Source code for data capture tool and data dictionary

B210624

21 June, 2022

My assessment repository:# https://github.com/B210624/B210624_Working_with_data_types_and_structures_in_Python_and_R.git ## Load packages and data Let's load the packages and data needed for this script. . . .

```
library(tidyverse) #The *tidyverse* is a collection of R packages designed for  maniplating dataset..
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.
```

Overview

In this assignment, I will use and load the Hospital length of stay (LOS) data (LOS_model) from the NHSRdatasets package. 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 search variable for later research as indicators for length of stay in hospital. Background: The data are csv.files from the NHSRdatasets package forskills development. 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.

NHSRdatasets(Creation)

```
library(NHSRdatasets)
```

Store the NHS Hospital length of stay (LOS) data set (Storage)

```
data(LOS_model) #Load the LOS_model data.
write_csv(LOS_model, here("RawData", "los.csv"))
#Here is the code to store theNHS Hospital length of stay (LOS) data set.
```

Load the NHS Hospital length of stay (LOS) data set. Here is start of Synthesis

```
LOS_CollectedData=read_csv(here("RawData", "los.csv"))
# I load the NHS Hospital length of stay (LOS) data (LOS_model) from RawData folder.
glimpse(LOS_CollectedData) #The `glimpse()` function is good to see the columns/variables in a data fra
```

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

Overview of my dataset of NHS Hospital length of stay (LOS) data (LOS_model)

I can see the los tibble consists of 300 rows of data and 5 columns with different classes. I have one order variable and four integer variables (or factors). The dataset contains:
* **ID:** All patients who stay in hospital has individual ID.
* **Organisation:** the Organisation is the fictional hospita; where the patients stay. * **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.

Missing data

```
#Calculate how many NAs there are in each variable.
LOS_CollectedData %>% map(is.na) %>%
map(sum) # 'map()' is a function for applying a function to each element of a list.
```

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

```
#The 'is.na' function produces a matrix, consisting of logical values.
```

The data is complete. I do not need to worry about manipulating missing data. ### Build a data dictionary for the data collected by the data capture tool. ##Build a linker data frame Firstly, I build a linker data frame. To do this, we need to create two string vectors representing the different variable descriptions and types. #Variable descriptions

```
LOS_variable_description <- c("The ID column means that patient IDs which can be linked to the data col
"The organsation indicating that fictional hospital codes relates to the place where the patients stay.
"The LOS showing length of staying in hosptail. It means how long days the patients stay in hospital.",
"The death means that number of patient death as result of patients staying in hospital.",
"The Age meaning the patients age. It shows that age when the patients start staying in hosptail.")
print(LOS_variable_description)
```

```
## [1] "The ID column means that patient IDs which can be linked to the data collected to the original 
## [2] "The organsation indicating that fictional hospital codes relates to the place where the patient
## [3] "The LOS showing length of staying in hosptail. It means how long days the patients stay in hosp
## [4] "The death means that number of patient death as result of patients staying in hospital."
## [5] "The Age meaning the patients age. It shows that age when the patients start staying in hosptail
```

###Variable types I have three quantitative values (measured values) variables and one fixed values variables.

```
glimpse(LOS_CollectedData) # I also used to see the data from LOS_CollectedData data frame.
```

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

I have three quantitative values (measured values) variables and one fixed values variables.

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

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

```
#I create a string vector representing the different variable types.
#It is a vector of integers with values 0 or 1. I need to use 0
#for a variable with quantitative values (measured values) variables and 1 for fixed values variables.
```

Now I use the build_linker() function from the dataMeta package to constructs an intermediary (linker) data frame between the CollectedData and the data dictionary. It requires the LOS_CollectedData data frame and LOS_variable_description and LOS_variable_type string vectors.

```
library(dataMeta) #Load the dataMeta to use build linker functon for the connection #for making diction
linker<-build_linker(LOS_CollectedData,LOS_variable_description,LOS_variable_type)
print(linker)
```

```
##       var_name
## 1           ID
## 2 Organisation
## 3          Age
## 4          LOS
## 5        Death
##
## 1 The ID column means that patient IDs which can be linked to the data collected to the original LOS
## 2                                  The organsation indicating that fictional hospital codes relates to
## 3                                    The LOS showing length of staying in hosptail. It means how long
## 4                                                   The death means that number of patient death as resu
## 5                                         The Age meaning the patients age. It shows that age when the
##   var_type
## 1        0
## 2        1
## 3        0
## 4        0
## 5        0
```

###Making data dictionary

```
LOS_dictionary <- build_dict(my.data = LOS_CollectedData, linker = linker)
```

```
## Enter description for variable 'Age' and option '5 to 95':
## Enter description for variable 'Death' and option '0 to 1':
## Enter description for variable 'ID' and option '1 to 300':
## Enter description for variable 'LOS' and option '1 to 18':
## Enter description for variable 'Organisation' and option 'Trust1':
## Enter description for variable 'Organisation' and option 'Trust2':
```

```
## Enter description for variable 'Organisation' and option 'Trust3':
## Enter description for variable 'Organisation' and option 'Trust4':
## Enter description for variable 'Organisation' and option 'Trust5':
## Enter description for variable 'Organisation' and option 'Trust6':
## Enter description for variable 'Organisation' and option 'Trust7':
## Enter description for variable 'Organisation' and option 'Trust8':
## Enter description for variable 'Organisation' and option 'Trust9':
## Enter description for variable 'Organisation' and option 'Trust10':
```

```
#I use the build_dict() function from the dataMeta to constructs a data dictionary
#for a LOS_CollectedData data frame with the aid of the linker data frame between.
glimpse(LOS_dictionary) # Check the dictionary to see how looks like.
```

```
## Rows: 14
## Columns: 4
## $ `variable name`        <chr> "Age", "Death", "ID", "LOS", "Organisation", " ~
## $ `variable description` <chr> "The LOS showing length of staying in hosptail.~
## $ `variable options`     <chr> "5 to 95", "0 to 1", "1 to 300", "1 to 18", "Tr~
## $ notes                  <chr> "", "", "", "", "", "", "", "", "", "", "", "",~
```

```
# For my next task, it to save LOS_dictionary to my working data folder 'Data'
write_csv(LOS_dictionary, here("RawData", "LOS_CollectedData_DataDictionary.csv"))
```

Append data dictionary to the CollectedData

As metadata, I now incorporate attributes to the LOS_CollectedData using the 'incorporate_attr()' function from the dataMeta package. The function requires the LOS_CollectedData and dictionary and main_string main_string as inputs.

```
main_string <- "This data describes the NHS Hospital length of stay (LOS)
data set (LOS_model) from the *NHSRdatasets* package collected by the data capture tool."
main_string #Create main_string for attributes
```

```
## [1] "This data describes the NHS Hospital length of stay (LOS)\ndata set (LOS_model) from the *NHSRda
```

Incorporate attributes as metada

```
#I use the 'incorporate_attr()' function to return an R dataset containing metadata stored in its attri
LOS_complete_CollectedData <- incorporate_attr(my.data = LOS_CollectedData, data.dictionary = LOS_dicti
#Change the author name
attributes(LOS_complete_CollectedData)$author[1]<-"B210624"
LOS_complete_CollectedData
```

```
## # A tibble: 300 x 5
##       ID Organisation   Age   LOS Death
##  * <dbl> <chr>        <dbl> <dbl> <dbl>
##  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
```

```
attributes(LOS_complete_CollectedData)
```

```
## $row.names
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
## [109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
## [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
## [145] 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
## [163] 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
## [181] 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
## [199] 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
## [217] 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
## [235] 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
## [253] 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
## [271] 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
## [289] 289 290 291 292 293 294 295 296 297 298 299 300
##
## $names
## [1] "ID"           "Organisation" "Age"          "LOS"          "Death"
##
## $spec
## cols(
##   ID = col_double(),
##   Organisation = col_character(),
##   Age = col_double(),
##   LOS = col_double(),
##   Death = col_double()
## )
##
## $problems
## <pointer: 0x562af5fff390>
##
## $class
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
##
## $main
## [1] "This data describes the NHS Hospital length of stay (LOS)\ndata set (LOS_model) from the *NHSRda
##
## $dictionary
##    variable name
## 1            Age
## 2          Death
## 3             ID
## 4            LOS
## 5   Organisation
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
##
## 1                                    The LOS showing length of staying in hosptail. It means how long
## 2                                        The Age meaning the patients age. It shows that age when the
## 3  The ID column means that patient IDs which can be linked to the data collected to the original LOS
## 4                                                 The death means that number of patient death as res
## 5                                  The organsation indicating that fictional hospital codes relates t
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
##    variable options notes
## 1           5 to 95
## 2            0 to 1
## 3          1 to 300
## 4           1 to 18
## 5            Trust1
## 6            Trust2
## 7            Trust3
## 8            Trust4
## 9            Trust5
## 10           Trust6
## 11           Trust7
## 12           Trust8
## 13           Trust9
## 14          Trust10
##
## $last_edit_date
## [1] "2022-06-21 08:53:14 UTC"
##
## $author
## [1] "B210624"
```

```
save_it(LOS_complete_CollectedData, here("RawData", "LOS_complete_CollectedData"))
#I use the 'save_it()' function to save the LOS_CollectedData with attributes stored
# as metadata as an R dataset (.rds) into the 'current working directory'RawData' folder.
LOS_complete_CollectedData<-readRDS(here("RawData","LOS_complete_CollectedData.rds"))
# Here is the end of the process of Synthesis.
```

CollectingDataInteractiveJupyterWidgets_Assignment(B210624)

June 21, 2022

0.1 CollectingDataInteractiveJupyterWidgets_Assignment(B210624)

Author details: *Author:* B210623. *Contact details:*snnnnnn@ed.ac.uk. **My assessment repository**: https://github.com/B210624/B210624_Working_with_data_types_and_structures_in_Python_and_R.

Notebook and data information: Assignment shows data manipulation with interactive jupyter widgets with the NHS Hospital length of stay (LOS) data (LOS_model) and save it to my working 'Data' folder, and finally saving all the captured test data to my 'RawData'. It consists of string data, numerical data and character data. The data I selected is csv format from the NHSRdatasets package for skills development. This is a artificially generated hospital data with fictional patients at ten fictional hospitals, LOS, Age and Death numbers.

```
[ ]: #Load the 'pandas' package, which is an useful for data manipulation.
     import pandas as pd
     testData=pd.read_csv("../Data/los.csv")
     testData #This section is the process of Creation and Storage.
```

```
[ ]: #Data type: I check the data type in the testData data frame. The dtypes␣
      ↪function is imported from the pandas package to know the data types in the␣
      ↪testData. The dtypes function returns the data types in the data frame.
     result = testData.dtypes
     print("Output:")
     print(result)# This section is the start of the process of Synthesis.
```

```
[ ]: #Set up an empty data frame in the working data folder to gather the data␣
      ↪captured by the Juypter widgets.
     #ID:Integer, Organisation: String, Age: Integer, LOS: integer, Consent: Boolean
     dfTofill = pd.DataFrame({'ID': [0],'Organisation': ['NA'],'Age': [0], 'LOS':␣
      ↪[0], "Consent":[False]})
     dfTofill
```

```
[ ]: dfTofill.to_csv('../Data/LOS_Data_frame_CollectedData.csv',index=False)#Saved␣
      ↪the empty data frame to my Data folder
```

```
[ ]: CollectData=pd.read_csv("../Data/LOS_Data_frame_CollectedData.csv")
     CollectData #The empty data frame is now saved to the working 'Data' folder. I␣
      ↪can read in the empty data frame.
```

Indexing in Python: Indexing in Python is important to choose the individual data by positioning.

```
[ ]: index_number=301 #Remember to change for each record.
     dfTofill.iloc[0,0]=index_number
     dfTofill
```

```
[ ]: #Widgets: I use to create graphical user-frentdly interface like a button,␣
      ↪dropdown or textbox.
     import ipywidgets as widgets #Load the 'ipywidgets' package
     from IPython.display import display #The IPython.display package is loaded to␣
      ↪display different objects in Jupyter.
```

Consent: Consent is a critcal element for data protection compliance. Consent expain the freedom to perticiapate the clinical research for smooth process to get their data. The [General Data Protection Regulation]. Before the reseachers obtain any data, they need to get consent from the patient to process and share the data the researchers will collect with the data capture tool.

```
[ ]: # Boolean widgets: Boolean widgets are designed to display a boolean value.␣
      ↪This is Checkbox widget
     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
     )
     display(a)#Please check the box to claim that I have consent.
```

```
[ ]: dfTofill.iloc[0,4]=a.value # I added the value in the empty data frame.
     dfTofill
```

```
[ ]: print(result[1]) #String data type
     testData.head(n=1)
```

The organisation variable: the organisation variable includes the fictional organisation code. Synthesis start from here.

```
[ ]: print(result[2]) #String data type
```

```
[ ]: import numpy as np #Load the 'numpy' package
     testData.describe(include='all') #I use the `describe()` function from the␣
      ↪*numpy* Python package to calculate summary statistics for the testData data␣
      ↪frame. The numpy package is useful for statistical analysis.
```

```
[ ]: org=list(testData['Organisation'].unique()) # I used the pandas package␣
      ↪`unique()` function to get the unique
```

```
org                                           # Organisation codes in the test␣
 ↪data.
```

```
[ ]: testData.head(n=1)# Look at the head of data
```

Selection widgets:Several widgets can be used to display single selection lists. I specify the selectable options by passing a list.

```
[ ]: c=widgets.Select(# I made the values to specific vatiables in the empty frame␣
      ↪or existing frames.
         options=org,
         rows=len(org),
         description='Organisation:',
         disabled=False)
     display(c)
```

```
[ ]: dfTofill.iloc[0,1]=c.value # I added the value in the empty data frame.
     dfTofill
```

```
[ ]: print(result[1]) # The type variable is String data type
```

```
[ ]: type=list(testData['Organisation'].unique())
     type #I first use the *pandas* package `unique()` function to get the unique␣
      ↪department type in the test data.
     testData.head(n=10)
```

The numeric variable includes the number of the length of stay in hospital and and death numbers.

```
[ ]: print(result[2])# The type variable is numeric data type
     testData.head(n=1)
```

Numeric widgets: Widgets exist for displaying integers and floats, both bounded and unbounded. The integer widgets share a similar naming scheme to their floating point counterparts. By replacing Float with Int in the widget name, you can find the Integer equivalent.

```
[ ]: e=widgets.IntText(# I made the values to input specific vatiables in the empty␣
      ↪frame or existing frames.
         value=0,
         description='Age:',
         disabled=False)
     display(e)
```

```
[ ]: dfTofill.iloc[0,2]=e.value # I added the value in the empty data frame.
     dfTofill
```

The numeric variable of length of stay in hosptal

```
[ ]: print(result[3])# The type variable is numeric data type
     testData.head(1)
```

```
[ ]: f=widgets.IntText(# I made the values to enter specific vatiables in the empty␣
      ↪frame or existing frames.
         value=0,
         description='LOS:',
         disabled=False)
     display(f)
```

```
[ ]: dfTofill.iloc[0,3]=f.value # I added the value in the empty data frame.
     dfTofill
```

Concatenating the collected data to the CollectData data frame.

```
[ ]: # I use the `concat()` function from the pandas package to append CollectData,␣
      ↪dfTofill data frames and testData.
     CollectData  = pd.concat([CollectData,testData,dfTofill,])# CollectData is the␣
      ↪first data frame
     display(CollectData) # The testData is the second frame and dfTofill is the␣
      ↪last data frame
```

```
[ ]: # I input the extra data of the consent for the test data using iloc ()␣
      ↪function for data manipuation
     CollectData.iloc[1:,4]=a.value
     display(CollectData)
```

```
[ ]: #I consent to process and share the data before I save it to the working data␣
      ↪folder.The following line of code, will ensure that you have consent to save␣
      ↪data.
     CollectData=CollectData[CollectData['Consent'] == True]
     display(CollectData)
```

```
[ ]: # This end of syhtheis. Saving the data collected by your data-capture tool to␣
      ↪the working data folder:
     CollectData.to_csv('../Data/LOS_python_CollectedData.csv', index=False)
     # I saved my collected data frame saved to the working ‘Data’ folder to repeat␣
      ↪data amanipulation via the Notebook until the end of data collection and␣
      ↪interactions and then save the captured test data to my ‘RawData’ folder.
     CollectData.to_csv('../RawData/LOS_python_CollectedDataFinal.csv',␣
      ↪index=False)#Thank you very much for reading my assingment and marking.
```