Descriptive Report

B210624

04 July, 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.
library(scales)#*lubridate* provides tools that make it easier to manipulate dates in R.
library(caret)# caret is a set of functions that attempt to streamline the process for creating predict
```

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, ~
```

```
class(LOS_CollectedData)#Look at the data class
```

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

```
head(LOS_CollectedData)#Look at data on the top
```

```
## # A tibble: 6 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
```

```
tail(LOS_CollectedData)#Look at data on the bottom
```

```
## # A tibble: 6 x 5
##      ID Organisation   Age   LOS Death
##   <dbl> <chr>        <dbl> <dbl> <dbl>
## 1   295 Trust5          32     4     0
## 2   296 Trust6          32     6     0
## 3   297 Trust7          55     6     1
## 4   298 Trust8          21     3     0
## 5   299 Trust9          54     1     0
## 6   300 Trust10         93    15     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.

Let's tablulate the raw data for your report

```
LOS_CollectedData %>%
  # Set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(LOS, Age, Death), comma) %>%
  # Show the first 10 rows
  head(10) %>%
  # Format as a table
  kable()
```

ID	Organisation	Age	LOS	Death
1	Trust1	55.0	2.0	0.0
2	Trust2	27.0	1.0	0.0
3	Trust3	93.0	12.0	0.0
4	Trust4	45.0	3.0	1.0
5	Trust5	70.0	11.0	0.0
6	Trust6	60.0	7.0	0.0
7	Trust7	25.0	4.0	0.0
8	Trust8	48.0	4.0	0.0
9	Trust9	51.0	7.0	1.0
10	Trust10	81.0	1.0	0.0

Let's save the raw LOS_CollectedData to your 'RawData' folder

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

Selecting variables for your data capture tool

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

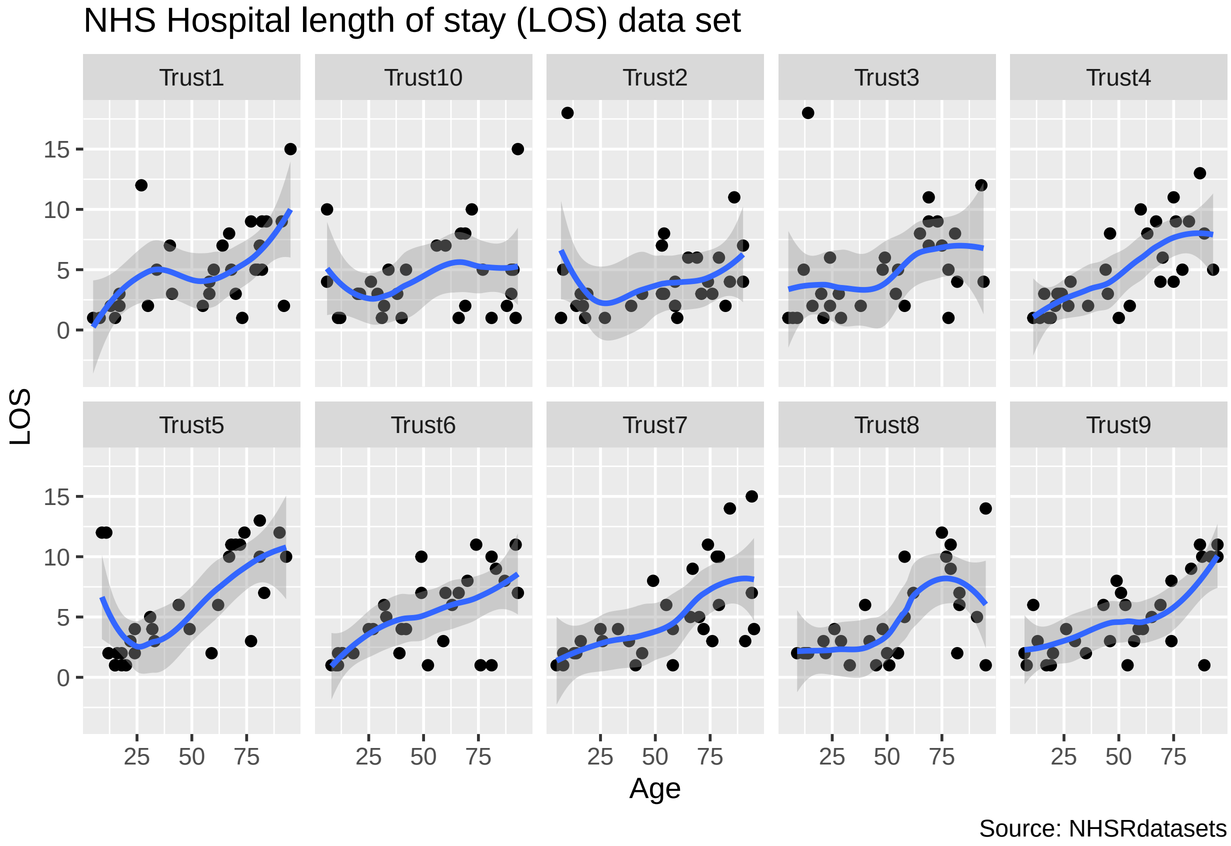
Let's tablulate the raw data for your report

```
LOS_CollectedData_parameters%>%
  # Set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(LOS, Age), comma) %>%
  # Show the first 10 rows
  head(10) %>%
  # Format as a table
  kable()
```

ID	Organisation	Age	LOS
1	Trust1	55.0	2.0
2	Trust2	27.0	1.0
3	Trust3	93.0	12.0
4	Trust4	45.0	3.0
5	Trust5	70.0	11.0
6	Trust6	60.0	7.0
7	Trust7	25.0	4.0
8	Trust8	48.0	4.0
9	Trust9	51.0	7.0
10	Trust10	81.0	1.0

Let's visualise the relationship between LOS and Age before we save or raw data file.

```
ggplot(LOS_CollectedData_parameters, aes(x=Age, y=LOS))  +
  facet_wrap(~Organisation, nrow=2)+
  geom_point() + geom_smooth() +
  labs(x = "Age",
       y = "LOS",
       title = "NHS Hospital length of stay (LOS) data set",
       caption = "Source: NHSRdatasets")
```



Let's tablulate the raw data for your report

```
LOS_CollectedData_parameters %>%
  # Set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(LOS, Age), comma) %>%
  # Show the first 10 rows
  head(10) %>%
  # Format as a table
  kable()
```

ID	Organisation	Age	LOS
1	Trust1	55.0	2.0
2	Trust2	27.0	1.0
3	Trust3	93.0	12.0
4	Trust4	45.0	3.0
5	Trust5	70.0	11.0
6	Trust6	60.0	7.0
7	Trust7	25.0	4.0
8	Trust8	48.0	4.0
9	Trust9	51.0	7.0
10	Trust10	81.0	1.0

Let's save provisional subsetted LOS_CollectedData to the 'RawData' folder

```
glimpse(LOS_CollectedData_parameters)
```

```
## Rows: 300
## Columns: 4
## $ 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, ~
```

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

Separating provisional ae_attendances_ENG_4hr_perfom data into training and testing sets

How many rows are in the LOS_CollectedData?

```
#The ae_attendances_ENG_4hr_perfom dataset is large with
nrow(LOS_CollectedData_parameters) #rows of data
```

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

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

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

##I use the createDataPartition() function from the *caret* package to splint our raw data into test and training data sets.

```
set.seed(333)
#Partitioning the raw data into the test and training data.
LOS_trainIndex <- createDataPartition(LOS_CollectedData_parameters$ID, p = LOS_prop,
                                  list = FALSE,
                                  times = 1)
head(LOS_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.
LOS_Train <- LOS_CollectedData_parameters[ LOS_trainIndex,]
nrow(LOS_Train)
```

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

There are 288 records in my training data.

Let's tabulate LOS_CollectedData_parameters training data for your report

```
LOS_Train %>%
  # set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(LOS, Age), comma) %>%
  # show the first 10 rows
  head(10) %>%
  # format as a table
  kable()
```

ID	Organisation	Age	LOS
1	Trust1	55.0	2.0
2	Trust2	27.0	1.0
3	Trust3	93.0	12.0
4	Trust4	45.0	3.0
5	Trust5	70.0	11.0
6	Trust6	60.0	7.0
7	Trust7	25.0	4.0
8	Trust8	48.0	4.0
9	Trust9	51.0	7.0
10	Trust10	81.0	1.0

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

```
write_csv(LOS_Train, here("Data", "LOS_CollectedData_parameters_train.csv"))
```

Let's extract the LOS_CollectedData_parameters test data

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

```
LOS_Test  <- LOS_CollectedData_parameters[-LOS_trainIndex,]
nrow(LOS_Test)
```

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

There are 12 records in my test data.

```
LOS_TestMarker  <- LOS_Test[1,]
```

```
LOS_TestMarker  %>%
  # set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(LOS, Age), comma) %>%
  # show the first 10 rows
  head(10) %>%
  # format as a table
  kable()
```

Let's tabulate LOS_CollectedData_parameters marker test data for your report

ID	Organisation	Age	LOS
43	Trust3	24	6

Our next task, it to save my LOS_CollectedData_parametersmarker test data to our working data folder 'Data'

```
write_csv(LOS_TestMarker, here("Data", "LOS_CollectedData_parameters_test_marker.csv"))
```

We then need to set aside the remaining records for me to test (or collect with my) mydata-capture tool.

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

```
LOS_Test  %>%
  # set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(LOS, Age), comma) %>%
  # show the first 10 rows
  head(10) %>%
  # format as a table
  kable()
```

Let's tabulate LOS_CollectedData_parameters test data for your report

ID	Organisation	Age	LOS
56	Trust6	11.0	1.0
72	Trust2	79.0	6.0
76	Trust6	66.0	7.0
85	Trust5	90.0	12.0
147	Trust7	38.0	3.0
162	Trust2	17.0	2.0
202	Trust2	69.0	6.0

ID	Organisation	Age	LOS
204	Trust4	27.0	2.0
252	Trust2	90.0	4.0
275	Trust5	44.0	6.0

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

```
write_csv(LOS_Test, here("Data", "LOS_test.csv"))
```