B202395_Assignment1_LoadingDataset_DataDictionary

B202395

18 June, 2022

Data

Link to GitHub repository The NHS England accident and emergency (A&E) attendances and admissions (ae_attendances) dataset from the NHSRdatasets package will be used to analyse changes in the percentage of A&E attendances which are within the 4-hour target at University Hospitals of Leicester NHS Trust (org_code: RWE) broken down by department type from 2016 to 2019. This analysis will help in resource planning during seasonal periods of poor performance. The variables of interest are: period (date), attendance (double precision), 4-hour breaches (double precision), performance (double precision) and type (character). Performance will be calculated from the ratio of breaches over attendance. An index will be assigned in R to enable any subsets of the data to be linked back to the original raw data. The subset of variables required will be partitioned in R into training and test data and saved in the 'Data' folder. Data collection of the test data will be done using Python using interactive Jupyter widgets. The data dictionary is in the 'RawData' folder(code at the bottom) and the data management plan is in the 'Outputs' folder.

Loading NHSR dataset

Load packages

```
#Loading the packages required
library(NHSRdatasets)
library(tidyverse)
library(here)
library(knitr)
library(scales)
library(lubridate)
library(caret)
library(gcookbook)
```

Load data

```
#Loading the ae_attendances data
data(ae_attendances)
```

Viewing the data

Basic overview of the data

```
#Creating the object ae
ae<-ae_attendances
#Checking the class of ae
class(ae)
#Viewing ae and eyeballing the data
ae
summary(ae)
glimpse(ae)
head(ae)
tail(ae, n=5)
```

The ae_attendances tibble consists of 12,765 rows of data and six columns: period(date), org_code (character), type (character), attendances (double precision), breaches (double precision) and admissions (double precision). Details about the variables in the ae_attendances dataset can be found in the NHSR community git hub. Link to NHS R community git hub.

Checking for missing data

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

True (1) would indicate missing data. This dataset does not contain missing data.

Indexing

As the data will be split into training and test data, an index is assigned so that the partitioned data sets can be linked to the raw data if required in the future.

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

Saving the ae_attendances raw data

```
#Saving the raw data into the 'RawData' folder
write_csv(ae, here("RawData", "ae_attendances.csv"))
```

Subsetting the data

Variables of interest

```
#Filter ae_attendances by org_code RWE, select the variables of interest
aerwe<-ae %>%
  filter(org_code=="RWE") %>%
  select(index, type, period, attendances, breaches)
#Viewing aerwe
aerwe
summary(aerwe)
```

The dataset was filtered by org_code RWE. The variables required for this analysis were selected: Index (integer), Type (character), Period (date), Breaches (double precision), Attendances (double precision) It is a tibble with 85 rows and 5 columns (variables listed above) with data from 1 April 2016 to 1 March 2019.

Calculating performance

```
#Calculating the performance for RWE
RWE_performance <- aerwe %>%
  group_by(index,period,type) %>%
  summarise_at(vars(attendances, breaches), sum) %>%
  mutate(performance = 1 - (breaches / attendances))
glimpse(RWE_performance)
summary(RWE_performance)
```

The ratio of breaches against attendances was used to calculate performance.

Saving aerwe performance data

```
#Saving aerwe in the RawData folder
write_csv(aerwe, here("RawData", "aerwe_attendances.csv"))
#Saving ENG_performancerwe in the RawData folder
write_csv(RWE_performance, here("RawData", "aerwe_attendances_4hr_perform.csv"))
```

Partitioning - training and test

```
#Checking how many rows of data there are in RWE_performance
nrow(RWE_performance)
#85

#Only 10-15 rows of test data is required. The next line of code calculates
#the proportion of the data that needs to be partitioned as training.
prop<-(1-(15/nrow(RWE_performance)))

#Display the proportion calculated
print(prop)
#0.8235294

#Set.seed is used as a reproducible random number generator. The number 333
#chosen is an arbitrary integer argument. Any integer can be chosen but
#using the same number will ensure that the results are reproducible.
set.seed(333)

#Partitioning the data into test and training
trainIndexrwe <- createDataPartition(RWE_performance$index, p = prop,
                                     list = FALSE,
                                     times = 1)

# All records that are in the trainIndexrwe are assigned to the training data.
aerweTrain <- RWE_performance[ trainIndexrwe,]

#Check the number of rows in the training data
nrow(aerweTrain)
#73
```

```
#Saving the training data into the Data folder
write_csv(aerweTrain, here("Data", "aerwe_attendances_4hr_perform_train.csv"))

#Extracting the test data
aerweTest  <- RWE_performance[-trainIndexrwe,]

#Check the number of rows in the test data
nrow(aerweTest)
#12

#Setting aside the first record from the test data for markers to test and
#evaluate the data capture tool
aerweTestMarker  <- aerweTest[1,]

#Saving the test data for markers into the Data folder
write_csv(aerweTestMarker, here("Data", "aerwe_attendances_4hr_perform_test_marker.csv"))

#Setting aside the remaining test records for collection with the data capture tool
aerweTest  <- aerweTest[2:nrow(aerweTest),]

#Saving the test data for the data capture tool in the Data folder
write_csv(aerweTest, here("Data", "aerwe_attendances_4hr_perform_test.csv"))
```

Data Dictionary

A data dictionary contains information about a dataset for example, details about the variables, its data collection and metadata. It makes the dataset accessible, understandable and reproducible. The *dataMeta* R package was used to create the metadata and append it to the original dataset's attributes list along with other information generally provided as metadata.

Loading packages and reading in the data

```
#Loading the packages
#install.packages('dataMeta')
library(dataMeta)
library (tidyverse)
library(here)
```

Reading collected data

```
#Reading in the collected data from the Raw data folder
CollectedDatarwe=read_csv(here("RawData", "CollectedDatarweFinal.csv"))
```

Viewing the data

```
#Using the glimpse() function from the *tibble* package to view the data
glimpse(CollectedDatarwe)
#In the example provided in the course material, CourseDataAll.csv has the
#variable period as a character. In CollectedDatarweFinal, the variable period
#is a date. It didn't appear correctly in the dictionary created below therefore
#mutating the variable period to a character.
```

```
CollectedDatarwe_amendeddate <-CollectedDatarwe %>% mutate(period=as.character(period))
glimpse(CollectedDatarwe_amendeddate)
```

Building the data dictionary

Build a linker data frame

The linker data frame requires the creation of two string vectors representing the different variable descriptions and the different variable types.

Variable descriptions

Creating a string vector representing the different variable descriptions.

```
variable_descriptionrwe <- c("The index column that allows us to link the data collected to the original
"The month that this activity relates to, stored as a date (1st of each month).",
"The department type for this activity.",
"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 performance ([1 - breaches]/attendances) calculated for the University Hospitals of Leicester NHS
"The consent from the end-user to process and share the data collected with the data capture tool.")
print(variable_descriptionrwe)
```

Variable types

A string vector representing the different variable types needs to be created. 0 will be assigned for variables with quantitative values (measured values) variables and 1 for fixed values (allowable values or codes) variables.

```
#Viewing the variable types in the collected data
glimpse(CollectedDatarwe_amendeddate)
```

```
#Creating a vector of integers based on the variable types
variable_typerwe <- c(0, 1, 1, 0, 0, 0, 1)
print(variable_typerwe)
```

Constructing an intermediary (linker) data frame between the CollectedDatarwe_amendeddate and the data dictionary using the `builder_linker()` function from the *dataMeta* package.

```
linkerrwe<-build_linker(CollectedDatarwe_amendeddate, variable_descriptionrwe, variable_typerwe)
print(linkerrwe)
```

Data dictionary

Constructing a data dictionary for a CollectedDatarwe_amendeddate data frame with the aid of the linkerrwe data frame between using the `build_dict()` function from the *dataMeta*.

```
dictionaryrwe <- build_dict(my.data = CollectedDatarwe_amendeddate, linker = linkerrwe)
#When running this code, it appeared with a prompt to add description for the
#variables - if added, this appears in the Notes column. Had to press enter to
#leave the entries blank, but the Notes column then gets populated by the next
#chunk of code. I was not able to get to the bottom of why this prompt appeared in
#my code but not in the course material example. Having looked it up on the
#internet, I suspect it's something to do with prompt defaults on my software.
#Maybe something to do with prompt_varopts.
glimpse(dictionaryrwe)
```

```
dictionaryrwe[15,4]<-"other: Other types of A&E/minor injury activity with designated accommodation for
dictionaryrwe[16,4]<-"1: Emergency departments are a consultant-led 24-hour service with full resuscita
dictionaryrwe[17,4]<-"2: Consultant-led mono speciality accident and emergency service (e.g. ophthalmol
```

```
glimpse(dictionaryrwe)
write_csv(dictionaryrwe, here("RawData", "CollectedDatarwe_amendeddate_DataDictionary.csv"))
```

Saving the data dictionary for CollectedDatarwe_amendeddate to the 'RawData' folder

Append data dictionary to the CollectedDatarwe_amendeddate

Using the 'incorporate_attr()' function to incorporate attributes as metadata to CollectedDatarwe_amendeddate.

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

Create main_string for attributes

Incorporate attributes as metadata

Using the 'incorporate_attr()' function to return an R dataset containing metadata stored in its attributes. The attributes to be added include: a data dictionary, number of columns, number of rows, the name of the author who created the dictionary and added it, the time when it was last edited, a brief description of the original dataset.

```
complete_CollectedDatarwe_amendeddate <- incorporate_attr(my.data = CollectedDatarwe_amendeddate, data.
main_string = main_string)
attributes(complete_CollectedDatarwe_amendeddate)$author[1]<-"B202395"
complete_CollectedDatarwe_amendeddate
attributes(complete_CollectedDatarwe_amendeddate)
```

Save the CollectedDatarwe_amendeddate with attributes

Using the 'save_it()' function to save the CollectedDatarwe_amendeddate with attributes stored as metadata as an R dataset (.rds) into the 'RawData' folder.

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