# Development of a data capture tool using data from NHS A&E departments

B209460

20/06/2022

## Introduction

This script will load the the NHS England accident and emergency attendances and admissions (ae_attendances) dataset from the NHSRdatasets package. It will briefly view, explore and tabulate the NHS England accident and emergency attendances and admissions (ae_attendances) data set and save it to the project 'RawData' folder. It will then examine the four-hour waiting time target performance for England as a whole, selecting a subset of the variables needed. It will then partitioned the data subset into training and testing data and save them to the project working 'Data' folder, ready for downstream exploratory analysis. A data capture tool will be developed within Python, utilising a selection of widgets for user input. Details will be provided below as well as links to python scripting.

A full data dictionary will also be enclosed within this script.

## Loading ae_attendances from NHSRdatasets

### Background to data

The NHS England accident and emergency attendances and admissions (ae_attendances). Is part of the NHSRdatasets package which has been created to support skills development in the NHS-R community. 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).

### Loading required packages

```
library(NHSRdatasets)
#Outlined above

library(tidyverse)
#The tidyverse is a collection of R packages designed for data science
#introduced by Hadley Wickham and his team that "share an underlying design
#philosophy, grammar, and data structures" of tidy data.

library(here)
#The here package enables easy file referencing in project-oriented workflows.
#In contrast to using `setwd()` function, which is fragile and dependent on the
#way you organise your files, here uses the top-level directory of a project
#to easily build paths to files.
```

```r
library(knitr)
##knitr is an R package that integrates code into text documents.
#Files can then be processed into a diverse array of document formats including
#pdfs, Word documents, etc.


library(caret)
#The caret package (short for Classification And REgression Training) is a set
#of functions that attempt to streamline the process for creating predictive
#models. The package contains tools for data splitting, pre-processing,
#feature selection, model tuning using resampling, variable importance
#estimation amongst others.

library(dplyr)
#The dplyr package is part of the tidyverse but required to be loaded
#separately.  The package provides operations to allow for easier
#data manipulation and transformation.

library(scales)
#The scales packages provides the internal scaling infrastructure to ggplot2 and
#its functions allow programmers to customise
#the transformations, breaks, guides and palettes used in visualisations.
#ggplot2 is a powerful package to draw graphics. It implements the grammar of
#graphics (and hence its name). ggplot2 is included in, and loaded with the
#tidyverse package

library(lubridate)
#lubridate provides tools that make it easier to manipulate dates in R.
#lubridate is part of Hadley's tidyverse ecosystem but is not loaded by the
#tidyverse package, which includes only what he thought were the core
#components. If you are going to be doing a lot of date manipulations,
#you need to load it separately.
```

## Uploading the data

```r
data(ae_attendances)
```

## Viewing the data and assessing for missing values

We will quickly view the data to understand its structure and assess the data quality.

```r
ae<-ae_attendances
#Store ae<-ae_attendances as a tbl_df.
ae
# review ae tibble with R
ae %>%
  map(is.na) %>%
map(sum)
# calculate the total missing values for each variable using map function
```

The dataset consists of 12,765 rows of data and six columns with different classes. A date variable, period, two character variables (or factors), org_code and type, and three numeric (double precision) variables, attendances, breaches and admissions.The data is also complete and so no further action required to handle missing data.

## Subsetting the data

Separating data into training and testing sets is vital for evaluating data collection and analysis tools. most of the data will be used for training, and a smaller portion of the data will be used for testing. To develop and evaluate our data capture tool, I will split the raw data into training and testing sets. We, therefore, need to add an index column to the raw data so we can link the partitioned data sets to the raw data if required in the future. We will use the `rowid_to_column()` function to convert row identities to a column named index.

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

## Formatting and saving raw data

To improve data legibility, The data has been formatted with a 1000 seperator for each of the numeric fields (attendances, breaches and admissions) and the date field has been formatted to "MMM/YY".

```
ae %>%
  # Set the period column to show in month-year format
  mutate_at(vars(period), format, "%b-%y") %>%
  # Set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(attendances, breaches, admissions), comma) %>%
  # Format as a table
  kable()
```

Data is then saved into the RawData folder within the project working directory

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

## Selecting variables of interest, subsetting and saving data

We will now use the dplyr package `select()` function to select the required variables. The dplyr package is loaded by the tidyverse package, as one of its core components. dplyr provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges

After, data will be saved into the RawData folder

```
ae<-ae %>% select(index, period, attendances, breaches, org_code)
write_csv(ae, here("RawData", "ae_attendances_VariablesForAnalysis.csv"))
```

## Seperating into test and train datasets

I will seperate the raw dataset into test and train datasets for the development of my data capture tool. I will select 10 rows of data from my raw dataset to test the data capture tool. I will first calultae the proportion that these ten rows represent of my overall dataset and then use the createDataPartition() function from the caret package to splint our raw data into test and training data sets.The 'set.seed()' function is a random number generator, which is useful for creating random objects that can be reproduced. This will make sure that every time we run this script, we will partition the raw data into the same test and training data. After brief formatting, the two seperate datasets will then be saved as test and training.

```
prop<-(1-(15/nrow(ae)))
# generates the proportion of test rows to training rows

set.seed(333)
#Partitioning the raw data into the test and training data.
trainIndex <- createDataPartition(ae$index, p = prop,
                                  list = FALSE,
                                  times = 1)
head(trainIndex, n=15)
```

```r
# All records that are in the trainIndex are assigned to the training data.
aeTrain <- ae[ trainIndex,]

aeTrain %>%
  # set the period column to show in Month-Year format
  mutate_at(vars(period), format, "%b-%y") %>%
  # set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(attendances, breaches), comma) %>%
  # show the first 10 rows
  head(10) %>%
  # format as a table
  kable()

#save ae_attendances_perform training data to working data folder 'Data'
write_csv(aeTrain, here("Data", "ae_attendances_perfom_train.csv"))

aeTest  <- ae[-trainIndex,]
nrow(aeTest)
aeTestMarker  <- aeTest[1,]

aeTestMarker  %>%
  # set the period column to show in Month-Year format
  mutate_at(vars(period), format, "%b-%y") %>%
  # set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(attendances, breaches), comma) %>%
  # show the first 10 rows
  head(10) %>%
  # format as a table
  kable()

#save our ae_attendances__perform marker test data to our working data folder
'Data'
write_csv(aeTestMarker, here("Data", "ae_attendances_test_marker.csv"))

### We then need to set aside the remaining records to test
#our data-capture tool.
aeTest  <- aeTest[2:nrow(aeTest),]

#### Let's tabulate ae_attendances_ENG_4hr_perfom test data for your report
aeTest  %>%
  # set the period column to show in Month-Year format
  mutate_at(vars(period), format, "%b-%y") %>%
  # set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(attendances, breaches), comma) %>%
  # show the first 10 rows
  head(10) %>%
  # format as a table
  kable()


### Our final task, is to save our ae_attendances_ENG_4hr_perform test data to
#our working data folder 'Data'
write_csv(aeTest, here("Data", "ae_attendances_test.csv"))
```

# Data Capture tool

The data capture tool will be constructuted using a python script and will capture each of the variables described in the above subset from th ae_attendances dataset from NHSRDatasets

The tool can be accessed via the Github repository, link below

## Consent

The tool will collect informed consent from participants using a widgit. The default option of which, is set to decline consent.

## Access

All scripts, Rawdata, Outputs and plans associated with this project can be found within the following Github repository.

link to Github repository

# Data dictionary for test data

The data dictionary associated with the test data collected using the Data Capture tool is saved within the RawData folder and contrains descriptions of each variable.

The dictionary was produced using the R dataMeta package, the steps for this can be followed using R script "Creating a data dictionary.r", which is saved within the project RScripts folder. It can also be re-created using the below code

```
library(dataMeta)
# The dataMeta package has a collection of functions that are designed to
#construct a data dictionary and append it to the original dataset as an
#attribute, along with other information generally provided in other software
#as metadata.

CollectedData=read_csv(here("RawData", "CollectedDataFinal.csv"))
#read in test data to create dictionary

head(CollectedData)
# review data set

CollectedData$period<-as.character(CollectedData$period)
#Convert date fields to character. This steps helps with some errors in creating
#the dictionary

glimpse(CollectedData)
#confirm changes

variable_description <- c("The index column that allows us to link the data
                          collected to the original ae_attendances data in the
                          'RawData' folder.",
                          "The month that this activity relates to, stored as a
                          date (1st of each month).",
                          "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 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 consent from the end-user to process and
                          share the data collected with the data capture tool.")
print(variable_description)
#Add the descriptions for each of the variables contained within the test data

variable_type <- c(0, 1, 0, 0, 1, 1)
#define each variable type (integer or other)

print(variable_type)
linker<-build_linker(CollectedData, variable_description, variable_type)
print(linker)
#The linker is an intermediary step before the creation of the dictionary


Dictionary <- build_dict(my.data = CollectedData, linker = linker,
                         prompt_varopts= FALSE, option_description = NULL)
#assembles the dictionary from constituent parts


glimpse(dictionary)
```