750	True
0	12390	2018-05-01	RA3	4390	348	True
0	12463	2018-04-01	RDE	7942	523	True

7.1.1 Saving the CollectData data frame

[130]:
```
CollectData.to_csv('../RefData/CollectedData_final.csv', index=False)
```

That is the CollectData data frame saved to the working 'Data' folder. You need to iterate through this Notebook until you have collected all of your test data and then save the captured test data to your 'RawData' folder.