THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

Technical Report

The NHS Case Study as an example to demonstrate the analytical capabilities by using the Jupiter Notebook with Python Programming Language as a tool to support the analysis

Vadims Suharevs

London. October 24, 2022

Contents

Background

The NHS have instructed the team that there is a requirement to conduct an analysis into its costs stating that the NHS incurs significant costs for patients, who do not attend their appointments. These extra costs could potentially be avoided by a reduction or elimination of the missed appointments.

Areas of analysis to consider are the staff capacity in the network and actual utilisation of resources.

This technical document does not answer the NHS question on missed appointments (*please refer to the enclosed PDF version of the PowerPoint Presentation or go to Appendix 3 of this report*).

Methodology

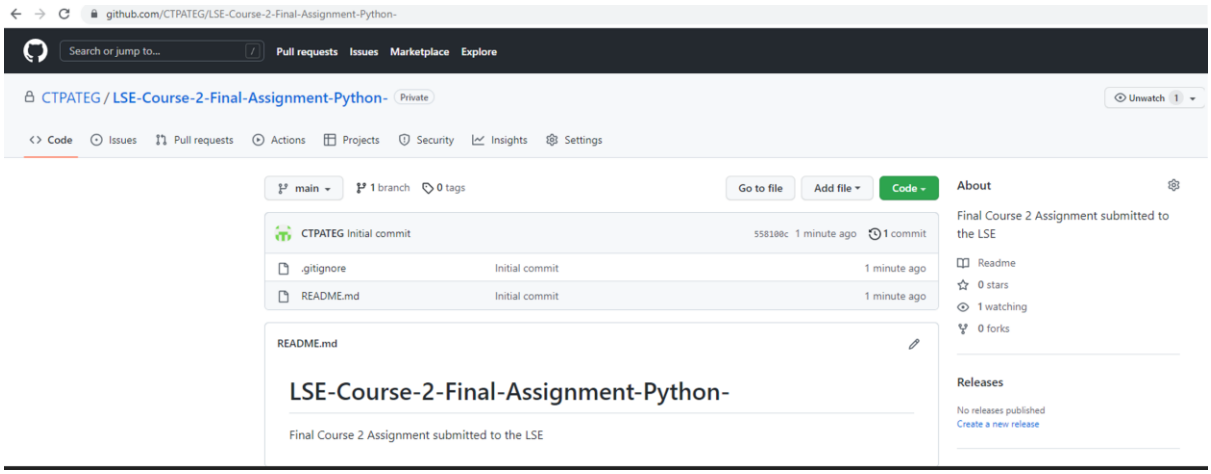
As per the academic requirement the report is a technical step by step document explaining the approach towards implementing the analysis.

Please note: The workflow of the report would be closely following the structure of the Jupiter Notebook.

1 Prework

The team had been provided with the datasets to explore the problem and make recommendations.

For the purposes of this analytics the project the team of analysts would be using Python environment via the Jupiter Notebook with various libraries to expand and support the analysis. The workbook would be saved on the GitHub repository (please follow the link: https://github.com/CTPATEG/LSE-Course-2-Final-Assignment-Python-).



2 Datasets and Data Exploration

2.1 Datasets and Starting out with Jupiter Notebook

The datasets provided are relatively “heavy” in memory - from 13mb to 38mb, and totalling close to 1 million lines on at least one of the datasets which is why Python is a tool of choice. The Jupiter notebook is structured in the way that all the activities can be run step by step from start to finish without having the need to go back to the prior sections. At the beginning of each section all the necessary libraries would be imported into the Jupiter Notebook so that if any extra analysis is required within a specific section – the library is already there and ready for use ***(Figure 1)***.

```
# Import all the potentially needed libraries for the assignment.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

from sklearn.metrics import r2_score, median_absolute_error, mean_absolute_error
from sklearn.metrics import median_absolute_error, mean_squared_error, mean_squared_log_error

# Ignore warnings.
import warnings
warnings.filterwarnings('ignore')

%matplotlib inline

# Get multiple outputs in the same cell.
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

Figure 1. Importing libraries into the Jupiter notebook

When exploring the three datasets (**actual_duration.csv *(Table 1)***, **appointments_regional.xlsx *(Table 2)*** , **national_categories.xlsx *(Table 3)***) in terms of the data quality, it became evident that it is not particularly realistic to join the datasets with a meaningful key to deep dive into the information.

Table 1. Actual Duration dataset exploration

```
# Import and sense-check the actual_duration.csv data set as ad.
ad = pd.read_csv('actual_duration.csv')

# View the DataFrame.
ad.head()
```

cb_location_code	sub_icb_location_ons_code	sub_icb_location_name	icb_ons_code	region_ons_code	appointment_date	actual_duration	count_of_appointments
00L	E38000130	NHS North East and North Cumbria ICB - 00L	E54000050	E40000012	01-Dec-21	31-60 Minutes	364
00L	E38000130	NHS North East and North Cumbria ICB - 00L	E54000050	E40000012	01-Dec-21	21-30 Minutes	619
00L	E38000130	NHS North East and North Cumbria ICB - 00L	E54000050	E40000012	01-Dec-21	6-10 Minutes	1698
00L	E38000130	NHS North East and North Cumbria ICB - 00L	E54000050	E40000012	01-Dec-21	Unknown / Data Quality	1277
00L	E38000130	NHS North East and North Cumbria ICB - 00L	E54000050	E40000012	01-Dec-21	16-20 Minutes	730

Table 2. Appointments Regional dataset exploration

```
# Import and sense-check the appointments_regional.csv data set as ar.
ar = pd.read_csv('appointments_regional.csv')

# View the DataFrame.
ar.head()
```

	icb_ons_code	appointment_month	appointment_status	hcp_type	appointment_mode	time_between_book_and_appointment	count_of_appointments
0	E54000034	2020-01	Attended	GP	Face-to-Face	1 Day	8107
1	E54000034	2020-01	Attended	GP	Face-to-Face	15 to 21 Days	6791
2	E54000034	2020-01	Attended	GP	Face-to-Face	2 to 7 Days	20686
3	E54000034	2020-01	Attended	GP	Face-to-Face	22 to 28 Days	4268
4	E54000034	2020-01	Attended	GP	Face-to-Face	8 to 14 Days	11971

Table 3. National Categories dataset exploration

```
# Import and sense-check the national_categories.xlsx data set as nc.
nc = pd.read_excel('national_categories.xlsx')

# View the DataFrame.
nc.head()
```

	appointment_date	icb_ons_code	sub_icb_location_name	service_setting	context_type	national_category	count_of_appointments	appointment_month
0	2021-08-02	E54000050	NHS North East and North Cumbria ICB - 00L	Primary Care Network	Care Related Encounter	Patient contact during Care Home Round	3	2021-08
1	2021-08-02	E54000050	NHS North East and North Cumbria ICB - 00L	Other	Care Related Encounter	Planned Clinics	7	2021-08
2	2021-08-02	E54000050	NHS North East and North Cumbria ICB - 00L	General Practice	Care Related Encounter	Home Visit	79	2021-08
3	2021-08-02	E54000050	NHS North East and North Cumbria ICB - 00L	General Practice	Care Related Encounter	General Consultation Acute	725	2021-08
4	2021-08-02	E54000050	NHS North East and North Cumbria ICB - 00L	General Practice	Care Related Encounter	Structured Medication Review	2	2021-08

2.2 Joining Datasets and Limitations

It is possible to join **national_categories.xlsx** *(Jupiter Notebook section 2.4)* with **actual_duration.csv** *(Jupiter Notebook section 2.2)* on *icb_ons_code* + *appointment_date* or *sub_icb_location_name* + *appointment_date,* but then it becomes evident that the dataset would become unusable from the perspective of the analysis, because the detail – the granularity of one or the other would be lost and the joining key would be too broad.

If we take **appointments_regional.csv** *(Jupiter Notebook section 2.3)*, the data is only available at monthly intervals, and even though the *icb_ons_code* is available, the *hcp_type* reference does not match the *service_setting* reference of **national_categories.xlsx**.

If we compare **appointments_regional.csv** and **actual_duration.csv**, the only valid key would be *appointment_month* and *icb_ons_code*, but all the detail and granularity useful for the analysis within these datasets would be lost.

Rolling up these datasets to a meaningful key selection would only result in using the *icb_ons_code* and sum the *count_of_appointments*, and would make no logical sense as this data for the period in question is readily available within the **national_categories.xlsx**. Therefore, there is no realistic transactional line level key for the join which could be used neither between each of the pairs of the reports, nor among all three reports together.

The conclusion from this preliminary analysis of the quality of the inputs suggests that the files should be worked on separately, but employing an arbitrary comparison between the datasets. Joining the

files together would bring more harm than good and would mislead the analysis or invalidate it altogether.

Further limitation with the dataset suggests that at an aggregated *icb_ons* level the Location Name is not available, which makes it difficult to roll the data up to the equivalent of the "Sold-To-Vendor" level, as the naming convention is only available at the "Ship-To-Vendor" level in commercial terms.

After exploring all three datasets within the Jupiter Notebook (please refer to *Sections 2.2, 2.3 and 2.4 on Jupiter Notebook*) it can be clearly confirmed there are no NaN (missing values within the datasets). The datasets are in the applicable formats, but if it would be required – the awareness is there that the format (i.e. date/month) could be changed for the ease of use in the analysis.

When exploring the spread of appointments data on each of the three datasets – it is evidenced that there is a substantial number of varied outliers at total level on each of the datasets (***Figure 2***). Removing these outliers could be misleading as there is a varied nature of data and a lot of variables add to the spread of data meaning that the capacity of a hospital in a city centre would be many times larger than the capacity in a rural area clinic. The above suggests that each of the datasets and subsets within them need to be looked at separately, individually and caution should be taken when discussing outliers, as crucial information might be removed if deleted.

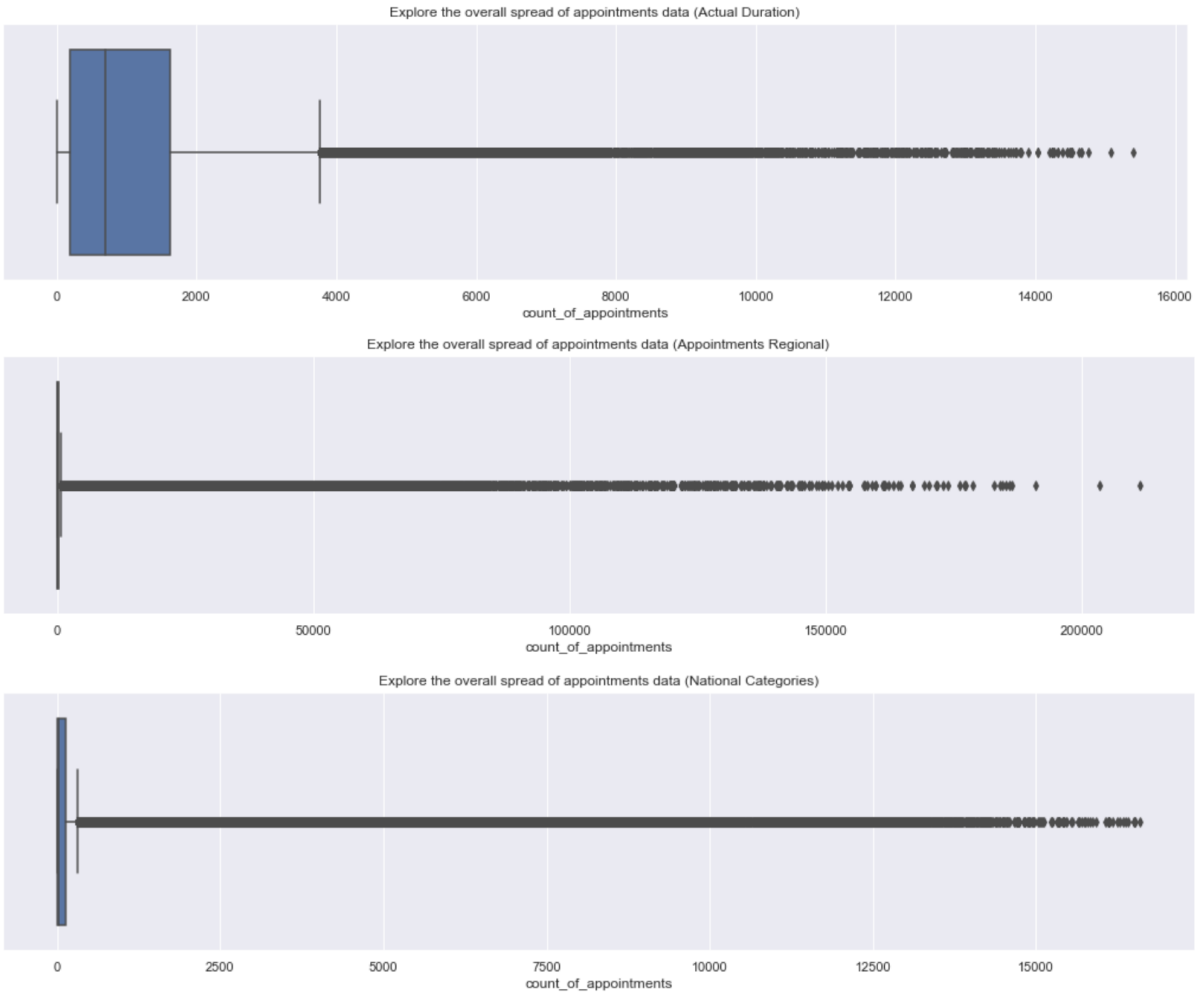


Figure 2. Boxplots of the overall spread of appointments data

2.3 Adding Value to Data (Extra Reference)

Section 2.5 on Jupiter Notebook suggests that there are 106 unique locations. These locations can be rolled up into the 42 *icb_ons_codes*. In order for the *icb_ons* location was more usable it had been decided to clean up the *sub_icb_location_name* and strip off the post code, which then matches the 42 *icb_ons_codes* and would be named *icb_location_name*. The reference file to be used further in some of the analysis snippets is called **icb_ons_name.csv** – the file needs to be copied into the Jupiter Notebook location when running all the code on the notebook (***Table 4***).

Table 4. Supporting reference file to be used for aggregations at icb_location_name level

```
# Import the supporting reference file created with the ons_code = ons name for the 42 locations.
# The ons codes does not add much value when talking with the client.
ons = pd.read_csv('icb_ons_name.csv')

# View the DataFrame.
ons.head()
```

	icb_ons_code	icb_location_name
0	E54000050	NHS North East and North Cumbria ICB
1	E54000048	NHS Lancashire and South Cumbria ICB
2	E54000057	NHS Greater Manchester ICB
3	E54000008	NHS Cheshire and Merseyside ICB
4	E54000061	NHS South Yorkshire ICB

We can then run the merge function to merge the **national_categories.xlsx, appointments_regional.csv** and **icb_ons_name.csv** datasets together to have the location name added (***Table 5***).

Table 5. Performing a MERGE function (inner join)

```
# Run the inner join function to add the icb location name to the nc dataset
nc_ons = pd.merge(nc, ons, how='inner', on='icb_ons_code')

# View the DataFrame.
nc_ons.head()
```

ıent_date	icb_ons_code	sub_icb_location_name	service_setting	context_type	national_category	count_of_appointments	appointment_month	icb_location_name
021-08-02	E54000050	NHS North East and North Cumbria ICB - 00L	Primary Care Network	Care Related Encounter	Patient contact during Care Home Round	3	2021-08	NHS North East and North Cumbria ICB
021-08-02	E54000050	NHS North East and North Cumbria ICB - 00L	Other	Care Related Encounter	Planned Clinics	7	2021-08	NHS North East and North Cumbria ICB
021-08-02	E54000050	NHS North East and North Cumbria ICB - 00L	General Practice	Care Related Encounter	Home Visit	79	2021-08	NHS North East and North Cumbria ICB
021-08-02	E54000050	NHS North East and North Cumbria ICB - 00L	General Practice	Care Related Encounter	General Consultation Acute	725	2021-08	NHS North East and North Cumbria ICB
021-08-02	E54000050	NHS North East and North Cumbria ICB - 00L	General Practice	Care Related Encounter	Structured Medication Review	2	2021-08	NHS North East and North Cumbria ICB

The analysis of the ***Figure 3*** is referenced in the Appendix 2.

```
# Determine the top five locations based on record count.
nc_ons['icb_location_name'].value_counts().head(20)
```

```
NHS Greater Manchester ICB                    59180
NHS Cheshire and Merseyside ICB               53744
NHS North East and North Cumbria ICB          53707
NHS Lancashire and South Cumbria ICB          43381
NHS West Yorkshire ICB                        42888
```

Figure 3. Number of transactions by icb_location

Then next step was to deep dive into the transaction counts and understand how many service settings, context types and national categories are there with nunique() function and value_counts() function (***Figure 4***).

```
# Determine the number of service settings.
print(f"The number of service settings: {nc['service_setting'].nunique()}")
nc['service_setting'].value_counts()
```

```
The number of service settings: 5

General Practice             359274
Primary Care Network         183790
Other                        138789
Extended Access Provision    108122
Unmapped                      27419
Name: service_setting, dtype: int64
```

```
# Determine the number of context types.
print(f"The number of context types: {nc['context_type'].nunique()}")
nc['context_type'].value_counts()
```

```
The number of context types: 3

Care Related Encounter    700481
Inconsistent Mapping       89494
Unmapped                   27419
Name: context_type, dtype: int64
```

Figure 4. nunique() and value_counts() functions for nc dataset

```
# Determine the number of appointment status.
print(
    f"The number of appointment statusses: {ar['appointment_status'].nunique()}")
ar['appointment_status'].value_counts()
```

```
The number of appointment statusses: 3

Attended    232137
Unknown     201324
DNA         163360
Name: appointment_status, dtype: int64
```

Figure 5. Appointments Regional dataset status quality

The analysis of the ***Figure 4*** and ***Figure 5*** is referenced in the Appendix 2.

3 Further Data Wrangling

When investigating the datasets further the next step was to understand which were the minimum and maximum dates available with the min() / max() functions to understand whether the datasets are comparable like for like, or the dates are inconsistent among the datasets (**Figure 6**).

```
# Determine the minimum and maximum dates in the ad DataFrame.
# Use appropriate docstrings.
print(ad['appointment_date'].min())
print(ad['appointment_date'].max())
print(
    f"The earliest date in the actual_duration dataset is {ad['appointment_date'].min()}")
print(
    f"The latest date in the actual_duration dataset is {ad['appointment_date'].max()}")
```

```
2021-12-01 00:00:00
2022-06-30 00:00:00
The earliest date in the actual_duration dataset is 2021-12-01 00:00:00
The latest date in the actual_duration dataset is 2022-06-30 00:00:00
```

```
# Determine the minimum and maximum dates in the nc DataFrame.
# Use appropriate docstrings.
print(nc['appointment_date'].min())
print(nc['appointment_date'].max())
print(
    f"The earliest date in the national_categories dataset is {nc['appointment_date'].min()}")
print(
    f"The latest date in the national_categories dataset is {nc['appointment_date'].max()}")
```

```
2021-08-01 00:00:00
2022-06-30 00:00:00
The earliest date in the national_categories dataset is 2021-08-01 00:00:00
The latest date in the national_categories dataset is 2022-06-30 00:00:00
```

```
# Determine the minimum and maximum dates in the nc DataFrame.
# Use appropriate docstrings.
print(ar['appointment_month'].min())
print(ar['appointment_month'].max())
print(
    f"The earliest month in the appointments_regional dataset is {ar['appointment_month'].min()}")
print(
    f"The latest month in the appointments_regional dataset is {ar['appointment_month'].max()}")
```

```
2020-01-01 00:00:00
2022-06-01 00:00:00
The earliest month in the appointments_regional dataset is 2020-01-01 00:00:00
The latest month in the appointments_regional dataset is 2022-06-01 00:00:00
```

Figure 6. The min() / max() functions applied to the datasets

For the above and also to filter the data – the to_datetime() function was used for the ease of filtering, then filtering applied on the date and the lambda function was applied for the specified variable, and then groupby() to isolate the service setting (***Figure 7***); (***Section 3.2 on Jupiter Notebook***).

```
# For each of these service settings, determine the number of records available for the period and the location.
# ['sub_icb_location_name' == 'NHS North West London ICB - W2U3Z']

# Create a DataFrame.
data = pd.DataFrame(nc)

# Change the date format.
data['appointment_date'] = pd.to_datetime(data['appointment_date'])

# Apply date filter between 01/01/2022 and 01/06/2022 (5 months) as stated on the assignment.
nc_date_filter = data[(data['appointment_date'] >= "2022-01-01")
                      & (data['appointment_date'] < "2022-06-01")]

# Use lambda function to filter out all the locations except NW London.
nc_nwl_lambda = nc_date_filter[nc_date_filter['sub_icb_location_name'].apply(lambda x:
                                                                             'NHS North West London ICB - W2U3Z' in x)]

# Group the result by service setting.
nc_subset = nc_nwl_lambda.groupby(['sub_icb_location_name', 'service_setting'])[['count_of_appointments']].sum() \
    .sort_values('count_of_appointments', ascending=False)
print(nc_subset)
print(nc_subset.sum())
```

```
                                                    count_of_appointments
sub_icb_location_name            service_setting
NHS North West London ICB - W2U3Z General Practice                4760966
                                 Unmapped                          387939
                                 Other                             151616
                                 Primary Care Network              108901
                                 Extended Access Provision          97409
count_of_appointments    5506831
dtype: int64
```

Figure 7. Demonstration of the to_datetime(), lambda, groupby() functions for filtering

After having looked at the total level, splitting out the data by month was the next activity. For doing this groupby() function with the dictionary to map() the month numbers to the month names was applied to generate the number of appointments in descending order, the same principle was also applied for the count of the total records (***Figure 8***); (***Section 3.3 on Jupiter Notebook***).

```
# Number of appointments per month == sum of count_of_appointments by month.
# Use the groupby() and sort_values() functions.
# Apply dictionary to map the month numbers to the month names.
# Sort by the highest number of appointments.
nc_group = nc.groupby([nc['appointment_date'].dt.year, nc['appointment_date'].dt.month.map
                       ({1: 'January', 2: 'February', 3: 'March', 4: 'April',
                         5: 'May', 6: 'June', 7: 'July', 8: 'August', 9: 'September',
                         10: 'October', 11: 'November', 12: 'December'})
                       ])[['count_of_appointments']].sum()\
    .sort_values('count_of_appointments', ascending=False)

print(nc_group)
print(nc_group.sum())
```

```
                                  count_of_appointments
appointment_date appointment_date
2021             November                      30405070
                 October                       30303834
2022             March                         29595038
2021             September                     28522501
2022             May                           27495508
                 June                          25828078
                 January                       25635474
                 February                      25355260
2021             December                      25140776
2022             April                         23913060
2021             August                        23852171
count_of_appointments    296046770
dtype: int64
```

Figure 8. Monthly number of appointments grouped using the dictionary via map() function.

4 Data Exploration via the use of Visualisations

To Further deep dive into the datasets we would start the visualisations journey. The datasets have been prepared and ready using the knowledge acquired in prior sections (***Section 4.1 on Jupiter Notebook***). The next step is to plot the countplot via the Seaborn library and format it to be user friendly (***Figure 9, Figure 10***).

```
# View monthly number of records.
sns.set(font_scale=1.5)
monthly_bar = plt.figure(figsize=(15, 12))
monthly_bar = sns.countplot(x='appointment_month', data=nc)

# Set legend and title.
monthly_bar.set_xlabel('Months')
monthly_bar.set_ylabel('No of Records')
monthly_bar.set_title('Number of Records by Month')
monthly_bar.bar_label(monthly_bar.containers[0])

# View output.
plt.show()
```

Figure 9. Plotting the countplot and formatting

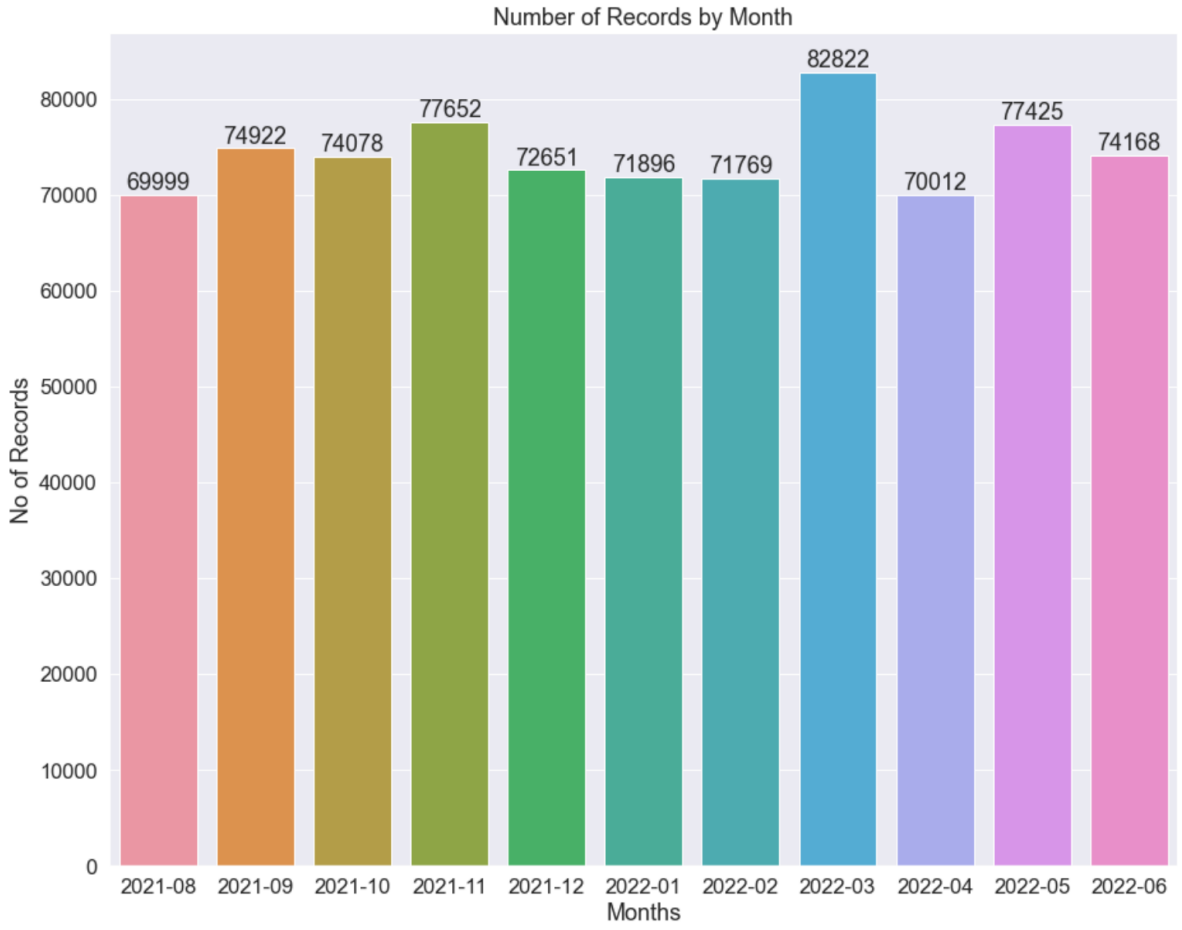


Figure 10. Countplot visual representation

Then further analysis (***Section 4.2 on Jupiter Notebook***) is required to deep dive into the slicing of data by Service Settings, Context Types and National Categories by creating DataFrame subsets grouped for each and then plotting them onto the lineplot via a sum().reset_index() on groupby() function (***Table 6, Figure 11***) as an example. After Having viewed the output the decision was made to save the subsets for each into the csv files for future use (**Figure 12**). Further formatting example used to not over clutter the lineplot by using setp() and legend() functions to move the legend outside of the lineplot (***Figure 13***).

Table 6. Example of the data preparation (DataFrame) to be plotted on the Line Plot for Service Settings

Service settings:

```
# Plot the appointments over the available date range, and review the service settings for months.
data = pd.DataFrame(nc)

nc_ss = data.groupby(['appointment_month', 'service_setting'])[
    ['count_of_appointments']].sum().reset_index()

# View output
print(nc_ss['service_setting'].nunique())
nc_ss.head()
```

```
5
```

	appointment_month	service_setting	count_of_appointments
0	2021-08	Extended Access Provision	160927
1	2021-08	General Practice	21575852
2	2021-08	Other	449101
3	2021-08	Primary Care Network	432448
4	2021-08	Unmapped	1233843

```
# Create a lineplot.
sns.set(font_scale=1.5)
nc_ss_line = plt.figure(figsize=(15, 12))
nc_ss_line = sns.lineplot(x='appointment_month', y='count_of_appointments', data=nc_ss,
                          hue='service_setting', markers=True, ci=None)

# Set legend and title.
nc_ss_line.set_xlabel('Months')
nc_ss_line.set_ylabel('Sum of Service Settings')
nc_ss_line.set_title('Service Settings by Month')
```

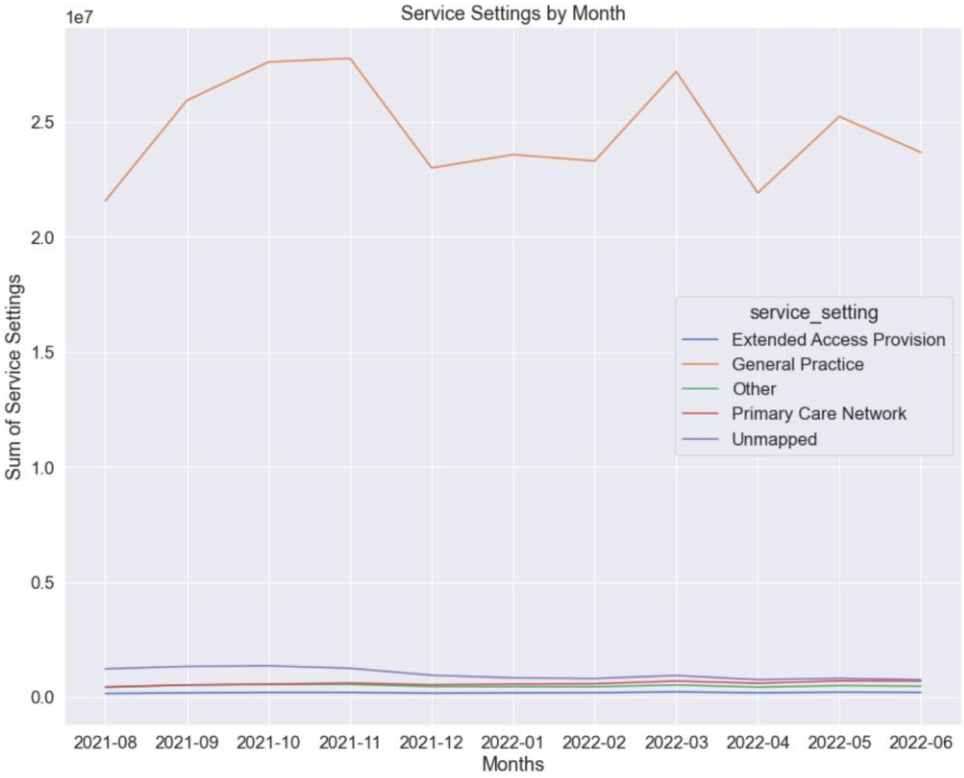


Figure 11. Lineplot for the Service Settings by Month

```
# Create a csv file to save the dataset for the future use if required.
nc_ss.to_csv('nc_ss.csv', index=False)
```

Figure 12. Example of the script to save a csv file.

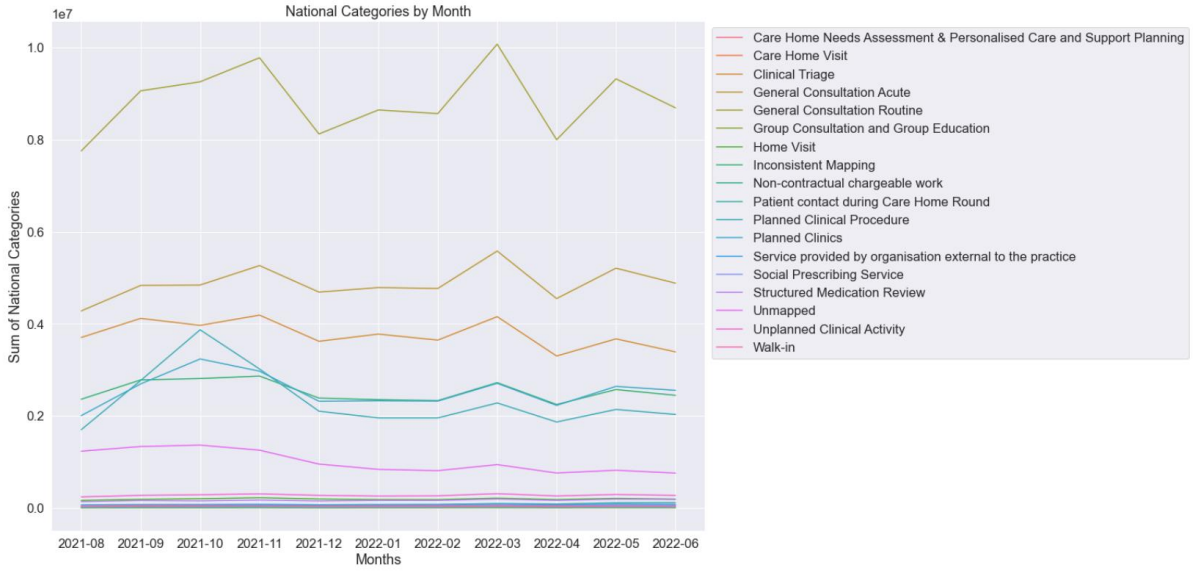


Figure 13. Reformatting the lineplot

The next objective was to run the seasonal patterns for the Service Settings and a similar process had been employed as evident on (***Figures 11, 12***) above, but isolated purely for the service setting sub-set (***Table 7***). The same principles have been applied to isolate the data, save the files and run visualisations for the Context Types and National Categories (***Section 4.3 on Jupiter Notebook***).

Table 7. Service Setting dataset isolated for further analysis and a copy saved as csv file

```
# Create a separate data set that can be used in future weeks.
nc_ss_day = nc.groupby(['appointment_date', 'appointment_month', 'service_setting'])[
    ['count_of_appointments']].sum().reset_index()
# View output.
nc_ss_day.head()
```

	appointment_date	appointment_month	service_setting	count_of_appointments
0	2021-08-01	2021-08	Extended Access Provision	438
1	2021-08-01	2021-08	General Practice	3411
2	2021-08-01	2021-08	Other	401
3	2021-08-01	2021-08	Primary Care Network	323
4	2021-08-01	2021-08	Unmapped	1054

```
# Create a csv file to save the dataset for the future use if required.
nc_ss_day.to_csv('nc_ss_day.csv', index=False)
```

5 Twitter

When working with the **twitter.csv** dataset several functions are not required as the data is already available and there is no need for data scraping online. However, for the illustrative purposes the BeautifulSoup had been imported and other basic data wrangling activities took place. The analysis of the (***Figure 14***) is referenced in the Appendix 2.

```
# Would it be useful to only look at retweeted and favourite tweet messages?
tweets['tweet_retweet_count'].value_counts().tail(10)
# The output means how many times (left) the number of tweets (right) has been retweeted.
```

```
14     1
79     1
20     1
39     1
19     1
303    1
57     1
40     1
54     1
169    1
Name: tweet_retweet_count, dtype: int64
```

```
# Would it be useful to only look at retweeted and favourite tweet messages?
tweets['tweet_favorite_count'].value_counts().tail(10)
# The output means how many times (left) the number of tweets (right) has been set as favourite.
```

```
8     1
13    1
11    1
7     1
20    1
28    1
14    1
18    1
9     1
42    1
Name: tweet_favorite_count, dtype: int64
```

Figure 14. The top ten tweets by the number of times retweeted and favourited

In order to try and generate meaningful insights the *tweet_full_text* column needs to be separated and stripped out of the noise and the hashtags need to be calculated to understand the volume of interests in the healthcare industry (***Section 5.2 on Jupiter Notebook***). After having used the code provided by the team to isolate the entries with the hashtags '#', and displaying the top 30 records, the next step was to properly format the dataset (***Figure 15***).

```
# Create a new DataFrame containing only the text.
tweets_base = pd.DataFrame(tweets)
tweets_text = tweets_base[['tweet_full_text']]
# View the DataFrame.
tweets_text.head()
```

```
# Display the first 30 records.
tags30 = pd.Series(tags).value_counts()
tags30.iloc[:30]
```

```
# Convert the series to a DataFrame in preparation for visualisation.
data = pd.DataFrame(tags30).reset_index()
display(data)
# Rename the columns.
dict = {'index': 'word', 0: 'count'}
data.rename(columns=dict, inplace=True)
display(data)
```

```
# Fix the count datatype.
data['count'] = data['count'].astype('int64')

# View the result.
data.head()
```

Figure 15. Formatting Twitter Text to run meaningful analysis

In order to clean up the visual it had been decided to limit the number of entries to the Top 20 using a horizontal barplot with clear evidence for ***#healthcare*** being in the commanding lead (***Figure 16***).

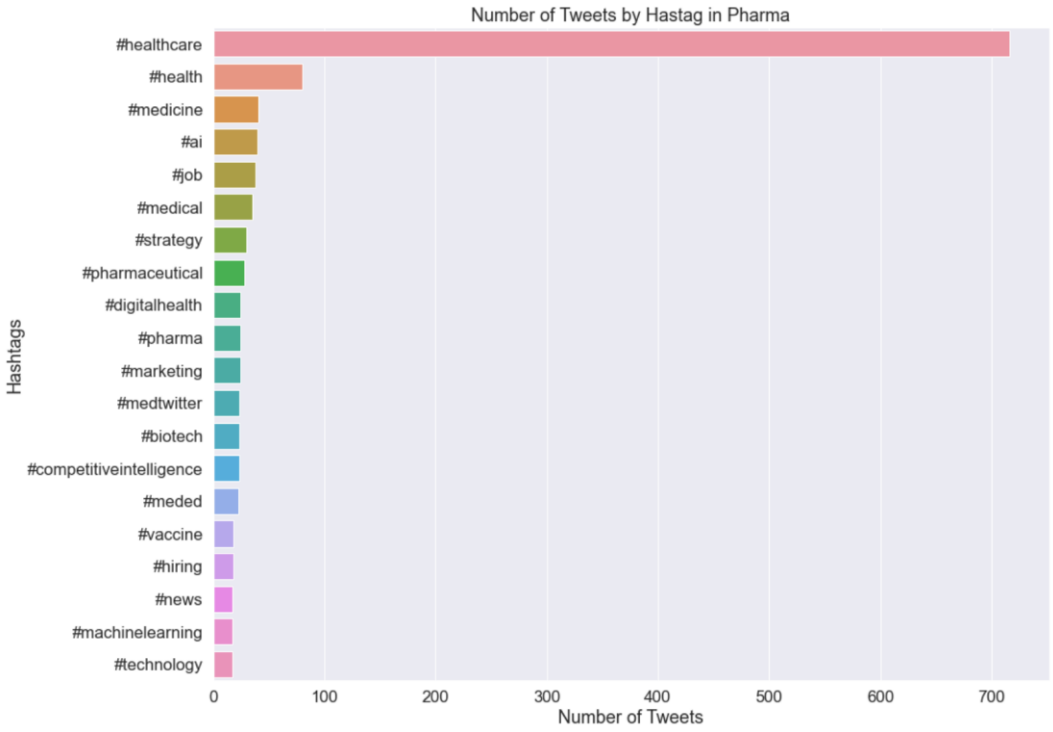


Figure 16. Volume of hashtags occurrence on Twitter

6 Further Deep Dive into the Data

After having run a detailed analysis on the **national_categories.xlsx** and having understood that the **actual_duration.csv** has got a very limited application and would not yield substantial added value for the analysis at hand, it had been decided to move focus back to the **appointments_regional.csv** and focus on the time horizon from August 2021 onwards (***Section 6.1 on Jupiter Notebook***). This includes the standard min() / max() date analysis, the filtering of the dataset and then creating an ar_agg subset of data with the relevant columns and variables, as well as further subsets to isolate General Practice, and then DNA (Did Not Attend).

Table 8. Treating the Appointments Regional file to fit the profile for further data exploration

```
# Print the min and max dates.
print(ar['appointment_month'].min())
print(ar['appointment_month'].max())
print(
    f"The earliest month in the appointments_regional dataset is {ar['appointment_month'].min()}")
print(
    f"The latest month in the appointments_regional dataset is {ar['appointment_month'].max()}")
```

```
2020-01
2022-06
The earliest month in the appointments_regional dataset is 2020-01
The latest month in the appointments_regional dataset is 2022-06
```

```
# Filter the data set to only look at data from 2021-08 onwards.
ar_month_filter = ar[ar['appointment_month'] >= "2021-08"]
ar_month_filter.head()
```

	icb_ons_code	appointment_month	appointment_status	hcp_type	appointment_mode	time_between_book_and_appointment	count_of_appointments
3652	E54000034	2021-08	Attended	GP	Face-to-Face	1 Day	6553
3653	E54000034	2021-08	Attended	GP	Face-to-Face	15 to 21 Days	2390
3654	E54000034	2021-08	Attended	GP	Face-to-Face	2 to 7 Days	10547
3655	E54000034	2021-08	Attended	GP	Face-to-Face	22 to 28 Days	937
3656	E54000034	2021-08	Attended	GP	Face-to-Face	8 to 14 Days	4961

Question 1: Should the NHS start looking at increasing staff levels?

```
# Create an aggregated data set to review the different features.
ar_agg = ar_month_filter.groupby(['appointment_month', 'hcp_type', 'appointment_status', 'appointment_mode',
                                  'time_between_book_and_appointment']).sum().reset_index()

# View the DataFrame.
ar_agg
```

	appointment_month	hcp_type	appointment_status	appointment_mode	time_between_book_and_appointment	count_of_appointments
0	2021-08	GP	Attended	Face-to-Face	1 Day	507835
1	2021-08	GP	Attended	Face-to-Face	15 to 21 Days	194726

Extra columns need to be created to understand the average daily number of appointments and use the result to apply to the final capacity calculation as stated by the client (***Table 9***). The function to build the output on (***Table 9***) is to use groupby(), then separate out only the columns that we are interested in via the DataFrame() command, after that conduct the calculations, and making sure the output is in the desired format to conduct further analysis (int, float).

Table 9. The script and the table showing the capacity utilisation based on the client requirement and supporting info

```
# Determine the total number of appointments per month.
data = pd.DataFrame(ar_agg)
ar_df = data.groupby(['appointment_month'])[
    ['count_of_appointments']].sum().reset_index()
ar_df.head()

# Add a new column to indicate the average utilisation of services.
# Monthly aggregate / 30 to get to a daily average by month.
ar_df = pd.DataFrame(ar_df, columns=[
                    'appointment_month', 'count_of_appointments', 'avg_daily_appt', 'utilisation'])
ar_df['avg_daily_appt'] = ar_df['count_of_appointments'] / 30
ar_df['avg_daily_appt'] = ar_df['avg_daily_appt'].astype('int64')

# Utilisation of the NHS 1,200,000 maximum daily capacity
ar_df['utilisation'] = round(ar_df['avg_daily_appt'] / 1200000, 1)
ar_df['utilisation'] = ar_df['utilisation'].astype('float')

# View the DataFrame.
ar_df.head()
```

	appointment_month	count_of_appointments
0	2021-08	23852171
1	2021-09	28522501
2	2021-10	30303834
3	2021-11	30405070
4	2021-12	25140776

	appointment_month	count_of_appointments	avg_daily_appt	utilisation
0	2021-08	23852171	795072	0.7
1	2021-09	28522501	950750	0.8
2	2021-10	30303834	1010127	0.8
3	2021-11	30405070	1013502	0.8
4	2021-12	25140776	838025	0.7

Following the achievement above the proportional utilisation had then been plotted onto the line plot to try and understand if the capacity utilisation was near its limits or not (***Figure 17***).

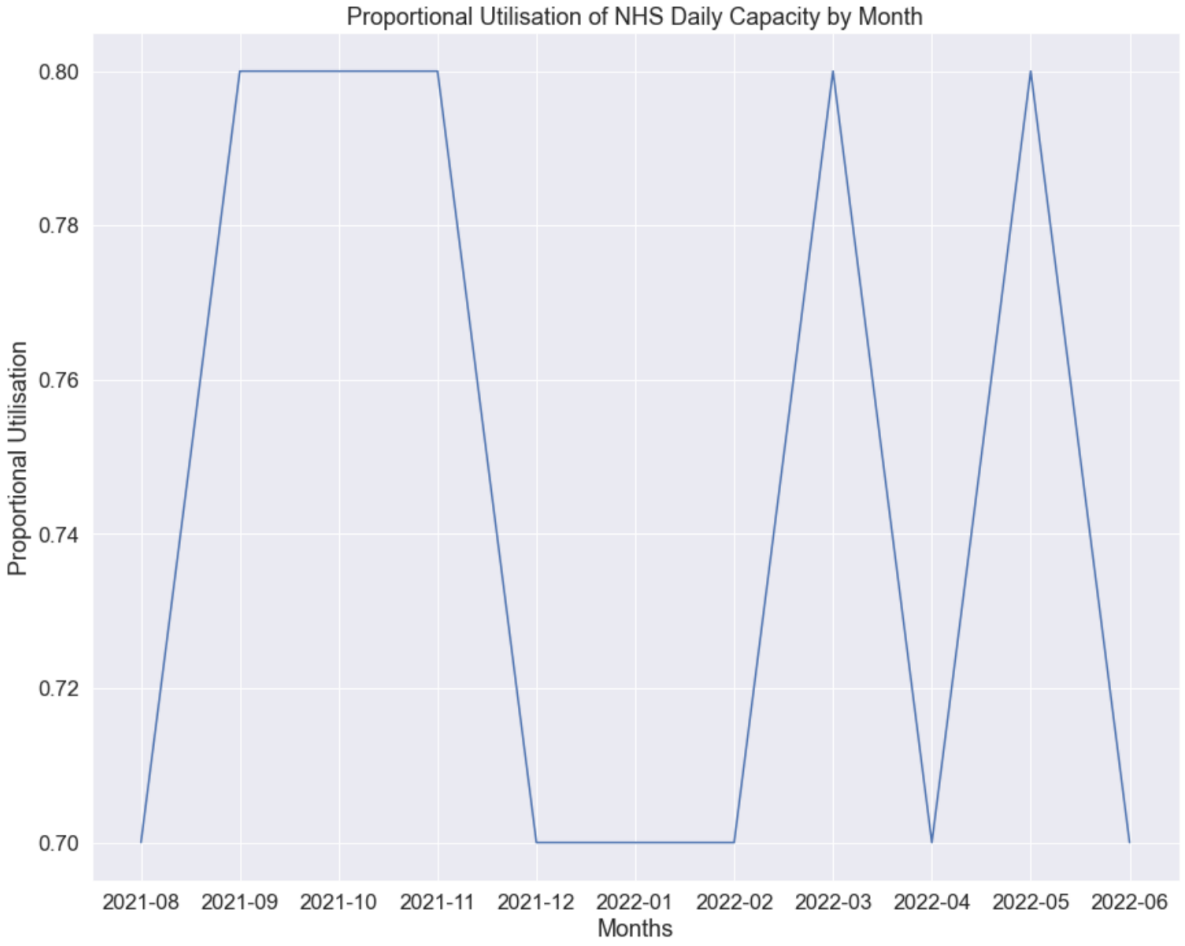
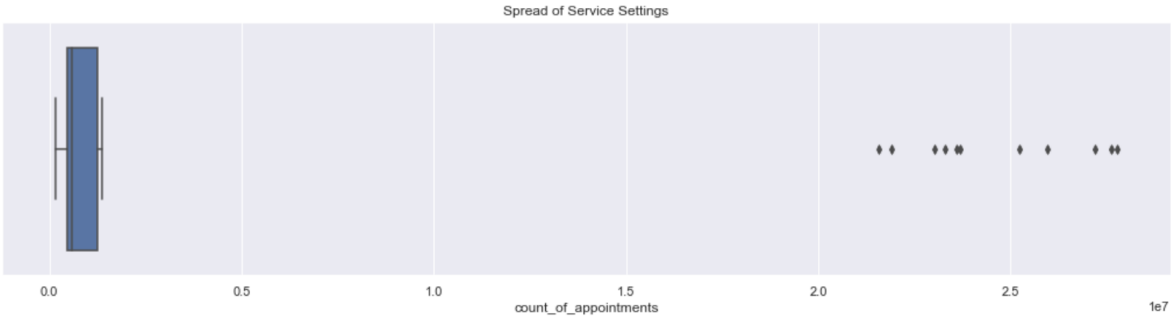


Figure 17. Proportional Utilisation of NHS Daily Capacity by Month

Further investigation led to analysing box plots for the subsets with and without the GP as a variable (***Figure 18***).

```
# Set figure size.
fig1 = plt.figure(figsize=(18, 4))

# Create a boxplot.
nc_ss_box = sns.boxplot(
    x=nc_appointments['count_of_appointments'], whis=1.5).set(title='Spread of Service Settings')
```




```
# Create a boxplot to investigate the service settings without GP.
nc_appointments_exgp = nc_appointments[nc_appointments['service_setting']
                                       != 'General Practice']

# Set figure size.
fig2 = plt.figure(figsize=(18, 4))

# Create a boxplot.
nc_ss_box_2 = sns.boxplot(
    x=nc_appointments_exgp['count_of_appointments'], whis=1.5).set(title='Spread of Service Settings ex GP')
```

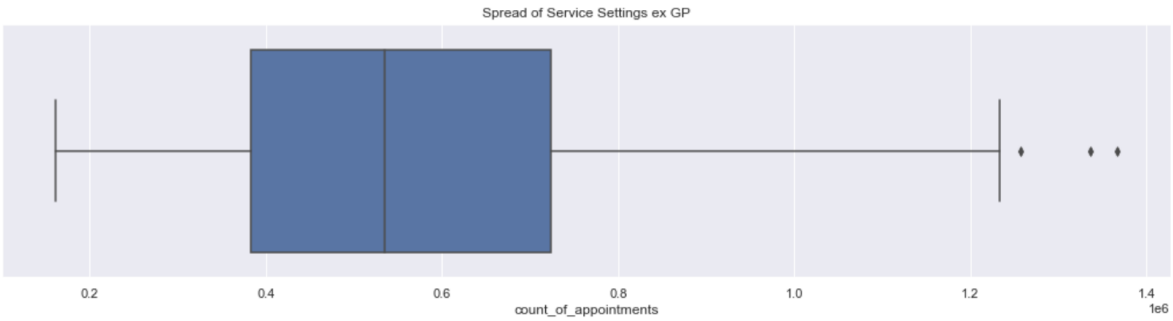


Figure 18. Box plot demonstration comparing the whole data set and once removing GP

7 Answering the NHS Questions

Please refer to the enclosed PDF version of the PowerPoint Presentation or go to Appendix 3 of this report.

Please also refer to the Section 7 of the Jupiter Notebook enclosed.

Technical Analysis Conclusions

All the required libraries (Pandas/Seaborn) and some optional ones (NumPy), as well as supporting functions have been used to aid with the analysis.

Data exploration functions such as isna(), info(), describe(), head(), min()/max() have been used to sense check the data loaded correctly.

Data wrangling functions including pd.DataFrame(), groupby(), sum(), reset_index() have been used extensively to treat the data and get it ready for further analysis.

Dictionaries and date filtering techniques tested by sub-setting data, using iloc[] function and nunique().

Formatting via to_datetime() and astype() implemented to treat the data.

Lambda function had been used to demonstrate the filtering capabilities using this method.

Data visualisation techniques using Seaborn and Matplotlib used extensively to create countplots, barplots, lineplots, boxplots to aid with the analysis.

Formatting for the visualisations used to set the font, background, as well as format axis and apply more readable legend and naming conventions.

Several subsets have been saved as csv files for future use for quick access to avoid importing the base files up to 817K lines and weight up to 38MB, which requires extra processing power and time.

A support reference file had been created to aid with aggregations at *icb_ons_code – _icb_ons_name*, which was not available and is one level above the *sub_icb_ons* name/code.

The detailed outcomes of the capacity utilisation analysis and specifically general practice deep dive would be available within the Appendix 3 and followed up on within the PPT presentation.

Appendix 1 (Technical)

The result can clearly show that the vast majority of the appointments is related to General Practice. Second largest is the “Unmapped” – the data quality issue (***Figure 7***).

Further analysis within **Section 6** led to splitting out the HCP types, appointment statuses, appointment modes, times between bookings/appointments and plotting them on the line plots following similar principles of data sub-setting as before and saving the sub-set files for future use.

Appendix 2 (Client Requirements)

Even though two of top 5 sub locations are in London, the highest number of transactions is in fact registered in Greater Manchester with 59K transactions (***Figure 3***).

The evidence suggests that 14.3% of all the lines on **national_categories.xlsx** are not mapped (117K records out of 817K) (***Figure 4***). The more problematic issue can be identified when looking at **appointments_regional.csv** for attended appointments where for 33.7% the status is unknown and 27.3% did not attend (***Figure 5***).

It can clearly be seen on ***Figure 6*** that the dates ranges are different which the client should be aware of, so in the further analysis the data would be filtered for the required specification (***Section 3.1 on Jupiter Notebook***).

Having investigated the *tweet_retweet_count* and *tweet_favorite_count* not much meaningful information came out of the analysis as there is insufficient detail to work with, so further information would be required (***Figure 14***) (***Section 5.1 on Jupiter Notebook***).

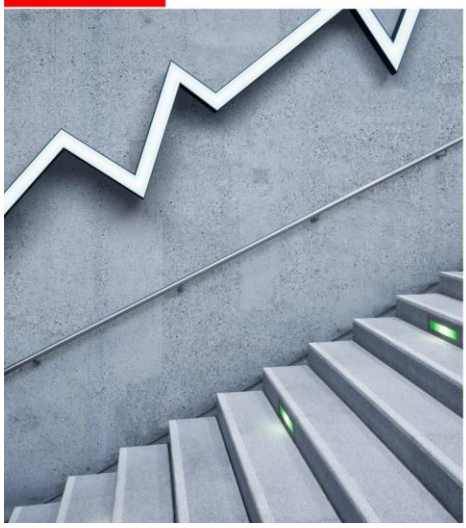
The client had requested to check the capacity utilisation based on the known information of 1,200,000 appointments per day capacity availability within the network (***Table 9***).

It can clearly be seen that, as requested by the client, the rounding to 1 decimal place is too broad, and suggestion would be to round the capacity to 2 decimal places (***Figure 17***).

The additional limitation is that the criteria for the capacity utilisation is too broad, because it does not take into account regional capabilities, various remote areas and national categories, as well as staff availability at a given practice (***Figure 17***). It is also quite difficult to estimate what the capacity would look like based on the various factors like appointment mode, or service settings and context types, because it had been evidenced throughout the analysis that the data is missing some values.

Appendix 3 (PPT, Section 7 of Jupiter Notebook)

THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE



NHS Case Study Analysis and Interim Conclusions

VADIMS SUHAREVS

24/10/2022

Objectives of the Analysis:

Investigate whether the staff capacity is adequate in the network

Actual utilization of resources

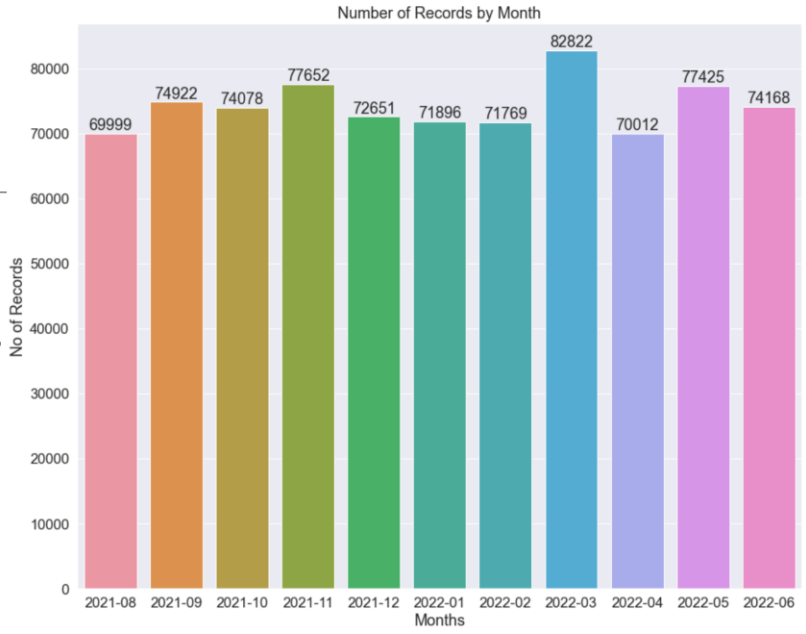
Investigate reasons behind missed appointments within General Practice and where the problem occurs

Twitter Exploration

Analysis of the social media trends related to NHS suggests there are no adverse hashtags evident and the top two trending hashtags are #healthcare and #health by a considerable margin.

Overall number of records per month (NC)

A relatively stable picture throughout the year with minor spikes in November, March and May



Total Monthly Appointments

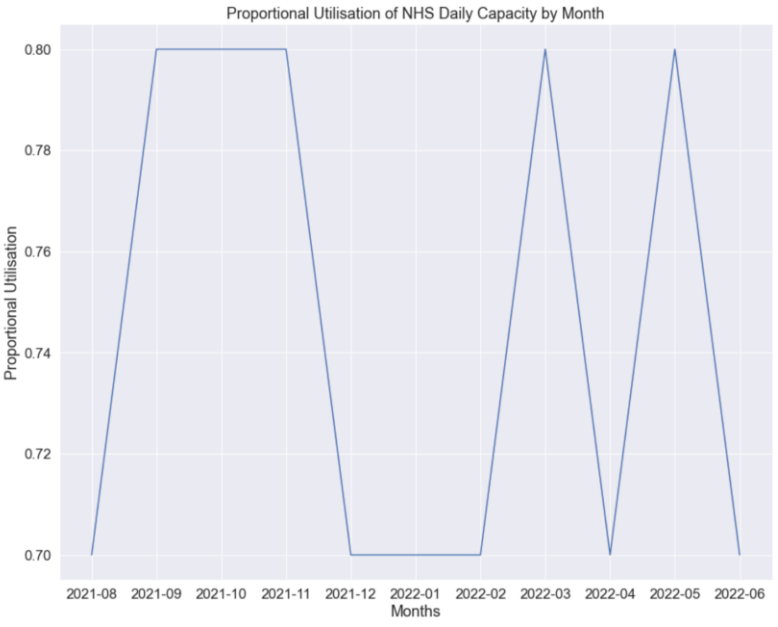
Peak is identified between October and November with more than 3 million monthly appointments.

During other months appointments are lower than 3 million.

Proportional Monthly Utilisations

During the Peak identified (October-November) the capacity stays well under the 100% critical level of total utilization.

Other peaks evident are September, March and May.



Exploring Service Settings for GP (National Categories)

No outliers identified within the dataset

Appointments for the GP Only (Regional)

```
Number of Records 2021-08 onwards 2022-06: 223418
Number of Records 2021-08 onwards 2022-06 GP Only: 84289
Attended    32992
Unknown     27585
DNA         23712
Name: appointment_status, dtype: int64
Face-to-Face    26612
Telephone       25852
Home Visit      14870
Video/Online    11152
Unknown          5803
Name: appointment_mode, dtype: int64
Same Day                 12901
2 to 7 Days              12772
1 Day                    12247
8  to 14 Days            11757
15  to 21 Days           10603
22  to 28 Days            9930
More than 28 Days         9286
Unknown / Data Quality    4793
Name: time_between_book_and_appointment, dtype: int64
```

84,3K transactions

148.4 million appointments

50.1% of the total appointments (38% transactions) for the period

Appointments by status (GP)

2.6% DNA

(28.1% Transactions)

2.9% Unknown

(32.7% Transactions)

Appointments by mode (GP)

50.7% Face-to-Face

(31.6% Transactions)

47.3% Telephone

(30.7% Transactions)

1% Unkown

(7% Transactions)

Appointments by Book Between (GP)

57% Same Date Appointments

(15% Transactions)

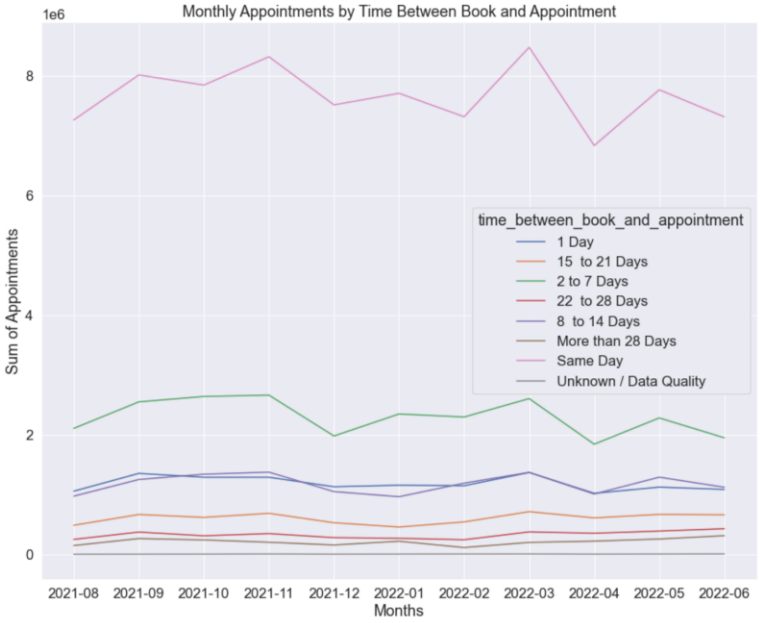
17% 2-7 Days Appointments

(15% Transactions)

Unknown:

0.1% Appointments

(5.7% of Transactions)



Appointments by ICB location (GP)

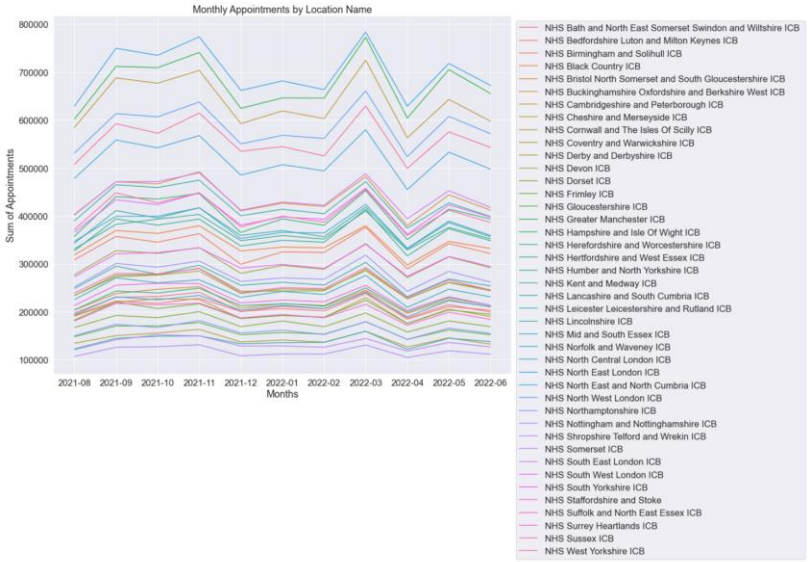
Transactions:

9% Manchester

8% Cheshire

7% North Cumbria

6% South Cumbria



Bookings not attended (GP only)

```
Number of Records 2021-08 onwards 2022-06 GP DNA Only: 23712
Face-to-Face     8492
Telephone        8148
Home Visit       2779
Video/Online     2550
Unknown          1743
Name: appointment_mode, dtype: int64
Same Day                  3945
2 to 7 Days               3827
1 Day                     3605
8  to 14 Days             3352
15  to 21 Days            3022
22  to 28 Days            2806
More than 28 Days         2645
Unknown / Data Quality     510
Name: time_between_book_and_appointment, dtype: int64
```

23,7K Transactions

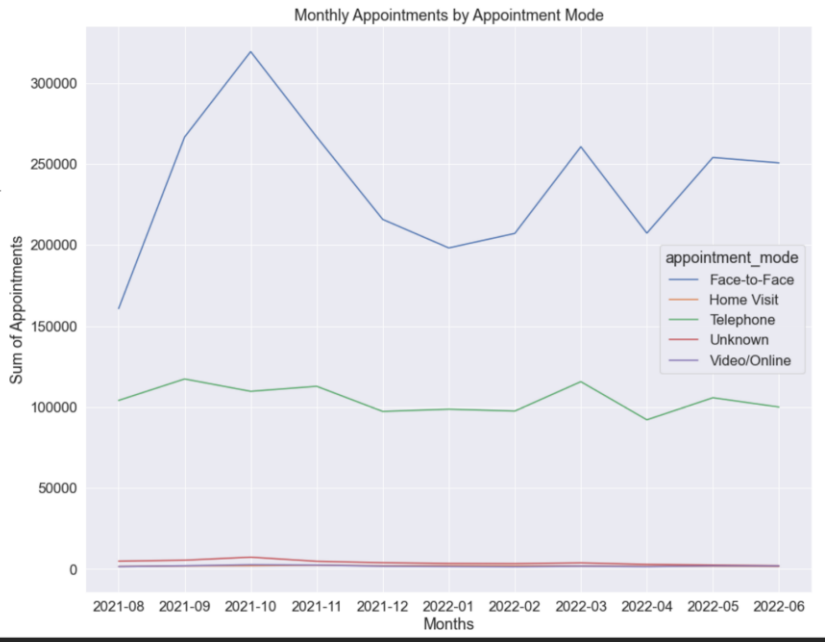
3.8 million appointments

1.3% of the total appointments for the period

Bookings not attended (Mode)

67.8% Face-to-Face

29.9% Telephone

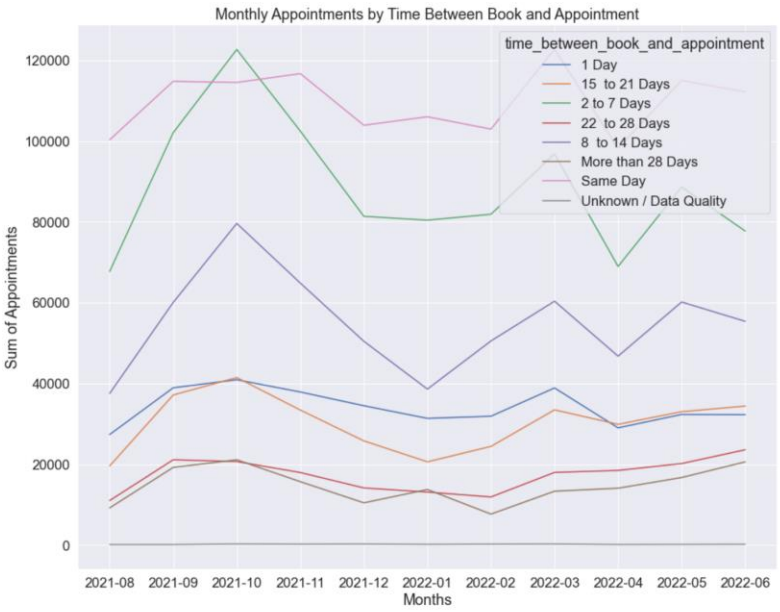


Bookings not attended (Book between)

31% Same Day

25% 2-7 Days

16% 8-14 Days



Top 10 ICB locations with missing appointments

Transactions:

Manchester 59K

Cheshire and Merseyside 54K

Others on the list are not in top 5 by transactions volume.

icb_location_name	count_of_appointments
NHS Greater Manchester ICB	279067
NHS Cheshire and Merseyside ICB	226688
NHS North East London ICB	195552
NHS North West London ICB	184873
NHS South East London ICB	178564
NHS North East and North Cumbria ICB	173869
NHS South West London ICB	162888
NHS North Central London ICB	149130
NHS Kent and Medway ICB	133973
NHS Hampshire and Isle Of Wight ICB	123769

Conclusions

No adverse comments on social media (Twitter) trending, so the complaints are taken care of, assuming that the twitter dataset corresponds to the same time horizon.

During the peak the capacity stays under the critical level of total network utilization, but further information would be required if more granular level is to be investigated

Out of the total number of general practice appointments – 2.6% did not attend, but this forms 28.1% of the transactions

3.8 million appointments not attended which forms 1.3% of the total appointments for the period and seems small, but in the absolute terms catering for 3.8 million failed appointments would be costly

The disparity between the appointment request and the short notice availability needs to be assessed as 56% of the appointment are missed when placed up to a week prior to the appointment.

30% of all the GP transactions are registered with 4 locations (42 locations in total)

List of the Top 10 locations where the problem with missing appointments persists provided to the NHS. Focus should be given to the Greater Manchester ICB, Cheshire and Merseyside, and London branches should be carefully monitored.

The datasets isolating GP and Unattended Appointments are available for download and upon request

Recommendations

The current analytics research is an interim analysis answering the question “where the problem exists?”

The next phase for the NHS would be to get back into its network and start asking questions “why it is happening?”

To answer the question how to reduce avoidable costs and reduce/eliminate unattended appointments the NHS need to provide the deeper and more granular data from within its value chain to the project team to start the next phase of the analysis.

Appendix 4 (Speech)

1

Hello, my name is Vadim and I would like to walk you through the NHS Case study analysis and interim conclusions. You will understand why interim by the end of this presentation.

2

The NHS have instructed the team that there is a requirement to conduct an analysis into its costs stating that the NHS incurs significant costs for patients, who do not attend their appointments. These extra costs could potentially be avoided by a reduction or elimination of the missed appointments.

Areas of analysis to consider are the staff capacity in the network and actual utilisation of resources.

3

The investigation started with the social media trends related to NHS and it suggests there are no adverse/negative hashtags evident and the top two trending hashtags are #healthcare and #health by a considerable margin. Having investigated the tweet_retweet_count and tweet_favorite_count not much meaningful information came out of the analysis as there is insufficient detail to work with, so further information would be required.

4

Next the overall number of records had been examined by month. A relatively stable picture throughout the year with minor spikes in November, March and May.

5

When plotting the total number of appointments on the same period – the peak is confirmed and identified between October and November with More than 3 million monthly appointments. During other months appointments are lower than 3 million.

6

The client had requested to check the capacity utilisation based on the known information of 1,200,000 appointments per day capacity availability.

During the identified peak the capacity stays well under the 100% critical level of total utilization. Other peaks are also evident for March and May. It can clearly be seen that, as requested by the client, the rounding to 1 decimal place is too broad, and suggestion would be to round the capacity to 2 decimal places to add extra granularity.

The additional limitation here is that the criteria for the capacity utilisation is too broad, because it does not take into account regional capabilities, various remote areas and national categories, as well as staff availability at a given practice. It is also quite difficult to estimate what the capacity would look like based on the various factors like appointment mode, or service settings and context types as the capacity would vary by region and the type of surgery.

7

The NHS have asked to investigate the detail behind the general practitioner appointments profile. We start with looking at the service setting on national categories dataset which shows that there are no outliers and the data are with the acceptable limits.

8

Then we look more closely at regional appointments for general practice only. It can be evidenced that it contains 84,3K transactions which make up 148.4 million appointments and comprise 50.1% of the total appointments and 38% transactions for the period.

9

Based on the prior information we start digging deeper into the general practice. We can see that 2.6% of the appointments were not attended, 28.1% transactions and unknown status is with the 2.9% of the appointments, 32.7% transactions. We will come back to this later as we are interested in the DNA cohort as the unknown at this stage is a data quality issue and cannot be assumed that it is related to the patients who did not attend their appointment. Throughout the analysis we can already see some of the data is missing some values and inconsistent.

10

When we look at the appointments by mode - 50.7% are Face-to-Face or 31.6% of Transactions, 47.3% by Telephone or 30.7% Transactions. 7% transactions are unknown, confirming that the data lacks quality or there are issues with collecting data.

11

Based on the appointments booking schedule 57% are Same Day appointments, 15% of Transactions and 17% are 2-7 days pre-registered, 15% Transactions. 0.1% of appointments make up 5.7% of unknown transactions, which can be deemed as a flawed data management practice.

12

If we investigate the ICB locations transactions, the Top 4 locations amount to 30% of transactions. The list is on the screen with Manchester leading the way.

13

As mentioned earlier, we then go back to the transactions for bookings that were not attended. We have got the drill down of the records on the left-hand side. The 23,7K Transactions make up 3.8 million appointments and are 1.3% of the total appointments for the period which clearly is a lot in the absolute terms.

14

We then dig deeper into the modes of unattended general practice appointments and can see that 67.8% are Face-to-Face, and 29.9% are Telephone.

15

Looking at the other dimension – at when the booking had been made - 31% Same Day missed appointments, 25% 2-7 Days and 16% 8-14 Days. This needs to be clearly looked at and the disparity between the appointment request and the same day and short notice, up to a week availability needs to be assessed.

16

Zooming out - the highest number of transactions overall (not only general practice) is placed by the Greater Manchester ICB – 59 thousand, as well as the highest number of missed appointments 279K are attributed to the same ICB. Cheshire and Merseyside second ones following on both accounts as well – 54K and 227K.

17

Based on the analysis undertaken the conclusions are as follows:

No adverse comments on social media (Twitter) trending, so the complaints are taken care of, assuming that the twitter dataset corresponds to the same time horizon.

During the peak the capacity stays under the critical level of total network utilization, but further information would be required if more granular level is to be investigated

Out of the total number of general practice appointments – 2.6% did not attend, but this forms 28.1% of the transactions

3.8 million appointments not attended which forms 1.3% of the total appointments for the period and seems small, but in the absolute terms catering for 3.8 million failed appointments would be costly

The disparity between the appointment request and the short notice availability needs to be assessed as 56% of the appointment are missed when placed up to a week prior to the appointment.

30% of all the GP transactions are registered with 4 locations (42 locations in total)

List of the Top 10 locations where the problem with missing appointments persists provided to the NHS. Focus should be given to the Greater Manchester ICB, Cheshire and Merseyside, and London branches should be carefully monitored.

The datasets isolating GP and Unattended Appointments are available for download and upon request

18

The current analytics research is an interim analysis answering the question "where the problem exists?"

The next phase for the NHS would be to get back into its network and start asking questions "why it is happening?"

To answer the question how to reduce avoidable costs and reduce/eliminate unattended appointments the NHS need to provide the deeper and more granular data from within its value chain to the project team to start the next phase of the analysis.