Title: ‘**Combined R Markdown document**’

Author: “B210741”

Date: “20 June, 2022”

1. Link to Github repository:

https://github.com/B210741/B210741_assessment.git (https://github.com/B210741/B210741_assessment.git)

2. Constructing a data dictionary and appending it to the data

Loading NHSRdatasets

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

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

Exploring the ae_attendances data

The dataset set I have chosen to manage from the NHSRdatasets package is the NHS England accident and emergency (A&E) attendances and admissions (ae_attendances) data. The ae_attendances data includes reported attendances, four-hour breaches and admissions for all A&E departments in England for 2016/17 through 2018/19 (Apr-Mar). We previously selected a subset of the variables needed for my data capture tool, including period, attendances and breaches, and subsetted the data into test and training data.

```
data(ae_attendances)
ae<-ae_attendances
class(ae)
```

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

```
ae
```

```
## # A tibble: 12,765 × 6
##    period     org_code type  attendances breaches admissions
##    <date>     <fct>    <fct>       <dbl>    <dbl>      <dbl>
##  1 2017-03-01 RF4      1           21289     2879       5060
##  2 2017-03-01 RF4      2             813       22          0
##  3 2017-03-01 RF4      other        2850        6          0
##  4 2017-03-01 R1H      1           30210     5902       6943
##  5 2017-03-01 R1H      2             807       11          0
##  6 2017-03-01 R1H      other       11352      136          0
##  7 2017-03-01 AD913    other        4381        2          0
##  8 2017-03-01 RYX      other       19562      258          0
##  9 2017-03-01 RQM      1           17414     2030       3597
## 10 2017-03-01 RQM      other        7817       86          0
## # … with 12,755 more rows
```

```
glimpse(ae)
```

```
## Rows: 12,765
## Columns: 6
## $ period      <date> 2017-03-01, 2017-03-01, 2017-03-01, 2017-03-01, 2017-03-0…
## $ org_code    <fct> RF4, RF4, RF4, R1H, R1H, R1H, AD913, RYX, RQM, RQM, RJ6, R…
## $ type        <fct> 1, 2, other, 1, 2, other, other, other, 1, other, 1, other…
## $ attendances <dbl> 21289, 813, 2850, 30210, 807, 11352, 4381, 19562, 17414, 7…
## $ breaches    <dbl> 2879, 22, 6, 5902, 11, 136, 2, 258, 2030, 86, 1322, 140, 0…
## $ admissions  <dbl> 5060, 0, 0, 6943, 0, 0, 0, 0, 3597, 0, 2202, 0, 0, 0, 3360…
```

```
head(ae)
```

```
## # A tibble: 6 × 6
##   period     org_code type  attendances breaches admissions
##   <date>     <fct>    <fct>       <dbl>    <dbl>      <dbl>
## 1 2017-03-01 RF4      1           21289     2879       5060
## 2 2017-03-01 RF4      2             813       22          0
## 3 2017-03-01 RF4      other        2850        6          0
## 4 2017-03-01 R1H      1           30210     5902       6943
## 5 2017-03-01 R1H      2             807       11          0
## 6 2017-03-01 R1H      other       11352      136          0
```

```
tail(ae, n=4)
```

```
## # A tibble: 4 × 6
##   period     org_code type  attendances breaches admissions
##   <date>     <fct>    <fct>       <dbl>    <dbl>      <dbl>
## 1 2018-04-01 RA3      1            3825      476       1016
## 2 2018-04-01 AXG      other        2980       24          0
## 3 2018-04-01 NLX24    other        1538        0          0
## 4 2018-04-01 RA4      1            4388       82       1292
```

```
#There are no missing data. The data is complete.
ae %>%
  map(is.na) %>%
map(sum)
```

```
## $period
## [1] 0
##
## $org_code
## [1] 0
##
## $type
## [1] 0
##
## $attendances
## [1] 0
##
## $breaches
## [1] 0
##
## $admissions
## [1] 0
```

```
#Add an index link column to ae_attendances data
ae <- rowid_to_column(ae, "index")

#Tabulate the raw data for the report

ae %>%
  # Set the period column to show in month-year format
  mutate_at(vars(period), format, "%b-%y") %>%
  # Set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(attendances, breaches, admissions), comma) %>%
  # Show the first 10 rows
  head(10) %>%
  # Format as a table
  kable()
```

index	period	org_code	type	attendances	breaches	admissions
1	Mar-17	RF4	1	21,289.0	2,879.0	5,060.0
2	Mar-17	RF4	2	813.0	22.0	0.0
3	Mar-17	RF4	other	2,850.0	6.0	0.0
4	Mar-17	R1H	1	30,210.0	5,902.0	6,943.0
5	Mar-17	R1H	2	807.0	11.0	0.0
6	Mar-17	R1H	other	11,352.0	136.0	0.0
7	Mar-17	AD913	other	4,381.0	2.0	0.0
8	Mar-17	RYX	other	19,562.0	258.0	0.0

index	period	org_code	type	attendances	breaches	admissions
9	Mar-17	RQM	1	17,414.0	2,030.0	3,597.0
10	Mar-17	RQM	other	7,817.0	86.0	0.0

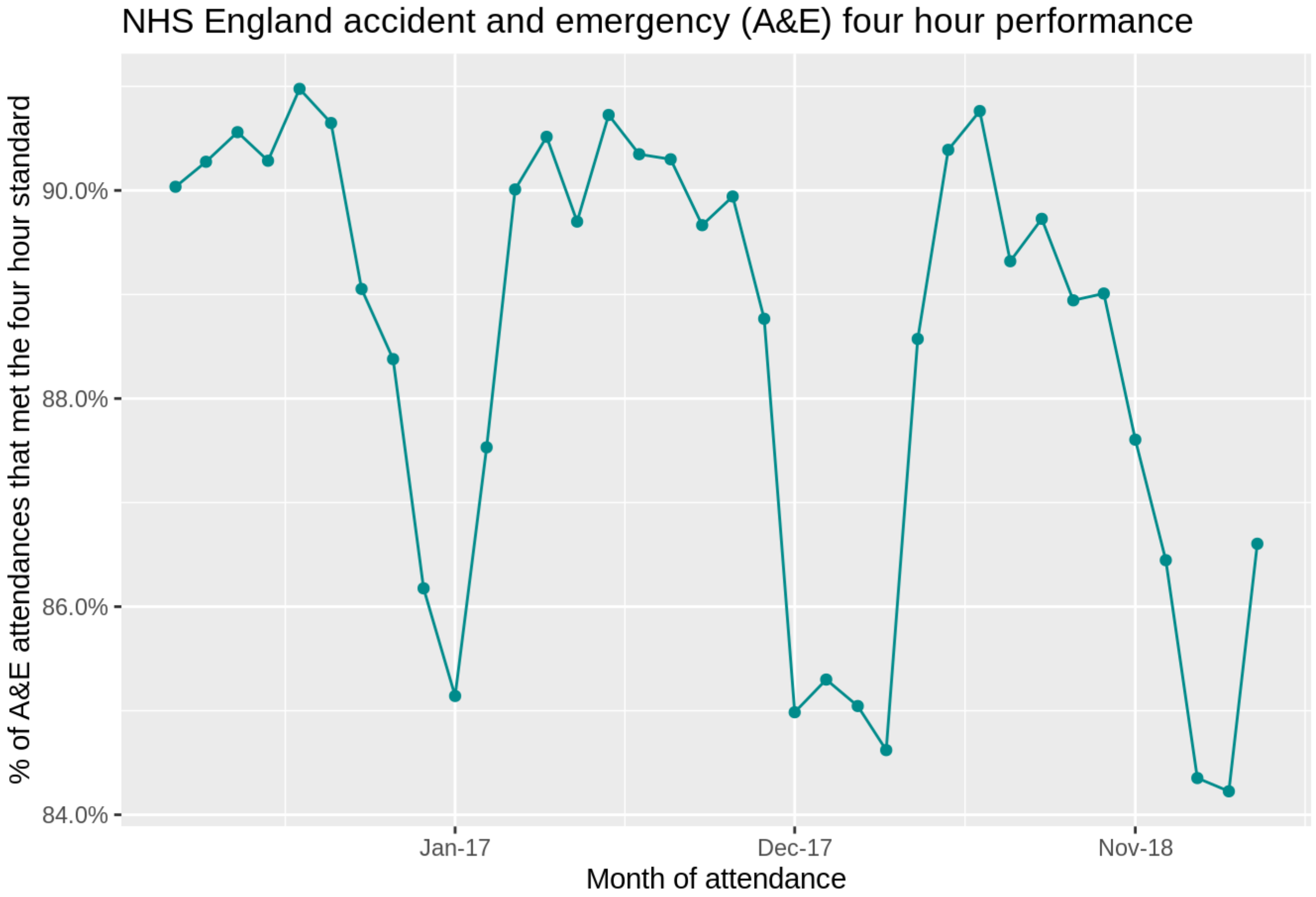
```
#Calculate monthly four hour waiting time target performance for England as a whole

ENG_performance <- ae %>%
  group_by(period) %>%
  summarise_at(vars(attendances, breaches), sum) %>%
  mutate(performance = 1 - breaches / attendances)
glimpse(ENG_performance)
```

```
## Rows: 36
## Columns: 4
## $ period      <date> 2016-04-01, 2016-05-01, 2016-06-01, 2016-07-01, 2016-08-0…
## $ attendances <dbl> 1867781, 2070340, 1958802, 2079034, 1932901, 1952464, 2001…
## $ breaches    <dbl> 186122, 201329, 184912, 201973, 174419, 182597, 219137, 22…
## $ performance <dbl> 0.9003513, 0.9027556, 0.9055994, 0.9028525, 0.9097631, 0.9…
```

```
#Visualise monthly four hour waiting time target

ggplot(ENG_performance, aes(period, performance)) +
  geom_line(color = "darkcyan") +
  geom_point(color = "darkcyan") +
  scale_y_continuous(labels = percent) +
  scale_x_date(date_labels = "%b-%y", date_breaks = "11 month")+
  labs(x = "Month of attendance",
       y = "% of A&E attendances that met the four hour standard",
       title = "NHS England accident and emergency (A&E) four hour performance",
       caption = "Source: NHSRdatasets")
```



Source: NHSRdatasets

```
#Select the ae_attendances data subset for further exploratory analysis

ae %>%
  # set the period column to show in Month-Year format
  mutate_at(vars(period), format, "%b-%y") %>%
  # set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(attendances, breaches), comma) %>%
  # show the first 10 rows
  head(10) %>%
  # format as a table
  kable()
```

index	period	org_code	type	attendances	breaches	admissions
1	Mar-17	RF4	1	21,289.0	2,879.0	5060
2	Mar-17	RF4	2	813.0	22.0	0
3	Mar-17	RF4	other	2,850.0	6.0	0
4	Mar-17	R1H	1	30,210.0	5,902.0	6943
5	Mar-17	R1H	2	807.0	11.0	0
6	Mar-17	R1H	other	11,352.0	136.0	0
7	Mar-17	AD913	other	4,381.0	2.0	0
8	Mar-17	RYX	other	19,562.0	258.0	0
9	Mar-17	RQM	1	17,414.0	2,030.0	3597

index	period	org_code	type	attendances	breaches	admissions
10	Mar-17	RQM	other	7,817.0	86.0	0

```
#save provisional subsetted ae_attendances data to the ‘RawData’ folder

write_csv(ae, here("RawData", "ae_attendances_ENG_4hr_perfom.csv"))
```

Separating provisional ae_attendances_ENG_4hr_perfom data into training and testing sets

Splint the raw data into test and training data sets

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

```
## [1] 12765
```

```
#[1] 12765

prop<-(1-(15/nrow(ae)))
#The proportion of the raw that needs to be assigned to the training data to ensure there is
 only 10 to 15 records in the test data is:

print(prop)
```

```
## [1] 0.9988249
```

```
#[1] 0.9988249

set.seed(333)

#Partitioning the raw data into the test and training data.
trainIndex <- createDataPartition(ae$index, p = prop,
                                  list = FALSE,
                                  times = 1)
head(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.

aeTrain <- ae[ trainIndex,]
nrow(aeTrain)
```

```
## [1] 12753
```

```
#[1] 12753 records
```

```
CollectedData=read_csv(here("RawData", "CollectedDataAll.csv"))

glimpse(CollectedData)
```

```
## Rows: 11
## Columns: 9
## $ index       <dbl> 1155, 2059, 3468, 4153, 4820, 7243, 8057, 8957, 10214, 103…
## $ period      <chr> "01/12/2016", "01/10/2016", "01/05/2016", "01/03/2018", "0…
## $ org_code    <chr> "C82010", "RDZ", "RVR", "RQM", "R1F", "RE9", "RQM", "RNL",…
## $ type        <chr> "other", "1", "2", "other", "other", "1", "1", "1", "other…
## $ attendances <dbl> 200, 6452, 417, 9376, 245, 5170, 15957, 7258, 3197, 2033, …
## $ breaches    <dbl> 0, 360, 0, 112, 0, 235, 1309, 1374, 0, 8, 1
## $ admissions  <dbl> 0, 1814, 6, 0, 0, 1269, 3375, 1947, 0, 105, 0
## $ performance <dbl> 1.00, 0.94, 1.00, 0.99, 1.00, 0.95, 0.92, 0.81, 1.00, 1.00…
## $ consent     <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE…
```

Let’s tabulate ae_attendances_ENG_4hr_perfom training data for your report

```
aeTrain %>%
  # set the period column to show in Month-Year format
  mutate_at(vars(period), format, "%b-%y") %>%
  # set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(attendances, breaches), comma) %>%
  # show the first 10 rows
  head(10) %>%
  # format as a table
  kable()
```

index	period	org_code	type	attendances	breaches	admissions
1	Mar-17	RF4	1	21,289.0	2,879.0	5060
2	Mar-17	RF4	2	813.0	22.0	0
3	Mar-17	RF4	other	2,850.0	6.0	0
4	Mar-17	R1H	1	30,210.0	5,902.0	6943
5	Mar-17	R1H	2	807.0	11.0	0
6	Mar-17	R1H	other	11,352.0	136.0	0
7	Mar-17	AD913	other	4,381.0	2.0	0
8	Mar-17	RYX	other	19,562.0	258.0	0
9	Mar-17	RQM	1	17,414.0	2,030.0	3597
10	Mar-17	RQM	other	7,817.0	86.0	0

```
write_csv(aeTrain, here("Data", "ae_attendances_ENG_4hr_perfom_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.
aeTest  <- ae[-trainIndex,]
nrow(aeTest)
```

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

```
#[1] 12
#There are 12 records in your test data
```

Set aside the first record from the ae_attendances_ENG_4hr_perfom test data

```
aeTestMarker  <- aeTest[1,]
```

Tabulate ae_attendances_ENG_4hr_perfom marker test data for the report

```
aeTestMarker  %>%
  # set the period column to show in Month-Year format
  mutate_at(vars(period), format, "%b-%y") %>%
  # set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(attendances, breaches), comma) %>%
  # show the first 10 rows
  head(10) %>%
  # format as a table
  kable()
```

index	period	org_code	type	attendances	breaches	admissions
115	Mar-17	NLO12	other	309	1	0

Save the ae_attendances_ENG_4hr_perfom marker test data to the working data folder ‘Data’

```
write_csv(aeTestMarker, here("Data", "ae_attendances_ENG_4hr_perfom_test_marker.csv"))
```

Setting aside the remaining records for testing data-capture tool.

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

Tabulate ae_attendances_ENG_4hr_perfom test data for the report

```
aeTest  %>%
  # set the period column to show in Month-Year format
  mutate_at(vars(period), format, "%b-%y") %>%
  # set the numeric columns to have a comma at the 1000's place
  mutate_at(vars(attendances, breaches), comma) %>%
  # show the first 10 rows
  head(10) %>%
  # format as a table
  kable()
```

index	period	org_code	type	attendances	breaches	admissions
1155	Dec-16	C82010	other	200	0.0	0
2059	Oct-16	RDZ	1	6,452	360.0	1814
3468	May-16	RVR	2	417	0.0	6
4153	Mar-18	RQM	other	9,376	112.0	0
4820	Feb-18	R1F	other	245	0.0	0
7243	Jul-17	RE9	1	5,170	235.0	1269
8057	Apr-17	RQM	1	15,957	1,309.0	3375
8957	Feb-19	RNL	1	7,258	1,374.0	1947
10214	Oct-18	RJ1	other	3,197	0.0	0
10328	Oct-18	RKB	2	2,033	8.0	105

Save the ae_attendances_ENG_4hr_perfom test data to the working data folder ‘Data’

```
write_csv(aeTest, here("Data", "ae_attendances_test.csv"))
```

The CollectedData dataset contains:

index: the index column that allows us to link the data collected to the original ae_attendances data in the ‘RawData’ folder.

period: the month that this activity relates to, stored as a date (1st of each month).

org_code: the Organisation data service (ODS) code for the organisation.

type: the Department Type for this activity, either

*1: Emergency departments are a consultant-led 24-hour service with full resuscitation facilities and designated accommodation for the reception of accident and emergency patients,

*2: Consultant-led mono speciality accident and emergency service (e.g. ophthalmology, dental) with designated accommodation for the reception of patients, or

*other: Other types of A&E/minor injury activity with designated accommodation for the reception of accident and emergency patients.

attendances: the number of attendances for this department type at this organisation for this month.

breaches: the number of attendances that breached the four-hour target.

admissions: the number of attendances that resulted in an admission to the hospital.

performance: the performance ([1 - breaches]/attendances) calculated for the whole of England.

consent: the consent from the end-user to process and share the data collected with the data capture tool.

3. Constructing a data dictionary and appending it to the data

Build a linker data frame

Variable descriptions

Create a string vector representing the different variable descriptions

```
variable_description <- c("The index column that allows us to link the data collected to the
 original ae_attendances data in the 'RawData' folder.",
"The month that this activity relates to, stored as a date (1st of each month).",
"The Organisation data service (ODS) code for the organisation. If you want to know the organ
isation associated with a particular ODS code, you can look it up from the following address:
https://odsportal.digital.nhs.uk/Organisation/Search.",
"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 number of attendances that resulted in an admission to the hospital.",
"The performance ([1 - breaches]/attendances) calculated for the whole of England.",
"The consent from the end-user to process and share the data collected with the data capture
 tool.")

print(variable_description)
```

```
## [1] "The index column that allows us to link the data collected to the original ae_attenda
nces data in the 'RawData' folder."
## [2] "The month that this activity relates to, stored as a date (1st of each month)."
## [3] "The Organisation data service (ODS) code for the organisation. If you want to know th
e organisation associated with a particular ODS code, you can look it up from the following a
ddress: https://odsportal.digital.nhs.uk/Organisation/Search."
## [4] "The department type for this activity."
## [5] "The number of attendances for this department type at this organisation for this mont
h."
## [6] "The number of attendances that breached the four-hour target."
## [7] "The number of attendances that resulted in an admission to the hospital."
## [8] "The performance ([1 - breaches]/attendances) calculated for the whole of England."
## [9] "The consent from the end-user to process and share the data collected with the data c
apture tool."
```

Variable types

```
glimpse(CollectedData)
```

```
## Rows: 11
## Columns: 9
## $ index       <dbl> 1155, 2059, 3468, 4153, 4820, 7243, 8057, 8957, 10214, 103…
## $ period      <chr> "01/12/2016", "01/10/2016", "01/05/2016", "01/03/2018", "0…
## $ org_code    <chr> "C82010", "RDZ", "RVR", "RQM", "R1F", "RE9", "RQM", "RNL",…
## $ type        <chr> "other", "1", "2", "other", "other", "1", "1", "1", "other…
## $ attendances <dbl> 200, 6452, 417, 9376, 245, 5170, 15957, 7258, 3197, 2033, …
## $ breaches    <dbl> 0, 360, 0, 112, 0, 235, 1309, 1374, 0, 8, 1
## $ admissions  <dbl> 0, 1814, 6, 0, 0, 1269, 3375, 1947, 0, 105, 0
## $ performance <dbl> 1.00, 0.94, 1.00, 0.99, 1.00, 0.95, 0.92, 0.81, 1.00, 1.00…
## $ consent     <lgl> TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE…
```

We have five quantitative values (measured values) variables and four fixed values (allowable values or codes) variables.

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

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

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

print(linker)
```

```
##      var_name
## 1       index
## 2      period
## 3    org_code
## 4        type
## 5 attendances
## 6    breaches
## 7  admissions
## 8 performance
## 9     consent
##
var_desc
## 1
The index column that allows us to link the data collected to the original ae_attendances dat
a in the 'RawData' folder.
## 2
The month that this activity relates to, stored as a date (1st of each month).
## 3 The Organisation data service (ODS) code for the organisation. If you want to know the o
rganisation associated with a particular ODS code, you can look it up from the following addr
ess: https://odsportal.digital.nhs.uk/Organisation/Search.
## 4
The department type for this activity.
## 5
The number of attendances for this department type at this organisation for this month.
## 6
The number of attendances that breached the four-hour target.
## 7
The number of attendances that resulted in an admission to the hospital.
## 8
The performance ([1 - breaches]/attendances) calculated for the whole of England.
## 9
The consent from the end-user to process and share the data collected with the data capture t
ool.
##   var_type
## 1        0
## 2        1
## 3        1
## 4        1
## 5        0
## 6        0
## 7        0
## 8        0
## 9        1
```

Data dictionary

Using the build_dict() function from the dataMeta to constructs a data dictionary for a CollectedData data frame with the aid of the linker data frame between.

```
#dictionary <- build_dict(my.data = CollectedData, linker = linker)
dictionary <- build_dict(my.data = CollectedData, linker = linker, option_description = NULL,
prompt_varopts = FALSE)
glimpse(dictionary)
```

```
## Rows: 29
## Columns: 3
## $ variable_name        <chr> "admissions", "attendances", "breaches", "consent…
## $ variable_description <chr> "The number of attendances that resulted in an ad…
## $ variable_options     <chr> "0 to 3375", "200 to 15957", "0 to 1374", "TRUE",…
```

```
dictionary[6,4]<-"C82010: Prescribing Cost Centre - OAKHAM MEDICAL PRACTICE."
dictionary[7,4]<-"RDZ: NHS Trust - The Royal Bournemouth and Christchurch Hospitals NHS Found
ation Trust."
dictionary[8,4]<-"RVR: NHS Trust - EPSOM AND ST HELIER UNIVERSITY HOSPITALS NHS TRUST."
dictionary[9,4]<-"RQM: NHS Trust - CHELSEA AND WESTMINSTER HOSPITAL NHS FOUNDATION TRUST."
dictionary[10,4]<-"R1F: NHS Trust - ISLE OF WIGHT NHS TRUST."
dictionary[11,4]<-"RE9: NHS Trust - SOUTH TYNESIDE NHS FOUNDATION TRUST."
dictionary[12,4]<-"RNL: NHS Trust - NORTH CUMBRIA UNIVERSITY HOSPITALS NHS TRUST."
dictionary[13,4]<-"RJ1 - NHS Trust - GUY'S AND ST THOMAS' NHS FOUNDATION TRUST."
dictionary[14,4]<-"RKB - NHS Trust - UNIVERSITY HOSPITALS COVENTRY AND WARWICKSHIRE NHS TRUS
T."
dictionary[15,4]<-"NLO12 - Independent Sector H/c Provider Site - OAKHAM URGENT CARE CENTRE."
dictionary[27,4] <-"other: Other types of A&E/minor injury activity with designated accommoda
tion for the reception of accident and emergency patients."
dictionary[28,4] <-"1: Emergency departments are a consultant-led 24-hour service with full r
esuscitation facilities and designated accommodation for the reception of accident and emerge
ncy patients."
dictionary[29,4] <-"2: Consultant-led mono speciality accident and emergency service (e.g. op
hthalmology, dental) with designated accommodation for the reception of patients."
```

Save the data dictionary for CollectedData to the ‘RawData’ folder

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

Incorporating attributes as metadata to the CollectedData as metadata using the ‘incorporate_attr()’ function from the dataMeta package.

```
main_string <- "This data describes the NHS England accident and emergency (A&E) attendances
 and breaches of four-hour wait time target data from the *NHSRdatasets* package collected by
the data capture tool."

main_string
```

```
## [1] "This data describes the NHS England accident and emergency (A&E) attendances and brea
ches of four-hour wait time target data from the *NHSRdatasets* package collected by the data
capture tool."
```

Incorporate attributes as metadata

Using the ‘incorporate_attr()’ function to return an R dataset containing metadata stored in its attributes. The attributes we are going to add 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_CollectedData <- incorporate_attr(my.data = CollectedData, data.dictionary = diction
ary,
main_string = main_string)

#Change the author name
attributes(complete_CollectedData)$author[1]<-"B210741"
complete_CollectedData
```

```
## # A tibble: 11 × 9
##    index period     org_code type  attendances breaches admissions performance
##  * <dbl> <chr>      <chr>    <chr>       <dbl>    <dbl>      <dbl>       <dbl>
##  1  1155 01/12/2016 C82010   other         200        0          0        1
##  2  2059 01/10/2016 RDZ      1            6452      360       1814        0.94
##  3  3468 01/05/2016 RVR      2             417        0          6        1
##  4  4153 01/03/2018 RQM      other        9376      112          0        0.99
##  5  4820 01/02/2018 R1F      other         245        0          0        1
##  6  7243 01/07/2017 RE9      1            5170      235       1269        0.95
##  7  8057 01/04/2017 RQM      1           15957     1309       3375        0.92
##  8  8957 01/02/2019 RNL      1            7258     1374       1947        0.81
##  9 10214 01/10/2018 RJ1      other        3197        0          0        1
## 10 10328 01/10/2018 RKB      2            2033        8        105        1
## 11 11767 01/06/2018 NLO12    other         336        1          0        1
## # … with 1 more variable: consent <lgl>
```

attributes(complete_CollectedData)

Save the CollectedData with attributes

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

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

4. Colleting data using interactive Jupyter widgets

Description of the code:

The data capture tool is created using Python coding language and interactive Jupyter widgets. Using the pandas package to import the data and setting an empty data frame to collect the data captured by the Jupyter widgets. Before any data is collected and saved, we need to ensure there is consent.

Description of the interactive widgets:

Boolean widgets for consent; DatePicker widget for Period variable; Selection widgets for Org_code variable; Radio buttons for Type variable; and Numeric widgets (IntText) for attendances and breaches variables.

Storage and archive:

The data is stored in the RawData folder to follow best practice for archiving and storing data.