B210533_assessment_R_markdown_file

B210533

20 June, 2022

Working with data types and structures in Python and R

Introduction

This is a project created as for the Working with Data Types and Structures in Python and R (2021-2022)[FLEX] course at University of Edinburgh. Prepared and submitted in June 2022. The data, code and analysis for this project can be accessed at: https://github.com/Exam-B210533/B210533_assessment

Loading NHSRdatasets

The data used in this project is from the NHSRdatasets package. This package has been created to support skills development in the NHS-R community and contains several free datasets. The ons_mortality dataset has been used and contains a provisional record of deaths registered weekly in England and Wales from January 2010 to end of March 2020.

All of the variables in the dataset are needed including categorisations (category 1 and category 2), counts, date and week number, and the data was subsetted into test and training data. The Jupyter Notebook "./ipynbScripts/data_collection_tool.ipynb" was used to to collect the data.

Load packages and data

Load the packages and data needed for this script.

```
library(NHSRdatasets)
library(tidyverse)
library(here)
library(knitr)
library(lubridate)
library(caret)
```

ONS Mortality Data

This project uses the ons_mortality data from the NHSRdatasets package.

```
data(ons_mortality)
mort <- ons_mortality
class(mort)
```

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

```
glimpse(mort)
```

```
## Rows: 18,803
## Columns: 5
## $ category_1 <chr> "Total deaths", "Total deaths", "Total deaths", "Total deat~
```

```
## $ category_2 <chr> "all ages", "all ages", "all ages", "all ages", "all ages",~
## $ counts     <dbl> 12968, 12541, 11762, 11056, 10524, 10117, 10102, 10295, 998~
## $ date       <date> 2010-01-08, 2010-01-15, 2010-01-22, 2010-01-29, 2010-02-05~
## $ week_no    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
```

The data is a table data frame with 18,803 rows of data and 5 columns. The Category 1 and Category 2 columns have multiple different character values and these should be examined to see the unique values in the set.

The data contains:

- **category_1**: character, containing the names of the groups for counts, e.g. "Total deaths", "all ages".
- **category_2**: character, subcategory of names of groups where necessary, e.g. details of region: "East", details of age bands "15-44".
- **counts**: integer, actual number of deaths in whole numbers.
- **date**: date, format is yyyy-mm-dd; all dates are a Friday.
- **week_no**: integer, each week in a year is numbered sequentially.

Date Range

The dataset contains values for dates between **08-01-2010** and **03-04-2020**:

```
min(mort$date, na.rm = TRUE)
```

```
## [1] "2010-01-08"
```

```
max(mort$date, na.rm = TRUE)
```

```
## [1] "2020-04-03"
```

Unique Categories

Two columns have character data with category data. To find the unique values for these columns the unique() function is used:

Category 1 unique values

```
unique_cat1 = unique(mort[c("category_1")])
kable(unique_cat1)
```

category_1
Total deaths
All respiratory diseases (ICD-10 J00-J99) ICD-10
Persons
Males
Females
Region
NA
average of same week over 5 years
Deaths where COVID-19 was mentioned on the death certificate (ICD-10 U07.1 and U07.2)

Category 2 unique values

```
unique_cat2 = unique(mort[c("category_2")])
unique_cat2_list <- data.frame(unique_cat2, freq=1:42)
kable(
  list(unique_cat2[1:10,], unique_cat2[11:20,],unique_cat2[21:31,],unique_cat2[32:42,]),
  booktabs = TRUE)
```

category_2	category_2	category_2	category_2
all ages	North East	v 2010	40-44
average of same week over 5 years	North West	v 2013 (IRIS)	45-49
v 2001	Yorkshire and The Humber	NA	50-54
Under 1 year	East Midlands	1-4	55-59
01-14	West Midlands	5-9	60-64
15-44	East	10-14	65-69
45-64	London	15-19	70-74
65-74	South East	20-24	75-79
75-84	South West	25-29	80-84
85+	Wales	30-34	85-89
		35-39	90+

Add an index to the data set

It's necessary to separate the data into training and testing data sets. The test data will be used to evaluate the data collection and analysis tools. A data index column is added to the raw data so that partitioned data can be linked back to the raw data if needed in the future.

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

Tabulate the raw data

The raw data is then tabulated to make the data more readable.

```
kable(head(mort, n = 5),
      digits = 3,
      format.args = list(big.mark = ","),
      caption= "ONS England and Wales Mortality Data Full Data Set, 2010 - 2019")
```

Table 2: ONS England and Wales Mortality Data Full Data Set, 2010 - 2019

index	category_1	category_2	counts	date	week_no
1	Total deaths	all ages	12,968	2010-01-08	1
2	Total deaths	all ages	12,541	2010-01-15	2
3	Total deaths	all ages	11,762	2010-01-22	3
4	Total deaths	all ages	11,056	2010-01-29	4
5	Total deaths	all ages	10,524	2010-02-05	5

Save the raw data to the project

The raw data is saved to the project folder using the here() function.

```
write_csv(mort, here("RawData", "ons_mortality.csv"))
```

Select a subset of variables for analysis

This project does not require all of the data contained in ons___mortality. A subset of the data will be created for use in the project. This project will examine the variance between the actual mortality and the 5-year average by season. This will be done for data on Total deaths. A subset of data is needed containing "Total Deaths" for "All Ages" and the "average of same week over 5 years".

Examine the data to be combined

First, filter() is used to create an object with the "average of the same week over 5 years" data and view the result.

```
mort_5year_avg <- filter(mort,category_2 == "average of same week over 5 years")
glimpse(mort_5year_avg)
```

```
## Rows: 521
## Columns: 6
## $ index      <int> 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,~
## $ category_1 <chr> "Total deaths", "Total deaths", "Total deaths", "Total deat~
## $ category_2 <chr> "average of same week over 5 years", "average of same week ~
## $ counts     <dbl> 12050.0, 12600.0, 11692.2, 11069.2, 10840.4, 10602.4, 10774~
## $ date       <date> 2010-01-08, 2010-01-15, 2010-01-22, 2010-01-29, 2010-02-05~
## $ week_no    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
```

```
tail(mort_5year_avg)
```

```
## # A tibble: 6 x 6
##   index category_1   category_2                          counts date       week_no
##   <int> <chr>        <chr>                                <dbl> <date>       <int>
## 1 16098 Total deaths average of same week over 5 years  10124 2019-11-22      47
## 2 16099 Total deaths average of same week over 5 years  10164 2019-11-29      48
## 3 16100 Total deaths average of same week over 5 years  10585 2019-12-06      49
## 4 16101 Total deaths average of same week over 5 years  10622 2019-12-13      50
## 5 16102 Total deaths average of same week over 5 years  11499 2019-12-20      51
## 6 16103 Total deaths average of same week over 5 years   8014 2019-12-27      52
```

There are **521 rows** of data. All rows are Category 1 "Total Deaths" and run from Week 1 (08/01/2010) to Week 52 (27/12/2019). Next, the filter() function is used to create an object with only "Total deaths" and "all ages" to examine the length and dates of Total deaths and all ages data.

```
mort_total <- filter(mort,category_1 == "Total deaths" & category_2 == "all ages")
glimpse(mort_total)
```

```
## Rows: 535
## Columns: 6
## $ index      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
## $ category_1 <chr> "Total deaths", "Total deaths", "Total deaths", "Total deat~
## $ category_2 <chr> "all ages", "all ages", "all ages", "all ages", "all ages",~
## $ counts     <dbl> 12968, 12541, 11762, 11056, 10524, 10117, 10102, 10295, 998~
## $ date       <date> 2010-01-08, 2010-01-15, 2010-01-22, 2010-01-29, 2010-02-05~
## $ week_no    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
```

```
tail(mort_total)
```

```
## # A tibble: 6 x 6
##   index category_1   category_2 counts date       week_no
##   <int> <chr>        <chr>       <dbl> <date>       <int>
## 1 17776 Total deaths all ages    10816 2020-02-28       9
## 2 17777 Total deaths all ages    10895 2020-03-06      10
## 3 17778 Total deaths all ages    11019 2020-03-13      11
## 4 17779 Total deaths all ages    10645 2020-03-20      12
## 5 17780 Total deaths all ages    11141 2020-03-27      13
## 6 17781 Total deaths all ages    16387 2020-04-03      14
```

The total deaths for all ages has **535 observations** while the average data has **521 observations**. The

last date for 5-year average observation is 27-12-2019, so need to remove "all deaths" observations after this date. Now we create a subset of the data for "all deaths" and "all ages" up to and including the last week of December 2019.

```
mort_all <- filter(mort,category_1 == "Total deaths" & category_2 == "all ages" & date < "2019-12-31")
glimpse(mort_all)
```

```
## Rows: 521
## Columns: 6
## $ index      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
## $ category_1 <chr> "Total deaths", "Total deaths", "Total deaths", "Total deat~
## $ category_2 <chr> "all ages", "all ages", "all ages", "all ages", "all ages",~
## $ counts     <dbl> 12968, 12541, 11762, 11056, 10524, 10117, 10102, 10295, 998~
## $ date       <date> 2010-01-08, 2010-01-15, 2010-01-22, 2010-01-29, 2010-02-05~
## $ week_no    <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
```

Append the 5 year average

The 5 years average data for the relevant dates is appended as a column to the subset of Total deaths for all ages.

```
mort_all$mort_avg = mort_5year_avg$counts
```

Calculate the variance between actual deaths and the 5 year average

Another column is added with a new variable with the calculated difference between the actual mortality and the average for the 5 years preceding that date.

```
mort_all$variance_from_avg = mort_all$counts / mort_all$mort_avg
```

Create a year column

The data will be examined and plotted using the week and year for the observation. For this purpose it is helpful to have a year variable based on the date for the same observation.

```
mort_all$year = as.character(year(mort_all$date))
```

Build the subset data frame

The data frame for analysis is created with the needed data.

```
mort_all <- mort_all[, c("index", "week_no", "year", "date", "counts", "mort_avg", "variance_from_avg")]
head(mort_all)
```

```
## # A tibble: 6 x 7
##   index week_no year  date       counts mort_avg variance_from_avg
##   <int>   <int> <chr> <date>      <dbl>    <dbl>             <dbl>
## 1     1       1 2010  2010-01-08  12968   12050              1.08
## 2     2       2 2010  2010-01-15  12541   12600              0.995
## 3     3       3 2010  2010-01-22  11762   11692.             1.01
## 4     4       4 2010  2010-01-29  11056   11069.             0.999
## 5     5       5 2010  2010-02-05  10524   10840.             0.971
## 6     6       6 2010  2010-02-12  10117   10602.             0.954
```

Check for missing values

There were NA values in the count column when the raw data was examined so its prudent to check again to see if the subset is complete or whether null values need to be managed.

```
mort_all %>%
  map(is.na) %>%
  map(sum)
```

```
## $index
## [1] 0
##
## $week_no
## [1] 0
##
## $year
## [1] 0
##
## $date
## [1] 0
##
## $counts
## [1] 0
##
## $mort_avg
## [1] 0
##
## $variance_from_avg
## [1] 0
```

There are no missing values and so we can proceed with analysis.

Tabulate the subset

The subset data is tabulated to make the data more readable.

```
kable(head(mort_all, n = 5),
      digits = 3,
      format.args = list(big.mark = ","),
      caption= "England and Wales Mortality Data with Variance from Average, 2010 - 2019")
```

Table 3: England and Wales Mortality Data with Variance from Average, 2010 - 2019

index	week_no	year	date	counts	mort_avg	variance_from_avg
1	1	2010	2010-01-08	12,968	12,050.0	1.076
2	2	2010	2010-01-15	12,541	12,600.0	0.995
3	3	2010	2010-01-22	11,762	11,692.2	1.006
4	4	2010	2010-01-29	11,056	11,069.2	0.999
5	5	2010	2010-02-05	10,524	10,840.4	0.971

Save the subset to the project

The subset of data is saved to the project folder using the here()) function.

```
write_csv(mort_all, here("RawData", "ons_mortality_ENG_1019.csv"))
```

Create test and training subsets

A subset of the data is needed to use in evaluating the data capture tool.

Calculate the proportion of the data subset to use for testing purposes

```
prop<-(1-(15/nrow(mort_all)))
print(prop)
```

```
## [1] 0.9712092
```

Set a seed to be able to replicate the random generator in future

The 'set.seed()' function is a random number generator, which is useful for creating random selections 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.

```
set.seed(432)
```

Partition the raw data

The createDataPartition() function is used to generate an object using the previously identified proportion of index rows. The index object is used to create the training set.

```
train_index <- createDataPartition(mort_all$index, p = prop,
                                   list = FALSE,
                                   times = 1)
head(train_index)
```

```
##      Resample1
## [1,]         1
## [2,]         2
## [3,]         3
## [4,]         4
## [5,]         5
## [6,]         6
```

Training subset

Assign the training data to a new object

All records in the train_index are assigned to the training data.

```
mort_alltrain <- mort_all[ train_index,]
nrow(mort_alltrain)
```

```
## [1] 509
```

There are 509 records in the training set

Tabulate the training dataset

The subset data is then tabulated to make the data more readable.

```
kable(head(mort_alltrain, n = 5),
      digits = 3,
      format.args = list(big.mark = ","),
      caption= "Training Data: England and Wales Mortality Data and Variance from Average, 2010 - 2019")
```

Table 4: Training Data: England and Wales Mortality Data and Variance from Average, 2010 - 2019

index	week_no	year	date	counts	mort_avg	variance_from_avg
1	1	2010	2010-01-08	12,968	12,050.0	1.076
2	2	2010	2010-01-15	12,541	12,600.0	0.995
3	3	2010	2010-01-22	11,762	11,692.2	1.006
4	4	2010	2010-01-29	11,056	11,069.2	0.999
5	5	2010	2010-02-05	10,524	10,840.4	0.971

Save the raw data to the project

The raw data is saved to the project folder using the here() function.

```
write_csv(mort_alltrain, here("Data", "ons_mortality_ENG_1019_training.csv"))
```

Create the test subset

Allocate to the test subset all records that are not assigned to the training data.

```
mort_alltest  <- mort_all[-train_index,]
nrow(mort_alltest)
```

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

There are 12 records in the test data. One row needs to be set aside for use by the assignment markers to test and evaluate the data-capture tool.

Markers subset

Create the Markers subset

The following code takes the first row of the test subset and saves it to a new object.

```
mort_alltestMarker  <- mort_alltest[1,]
```

Tabulate the Markers subset

The subset data is then tabulated to make the data more readable.

```
kable(head(mort_alltestMarker, n = 10),
      digits = 3,
      format.args = list(big.mark = ","),
      caption= "Markers Test Data: England and Wales Mortality Data and Variance from Average, 2010 - 20
```

Table 5: Markers Test Data: England and Wales Mortality Data and Variance from Average, 2010 - 2019

index	week_no	year	date	counts	mort_avg	variance_from_avg
1,792	24	2011	2011-06-17	8,961	8,984.8	0.997

Save the the Markers subset

The raw data is saved to the project folder using the here()) function.

```
write_csv(mort_alltestMarker, here("Data", "ons_mortality_ENG_1019_test_marker.csv"))
```

Create the test subset

Remove the Markers row from the test set.

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

Tabulate the test subset

Tabulate the remaining test subset data.

```
kable(head(mort_alltrain, n = 5),
      digits = 3,
      format.args = list(big.mark = ","),
      caption= "Test Data: England and Wales Mortality Data and Variance from Average, 2010 - 2019")
```

Table 6: Test Data: England and Wales Mortality Data and Variance from Average, 2010 - 2019

index	week_no	year	date	counts	mort_avg	variance_from_avg
1	1	2010	2010-01-08	12,968	12,050.0	1.076
2	2	2010	2010-01-15	12,541	12,600.0	0.995
3	3	2010	2010-01-22	11,762	11,692.2	1.006
4	4	2010	2010-01-29	11,056	11,069.2	0.999
5	5	2010	2010-02-05	10,524	10,840.4	0.971

Save the test subset

```
write_csv(mort_alltest, here("Data", "ons_mortality_ENG_1019_test.csv"))
```

Create the data dictionary

This data dictionary provides information about data from the ONS Mortality data collected using the The Jupyter Notebook data collection tool saved at "./ipynbScripts/data_collection_tool.ipynb". The output is saved as an R dataset (.rds) in the 'RawData' folder.

Load packages

```
library(dataMeta)
library (tidyverse)
library(here)
```

Load the data

```
collected_data_final=read_csv(here("RawData", "collected_data_final.csv"))
```

View the data

```
glimpse(collected_data_final)
```

```
## Rows: 11
## Columns: 8
## $ index             <dbl> 1798, 1803, 5324, 5326, 7121, 10722, 10742, 12473, 1~
## $ week_no           <dbl> 30, 35, 19, 21, 48, 27, 47, 10, 28, 22, 30
## $ year              <dbl> 2011, 2011, 2013, 2013, 2014, 2016, 2016, 2017, 2018~
## $ date              <date> 2011-07-29, 2011-09-02, 2013-05-10, 2013-05-24, 2014~
## $ counts            <dbl> 0, 7717, 8814, 9530, 9928, 9138, 10603, 11077, 9293~
## $ mort_avg          <dbl> 8737.4, 7984.6, 9096.0, 9311.2, 9398.8, 8872.2, 9572~
## $ variance_from_avg <dbl> 0.967794, 0.966485, 0.968997, 1.023499, 1.056305, 1.~
## $ consent           <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE~
```

The collected_data_final data set contains:

- **category_1:** character, containing the names of the groups for counts, e.g. "Total deaths", "all ages".
- **category_2**: character, subcategory of names of groups where necessary, e.g. details of region: "East", details of age bands "15-44".
- **counts:** integer, actual number of deaths in whole numbers.
- **date:** date, format is yyyy-mm-dd; all dates are a Friday.
- **week_no:** integer, each week in a year is numbered sequentially.
- **mort_avg** float, average mortality for the 5-years prior to date
- **variance_from_avg** float, difference between actual mortality and 5-year average as a proportion
- **consent:** the consent from the end-user to process and share the data collected with the data capture tool.

Change the date format to character

The data dictionary functions used below work more effectively with character type data rather than date type data, tehrefore the date column is transformed.

```
collected_data_final$date <- as.character(collected_data_final$date)
```

Build a linker data frame

Create two string vectors representing the different variable descriptions and the different variable types.

Variable Descriptions Create the variable descriptions

```
variable_description <- c("The index column that allows us to link the data collected to the original or
                          "The week in the year that this activity relates to, with numbers starting fr
                          "The year that this activity relates to.",
                          "The date of the Friday of the week that this activity relates to.",
                          "The count of deaths in that week.",
                          "The average count of deaths in the same week using data from the previous 5-y
                          "The difference between the actual deaths in the week compared to the 5-year a
                          "The consent from the end-user to process and share the data collected with th
```

Variable Types There are six quantitative values variables and two fixed values (allowable values or codes) variables.

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

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

Build the Linker The collected_data_final variable names, variable_description and variable_type string vectors are brought together in a intermediary (linker) data frame.

```
linker<-build_linker(collected_data_final, variable_description, variable_type)
```

Create the data dictionary

The build_dict function is used to create a data frame using the linker (and the variable names, variable descriptions and variable types) to create the data dictionary.

```
dictionary <- build_dict(my.data = collected_data_final, linker = linker, option_description = NULL, pr
glimpse(dictionary)
```

```
## Rows: 18
## Columns: 3
## $ variable_name        <chr> "consent", "counts", "date", " ", " ", " ", " ", ~
## $ variable_description <chr> "The consent from the end-user to process and sha~
## $ variable_options     <chr> "TRUE", "0 to 11077", "2011-07-29", "2011-09-02",~
```

Save the data dictionary

the data is saved to the RawData folder.

```
write_csv(dictionary, here("RawData", "collected_data_dictionary.csv"))
```

Append data dictionary to the collected_data_final file

Create main_string for attributes

An introductory description for the data set is created to be added as meta data alongside the data dictionary below.

```
main_string <- "This data describes the NHS England Mortality data and running 5 years averages from th
main_string
```

```
## [1] "This data describes the NHS England Mortality data and running 5 years averages from the *NHSRda
```

Incorporate attributes as metada

```
complete_collected_data <- incorporate_attr(my.data = collected_data_final, data.dictionary = dictionary
                                            main_string = main_string)
#Change the author name
attributes(complete_collected_data)$author[1]<-"B210533"
attributes(complete_collected_data)
```

```
## $row.names
##  [1]  1  2  3  4  5  6  7  8  9 10 11
##
## $names
## [1] "index"             "week_no"           "year"
## [4] "date"              "counts"            "mort_avg"
## [7] "variance_from_avg" "consent"
```

```
##
## $spec
## cols(
##   index = col_double(),
##   week_no = col_double(),
##   year = col_double(),
##   date = col_date(format = ""),
##   counts = col_double(),
##   mort_avg = col_double(),
##   variance_from_avg = col_double(),
##   consent = col_logical()
## )
##
## $problems
## <pointer: 0x55918d22c9a0>
##
## $class
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
##
## $main
## [1] "This data describes the NHS England Mortality data and running 5 years averages from the *NHSRda
##
## $dictionary
##        variable_name
## 1            consent
## 2             counts
## 3               date
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14              index
## 15          mort_avg
## 16 variance_from_avg
## 17           week_no
## 18              year
##
## 1                       The consent from the end-user to process and share the data collected with th
## 2                                                                                    The count of
## 3                                                  The date of the Friday of the week that this
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
```

```
## 12
## 13
## 14 The index column that allows us to link the data collected to the original ons_mortality data in t
## 15                                      The average count of deaths in the same week using data from
## 16           The difference between the actual deaths in the week compared to the 5-year average as a
## 17                                The week in the year that this activity relates to, with numbers st
## 18                                                                                    The year that this
##       variable_options
## 1                 TRUE
## 2           0 to 11077
## 3           2011-07-29
## 4           2011-09-02
## 5           2013-05-10
## 6           2013-05-24
## 7           2014-11-28
## 8           2016-07-08
## 9           2016-11-25
## 10          2017-03-10
## 11          2018-07-13
## 12          2019-05-31
## 13          2019-07-26
## 14       1798 to 16029
## 15     7984.6 to 10816
## 16 0.966485 to 1.10771
## 17            10 to 48
## 18        2011 to 2019
##
## $last_edit_date
## [1] "2022-06-20 18:05:32 UTC"
##
## $author
## [1] "B210533"
```

Save collected_data_final with attributes

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

```
complete_collected_data <- readRDS(here("RawData", "complete_collected_data.rds"))
```

Save RDS

Create the data capture tool with Python

The data capture tool has been created using Python in Jupyter Notebook. The data is captured using interactive widgets. It has been tested using the test data. The output is saved in the B210533_assessment/RawData folder as collected_data_dictionary.csv.