Source code for data capture tool and data dictionary

B210624

18 June, 2022

My assessment repository:

https://github.com/B210624/B210624_Working_with_data_types_and_structures_in_Python_and_R.git

Overview

In this assignment, I will load the Hospital length of stay (LOS) data (LOS_model) from the NHSRdatasets package. Next I will shortly see, investigate and tabulate the NHS Hospital length of stay (LOS) data set and save it to my RawData folder. I will then check if the length of stay is related to some variables. I need variable as indicators for length of stay in hospital. If we can cotrl indcator, we can control the length of stay in hopital. I will then partitioned the data subset into training and testing data and save them to my working 'Data' folder, ready for downstream exploratory analysis.

Background

The data that you will be managing on the course are from the *NHSRdatasets* package. This package has been created to support skills development in the NHS-R community.

- **Hospital length of stay (LOS) data (LOS_model)** Artificially generated hospital data. Fictional patients at ten fictional hospitals, with LOS, age and date status data. Data were generated to learn generalised linear models (GLM) concepts, modelling either death or LOS.

Load packages and data

Let's load the packages and data needed for this script. . . .

```
library(NHSRdatasets)
library(tidyverse)
```

The *tidyverse* is a collection of R packages designed for maniplating datasets for clearning data.

```
library(here)
```

The *here* package is for easy filing in project-centered workflows.

```
library(knitr)
```

knitr is an R package that integrates code into text documents including PDF and Word documents formats.

```
library(scales)
```

The *scales* packages provides the internal scaling infrastructure to visualization

```
library(lubridate)
```

lubridate provides tools for easy manipulationg ofdates in R.

```
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.

NHSRdatasets

Here is the code to load the five NHSRdatasets from the *NHSRdatasets* R package.

```
#Load the LOS_model data.
data(LOS_model)
```

NHS Hospital length of stay (LOS) data (LOS_model)

In this assignment, I will be focusingon the NHS Hospital length of stay (LOS) data.

Let's have a look at the *NHS Hospital length of stay (LOS) data (LOS_model)

```
data(LOS_model)
los<-LOS_model
class(los)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

The tbl_df class is a subclass of data.frame. A data.frame is a table

```
los
```

```
## # A tibble: 300 x 5
##       ID Organisation   Age   LOS Death
##    <int> <ord>        <int> <int> <int>
##  1     1 Trust1          55     2     0
##  2     2 Trust2          27     1     0
##  3     3 Trust3          93    12     0
##  4     4 Trust4          45     3     1
##  5     5 Trust5          70    11     0
##  6     6 Trust6          60     7     0
##  7     7 Trust7          25     4     0
##  8     8 Trust8          48     4     0
##  9     9 Trust9          51     7     1
## 10    10 Trust10         81     1     0
## # ... with 290 more rows
```

An overview of my data is returned when a tibble is printed to the RStudio console. I can see the los tibble consists of 300 rows of data and 5 columns with different classes. I have one order variable, four variables (or factors). A factor is a variable used to categorise and store the data, having a limited number of different values. Factors are an important class for exploratory data analysis, visualisation, and statistical analysis with R.

The dataset contains:

- **ID:** All patients who stay in hospital has individual ID.

- **Organisation:** the Organisation is where the patients belong to . The Organisation is a unique code created by NHS NHS Digital.
- **Age:** Age is patient age which is numeric data type and integer by using class
- **Length of stay (LOS) :** the length of stay in hospital means how many days patients stay in hospital.
- **Death:** Death means the number of death in hospital.

Let's view at the los data

The `glimpse()` function is good to see the columns/variables in a data frame and show data type.

```
glimpse(los)
```

```
## Rows: 300
## Columns: 5
## $ ID           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17~
## $ Organisation <ord> Trust1, Trust2, Trust3, Trust4, Trust5, Trust6, Trust7, T~
## $ Age          <int> 55, 27, 93, 45, 70, 60, 25, 48, 51, 81, 58, 16, 21, 82, 1~
## $ LOS          <int> 2, 1, 12, 3, 11, 7, 4, 4, 7, 1, 4, 3, 1, 9, 12, 1, 4, 3, ~
## $ Death        <int> 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, ~
```

Here is the output of the 'glimpse()' function.

The `head()` function is to see the data at top n rows of a data frame.

```
head(los)
```

```
## # A tibble: 6 x 5
##      ID Organisation   Age   LOS Death
##   <int> <ord>        <int> <int> <int>
## 1     1 Trust1          55     2     0
## 2     2 Trust2          27     1     0
## 3     3 Trust3          93    12     0
## 4     4 Trust4          45     3     1
## 5     5 Trust5          70    11     0
## 6     6 Trust6          60     7     0
```

I can specify the number of rows we want to see in a data frame with the argument "n".

```
head(los, n=3)
```

```
## # A tibble: 3 x 5
##      ID Organisation   Age   LOS Death
##   <int> <ord>        <int> <int> <int>
## 1     1 Trust1          55     2     0
## 2     2 Trust2          27     1     0
## 3     3 Trust3          93    12     0
```

In the example above, we used n=3 to look at the first three rows of the ae_attendances data frame.

Missing data

It is common when collecting data for some entries to be missing, which can significantly impact. Therefore I need to check for missing data in the los data.

'map()' is a function for applying a function to each element of a list. The 'is.na' function produces a matrix, consisting of logical values (i.e. TRUE or FALSE), whereby TRUE shows a missing value.

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

```
## $ID
## [1] 0
##
## $Organisation
## [1] 0
##
## $Age
## [1] 0
##
## $LOS
## [1] 0
##
## $Death
## [1] 0
```

The data is complete. I do not need to worry about manipulating missing data.

I do not need to add the index link colum since ID already exists

Let's save the raw ae_attendances data to your 'RawData' folder

I will use the *here* package to build a directory to write the raw ae_attendances data to your 'RawData' folder to save the collected data.

```
write_csv(los, here("RawData", "los.csv"))
```

Selecting variables for your data capture tool

It extremenly important to note the full los dataset develop my data collection tool. For example, I think that age can be an indicator for length of stay in hospital. Therefore, I selected the age variable of integer. If we reduce the number of older people staying in hospital, we can control the length of stay in hospital.

Examining the four-hour waiting time target performance for England

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.

```
los<-los %>% select(ID, Organisation, LOS, Age)
```

Let's save provisional subsetted ae_attendances data to the 'RawData' folder

It is important to save the collected data. I name this file "los_age_data", and write it to the raw data folder.

```
glimpse(los)
```

```
## Rows: 300
## Columns: 4
## $ ID           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17~
## $ Organisation <ord> Trust1, Trust2, Trust3, Trust4, Trust5, Trust6, Trust7, T~
```

```
## $ LOS          <int> 2, 1, 12, 3, 11, 7, 4, 4, 7, 1, 4, 3, 1, 9, 12, 1, 4, 3, ~
## $ Age          <int> 55, 27, 93, 45, 70, 60, 25, 48, 51, 81, 58, 16, 21, 82, 1~
```

```
write_csv(los, here("RawData", "los_age_data.csv"))
```

Separating provisional los_age_data into training and testing sets

For developing and evaluating my data capture tool, I need to splint the raw data into test and training data sets.

How many rows are in the los_age_data dataset?

```
#The los_age_data is large with
los_age_data<-los
nrow(los_age_data) #rows of data
```

```
## [1] 300
```

Here shows the proportion (`prop`) of the raw data to assign to the training data:

```
prop<-(1-(15/nrow(los_age_data)))
#The proportion of the raw that needs to be assigned to the training data to ensure there is only 10 to
print(prop)
```

```
## [1] 0.95
```

I 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.

```
#the raw data into the same test and training data.
set.seed(333)
#Partitioning the raw data into the test and training data.
trainIndex_los<- createDataPartition(los_age_data$ID, p = prop,
                                     list = FALSE,
                                     times = 1)

# All records that are in the trainIndex are assigned to the training data.
los_age_Train <- los_age_data[trainIndex_los,]
nrow(los_age_Train)
```

```
## [1] 288
```

There are 288 records in your training data.

Our next task, it to save ae_attendances_ENG_4hr_perfom training data to your working data folder 'Data'

```
write_csv(los_age_Train, here("Data", "los_age_train.csv"))
```

Let's extract the ae_attendances_ENG_4hr_perfom test data

#All records that are not in the trainIndex (`-trainIndex`) are assigned to the test data.

```
los_age_Test <- los_age_data[-trainIndex_los,]
nrow(los_age_Test)
```

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

There are 12 records in my test data. I now need to set aside the first record from the los_age_test so that my markers can test and evaluate your data-capture tool.

```
los_age_TestMarker  <- los_age_Test[1,]
```

Our next task, it to save our ae_attendances_ENG_4hr_perfom marker test data to our working data folder 'Data'

```
write_csv(los_age_TestMarker, here("Data", "los_age_TestMarker.csv"))
```

We then need to set aside the remaining records for you to test (or collect with your) your data-capture tool.

```
losTest  <- los_age_Test[2:nrow(los_age_Test),]
```

Our final task, is to save our ae_attendances_ENG_4hr_perfom test data to our working data folder 'Data'

```
write_csv(losTest, here("Data", "los_test.csv"))
```

I ended my coding here.