Working with Data types and structures in Python and R Assessment#2 B210635

Link to GitHub repository

```r
"https://github.com/B210635/B210635_assessment"
```

```
## [1] "https://github.com/B210635/B210635_assessment"
```

Load packages

These packages will be uploaded because they are needed for the scripts.

```r
library(tidyverse) #This is a collection of R packages needed to tidy the data.
library(NHSRdatasets) #This is the package needed to load the NHSR data sets.
library(here) #This package is used to build path to the files.
library(knitr) #This package is used to integrate code into text documents in Rmarkdown.
library(scales) #This package provides the internal scaling infrastructure to ggplot2.
library(lubridate) #This package is for manipulating dates in R.
library(dataMeta) # This package will be used to construct the data dictionary
library(caret) #This package contains a set of functions that attempt to
#streamline the process for creating predictive models.
```

Loading NHS England accident and emergency attendances and admissions dataset

The NHS England accident and emergency attendances and admissions (ae_attendances) dataset has been chosen for the analysis to explore the monthly burden of admissions in emergency departments across England. The dataset contains reported attendances, four-hour breaches and admissions for all A&E departments in England for the years 2016/17 through 2018/19 (Apr-Mar).

Emergency departments are critical in life-saving processes; thus, health decision-makers need to rationally allocate resources (human, medical supplies, and logistics) to ensure efficiency and economy. Exploring the monthly trend in the burden of emergency admissions is a useful way of demonstrating evidence-based approach to resource allocation during varying periods of the year.

```r
data(ae_attendances) #Load the ae_attendances data.
```

Archive the raw dataset to the RawData folder

```r
#Create raw data object
ae <- ae_attendances # save the ae_attendances dataset as 'ae'

#Save all raw data objects as CSV files to RawData folder
write_csv(ae, here("RawData", "ae_attendances.csv"))
```

Create a provisional subset dataset for admissions burden in Emergency departments in England

Because the report focuses on the monthly burden of emergency admissions across all organizations in England,the following variables were selected from the original dataset: org_code, period, and admissions for analysis. An index column was first added to the original dataset for future use in the development of the data capture tool. The emergency department was filtered from the 'type' variable.

```
# Create subset data for admissions burden in Emergency departments in England and store it as 'ae1'**

ae1 <- rowid_to_column(ae, "index") %>%  # index original dataset
  filter(type==1)%>%   # filter original dataset to select only rows containing emergency department
   dplyr::select(index, org_code, period, admissions)   # select the pertinent variables
```

Explore the structure of the accident and emergency attendances and admissions provisional dataset

The resulting dataset contains 4,932 rows with four columns corresponding to four variables as follows: index stored as integer, org_code stored as factor, period stored as date, and admissions stored as numeric.

```
str(ae1) #use the 'str' function to explore structure of the provisional dataset
```

```
## tibble [4,932 x 4] (S3: tbl_df/tbl/data.frame)
##  $ index     : int [1:4932] 1 4 9 11 15 17 21 23 26 29 ...
##  $ org_code  : Factor w/ 274 levels "8J094","AAH",..: 112 69 169 129 192 127 172 230 138 83 ...
##  $ period    : Date[1:4932], format: "2017-03-01" "2017-03-01" ...
##  $ admissions: num [1:4932] 5060 6943 3597 2202 3360 ...
```

Save provisional subset dataset to the 'Data' folder as CSV file

```
write_csv(ae1, here("Data", "ae_attendances_admissions_emergency.csv"))
```

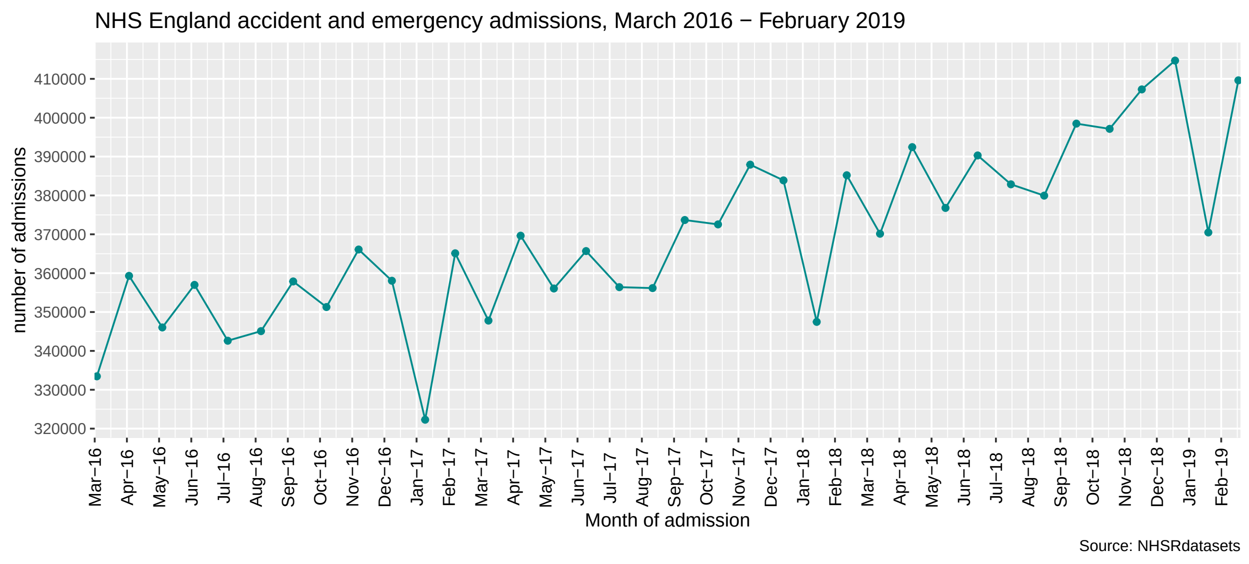
Show monthly trend of attendances that result in admissions across England at Emergency departments

The results showed overall increasing trend despite monthly fluctuations in the number of admissions at emergency departments. Noticeable dips were seen yearly during the month of January.

```
 # Graph of monthly trend of emergency admissions**

Emergency_AdmissionsTrend <- ae1%>%
  group_by(period)%>%
  summarise(admissions = sum(admissions))%>% # sum the number of admissions by period of admission
  ggplot(aes(period, admissions)) +
  geom_line(color = "darkcyan")+
   scale_x_date(date_breaks = "30 day", date_labels = "%b-%y", expand=c(0,2))+
  theme(axis.text.x = element_text(angle = 90, hjust = 0.4, vjust = 0.5, size = 10, color = "Black"))+
  scale_y_continuous(breaks = seq(0,500000, 10000))+
  geom_point(color = "darkcyan") +
  labs(x = "Month of admission",
       y = "number of admissions",
       title = "NHS England accident and emergency admissions, March 2016 - February 2019",
       caption = "Source: NHSRdatasets")

Emergency_AdmissionsTrend
```



```
ggsave("Emergency_AdmissionsTrend.png") # to save the graph to working directory
```

Separating provisional ae_attendances_admissions_emergency data into training and testing sets

This is to split the subsetted data into test and training data sets. First, we check the number of rows in the dataset and next calculate the proportion to assign to the training data

```
nrow(ae1) #rows of data
```

```
## [1] 4932
```

```
prop<-(1-(15/nrow(ae1))) #calculate proportion to assign
print(prop)
```

```
## [1] 0.9969586
```

Extracting the training data.

```
set.seed(333)

#Partitioning the raw data into the test and training data.
trainIndex <- createDataPartition(ae1$index, p = prop,
                                  list = FALSE,
                                  times = 1)
head(trainIndex)
```

```
##      Resample1
## [1,]         1
## [2,]         2
## [3,]         3
## [4,]         4
## [5,]         5
## [6,]         6
```

```
# All records that are in the trainIndex are assigned to the training data.
ae1Train <- ae1[ trainIndex,]
```

```
nrow(ae1Train)
```

```
## [1] 4920
```

There are 4,920 records in the training data. That is a large dataset!

Save the training dataset to the folder 'Data'

```
write_csv(ae1Train, here("Data", "ae_attendances_admissions_emergency_train.csv"))
```

Extract the test data

All records that are not in the trainIndex (`-trainIndex`) are assigned to the test data. There are 12 records in the training dataset.

```
ae1Test  <- ae1[-trainIndex,]
nrow(ae1Test)
```

```
## [1] 12
```

Put aside the first record from the test data for markers.

```
ae1TestMarker  <- ae1Test[1,]
```

Save the marker test data to the folder 'Data'

```
write_csv(ae1TestMarker, here("Data", "ae_attendances_admissions_emergency_marker.csv"))
```

To set aside the remaining records to test (or collect with) the data-capture tool.

```
ae1Test  <- ae1Test[2:nrow(ae1Test),]
```

To save the ae_attendances_admissions_emergency test data to the folder 'Data'

```
write_csv(ae1Test, here("Data", "ae_attendances_admissions_emergency_test.csv"))
```

#**Data dictionary**

This is to develop a data dictionary of the data collected using the data capture tool to provide detailed information about the variables and features of the collected dataset including its metadata. First, the data collected from the Python Jupyter notebook is loaded.

```
# Load the dataset that was collected in python and stored in the Raw data folder.
CollectedData=read_csv(here("RawData", "CollectedDataFinal.csv"))

# view the columns/variables of the collected dataset data frame  and their types
glimpse(CollectedData)
```

```
## Rows: 11
## Columns: 5
## $ index       <dbl> 2826, 2865, 3941, 5431, 5940, 6727, 7952, 8427, 10489, 1239~
## $ org_code    <chr> "RGT", "RM1", "RRK", "RJN", "RNZ", "RQ3", "RBL", "R1K", "RP~
## $ period      <date> 2016-07-01, 2016-07-01, 2016-04-01, 2017-12-01, 2017-11-01~
## $ admissions <dbl> 3123, 2891, 2744, 971, 1131, 754, 2491, 6522, 2065, 1090, 2~
## $ consent    <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,~
```

To build a linker dataframe by creating two string vectors

```
# Building a linker data frame by creating two string vectors representing the different variable descr

variable_description <- c("The index column that allows us to link the data collected to the original a
print(variable_description)
```

```
## [1] "The index column that allows us to link the data collected to the original ae_attendances data 
## [2] "The Organisation data service (ODS) code for the organisation."
## [3] "The month for this type of activity."
## [4] "The number of attendances that resulted in an admission to the emergency department."
## [5] "The consent from the end-user to process and share the data collected with the data capture tool
```

Variable types - to show the types of variables in the linker dataframe

The dataset consisted of two quantitative variables (index and admissions) represented by "0" and three fixed values variables (org-code, period, and consent) represented by "1".

```
# Variable types
# Use 0 to represent quantitative variable types and 1 to represent fixed values
variable_type <- c(0, 1, 1, 0,1)
print(variable_type)
```

```
## [1] 0 1 1 0 1
```

To construct a linker between the dataset collected with the data capture tool and the data dictionary

```
# Construct a link between the collected dataset and the data dictionary.
linker<-build_linker(CollectedData, variable_description, variable_type)
print(linker)
```

```
##     var_name
## 1      index
## 2   org_code
## 3     period
## 4 admissions
## 5    consent
##
## 1 The index column that allows us to link the data collected to the original ae_attendances data in 
## 2                                                          The Organisation data service (ODS) code 
## 3                                                                                   The month for t
## 4                                         The number of attendances that resulted in an admission to the 
## 5                               The consent from the end-user to process and share the data collected with t
##   var_type
## 1        0
## 2        1
## 3        1
## 4        0
## 5        1
```

To construct the dictionary and save the dictionary to the RawData folder

```
# construct dictionary using the build_dict() function from the dataMeta
CollectedData_DataDictionary <- build_dict(my.data = CollectedData, linker = linker, option_description
prompt_varopts = FALSE)

#View dictionary structure
glimpse(CollectedData_DataDictionary)
```

```
## Rows: 24
## Columns: 3
## $ variable_name        <chr> "admissions", "consent", "index", "org_code", " "~
## $ variable_description <chr> "The number of attendances that resulted in an ad~
## $ variable_options     <chr> "754 to 6522", "TRUE", "2826 to 12463", "RGT", "R~
```

```
# To save the CollectedData_DataDictionary dataset to the 'RawData' folder
write_csv(CollectedData_DataDictionary, here("RawData", "CollectedData_DataDictionary.csv"))
```

To create a main_string for describing the dataset attributes

```
## Create main_string for describing the CollectedData data frame.
main_string <- "This data describes the NHS England accident and emergency (A&E) admissions across emerg

main_string
```

```
## [1] "This data describes the NHS England accident and emergency (A&E) admissions across emergency dep
```

This is to incorporate the attributes as metadata to the collected dataset

```
## Incorporate attributes as metadata
complete_CollectedData <- incorporate_attr(my.data = CollectedData, data.dictionary = CollectedData_Data
main_string = main_string)

#Change the author name
attributes(complete_CollectedData)$author[1]<-"B210635"
# complete_CollectedData

#attributes of complete_CollectedData
attributes(complete_CollectedData)
```

```
## $row.names
##  [1]  1  2  3  4  5  6  7  8  9 10 11
##
## $names
## [1] "index"      "org_code"   "period"     "admissions" "consent"
##
## $spec
## cols(
##   index = col_double(),
##   org_code = col_character(),
##   period = col_date(format = ""),
##   admissions = col_double(),
##   consent = col_logical()
## )
```

```
##
## $problems
## <pointer: 0x56314be55370>
##
## $class
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
##
## $main
## [1] "This data describes the NHS England accident and emergency (A&E) admissions across emergency dep
##
## $dictionary
##    variable_name
## 1     admissions
## 2        consent
## 3          index
## 4       org_code
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15        period
## 16
## 17
## 18
## 19
## 20
## 21
## 22
## 23
## 24
##
## 1                                     The number of attendances that resulted in an admission to the
## 2                        The consent from the end-user to process and share the data collected with
## 3  The index column that allows us to link the data collected to the original ae_attendances data in
## 4                                                          The Organisation data service (ODS) code
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15                                                                                     The month for
## 16
## 17
## 18
## 19
## 20
## 21
## 22
## 23
## 24
##    variable_options
## 1       754 to 6522
## 2              TRUE
## 3     2826 to 12463
## 4               RGT
## 5               RM1
## 6               RRK
## 7               RJN
## 8               RNZ
## 9               RQ3
## 10              RBL
## 11              R1K
## 12              RPA
## 13              RA3
## 14              RDE
## 15            16983
## 16            16892
## 17            17501
## 18            17471
## 19            17379
## 20            17287
## 21            17956
## 22            17814
## 23            17652
## 24            17622
##
## $last_edit_date
## [1] "2022-07-05 13:59:58 UTC"
##
## $author
## [1] "B210635"
```

This is to save the complete_CollectedData with attributes to the RawData folder

```
##Save the complete_CollectedData with attributes to the RawData folder
save_it(complete_CollectedData, here("RawData", "complete_CollectedData"))
```

Data capture tool

```
# Link to the data capture tool
# https://github.com/B210635/B210635_assessment/blob/master/IPynbScripts/Working%20with%20Data%20types%
```

The data capture tool was developed in Python using interactive jupyter widgets to collect the data. Widget is an intuitive feature consisting of graphical user interface elements, such as a button, dropdown menus, or textboxes to collect or input user data. The test dataset used for data collection consisted of eleven rows and four columns or variables with data types as follows: index (integer), org_code (string), period (date), and admission (integer). Because consent is crucial to ensure data protection compliance in line with data regulations standards, a boolean variable 'consent' was added to the data capture tool using a Boolean widget (checkbox widget) with values as 'True' or 'False'. True corresponds to consent provided by the end-user to analyze and share the collect dataset. A datepicker widget was used to collect data for the period variable as date. The org_code variable was displayed as list and selection widgets were used to select the org_code value from the list. Because the admissions variable is an integer, the *'IntText widget'* was used to input the value. Eleven iterations were performed for each variable to collect the data captured by the Juypter widgets to an empty data frame.