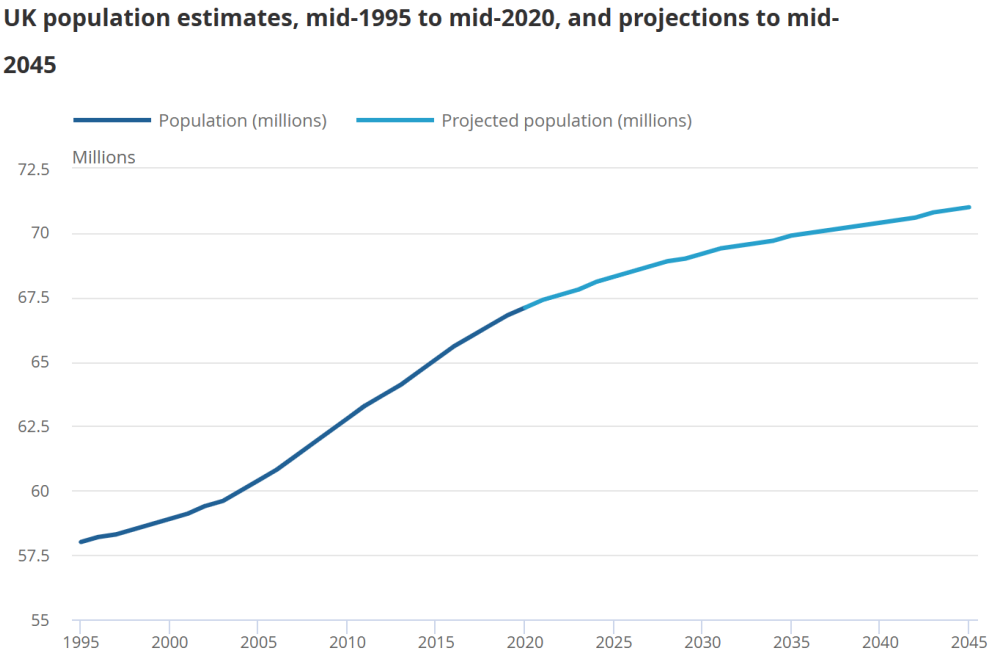
Utilisation and Capacity in GP Practices

Data Analysis by Alasdair Bell

Report Contents

1. Background

According to the Office for National Statistics, the UK's population is projected to increase by 3.2% to 69.2 million by mid-2030, with England growing at 3.5%. Additionally, the number of older people is expected to almost double by 2045.



National population projections - Office for National Statistics

Against the backdrop of a growing and ageing population, the NHS consistently faces issues concerning capacity and efficiency.

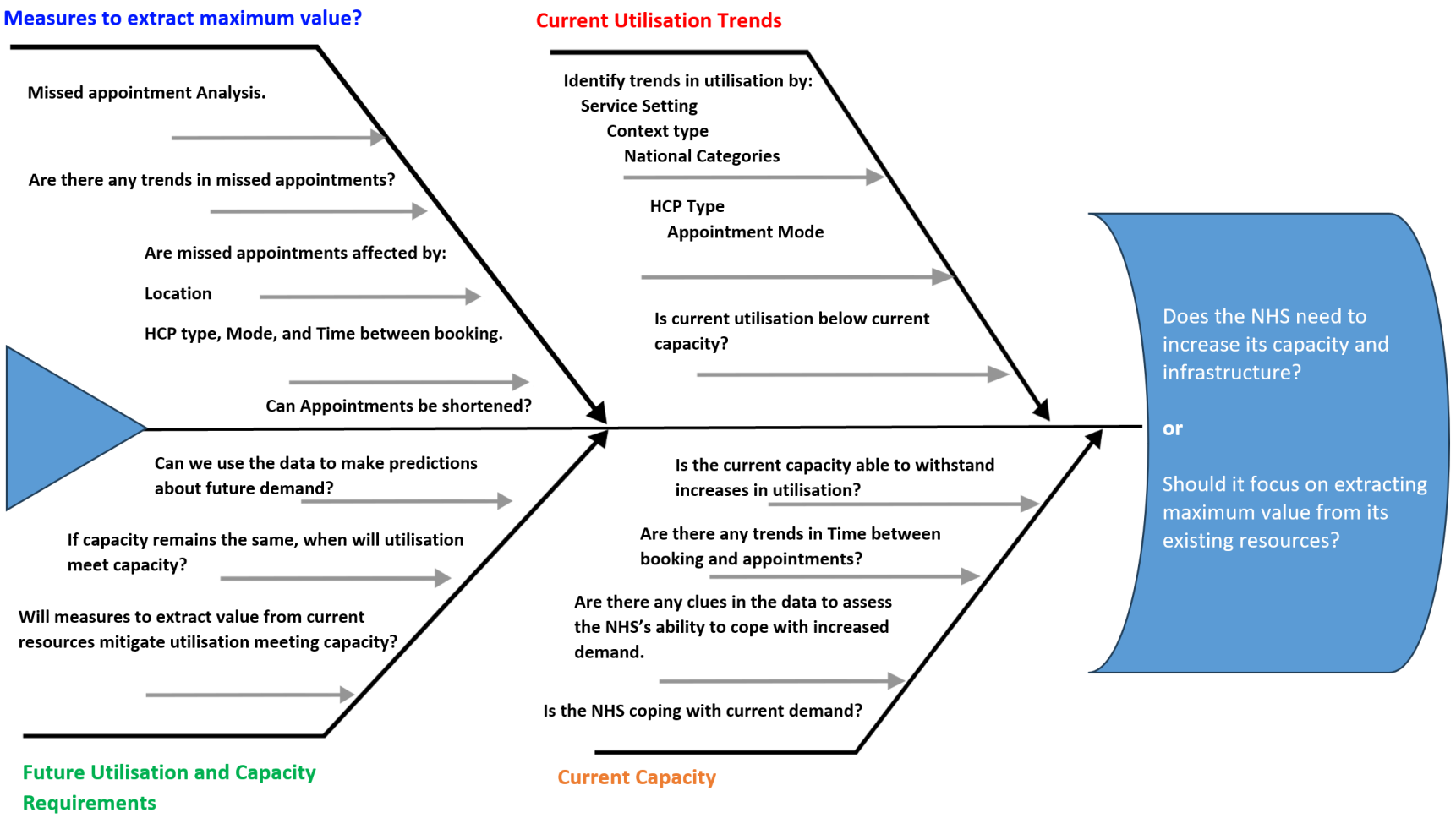
1.1 Organisational Problem

Given current trends in utilisation, does the NHS need to increase its capacity and infrastructure, or should it focus on extracting maximum value from its existing resources?

NHS

Diagnostic Analysis using Python.

From an Organisational Problem to a Data Problem



1.2 Scope of the Report and Limitations

The following report extracts the most pertinent and valuable insights from an exhaustive exploratory analysis in the accompanying Jupyter Notebook.

The data provided has several limitations:

Limitations	Assumptions
No financial data	Efficiency and cost effectiveness determined solely by analysis of appointments.
No Staffing data	Staffing requirements derived from appointment analysis e.g. time between booking and appointment.
The data contains multiple inconsistent, unmapped and unknown data.	The correctly apportioned data is sufficient for analysis and meaningful conclusions can be drawn.
The actual duration dataframe is limited to 7 months and has a materially significant unknown and data quality element.	The unknown data is comprised and proportioned broadly corressponding to the remaining durations.
Time between booking and appointment.	The author acknowledges delays may be at the patients request.
No data on hospitals and emergency services	General Practice is the focus of the study.
Seasonal Analysis restricted due to data timeframe	Monthly Analysis is sufficiently granular.
Primary data source Appointments Regional in monthly format only.	Monthly analysis is sufficient for all dataframes.
Twitter data search unspecified	The origin of the twitter data is unknown and therefore taken at face value.

1.3 Analytical Approach

A full analytical approach is contained in Appendix A.1

Section	Problem	Analytical Solution
Data Preparation	The data exists in CSV and Excel formats	Import data into Jupyter notebook using pandas and create dataframes.
	The data formats, types, shape and metadata is unknown.	Print a data dictionary using .shape() and .info() and .head() functions
	What are the characteristics of the dataframes.	Print descriptive statistics.
	The date is represented differently in each dataframe.	Assess the range, format and timespan of the dataframes and assimilate.
	The locations are represented differently in each dataframe.	Create an index of ONS codes and locations and map into each data frame so that a common means of comparison is available.
	Repetitive aggregations are required throughout the study requiring multiple dataframes, groupby elements, locations and date timeframes.	Create a multi functional aggregation function to reduce workload and repetition during the study that allows for filtering, flexibility and user inputs.
Appointment Analysis	The utilisation and appointment trends are unknown.	Use the multi function to thoroughly assess utilisation by grouping the data and apply linear regression to establish trends.
How effectively is the NHS coping with current Utilisation Trends	Isolate a measure of the NHS's current effectiveness.	Use this measure to assess whether the NHS is operating within its current capacity, adjust the measure to account for trends in utilisation.
Measures to extract value from current resources.	How can the NHS maximise the value of it's existing resources?	Analyse missed appointments and explore the potential to enact strategies to reduce missed appointments. Analyse the actual duration of appointments and assess whether the time is being used effectively.
Capacity and Utilisation of Resources	How close is the NHS to operating at full capacity?	Use the analysis of appointments to gauge whether the NHS is nearing the 1.2m appointments per day capacity
	Can the measures outlined above create more spare utilitity?	Assess the impact of reducing missed appointments on utilistation in the context of capacity.
	What impact will measures to extract existing value have on creating spare capacity?	Plot the difference between utilisation and actual utilisation when missed appointments have been reduced.
Patterns and Predictions	If the NHS can maximise it's existing resources, will this negate any requirement to increase capacity, given the trends in utilisation.	Assess the potential benefits of reducing missed appointments in the context of utilisation trends. Extrapolate the trends to assess potential convergence points of utilisation and capacity. Is this convergence point materially affected by measures to extract value form current resources.

2. Data Preparation

2.1 Importing Data and Basic Exploration

Relevant libraries were imported:

```
# Import the necessary libraries.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import FuncFormatter
```

The raw data was imported into Jupyter using the following code and aliases:

```
ad = pd.read_csv('actual_duration.csv')
ar = pd.read_csv('appointments_regional.csv')
nc = pd.read_excel('national_categories.xlsx')
```

Metadata for each dataframe was assessed constituting a data dictionary:

- Number of rows and columns (df.shape),
- Column names, null count, and data type (df.info()).
- The first 5 rows were displayed:

E.g.

```
# View the shape of the DataFrame. Review metadata incl columns, null values and data types. Sense check top 5 rows.

print(ar.shape)
print(ar.info())
ar.head()
```

```
(596821, 7)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 596821 entries, 0 to 596820
Data columns (total 7 columns):
 #   Column                            Non-Null Count   Dtype
---  ------                            --------------   -----
 0   icb_ons_code                      596821 non-null  object
 1   appointment_month                 596821 non-null  object
 2   appointment_status                596821 non-null  object
 3   hcp_type                          596821 non-null  object
 4   appointment_mode                  596821 non-null  object
 5   time_between_book_and_appointment  596821 non-null  object
 6   count_of_appointments             596821 non-null  int64
dtypes: int64(1), object(6)
memory usage: 31.9+ MB
None
```

	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

The data was assessed using the checklist (Appendix A.2).

2.2 Descriptive Statistics

The descriptive statistics were determined and printed side by side for comparison:

```
# Compare the descriptive statistics of the 3 dataframes side by side for easy comparison.

descriptive_stats = pd.concat([ad.describe(),
      ar.describe(),
      nc.describe()], axis=1).round(2)
descriptive_stats.columns = ['ad stats', 'ar stats', 'nc stats']
print(descriptive_stats)
```

```
        ad stats   ar stats   nc stats
count  137793.00  596821.00  817394.00
mean     1219.08    1244.60     362.18
std      1546.90    5856.89    1084.58
min         1.00       1.00       1.00
25%       194.00       7.00       7.00
50%       696.00      47.00      25.00
75%      1621.00     308.00     128.00
max     15400.00  211265.00   16590.00
```

Observations

The minimum and maximum values in each dataframe differ considerably as the appointments in each record are already aggregated. By using the count function, we assume these extremes to be equivalent ie they each count for 1 within the count function. Therefore, the analysis will be based solely on sum aggregations and any request for count of records will assume that record means appointment.

There is a significant difference in the mean of the NC dataframe as it has a larger number of records.

2.3 Date Assimilation, Range and Timespan

The data frames were assessed for date format, range and time span:
E.g.

Check date formats:

```
# View the first and last five rows of appointment_date for the ad DataFrame to determine the date format.

print(ad['appointment_date'].head())
print(ad['appointment_date'].tail())
```

```
0    01-Dec-21
1    01-Dec-21
2    01-Dec-21
3    01-Dec-21
4    01-Dec-21
Name: appointment_date, dtype: object
137788    30-Jun-22
137789    30-Jun-22
137790    30-Jun-22
137791    30-Jun-22
137792    30-Jun-22
Name: appointment_date, dtype: object
```

Formats were corrected:

```
# Change the date format of ad['appointment_date'].

ad['appointment_date'] = pd.to_datetime(ad['appointment_date'])

# Sense check date conversion. Tail as head could be incorrect ie 1 December or 12 January.
print(ad['appointment_date'].dtypes)
print(ad['appointment_date'].head())
print(ad['appointment_date'].tail())
```

```
datetime64[ns]
0   2021-12-01
1   2021-12-01
2   2021-12-01
3   2021-12-01
4   2021-12-01
Name: appointment_date, dtype: datetime64[ns]
137788   2022-06-30
137789   2022-06-30
137790   2022-06-30
137791   2022-06-30
137792   2022-06-30
Name: appointment_date, dtype: datetime64[ns]
```

Appointment_months added where necessary:

```
# Add a column to show appointment month

ad['appointment_month'] = ad['appointment_date']
ad['appointment_month'] = pd.to_datetime(ad['appointment_month']).dt.strftime('%Y-%m')
print(ad['appointment_month'].head())
```

```
0    2021-12
1    2021-12
2    2021-12
3    2021-12
4    2021-12
Name: appointment_month, dtype: object
```

The date ranges and timespans were determined:

```
# Determine the minimum and maximum dates in the ad DataFrame.

ad['appointment_date'] = pd.to_datetime(ad['appointment_date'])
ad_min_date = ad['appointment_date'].dt.date.min()
ad_max_date = ad['appointment_date'].dt.date.max()

# What is the timespan of the ad dataframe

time_span = np.timedelta64(ad_max_date - ad_min_date)

# Convert to days and months

days = time_span.astype('timedelta64[D]').astype(int)

months = days // 30

# Calculate remaining days
remaining_days = days % 30

print(f"The earliest date in the ad dataframe is {ad_min_date}")
print(f"The latest date in the ad dataframe is {ad_max_date}")
print((f"Time Span: {months} months and {remaining_days} day(s)"))
```

```
The earliest date in the ad dataframe is 2021-12-01
The latest date in the ad dataframe is 2022-06-30
Time Span: 7 months and 1 day(s)
```

The datasets span different time periods and have different formats.

DataFrame	Date from	Date to	Time Span
actual_duration	01 December 2021	30 June 2022	7 months
appointments_regional	Jan-20	Jun-22	30 months
national_categories	01 August 2021	30 June 2022	11 months

The above actions enable analysis based on a common monthly appointment timeframe.

2.4 Multi-functional Aggregation Function

sum_appointments_multi()

- Aggregating the number of appointments in the datasets is required in multiple questions.
- A function was created that allows for the selection of a dataframe and groupby element to return the aggregated (sum) of appointments.
- The function has been created iteratively and certain functionality has been added when required, such as location and date filters.
- The function allows technical and non-technical users to specify the dataframe and parameters.
- Several questions posed by the NHS are specific in nature. The function allows for the selection of different parameters and therefore widens the scope to accommodate future variations of the business questions, providing flexibility.
- Allows for the dataframe created in the function to be used outside the function.

A full breakdown of the code is contained in Appendix A.4.

2.5 Locations: Index and Mapping

The locations are represented differently across the data sets.

	Regional		ICB		Sub ICB		
DataFrame	ONS Code	Location Name	ONS Code	Location Name	ONS Code	Location Code	Location Name
actual_duration	Yes	No	Yes	No	Yes	Yes	Yes
appointments_regional	No	No	Yes	No	No	No	No
national_categories	No	No	Yes	As prefix to Sub ICB	No	As suffix to Sub ICB	Yes

The common grouping between all data sets is ONS code for ICBs and this will provide the most optimal method of comparing datasets based on location.

ICB Code and Location Index

An index of ONS codes and their respective location names was created.

Extract:

icb_ons_code	icb_location_name
E54000008	NHS Cheshire and Merseyside ICB
E54000010	NHS Staffordshire and Stoke-on-Trent ICB
E54000011	NHS Shropshire Telford and Wrekin ICB
E54000013	NHS Lincolnshire ICB
E54000015	NHS Leicester Leicestershire and Rutland ICB
E54000018	NHS Coventry and Warwickshire ICB
E54000019	NHS Herefordshire and Worcestershire ICB
E54000022	NHS Norfolk and Waveney ICB
E54000023	NHS Suffolk and North East Essex ICB
E54000024	NHS Bedfordshire Luton and Milton Keynes ICB
E54000025	NHS Hertfordshire and West Essex ICB
E54000026	NHS Mid and South Essex ICB
E54000027	NHS North West London ICB
E54000028	NHS North Central London ICB
E54000029	NHS North East London ICB

Mapping location names

A mapping function was created (using dataframe and column position as parameters) to input location names to their respective codes in nc, ar and ad.

Locations

What are the Top 5 locations in terms of appointments?

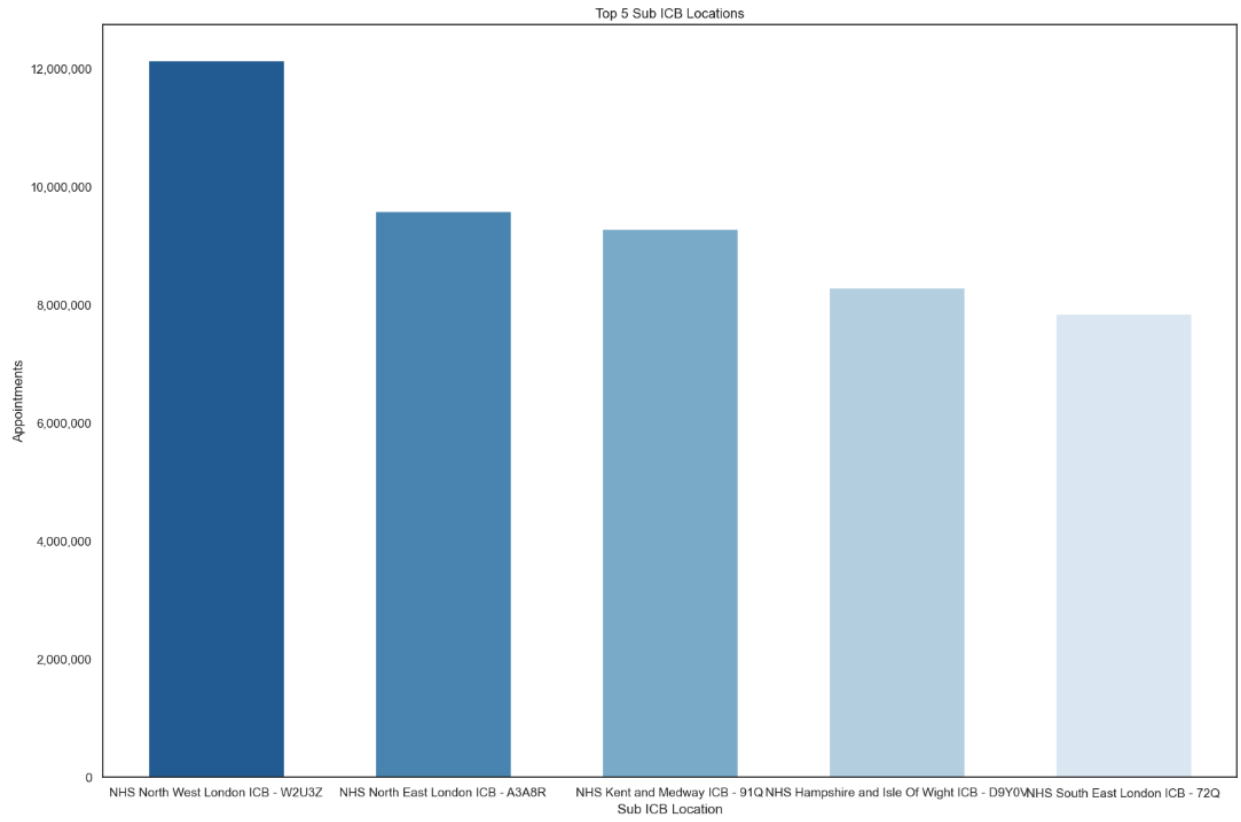
Sub ICBS

Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
sub_icb_locations	nc	sub_icb_location_name	Sum	None	Top 5 by count_of_appointments		Barplot: Top 5 sub icb locations

```
Number of sub_icb_locations:  106
```

	sub_icb_location_name	count_of_appointments
76	NHS North West London ICB - W2U3Z	12142390
67	NHS North East London ICB - A3A8R	9588891
47	NHS Kent and Medway ICB - 91Q	9286167
36	NHS Hampshire and Isle Of Wight ICB - D9Y0V	8288102
82	NHS South East London ICB - 72Q	7850170



There are 106 sub ICB locations. North West London – W2U3Z has the highest number of appointments.

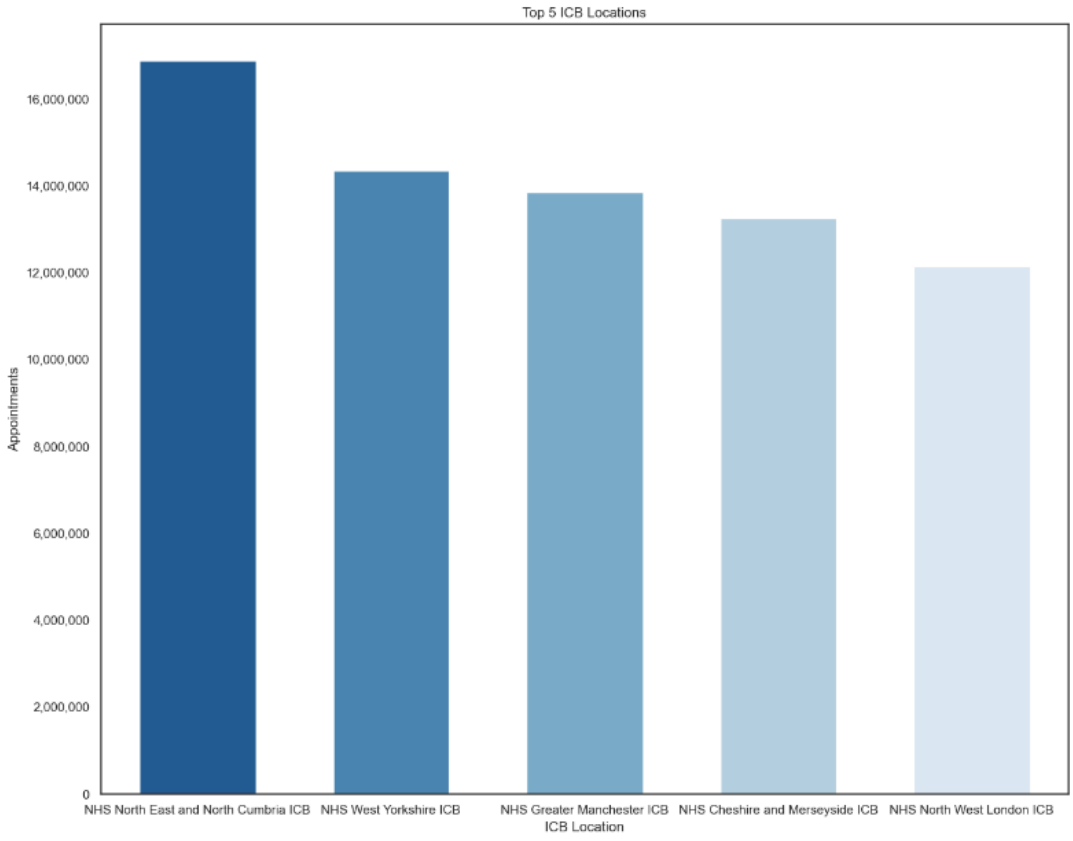
ICBs

Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
icb_locations	nc	icb_location_name	Sum	None	Top 5 by count_of_appointments		Barplot: Top 5 icb locations

```
Number of icb_locations:  42
```

	icb_location_name	count_of_appointments
28	NHS North East and North Cumbria ICB	16882235
41	NHS West Yorkshire ICB	14358371
15	NHS Greater Manchester ICB	13857900
7	NHS Cheshire and Merseyside ICB	13250311
29	NHS North West London ICB	12142390



There are 42 ICB locations. NHS North East and North Cumbria has the highest number of appointments.

Regions

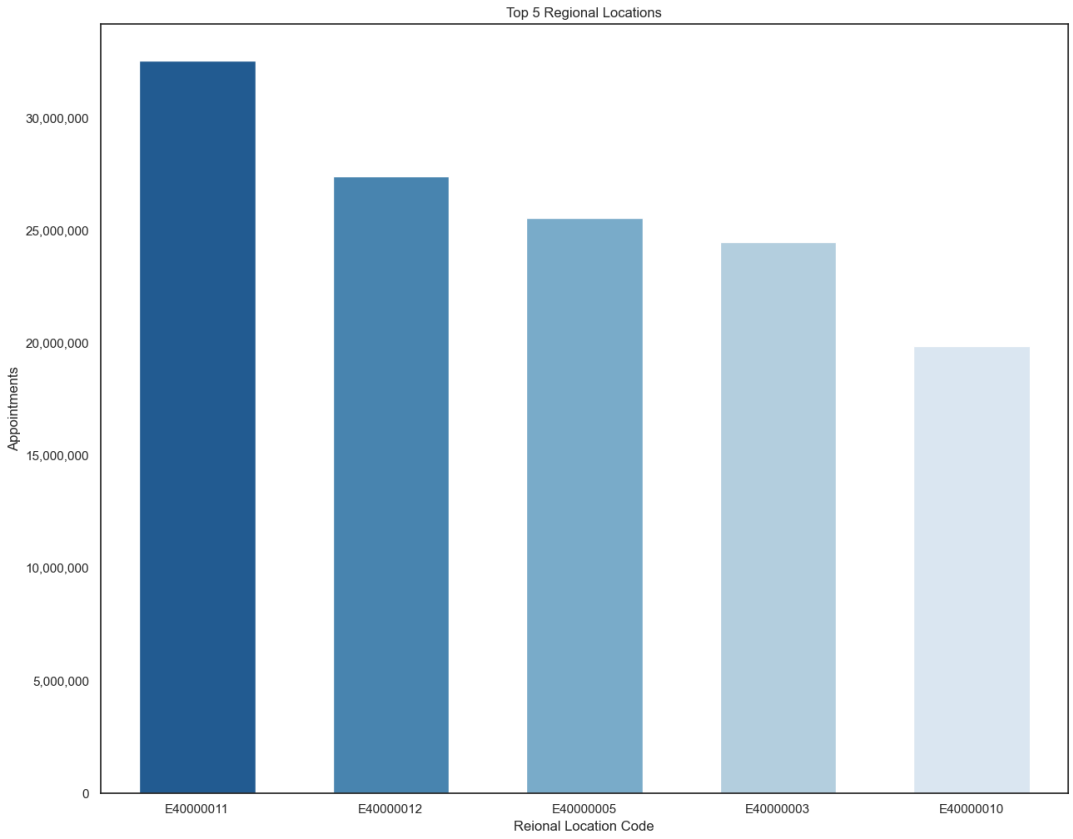
Actual_duration contains regional codes.

Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
regions	ad	region_ons_code	Sum	None	Top 5 by count_of_appointments		Barplot: Top 5 regional locations

```
Number of regions:  7
```

	region_ons_code	count_of_appointments
5	E40000011	32574555
6	E40000012	27425610
1	E40000005	25577953
0	E40000003	24488519
4	E40000010	19880924



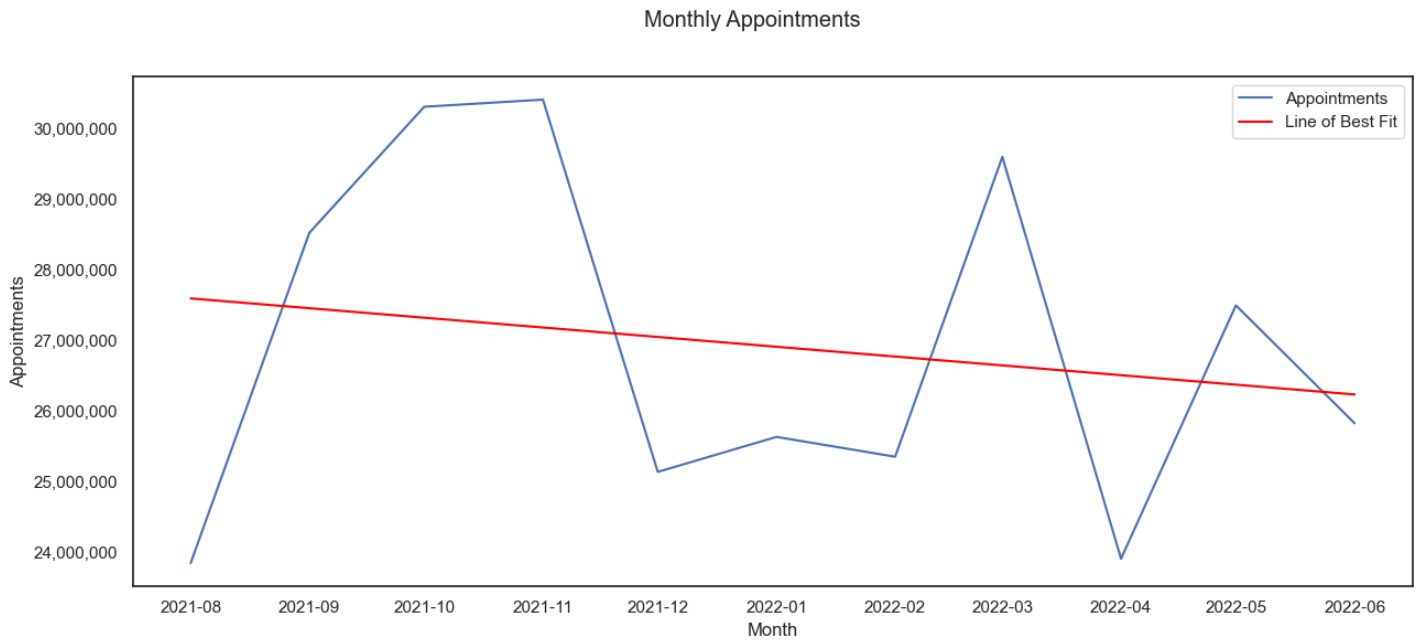
There are 7 Regional locations. E40000011 (Midlands) has the highest number of appointments.

3. Appointment Analysis

3.1 Appointments by service settings, context types, national categories.

General NC dataframe.

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
nc_general	nc	appointment_month	Sum				Lineplot: Monthly Appointments Full



- The busiest times of year are October and November. We see a considerable increase in appointments in the transition to Autumn. There is also a significant increase in appointments in February through March.
- The linear regression line indicates a downtrend in total appointments in the NC dataframe.

Service Settings

Call the sum_appointment_multi function and create new df(s).

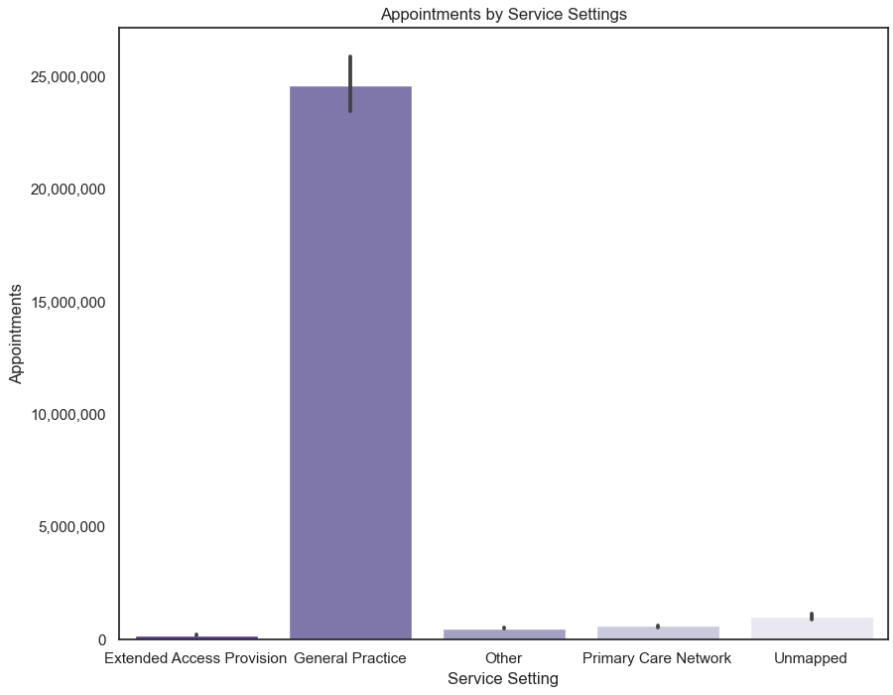
New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
nc_ss	nc	appointment_month	Sum	None	None	Count unique service settings	Barplot: Service settings by count_of _appointments
		service_setting					Lineplot: appointment_month by count_of_appointments
							Barplot: Seasonal Service settings by count_of _appointments

```
Number of nc_ss:  55
Number of service settings: 5
```

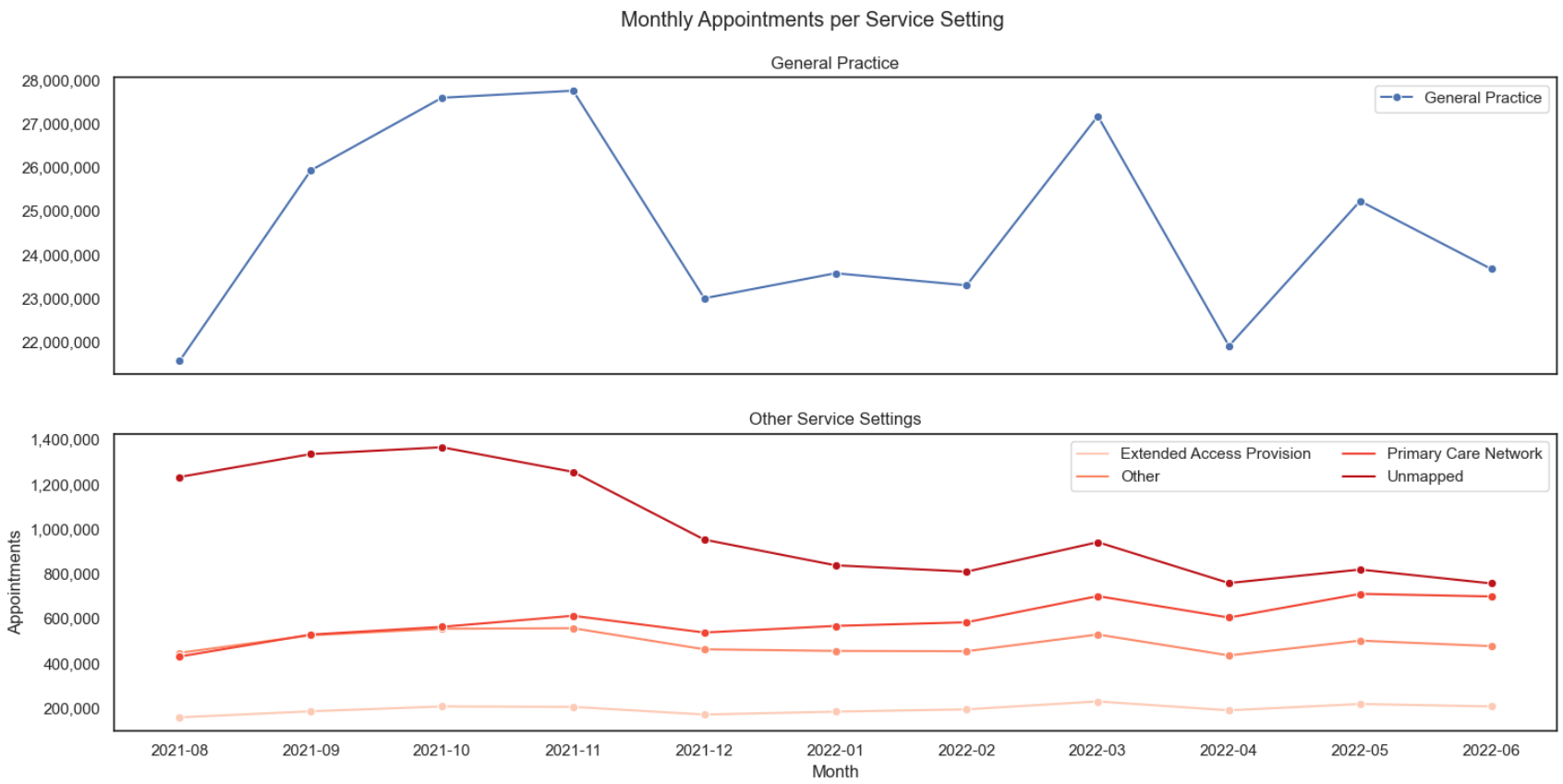
	appointment_month	service_setting	count_of_appointments
0	2021-08	Extended Access Provision	160927
1	2021-08	General Practice	21575852
2	2021-08	Other	449101
3	2021-08	Primary Care Network	432448
4	2021-08	Unmapped	1233843
5	2021-09	Extended Access Provision	187906
6	2021-09	General Practice	25940821
7	2021-09	Other	527174
8	2021-09	Primary Care Network	530485
9	2021-09	Unmapped	1336115

```python
# Count number of service_settings
ss = nc['service_setting'].nunique()
print(f"Number of service settings: {ss}")
```

```
Number of service settings: 5
```



SS Monthly Trends



Insights

- There are 5 service settings. General practice is the largest and in line with the general trend.
- Unmapped has been steadily declining over the period.

Context Types

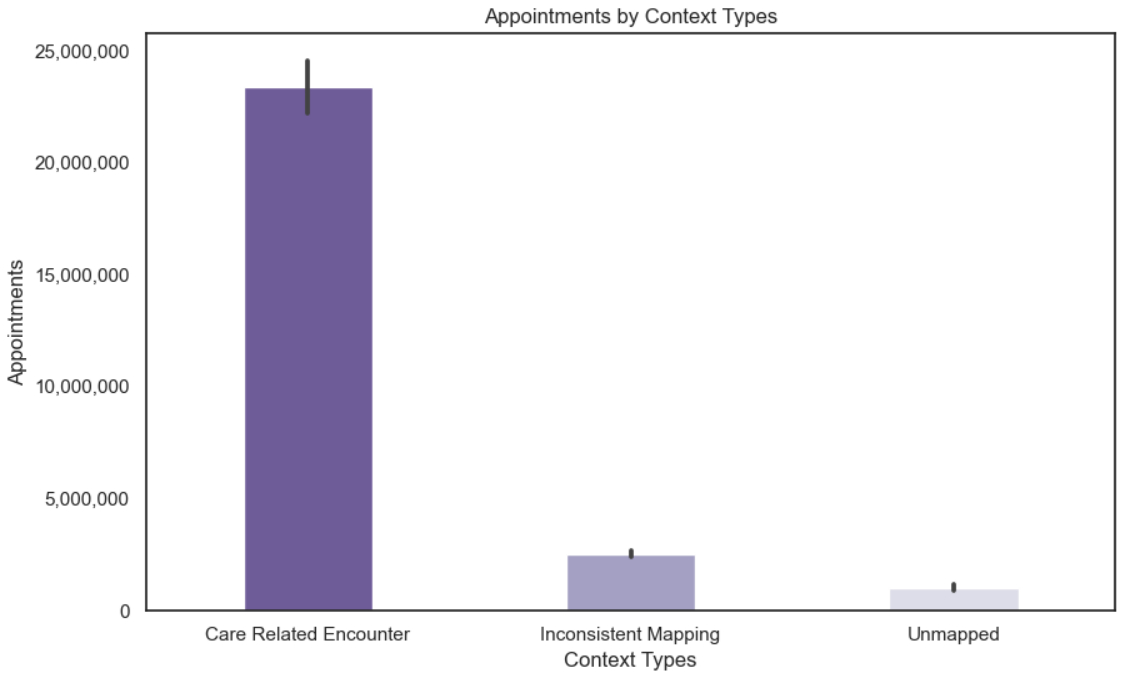
Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
nc_ct	nc	appointment_month	Sum	None	None		Barplot: context_type by count_of _appointments
		context_type					Lineplot: appointment_month by count_of_appointments
							Barplot: Seasonal context_type by count_of _appointments

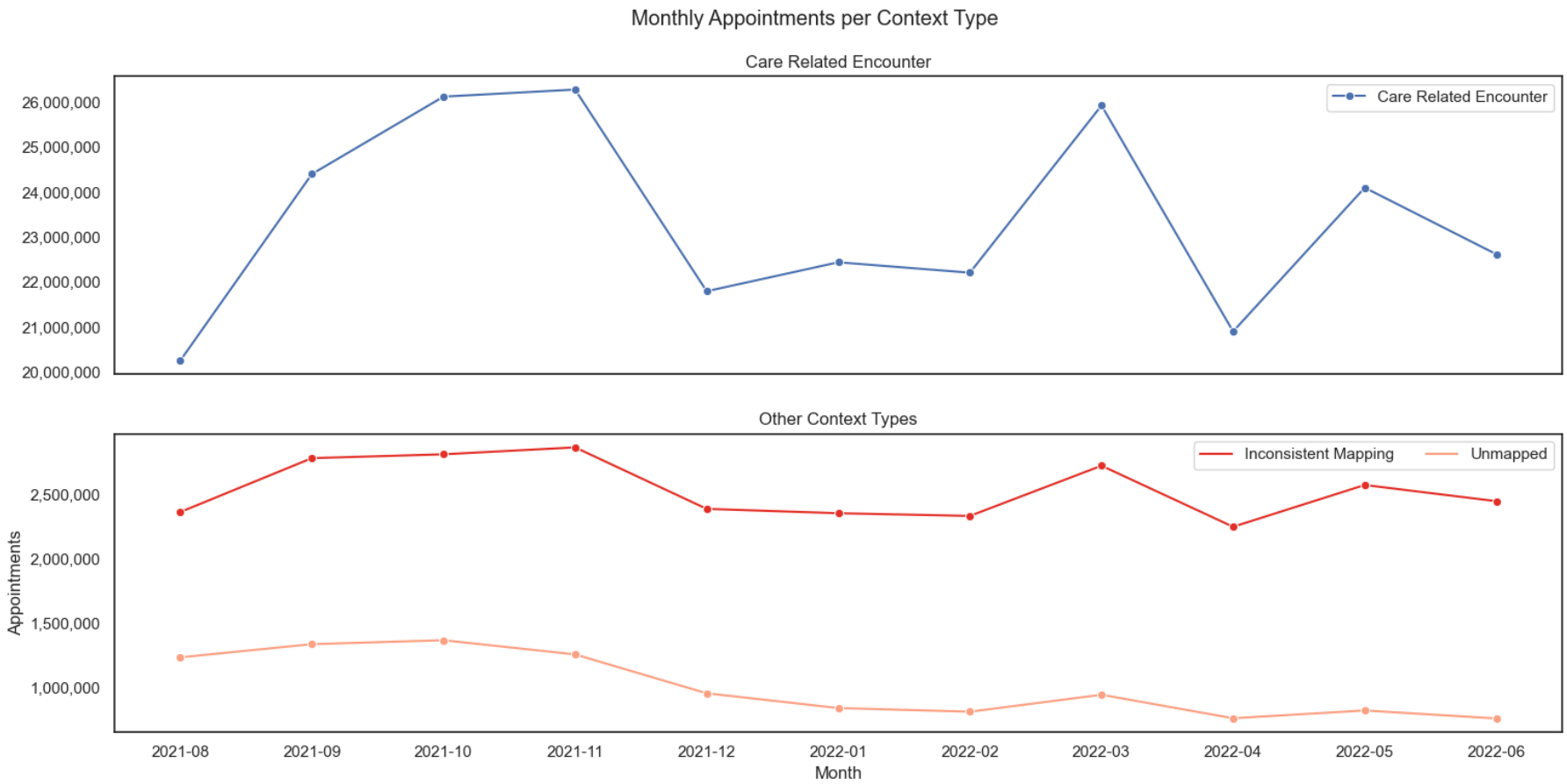
	appointment_month	context_type	count_of_appointments
0	2021-08	Care Related Encounter	20255235
1	2021-08	Inconsistent Mapping	2363093
2	2021-08	Unmapped	1233843
3	2021-09	Care Related Encounter	24404251
4	2021-09	Inconsistent Mapping	2782135
5	2021-09	Unmapped	1336115
6	2021-10	Care Related Encounter	26125201
7	2021-10	Inconsistent Mapping	2811977
8	2021-10	Unmapped	1366656
9	2021-11	Care Related Encounter	26282778

```
# Count number of context types
ct = nc['context_type'].nunique()
print(f"Number of context types: {ct}")
```

```
Number of context types: 3
```



CT Monthly Trends



- Care-related encounters are broadly in line with the general trend.
- Inconsistent mapping and unmapped are ambiguous and form a materially significant proportion of total appointment.

National Categories

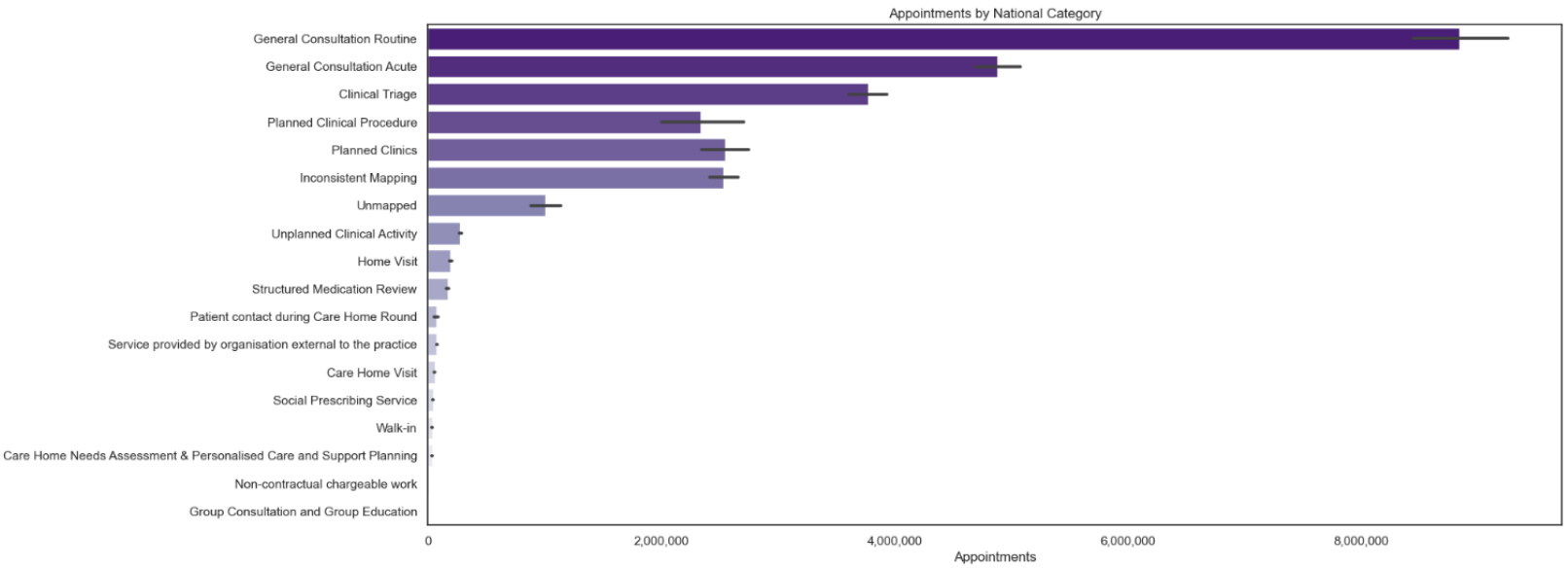
Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
nc_nc	nc	appointment_month	Sum	None	None	**Split into 4 tiers to better visualise in lineplot.**	Barplot: National category by count_of _appointments
		national_category					Lineplot: appointment_month by count_of_appointments
							Barplot: Seasonal National category by count_of _appointments

```
# Count number of national categories
ncs = nc['national_category'].nunique()
print(f"Number of national categories: {ncs}")
```

```
Number of national categories: 18
```

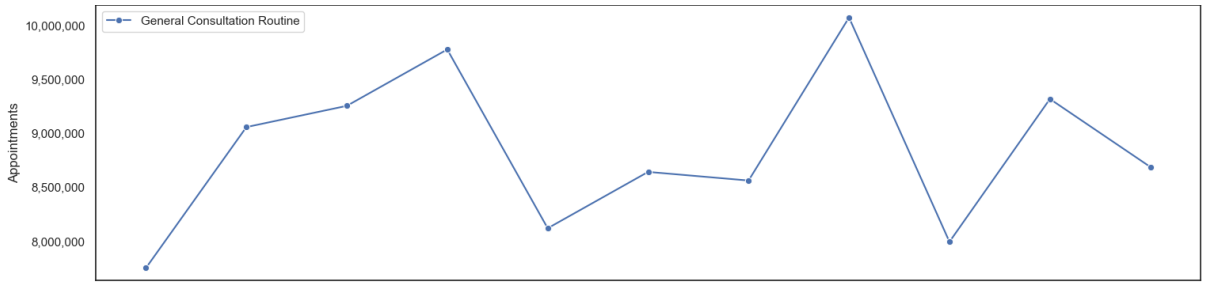
	appointment_month	national_category	count_of_appointments
0	2021-08	Care Home Needs Assessment & Personalised Care and Support Planning	29676
1	2021-08	Care Home Visit	47583
2	2021-08	Clinical Triage	3704207
3	2021-08	General Consultation Acute	4280920
4	2021-08	General Consultation Routine	7756045
5	2021-08	Group Consultation and Group Education	5161
6	2021-08	Home Visit	165061
7	2021-08	Inconsistent Mapping	2363093
8	2021-08	Non-contractual chargeable work	10775
9	2021-08	Patient contact during Care Home Round	31316

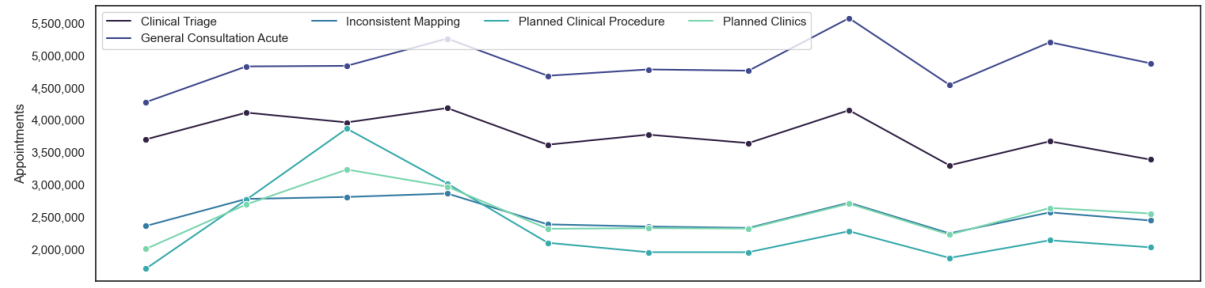


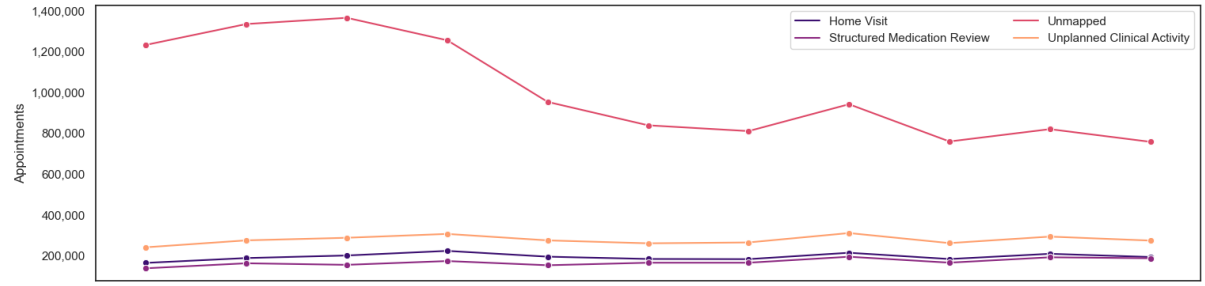
NC Monthly Trends

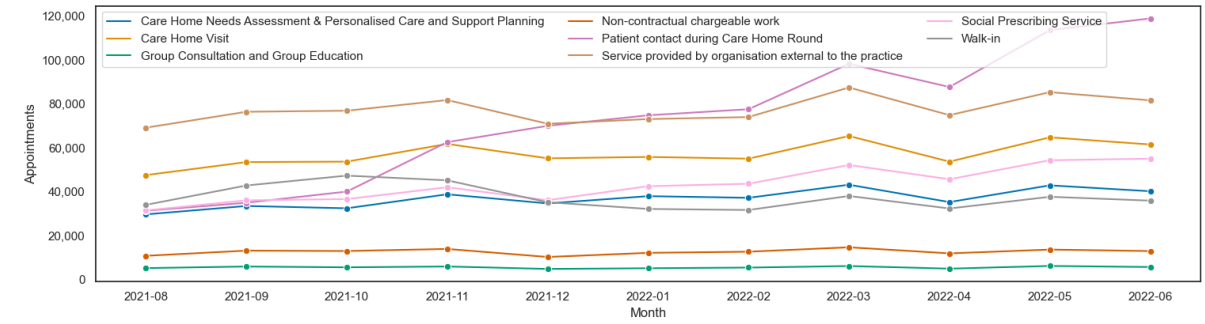
There are 18 national categories. General Consultation Routine forms the most appointments. The categories differ widely in the number of appointments. 4 tiers were created based on the count_of_appointments so that each national category can be better visualised.

```
# Create 3 categories based on sum of appointments in the dataframe.(General Consultation Routine has it's own tier)

nc_nc_sums = nc_nc.groupby('national_category')['count_of_appointments'].sum()

tier_2 = category_sums[(nc_nc_sums > 20000000) & (nc_nc_sums < 55000000)].index.tolist()
tier_3 = category_sums[(nc_nc_sums > 1000000) & (nc_nc_sums < 20000000)].index.tolist()
tier_4 = category_sums[nc_nc_sums < 1000000].index.tolist()
```









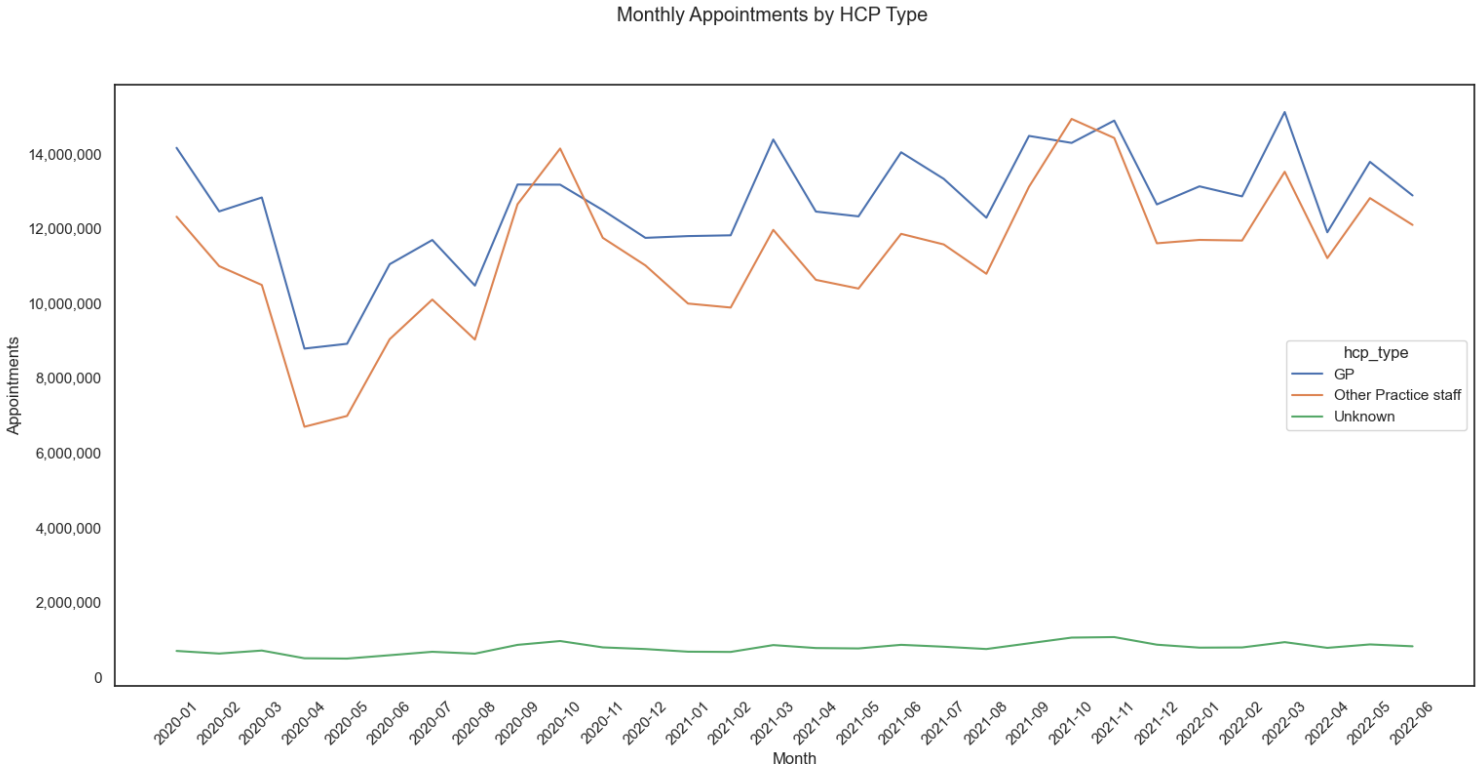
- General consultation routine is broadly in line with the overall trend.
- Unmapped steadily declining.
- Notable increase in ‘patient contact during care home round’.

3.2 Appointments by HCP type and Mode.

How do the healthcare professional types differ over time?

Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ar_hcp	ar	appointment_month	Sum	None	None		Lineplot: Monthly Appointments by HCP Type
		hcp_type					

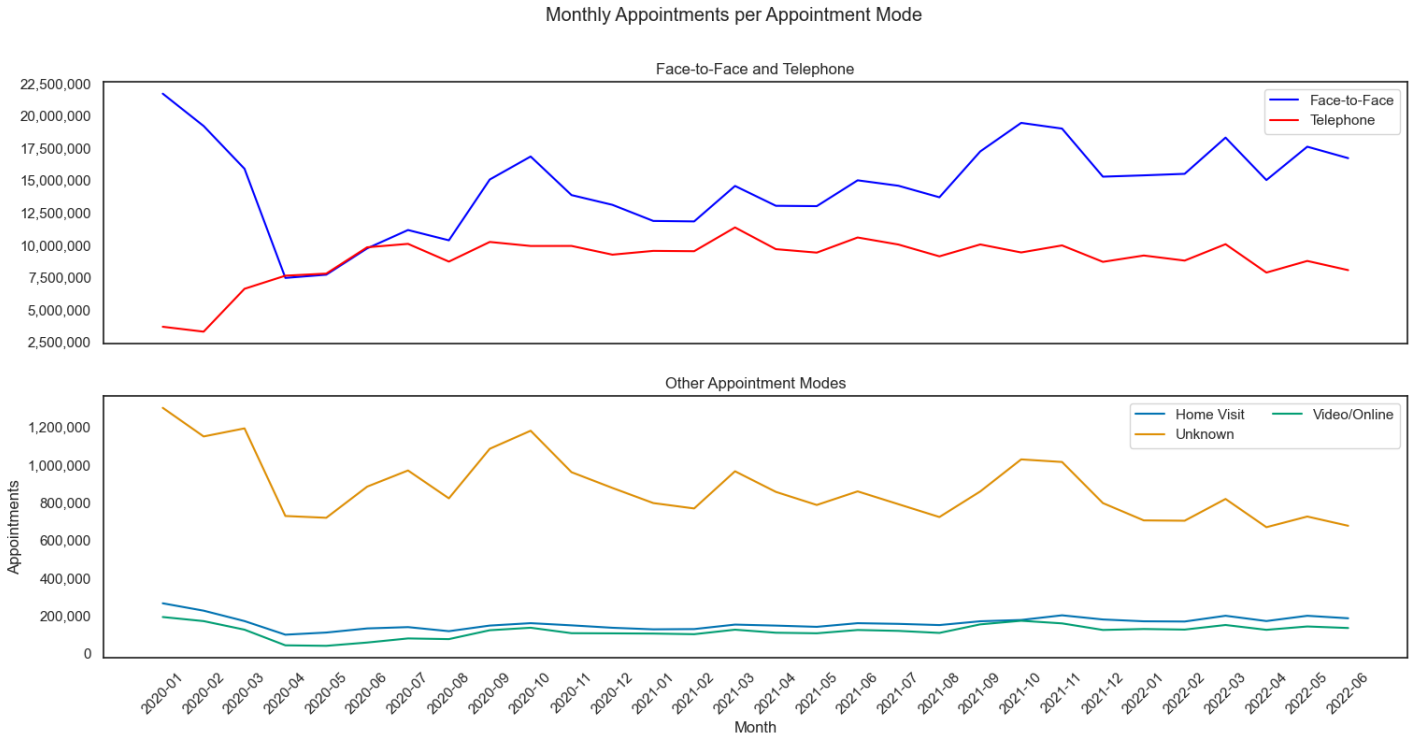


- HCP types are broadly in line with total appointments, peaking in Autumn and late winter / early spring.
- Generally, GP appointments exceed other practice staff. In the early autumn, other practice appointments overtake GP appointments as the primary HCP type.

Are there changes in terms of appointment mode and the busiest months?

Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ar_mode	ar	appointment_month	Sum	None	None	Lineplot split to enable higher resolution veiwing.	Lineplot: Monthly Appointments by appointment_mode
		appointment_mode					



- In general, face-to-face appointments exceed all other appointment modes.
- In the busiest months, face-to face appointments see the greatest increase of appointment modes.
- At the beginning of the dataset, there was a marked decrease in face-to-face appointments and a corresponding increase in telephone appointments, perhaps due to the pandemic.
- Since mid-February 2021, telephone appointments have steadily decreased diverging from the steadily increasing face-to-face appointments.
- Home visits and video/online appointments remain low and steady**.**

3.3 Total Appointments

Which month had the highest number of appointments?

Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
nc_monthly	nc	appointment_month	Sum	None	Sorted by count_of_appointments		

	appointment_month	count_of_appointments
3	2021-11	30405070
2	2021-10	30303834
7	2022-03	29595038
1	2021-09	28522501
9	2022-05	27495508

November 2021.

What was the total number of appointments per month for each dataset?

Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ar_monthly	ar	appointment_month	Sum	None	None	Merged ar_monthly, nc_monthly and ad_monthly	Lineplot: Monthly Appointments per Data Set
nc_monthly	nc	appointment_month	Sum	None	None	Merged ar_monthly, nc_monthly and ad_monthly	Lineplot: Monthly Appointments per Data Set
ad_monthly	ad	appointment_month	Sum	None	None	Merged ar_monthly, nc_monthly and ad_monthly	Lineplot: Monthly Appointments per Data Set

Merge dataframes and rename columns to compare.

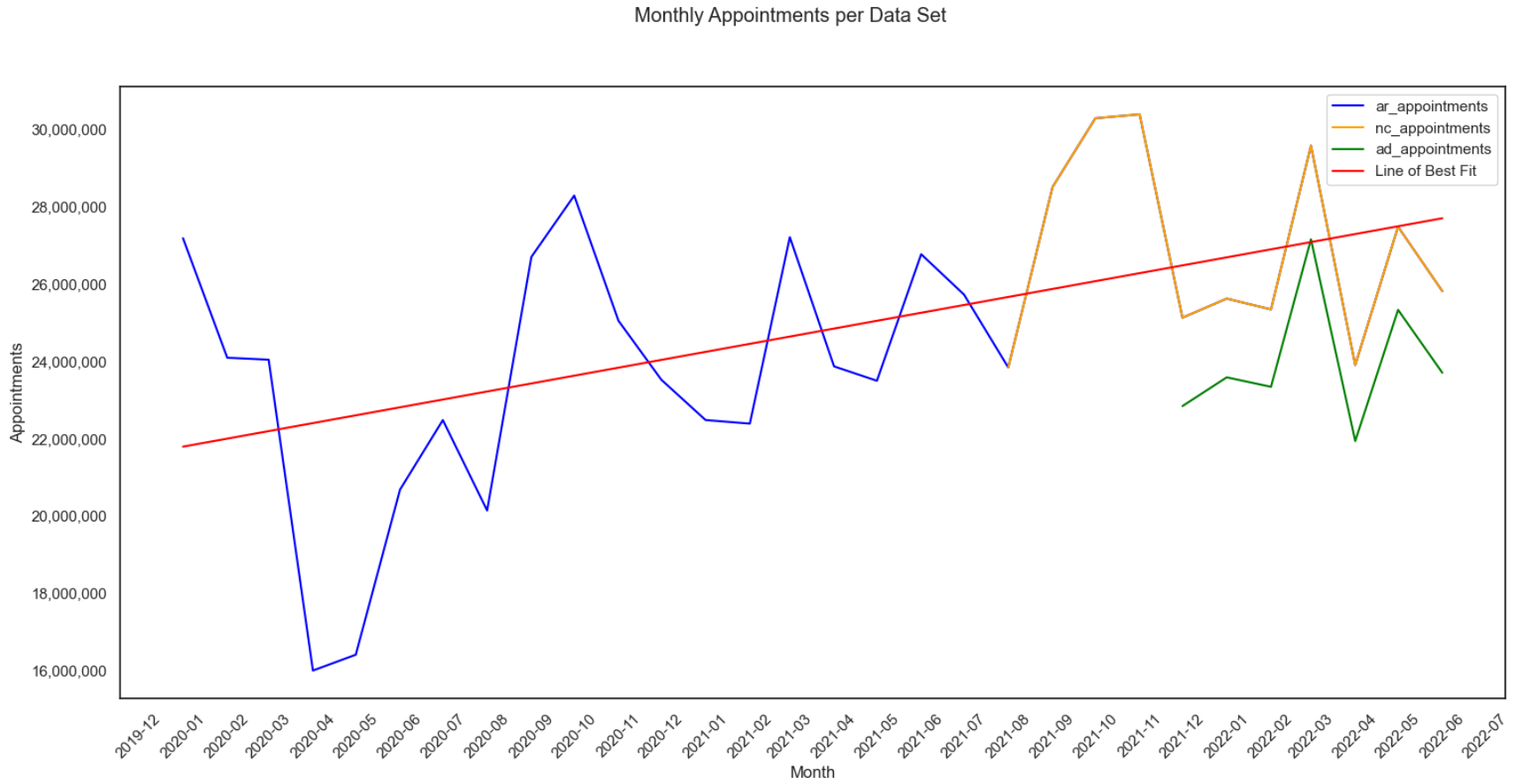
```
# Merge and compare similar dates.

monthly_merge = pd.merge(ar_monthly, nc_monthly, on='appointment_month', how='left')
monthly_merge = pd.merge(monthly_merge, ad_monthly, on='appointment_month', how='left')

# Change column names
monthly_merge.columns = ['appointment_month', 'ar_appointments', 'nc_appointments', 'ad_appointments']
monthly_merge
```

Output:

```
Number of ar_monthly:  30
Number of nc_monthly:  11
Number of ad_monthly:  7
```

	appointment_month	ar_appointments	nc_appointments	ad_appointments
0	2020-01	27199296	NaN	NaN
1	2020-02	24104621	NaN	NaN
2	2020-03	24053468	NaN	NaN
3	2020-04	16007881	NaN	NaN
4	2020-05	16417212	NaN	NaN
5	2020-06	20690805	NaN	NaN
6	2020-07	22491437	NaN	NaN
7	2020-08	20150520	NaN	NaN
8	2020-09	26714255	NaN	NaN
9	2020-10	28301932	NaN	NaN
10	2020-11	25061602	NaN	NaN
11	2020-12	23535936	NaN	NaN
12	2021-01	22492069	NaN	NaN
13	2021-02	22399569	NaN	NaN
14	2021-03	27225424	NaN	NaN
15	2021-04	23879932	NaN	NaN
16	2021-05	23508395	NaN	NaN
17	2021-06	26784182	NaN	NaN
18	2021-07	25739219	NaN	NaN
19	2021-08	23852171	23852171.0	NaN
20	2021-09	28522501	28522501.0	NaN
21	2021-10	30303834	30303834.0	NaN
22	2021-11	30405070	30405070.0	NaN
23	2021-12	25140776	25140776.0	22853483.0
24	2022-01	25635474	25635474.0	23597196.0
25	2022-02	25355260	25355260.0	23351939.0
26	2022-03	29595038	29595038.0	27170002.0
27	2022-04	23913060	23913060.0	21948814.0
28	2022-05	27495508	27495508.0	25343941.0
29	2022-06	25828078	25828078.0	23715317.0

Diagnostic Analysis using Python.



Insights

- The busiest times of year are October and November. We see a considerable increase in appointments in the transition to Autumn. There is also a significant increase in appointments in February through March.
- The nc and ad dataframes form only part of a wider uptrend as depicted by the line of best fit. One might expect October / November 2022 to exceed 30,000,000 per month.
- Missed appointments aren't included in the actual duration figures.

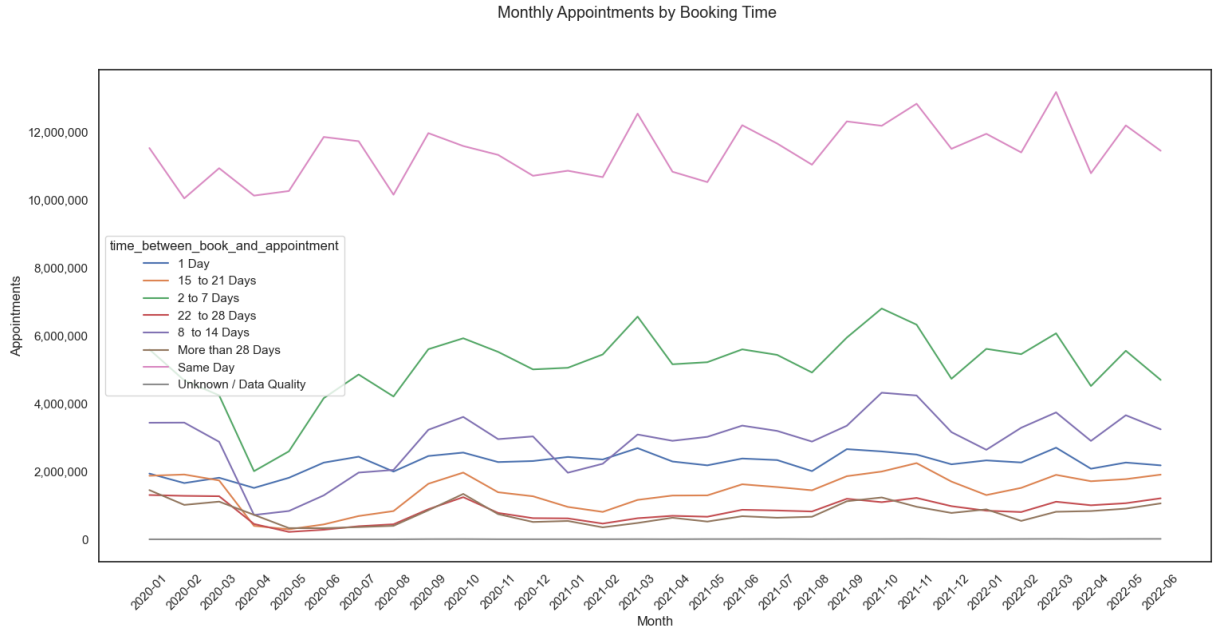
4. How effectively is the NHS is Coping with Increasing Utilisation.

4.1 Time Between Booking and Appointment

Are there any trends in time between booking an appointment?

Call the sum_appointment_multi function and create new df(s).

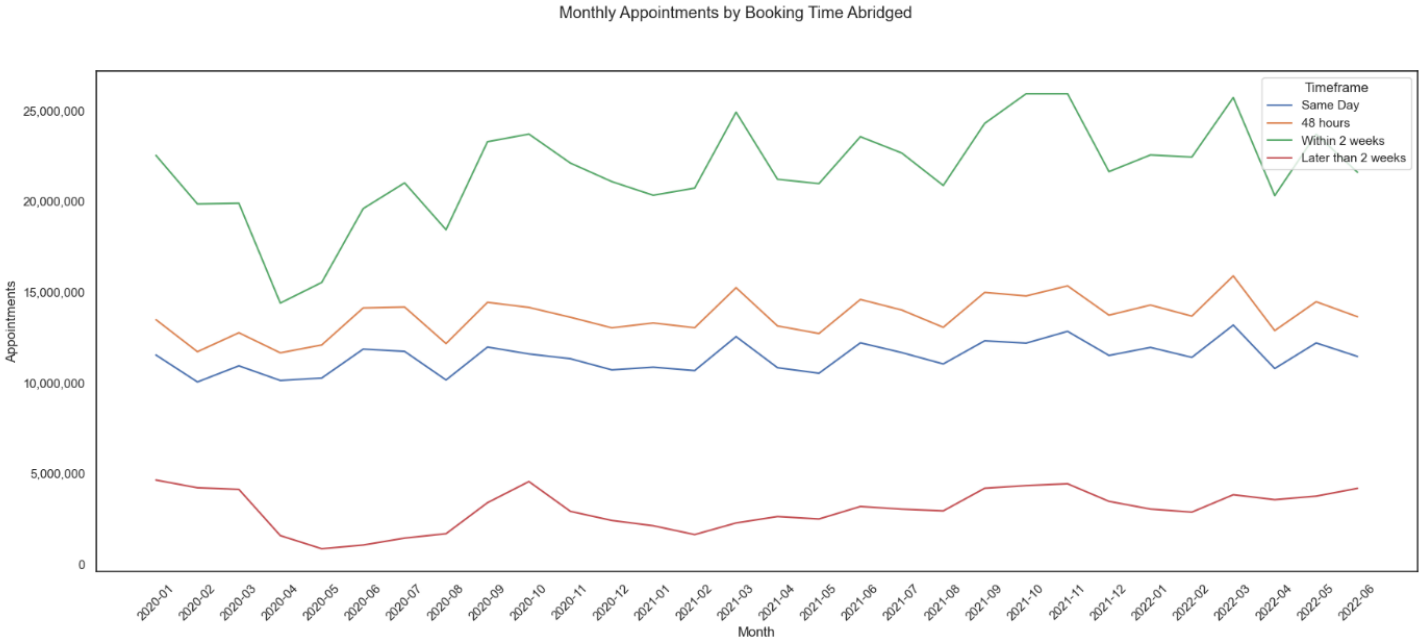
New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ar_booking	ar	appointment_month	Sum	None	None	Pivoted to create pivot_ar_booking	Lineplot: Monthly Appointments by Booking Time
		time_between_book_and_appointment		None			
pivot_ar_booking	ar_booking	Timeframes grouped to Same day, 48 hours, Within 2 weeks and later than 2 weeks.	Sum	None	None	% calculations made on total appointments.	Lineplot: Monthly Appointments by Booking Time Abridged
pivot_ar_booking_perc	ar_booking	Timeframes grouped to Same day (%), 48 hours (%), Within 2 weeks (%) and later than 2 weeks (%).	Sum	None	None		Lineplot: Monthly Appointments by Booking Time %
average_wait	pivot_ar_booking_perc		Mean	None	None		



Most patients are seen on the same day as booking the appointment. It is comparatively rare to wait longer than 28 days.

The ar_booking df was pivoted and regrouped into the timeframes in accordance with government target of 2 weeks (source: [New plan to make it easier for patients to see their GP - GOV.UK (www.gov.uk)](). Same day, 48 hours and Later than 2 weeks also included.

time_between_book_and_appointment	appointment_month	Same Day	48 hours	Within 2 weeks	Later than 2 weeks	Same day (%)	48 hours (%)	Within 2 weeks (%)	Later than 2 weeks (%)
0	2020-01	11541694	13487405	22536951	4654103	42.446659	49.602362	82.883698	17.116302
1	2020-02	10058743	11724819	19865987	4230033	41.744417	48.658737	82.445097	17.554903
2	2020-03	10947395	12770321	19904530	4140639	45.528459	53.109716	82.779747	17.220253
3	2020-04	10139567	11662644	14404742	1594870	63.373831	72.893293	90.031821	9.968179
4	2020-05	10274928	12096738	15540879	869099	62.613905	73.715748	94.703838	5.296162



- Most patients are seen within 2 weeks of booking and this is increasing in line with the general trend.
- The later than 2 weeks threshold is plateauxed with notable upticks in busier periods.

We have seen that the number of appointments occurring within 2 weeks of booking is increasing and that the number of appointments occurring after 2 weeks is plateauxed.

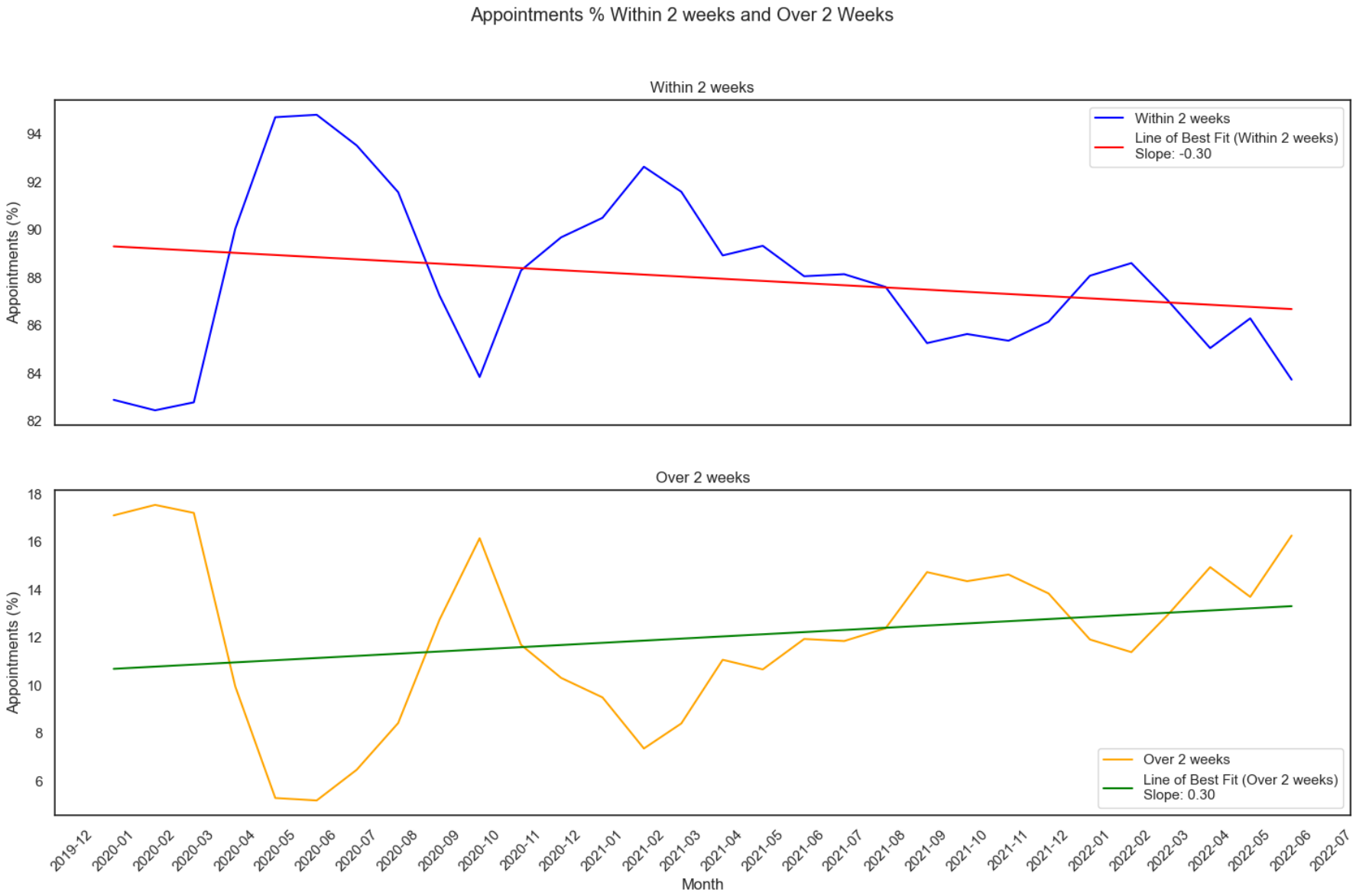
However, the total number of appointments is also increasing. To understand the change in real terms, we must standardise for the increase in utilisation.

The number of appointments for the new timeframes were expressed as a percentage of the total monthly appointments.

	Timeframe Average
time_between_book_and_appointment	
Same day (%)	46.81
48 hours (%)	56.01
Within 2 weeks (%)	87.99
Later than 2 weeks (%)	12.01

NHS

Diagnostic Analysis using Python.



Insights

In real terms, the NHS's ability to see patients within 2 weeks is falling and, by necessity, the patients waiting over 2 weeks is increasing.

When appointment demand is high the 'Within 2 week' threshold is more difficult to maintain.

This is a strong indication that capacity is being stretched.

5. Measures to Extract Value from Existing Resources.

5.1 Missed Appointment Analysis.

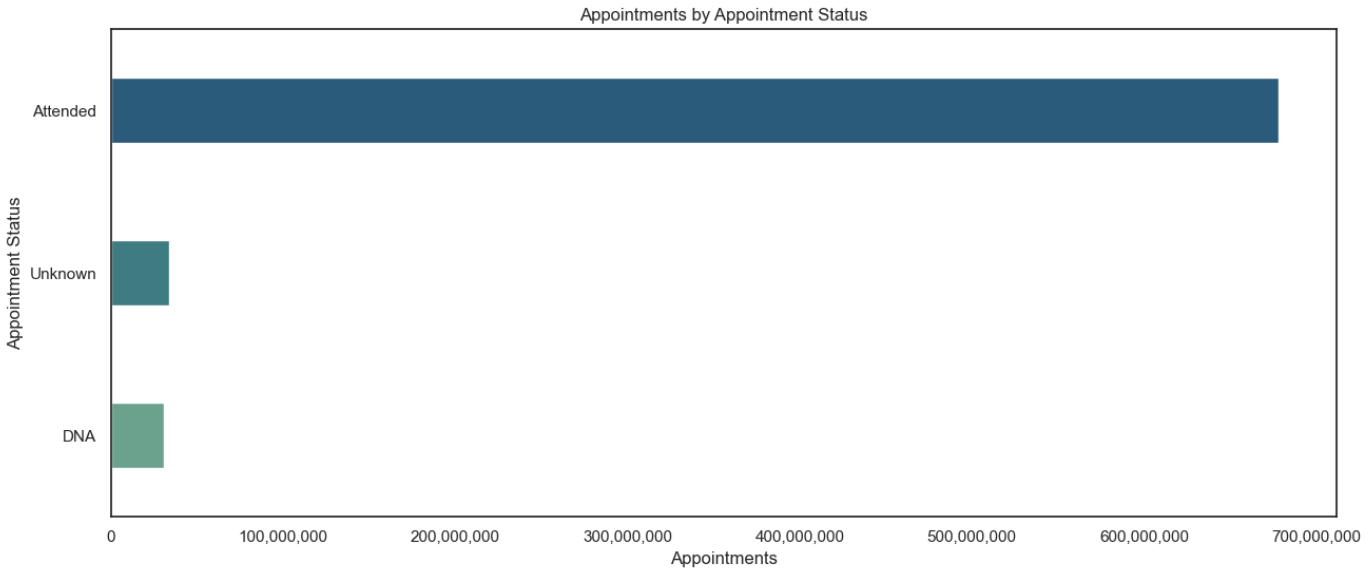
"The NHS incurs significant, potentially avoidable, costs when patients miss general practitioner (GP) appointments."

According to the BMJ it is estimated that 1 in 20 appointments are missed, costing on average £30 per appointment (D.Oliver, https://doi.org/10.1136/bmj.l545 , 2019).

1. **How many appointment statuses are there and what is the count of appointments in each status?**

Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ar_as	ar	appointment_status	Sum	None	Sorted by count_of_appointments		Barplot: Appointments by Appointment Status



There are 3 appointment statuses, the majority being "Attended". A significant proportion of total appointments were missed. Due to the "unknown" status, missed appointments may be considerably higher.

The following analysis uses appointment status as a percentage of total appointments within a category standardize the data for comparison:

2. As a percentage of total appointments, which ICB locations are the poorest performers regarding missed appointments.

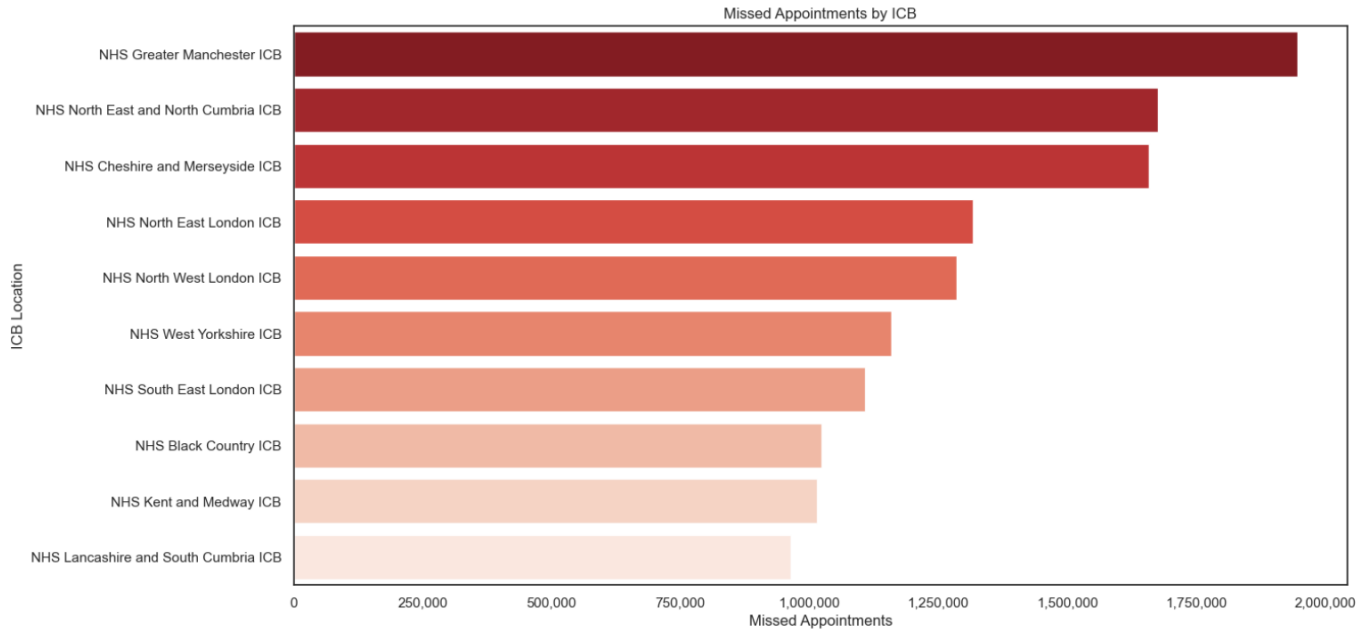
Call the sum_appointment_multi function and create new df(s).

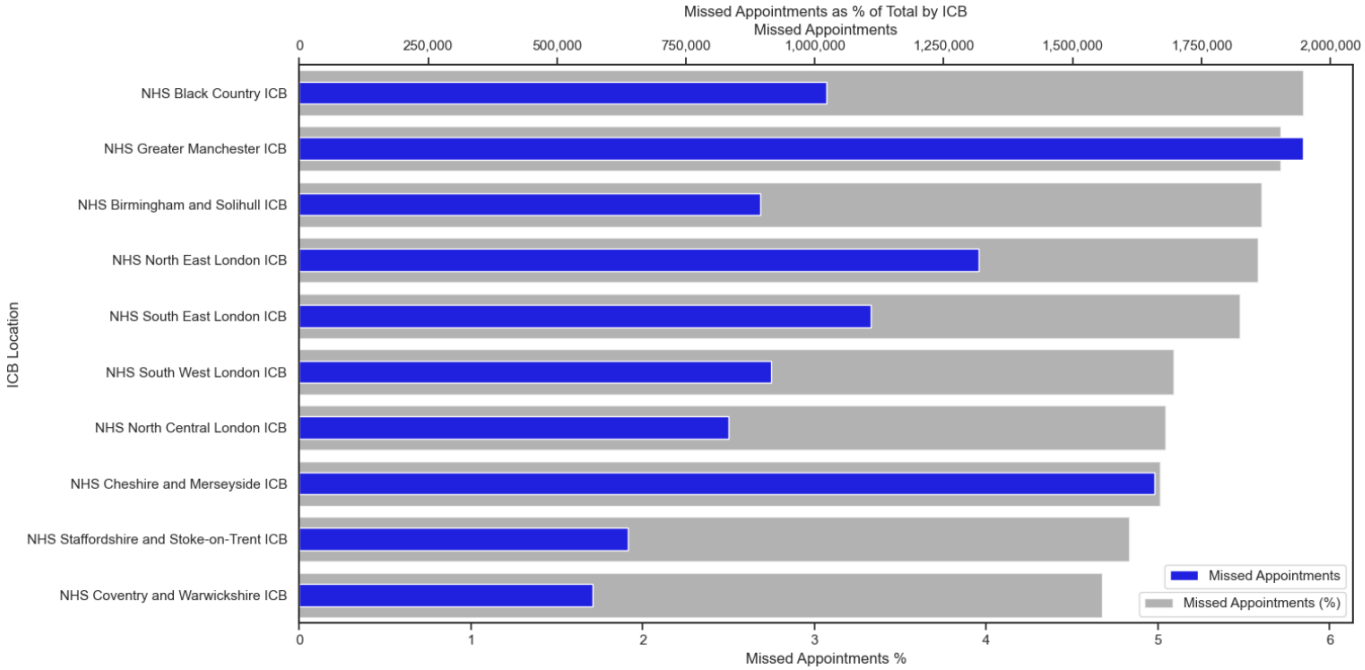
New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ar_icb_as	ar	icb_location_name	Sum	None	None	Pivoted.	Barplot: Missed Appointments by ICB
pivot_ar_icb_as	ar_icb_as	appointment_status	Sum	None	By Top 10 DNA & Top 10 DNA%	Calculated columns added: Total, % appointment_status as proprtion of total.	Barplot: Missed Appointments as % of Total by ICB

The returned df was then pivoted to split the appointment statuses into columns. A total column was added, and a percentage of total column calculated. The pivoted df was sorted in descending order by DNA % and the index reset. The top 10 poorest performing locations in terms of missed appointments were displayed.

appointment_status	icb_location_name	Attended	DNA	Unknown	Total	Attended (%)	DNA (%)	Unknown (%)
1	NHS Greater Manchester ICB	30310360	1946114	1824109	34080583	88.937328	5.710331	5.352341
23	NHS North East and North Cumbria ICB	39374437	1676790	2032308	43083535	91.390915	3.891951	4.717134
7	NHS Cheshire and Merseyside ICB	29881646	1658787	1559449	33099882	90.277198	5.011459	4.711343
3	NHS North East London ICB	21267205	1317005	1010049	23594259	90.137202	5.581888	4.280910
15	NHS North West London ICB	26848972	1286262	1245533	29380767	91.382815	4.377905	4.239280
36	NHS West Yorkshire ICB	33182630	1159691	1748066	36090387	91.943126	3.213296	4.843578
4	NHS South East London ICB	18158276	1107753	975423	20241452	89.708367	5.472695	4.818938
0	NHS Black Country ICB	15650234	1023176	840640	17514050	89.358167	5.842030	4.799804
12	NHS Kent and Medway ICB	20426411	1015035	1182532	22623978	90.286558	4.486545	5.226897
13	NHS Lancashire and South Cumbria ICB	19582053	964458	975892	21522403	90.984510	4.481182	4.534308

appointment_status	icb_location_name	Attended	DNA	Unknown	Total	Attended (%)	DNA (%)	Unknown (%)
0	NHS Black Country ICB	15650234	1023176	840640	17514050	89.358167	5.842030	4.799804
1	NHS Greater Manchester ICB	30310360	1946114	1824109	34080583	88.937328	5.710331	5.352341
2	NHS Birmingham and Solihull ICB	14361165	893541	697763	15952469	90.024717	5.601271	4.374013
3	NHS North East London ICB	21267205	1317005	1010049	23594259	90.137202	5.581888	4.280910
4	NHS South East London ICB	18158276	1107753	975423	20241452	89.708367	5.472695	4.818938
5	NHS South West London ICB	16330355	914695	727916	17972966	90.860657	5.089282	4.050061
6	NHS North Central London ICB	15038067	831775	625576	16495418	91.165116	5.042461	3.792423
7	NHS Cheshire and Merseyside ICB	29881646	1658787	1559449	33099882	90.277198	5.011459	4.711343
8	NHS Staffordshire and Stoke-on-Trent ICB	11904788	637789	653638	13196215	90.213656	4.833121	4.953223
9	NHS Coventry and Warwickshire ICB	11122652	568621	476919	12168192	91.407598	4.673011	3.919391





The above outlines the poorest performing ICB locations in terms of the missed appointments rate. The NHS might consider focusing on improving any ICB location above the overall average of 4.60% DNAs to total appointments. In addition, those ICB locations that have high DNAs in magnitude and proportion should be investigated with a higher degree of emphasis.

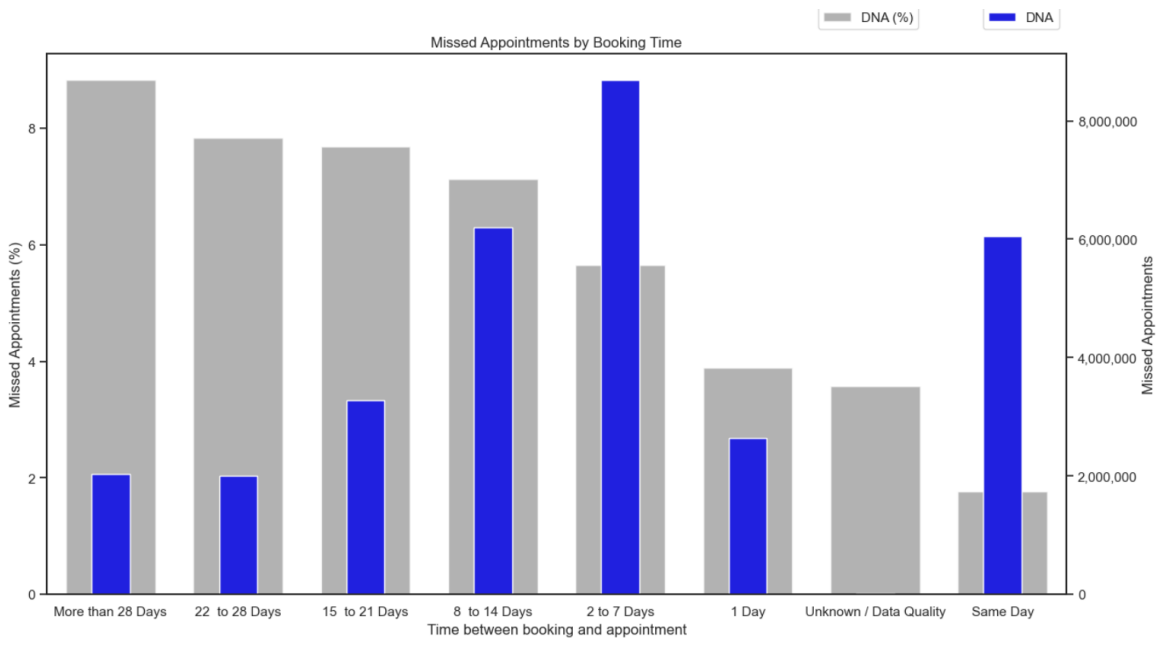
3. Does the time between booking an appointment and the appointment date influence the likelihood that an appointment will be missed.

Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ar_tb_as	ar	time_between_book_and_appointment	Sum	None	None	Pivoted.	Barplot: Missed Appointments by Booking Time
pivot_ar_tb_as	ar_tb_as	appointment_status	Sum	None	DNA (%)	Calculated columns added: Total, % appointment_status as proportion of total.	

The new df was pivoted in the same fashion as ar_icb_as.

appointment_status	time_between_book_and_appointment	Attended	DNA	Unknown	Total	Attended (%)	DNA (%)	Unknown (%)
0	More than 28 Days	16699531	2036154	4315302	23050987	72.446056	8.833262	18.720682
1	22 to 28 Days	20798309	1999990	2738242	25536541	81.445287	7.831875	10.722838
2	15 to 21 Days	35842753	3282752	3585069	42710574	83.920092	7.686040	8.393868
3	8 to 14 Days	75092108	6193368	5561043	86846519	86.465306	7.131395	6.403300
4	2 to 7 Days	138103022	8697476	6994033	153794531	89.797096	5.655257	4.547647
5	1 Day	62556833	2634536	2524728	67716097	92.381038	3.890561	3.728402
6	Unknown / Data Quality	283003	14353	104749	402105	70.380373	3.569466	26.050161
7	Same Day	328380317	6052604	8314250	342747171	95.808323	1.765909	2.425768



The 2 to 7 days bracket had the highest number of DNAs. However, as a proportion, the longer the period between booking and the appointment, the less likely the patient is to attend, with more than 28 days being the least likely.

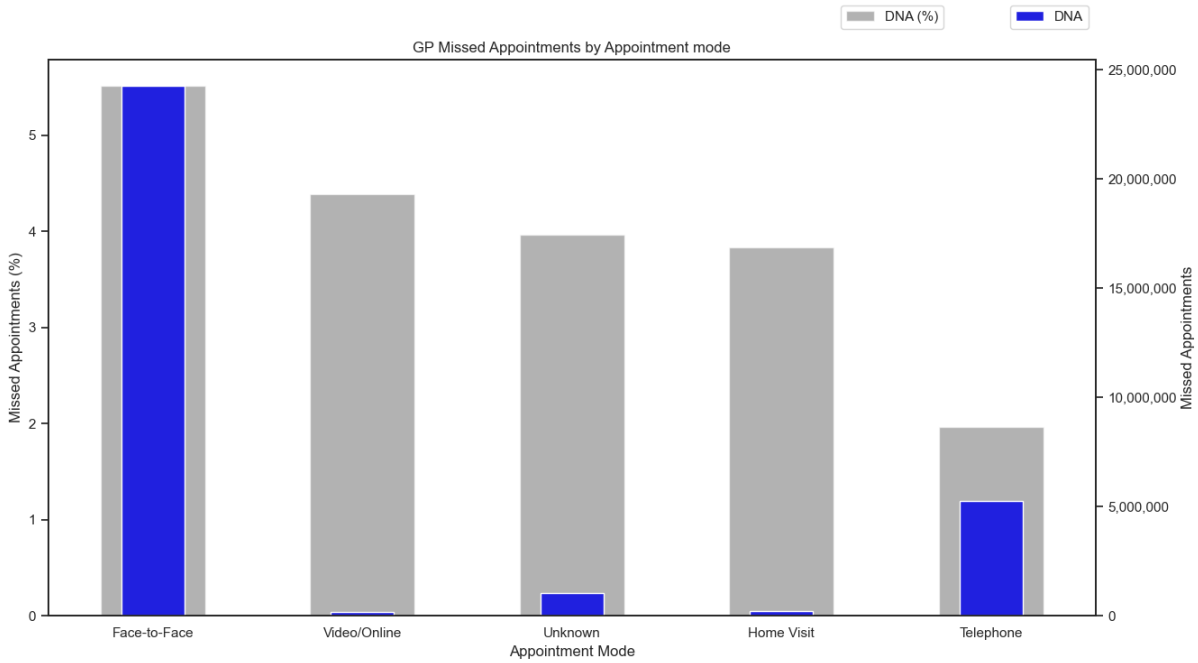
4. Which appointment mode is most likely to be missed?

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ar_mode_as	ar	appointment_mode	Sum	None	None	Pivoted.	Barplot: Missed Appointments by Appointment mode
pivot_ar_ar_mode_as	ar_mode_as	appointment_status	Sum	None	DNA (%)	Calculated columns added: Total, % appointment_status as proportion of total.	

The new df was pivoted in the same fashion as ar_icb_as.

appointment_status	appointment_mode	Attended	DNA	Unknown	Total	Attended (%)	DNA (%)	Unknown (%)
0	Unknown	5336775	209413	329802	5875990	90.823419	3.563876	5.612705
1	Face-to-Face	167801433	6154242	6961552	180917227	92.750390	3.401689	3.847921
2	Video/Online	1744346	52317	76621	1873284	93.117007	2.792796	4.090197
3	Home Visit	1467308	49109	408836	1925253	76.213776	2.550782	21.235443
4	Telephone	181497143	2963854	4597389	189058386	96.000578	1.567692	2.431730

Note that home visits have a 21% “unknown” with regards to appointment status. This is considerable and worthy of further investigation.



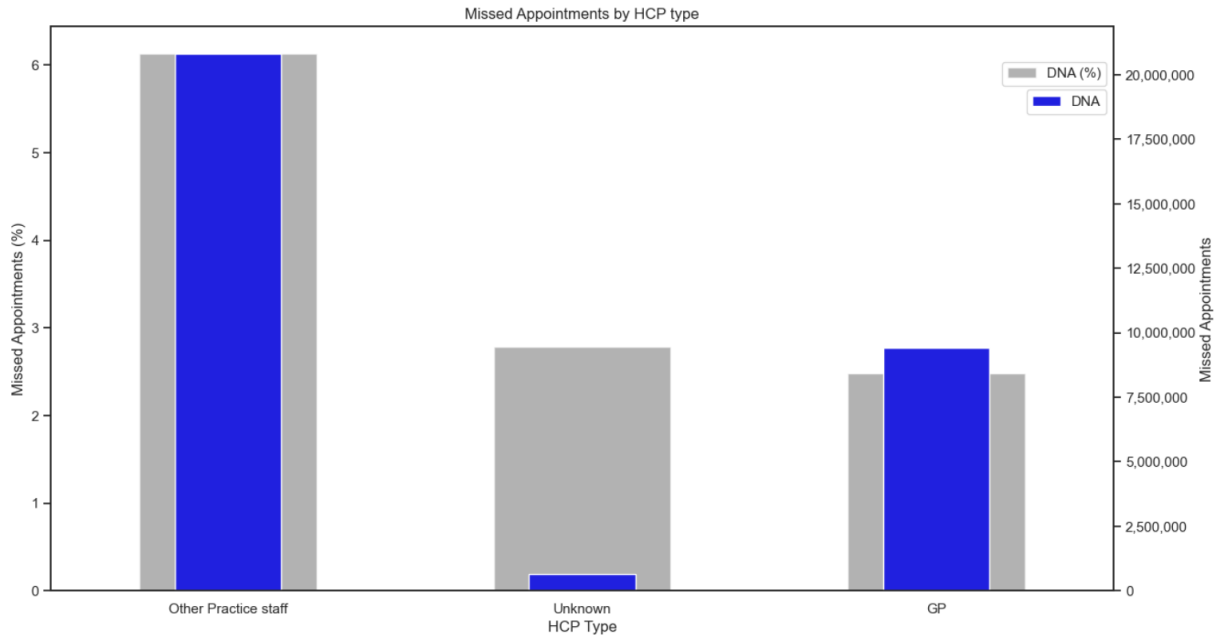
Face to face appointments are the most unattended both in magnitude and proportion.

5. Which hcp type attracts the most missed appointments?

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ar_hcp_as	ar	hcp_type	Sum	None	DNA (%)	Pivoted.	Barplot: Missed Appointments by HCP type.
pivot_ar_hcp_as	ar_hcp_as	appointment_status	Sum	None		Calculated columns added: Total, % appointment_status as proportion of total.	

The new df was pivoted in the same fashion as ar_icb_as.

appointment_status	hcp_type	Attended	DNA	Unknown	Total	Attended (%)	DNA (%)	Unknown (%)
0	Other Practice staff	299129577	20829335	19691623	339650535	88.069809	6.132578	5.797613
1	Unknown	20779294	652963	2071593	23503850	88.408044	2.778111	8.813845
2	GP	357847005	9428935	12374200	379650140	94.257045	2.483585	3.259369



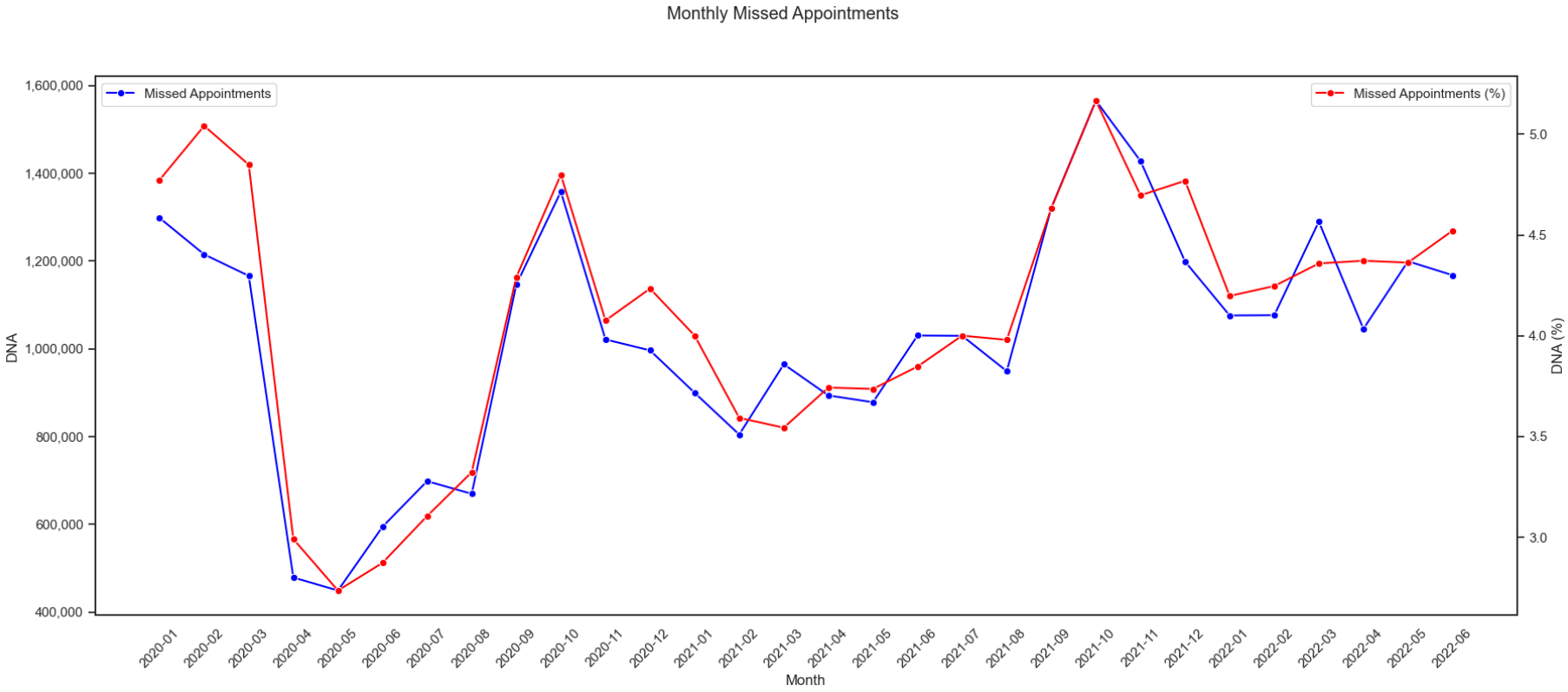
Although Other Practice staff has a lower total number of appointments than GP appointments, they attract significantly more DNAs both in terms of magnitude and proportion. Perhaps this is due to lower severity of the patients' complaint and the perceived importance level of a doctor's appointment vs other practice staff.

6. Does the time of year affect the likelihood of missed appointments?

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ar_am_as	ar	appointment_month	Sum	None	None	Pivoted.	Lineplot: Monthly Missed Appointments
pivot_ar_am_as	ar_am_as	appointment_status	Sum	None	DNA (%)	Calculated columns added: Total, % appointment_status as proportion of total.	

A dual axis lineplot was created to illustrate DNAs and DNAs % to assess for discrepancies and monthly missed appointments.

Diagnostic Analysis using Python.



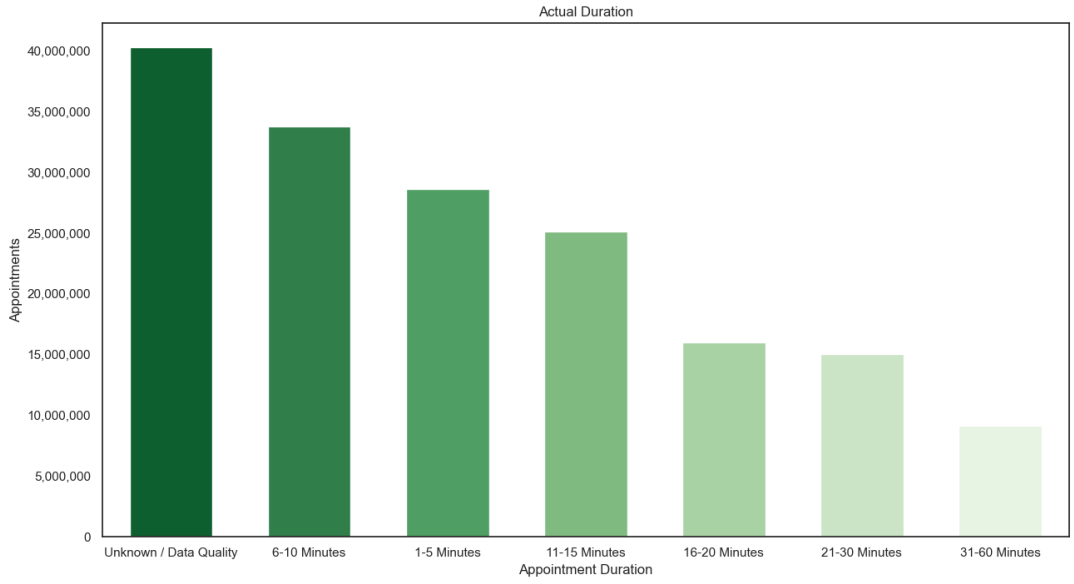
The number of missed appointments and % of missed appointments as a proportion of total appointments are broadly in line. The transition from summer to autumn tends to see a sharp increase in the number and proportion of missed appointments which then tends to sharply decrease as autumn transitions to winter in line with the general monthly trend for all appointments.

Note that from May 2020, the lineplot makes higher highs and higher lows indicating an uptrend in missed appointments.

5.2 Appointments by actual duration

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ad_duration	ad		Sum	None	By count_of_appointments	None	Lineplot: Monthly Appointments Full

	actual_duration	count_of_appointments
6	Unknown / Data Quality	40284086
5	6-10 Minutes	33800815
0	1-5 Minutes	28600865
1	11-15 Minutes	25160882
2	16-20 Minutes	16004247
3	21-30 Minutes	15026365
4	31-60 Minutes	9103432



Approximately 40 million appointments run past the 15-minute mark. There are too many appointments in the unknown/data quality and the data only spans 7 months. This reduces the value of the ad data frame. Measures should be implemented to address the data quality of the actual duration dataset.

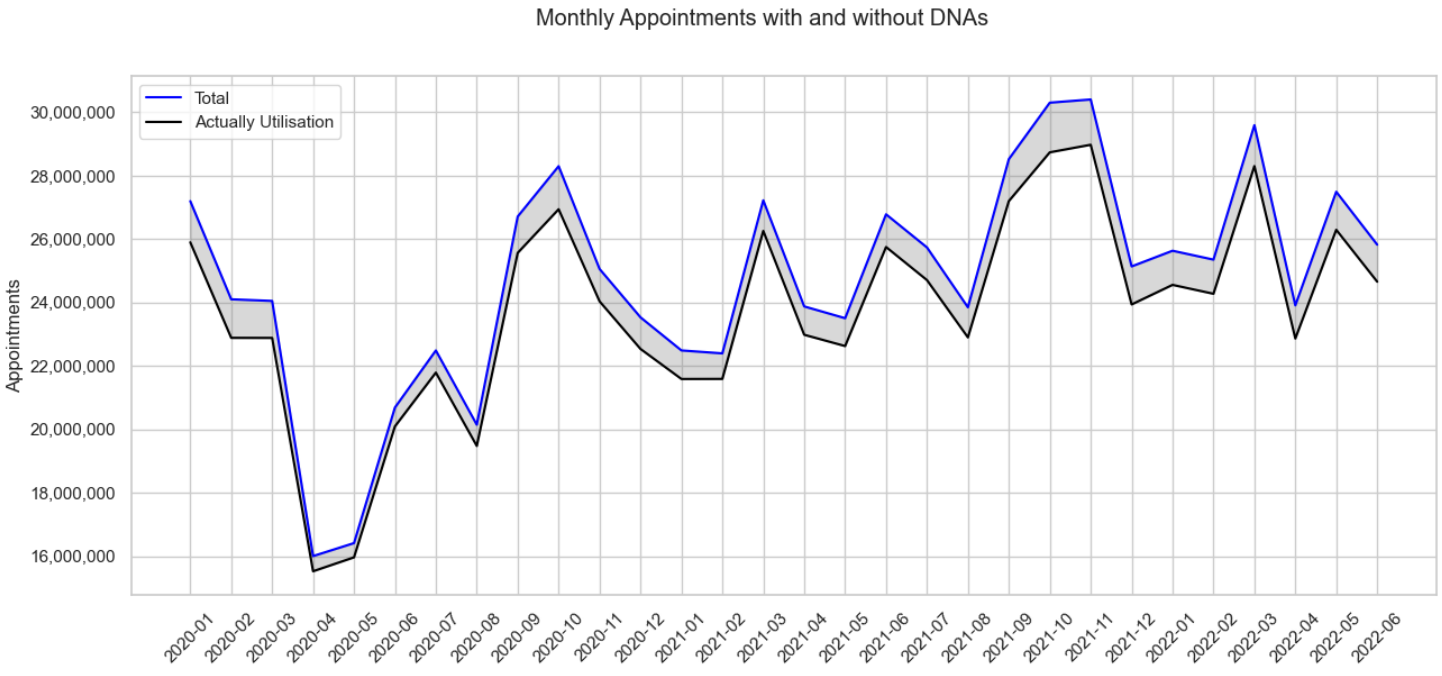
6. Capacity and Utilisation of Resources

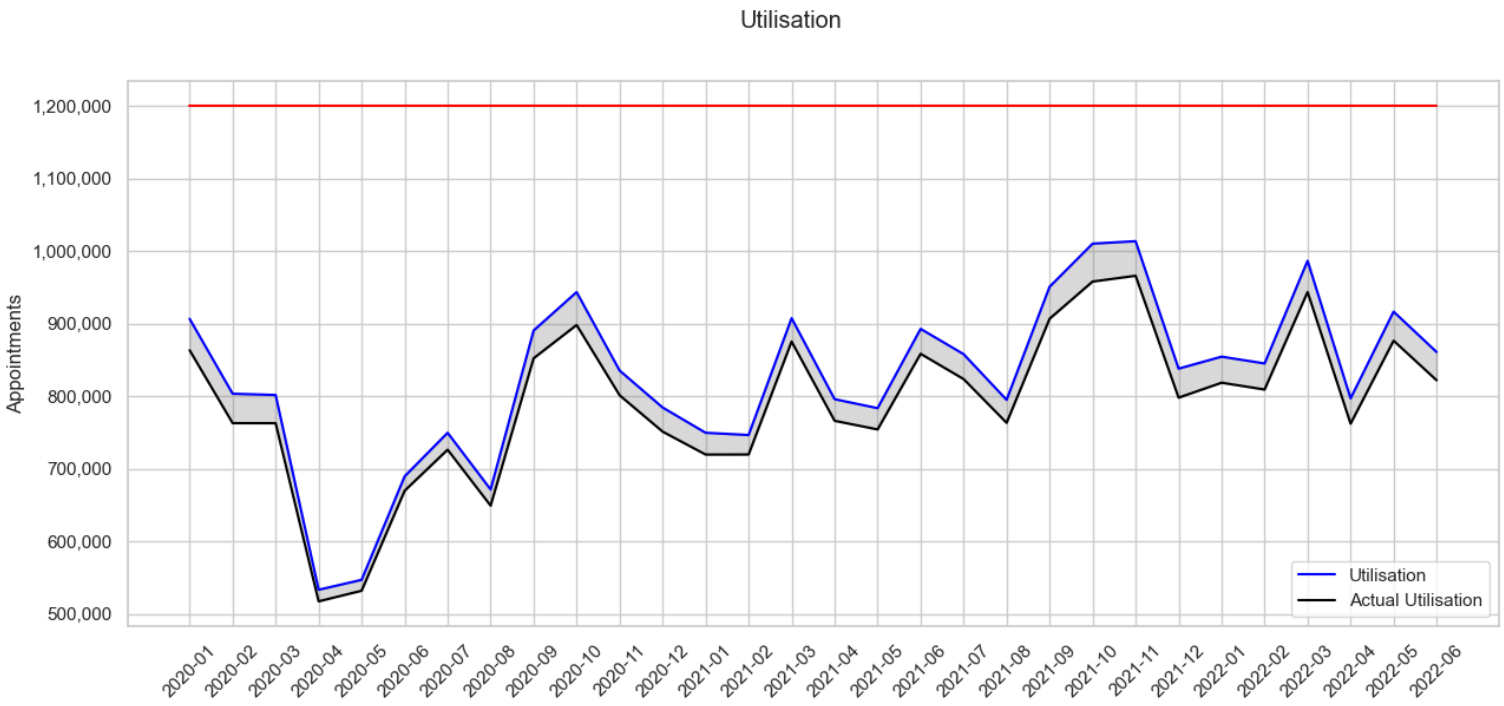
Call the sum_appointment_multi function and create new df(s).

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
ar_monthly_ut	ar	appointment_month, appointment_status	Sum		None	Pivotted	Lineplot: Monthly Appointments with and without DNAs. Utilisation.
pivot_ar_monthly_ut	ar_monthly_ut	appointment_month	Sum		None	5 Columns added: 1. Total (sum of attended, DNA and unknown). 2. Actual Utilisation (Total-DNA) 3. Utilisation (Total /30. 4. Av. Actual Utilisation (Actual Utilisation /30). 5. capacity (1.2 mil)	Lineplot: Monthly Utilisation.

appointment_status	appointment_month	Attended	DNA	Unknown	Total	Actual Utilisation	utilisation	Av. Actual Utilisation	capacity
0	2020-01	24538291	1298269	1362736	27199296	25901027	906643.2	863367.6	1200000
1	2020-02	21640067	1215154	1249400	24104621	22889467	803487.4	762982.2	1200000
2	2020-03	20718865	1166314	2168289	24053468	22887154	801782.3	762905.1	1200000
3	2020-04	13982824	478766	1546291	16007881	15529115	533596.0	517637.2	1200000
4	2020-05	14962850	449057	1005305	16417212	15968155	547240.4	532271.8	1200000

The chart below shows monthly total appointments actual appointments utilised to see patients (Total-DNAs)



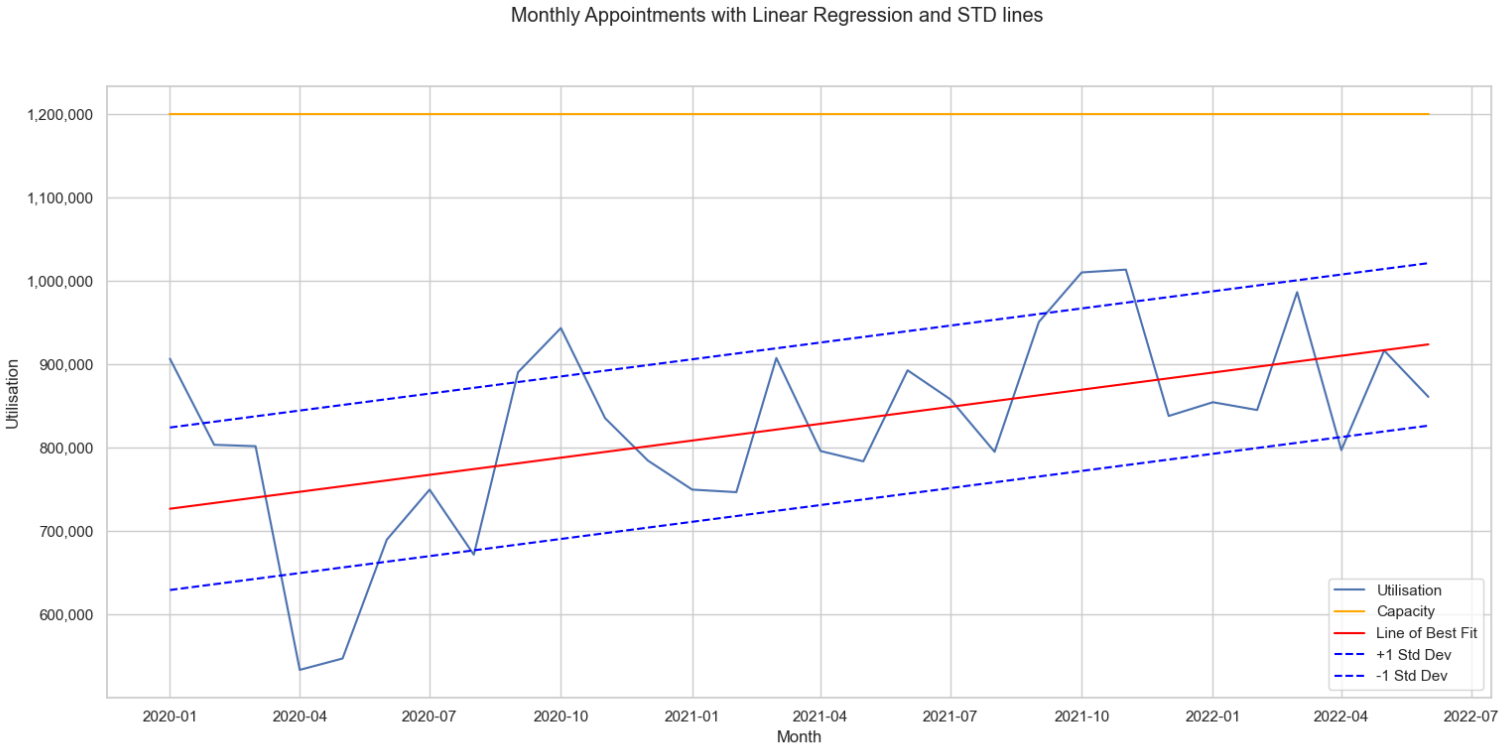


The utilization of appointments is within the capacity of 1.2 million appointments stated by the NHS. The shaded area represents the spare capacity available if missed appointments are reduced.

7. Patterns and Predictions

Call the sum_appointment_multi function and create new df(s).

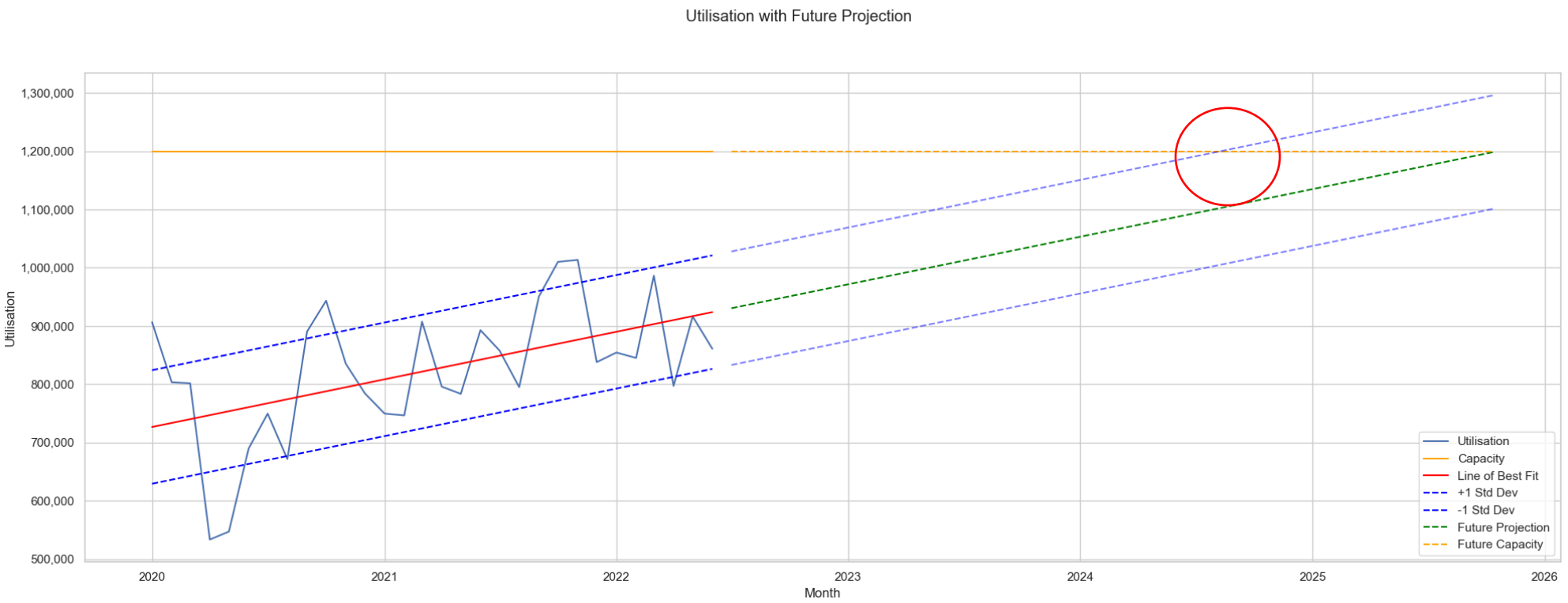
The chart below illustrates the uptrend and adds a +/- standard deviation channel to account for peaks and troughs either side of the LinReg line.



Based on Historic evidence, can we predict when this capacity might be under threat?

The linear regression line was extended to extrapolate a potential convergence point at which the capacity of 1.2 million appointments might be reached.

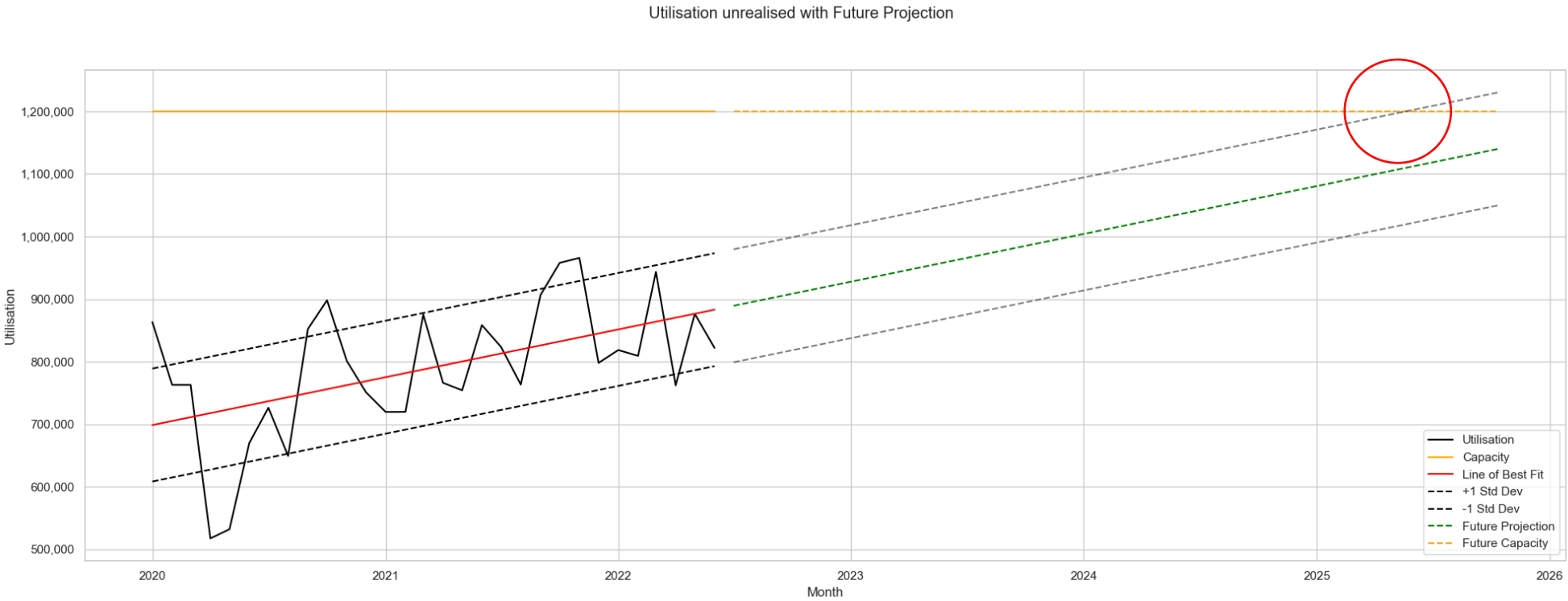
Diagnostic Analysis using Python.



If the Autumn peaks continue to break previous highs and the Spring troughs exceed previous lows, we might expect the capacity of 1.2 million to be reached in Autumn 2024 as illustrated by the upper standard deviation line.

Therefore, the NHS might consider actioning a recruitment drive if the 2022 Autumn figures exceed 1.05 million and the early spring low exceeds 800,000.

If the missed appointments are eradicated reaching capacity will be postponed by a year.



8. Insights and Recommendations

- The national category data suggests a downtrend in monthly appointments, but when merged with the AR data, utilization shows a significant uptrend.
- The NHS faces challenges in seeing patients within two weeks, indicating that capacity is under pressure.
- Analysis of missed appointments reveals ICB locations with over 4.6% missed appointments may need attention.
- Missed appointments are affected by 'time between booking and appointment', appointment mode and HCP type. Strategies like timely reminders, remote appointments and emphasizing non-GP staff appointments can mitigate missed appointments.
- Despite operating at 1.2 million appointments daily, reducing missed appointments could create a utilization-capacity margin.
- Without intervention, utilization may meet capacity in autumn 2024, but strategies could postpone this by a year.
- Short-term capacity increases will be required.

Further Research

- **Staffing Ratios:** Trends in GP and other practice staff to patient or population ratios.
- **Surveys and Interviews:** Workforce perceptions of staffing and capacity. Patients feedback surveys on efficiency and satisfaction.
- Data on vacancy rates and staff departures.
- Financial data and cost benefit analysis.

Appendix

A.1 Full Analytical Approach Plan

A comprehensive and iterative Analytical Approach was developed:

Section	Business questions addressed	Analytical questions posed	Analytical Approach	Dataframe	Observations and Limitations
Date Range and Assimilation	**What is the date range of the provided data sets.**		Create appointment month columns.	All	
		Between what dates were appointments scheduled?	Check and convert all date formats into datetime. Use min(), max() functions to find the earliest and latest dates in the dataset.	All	
		What are the timespans that the datasets cover?	Use np.timedelta64 function to find the timespan and convert to months.	All	
Locations	**What is the number of locations?**	How many locations are there in each location subdivision?	Analyse which datasets hold which location information. Identify a common location type shared by all datasets. Create an index relating ONS codes to actual locations for ease of identification.		There are 3 categories of location: Regional, ICB and sub ICB.
Create a multi-function that enables grouping of count of appointments based on columns in user selected dataframe returning sum of appointments in each group.					
Total appointments	**What is the number of appointments and records per month?**	Which month had the highest number of appointments?	Use multi function to groupby and sort	All	Why records? The national categories dataset has records with 1 appointment all the way to 16590 appointments in one record.
		What was the total number of appointments per month per data source?	Align the dataframes by month, merge and compare the sum of appointments. Check for discrepancies between nc, ar and ad.	All	By counting the number of records we assume these extremes to be equivalent i.e. they each count for 1 within the count function.

Section	Business questions addressed	Analytical questions posed	Analytical Approach	Dataframe	Observations and Limitations
Appointments by service setting, context type and national category.	**What is the number of service settings, context types, national categories, and in the data sets?**	How many service setting, context types, national categories and appointment statuses are there?	Count unique values using nunique() function.	NC	Assess the levels of unmapped and inconsistent mapping and there potential impact on the analysis.
		The count of each category within the groups is limited. Widen the scope of the question to include the number of appointments in each category ranked in order.	Use multi-function to group of count of appointments based on service settings, context type and national category.	NC	
			Enable user inputs to select the column to group by.	NC	
	Which service settings reported the most appointments for a specific period? E.g. North West London from 1 January to 1 June 2022?	What is the sum of appointments grouping for service setting and filtering for specific location and dates?	Update function to enable the user to filter by dates and specific locations codes.	NC	
	What monthly and seasonal trends are evident, based on the number of appointments for service settings, context types, and national categories?	How does the number of appointments in service settings, context types, and national categories change over the 11 month period.	Group datasets for month and the requested elements. Use lineplots to illustrate trends over the 11 month period	NC	
		How does the number of appointments in service settings, context types, and national categories change seasonally.	Compare the requested elements on a seasonal basis. We can only compare seasonality with the 1 year. I.e. are appointments more frequent in winter vs summer.	NC	Seasonal trends cannot be ascertained as we only have 11 months of data in the National Category data set. We can compare seasons within the year and ascertain the busiest season for the data provided.

NHS

Diagnostic Analysis using Python.

Section	Business questions addressed	Analytical questions posed	Analytical Approach	Dataframe	Observations and Limitations
Appointments by HCP type, mode and Time between booking		How do the healthcare professional types differ over time?	Groupby hcp_type and appointment month and plot.	AR	
		Are there changes in terms of appointment mode and the busiest months?	Define busiest months, groupby appointment month, appointment mode	AR	
		Are there any trends in time between booking an appointment?	Groupby time between booking and appointment_month.	AR	
Missed appointment analysis	**What insights can be gained by looking at missed appointments?**	How many appointment statuses are there and how many appointments are in each.	Count unique values using nunique() function. Use function to groupby and count appointments in each appointment status.	AR	
		As a percentage of total appointments, which ICB locations are the poorest performers with regard to missed appointments.	Group ar by location and appointment status. Pivot the result to show Attended, DNA and Unknown as separate columns. Add columns to illustrate % of total appointments. Rank in terms of DNA%.	AR	
		Does the time between booking an appointment and the appointment date have an effect on the likelihood that an appointment will be missed.	As above	AR	
		Which GP appointment mode is most likely to be missed?	As above	AR	
		Which hcp type attracts the most missed appointments?	As above	AR	
		Does the time of year affect the likelihood of missed	As above	AR	

NHS

Diagnostic Analysis using Python.

Section	Business questions addressed	Analytical questions posed	Analytical Approach	Dataframe	Observations and Limitations
Appointment by actual duration		How long do appointments last? Can reducing lengthy appointments increase capacity?	Groupby actual duration and plot as a barplot.		
Twitter	**What are the top trending hashtags (#) on the supplied Twitter data set and how can this be used in the decision-making process?**	Which tweets have user engagement?	Identify popular or influential content by filtering for retweets and favourites.	tweets	Outline how twitter data can be used to benefit the NHS going forward and raise awareness to limitations.
		Which hashtags occur most frequently within the tweets that have passed the user engagement test above?	Verify equivalent hashtags. Identify the number of times hashtags occur in the dataset and rank by frequency of occurrence.	tweets	
Capacity and Utilisation of Resources	**Was there adequate staff and capacity in the networks?**	What added utility can be gained by reducing missed appointments (DNAs)?	Compare the daily utilisation to the 1.2 mil provided by the NHS. Calculate the Actual utilisation by subtracting confirmed DNAs from the total.	ar	
	What was the actual utilisation of resources?		Illustrate graphically.	ar	
Patterns and Predictions	**What are the most important patterns visible in the data relating to the use case?**	What is the overall trend in the data?	Use linear regression to isolate trend.	ar	
		If the trend continues, can we predict when capacity might be threatened?	Extend the trend into the future and assess potential convergence.	ar	
Insights	**What insights can be gained from the data, and what recommendations can be made to the NHS based on these insights?**		Summarise the pertinent insights from the analysis and prepare for presentation.		

A.2 Data Cleaning Checklist

Stage
Data Accuracy
Check for spelling errors.
Values out of range
Incorrect or invalid data types
Blank cells or spaces
Incorrect use of nulls
Incorrect calculations
Mistypes and other format errors.
Data Completeness
Blanks in a required field
Partial or incomplete data
Incorrect or invalid calculations
Missing values.
Data Consistency
Precision
Structure of data
Case sensitivity
Data type
General Consistency
Data Uniqueness
Entries with the same spelling but in a different case.
Entries with different spelling.
Different words but with the same meaning.
Words with alternate representations.
Duplicate Values
Data timeliness
Correct date format and type.

A.3 Dataframe list

New df name	Origin df	Grouped by	Aggregation	Filter	Sort	Further manipulation	Plot
sub_icb_locations	nc	sub_icb_location_name	Sum	None	Top 5 by count_of_appointments		Barplot: Top 5 sub icb locations
icb_locations	nc	icb_location_name	Sum	None	Top 5 by count_of_appointments		Barplot: Top 5 icb locations
regions	ad	region_ons_code	Sum	None	Top 5 by count_of_appointments		Barplot: Top 5 regional locations
nc_monthly	nc	appointment_month	Sum	None	Sorted by count_of_appointments		
ar_monthly	ar	appointment_month	Sum	None	None	Merged ar_monthly, nc_monthly and ad_monthly	Lineplot: Monthly Appointments per Data Set
nc_monthly	nc	appointment_month	Sum	None	None	Merged ar_monthly, nc_monthly and ad_monthly	Lineplot: Monthly Appointments per Data Set
ad_monthly	ad	appointment_month	Sum	None	None	Merged ar_monthly, nc_monthly and ad_monthly	Lineplot: Monthly Appointments per Data Set
nc_monthly	nc	appointment_month	Sum	None	None	Merged ar_monthly, nc_monthly and ad_monthly	Lineplot: Monthly Appointments per Data Set
ad_monthly	ad	appointment_month	Sum	None	None	Merged ar_monthly, nc_monthly and ad_monthly	Lineplot: Monthly Appointments per Data Set
nc_ss	nc	appointment_month	Sum	None	None	Count unique service settings	Barplot: Service settings by count_of _appointments
		service_setting					Lineplot: appointment_month by count_of_appointments
							Barplot: Seasonal Service settings by count_of _appointments
nwl_service_settings	nc	service_setting	Sum	Location: E54000027	Sorted by count_of_appointments		Barplot: Appointments by Service Settings between
				Dates: 2022-01-01 to 2022-06-01			01 Jan 2022 and 01 June 2022 for North West London
nc_ct	nc	appointment_month	Sum	None	None		Barplot: context_type by count_of _appointments
		context_type					Lineplot: appointment_month by count_of_appointments
							Barplot: Seasonal context_type by count_of _appointments
nc_nc	nc	appointment_month	Sum	None	None	**Split into 4 tiers to better visualise in lineplot.**	Barplot: National category by count_of _appointments
		national_category					Lineplot: appointment_month by count_of_appointments
							Barplot: Seasonal National category by count_of _appointments

ar_hcp	ar	appointment_month	Sum	None	None		Lineplot: Monthly Appointments by HCP Type
		hcp_type					
ar_mode	ar	appointment_month	Sum	None	None	Lineplot split to enable higher resolution veiwing.	Lineplot: Monthly Appointments by appointment_mode
		appointment_mode					
ar_booking	ar	appointment_month	Sum	None	None	Pivoted to create pivot_ar_booking	Lineplot: Monthly Appointments by Booking Time
		time_between_book_and_appointment		None			
pivot_ar_booking	ar_booking	Timeframes grouped to Same day, 48 hours, Within 2 weeks and later than 2 weeks.	Sum	None	None	% calculations made on total appointments.	Lineplot: Monthly Appointments by Booking Time Abridged
pivot_ar_booking_perc	ar_booking	Timeframes grouped to Same day (%), 48 hours (%), Within 2 weeks (%) and later than 2 weeks (%).	Sum	None	None		Lineplot: Monthly Appointments by Booking Time %
average_wait	pivot_ar_booking_perc		Mean	None	None		
ar_as	ar	appointment_status	Sum	None	Sorted by count_of_appointments		Barplot: Appointments by Appointment Status
ar_icb_as	ar	icb_location_name	Sum	None	None	Pivoted.	Barplot: Missed Appointments by ICB
pivot_ar_icb_as	ar_icb_as	appointment_status	Sum	None	By Top 10 DNA & Top 10 DNA%	Calculated columns added: Total, % appointment_status as proprtion of total.	Barplot: Missed Appointments as % of Total by ICB
ar_tb_as	ar	time_between_book_and_appointment	Sum	None	None	Pivoted.	Barplot: Missed Appointments by Booking Time
pivot_ar_tb_as	ar_tb_as	appointment_status	Sum	None	DNA (%)	Calculated columns added: Total, % appointment_status as proportion of total.	
ar_mode_as	ar	appointment_mode	Sum	None	None	Pivoted.	Barplot: Missed Appointments by Appointment mode
pivot_ar_ar_mode_as	ar_mode_as	appointment_status	Sum	None	DNA (%)	Calculated columns added: Total, % appointment_status as proportion of total.	
ar_hcp_as	ar	hcp_type	Sum	None	DNA (%)	Pivoted.	Barplot: Missed Appointments by HCP type.
pivot_ar_hcp_as	ar_hcp_as	appointment_status	Sum	None		Calculated columns added: Total, % appointment_status as proportion of total.	
ar_am_as	ar	appointment_month	Sum	None	None	Pivoted.	Lineplot: Monthly Missed Appointments
pivot_ar_am_as	ar_am_as	appointment_status	Sum	None	DNA (%)	Calculated columns added: Total, % appointment_status as	

tweets	tweets.csv	None	None		None	Filtered as below	None
tweets_popular	tweets	None	None	tweet_retweet_count > 0 or tweet_favorite_count > 0	None		None
tweets_text	tweets_popular	None	None	tweet_full_text only.	None		None
tags_count	tags_series	None	None	Filtered by for loop	By value_count of # occurrence.	Top 30 printed	None
tags_count_above_10	tags_count	None	None	Filtered for count above 10.	None	Searched for equivalent tags and tags mapped.	Barplot: Top Trending Hashtags
ar_monthly_ut	ar	appointment_month	Sum	date_filter = 'y' start_date = '2021-08' end_date = '2022-06'	DNA (%)	Added 2 columns. Utilisation = monthly appointments / 30 and capacity 1,200,000.	Lineplot: Monthly Appointments
ar_agg	ar	appointment_month	Sum	date_filter = 'y' start_date = '2021-08' end_date = '2022-06'	None		
		hcp_type					
		appointment_status					
		appointment_mode					
		time_between_book_and_appointment					
ar_monthly_ut	ar	appointment_month, appointment_status	Sum		None	Pivotted	Lineplot: Monthly Appointments, Utilisation.
pivot_ar_monthly_ut	ar_monthly_ut	appointment_month	Sum		None	5 Columns added: 1. Total (sum of attended, DNA and unknown). 2. Actual Utilisation (Total-DNA) 3. Utilisation (Total /30. 4. Av. Actual Utilisation (Actual Utilisation /30). 5. capacity (1.2 mil)	Lineplot: Monthly Utilisation.
nc_general	nc	appointment_month	Sum				Lineplot: Monthly Appointments Full
ad_duration	ad		Sum	None	By count_of_appointments	None	Lineplot: Monthly Appointments Full

A.4 Location Index

icb_ons_code	icb_location_name
E54000008	NHS Cheshire and Merseyside ICB
E54000010	NHS Staffordshire and Stoke-on-Trent ICB
E54000011	NHS Shropshire Telford and Wrekin ICB
E54000013	NHS Lincolnshire ICB
E54000015	NHS Leicester Leicestershire and Rutland ICB
E54000018	NHS Coventry and Warwickshire ICB
E54000019	NHS Herefordshire and Worcestershire ICB
E54000022	NHS Norfolk and Waveney ICB
E54000023	NHS Suffolk and North East Essex ICB
E54000024	NHS Bedfordshire Luton and Milton Keynes ICB
E54000025	NHS Hertfordshire and West Essex ICB
E54000026	NHS Mid and South Essex ICB
E54000027	NHS North West London ICB
E54000028	NHS North Central London ICB
E54000029	NHS North East London ICB
E54000030	NHS South East London ICB
E54000031	NHS South West London ICB
E54000032	NHS Kent and Medway ICB
E54000034	NHS Frimley ICB
E54000036	NHS Cornwall and The Isles Of Scilly ICB
E54000037	NHS Devon ICB
E54000038	NHS Somerset ICB
E54000039	NHS Bristol North Somerset and South Glouceste...
E54000040	NHS Bath and North East Somerset Swindon and W...
E54000041	NHS Dorset ICB
E54000042	NHS Hampshire and Isle Of Wight ICB
E54000043	NHS Gloucestershire ICB
E54000044	NHS Buckinghamshire Oxfordshire and Berkshire ...
E54000048	NHS Lancashire and South Cumbria ICB
E54000050	NHS North East and North Cumbria ICB
E54000051	NHS Humber and North Yorkshire ICB
E54000052	NHS Surrey Heartlands ICB
E54000053	NHS Sussex ICB
E54000054	NHS West Yorkshire ICB
E54000055	NHS Birmingham and Solihull ICB
E54000056	NHS Cambridgeshire and Peterborough ICB
E54000057	NHS Greater Manchester ICB
E54000058	NHS Derby and Derbyshire ICB
E54000059	NHS Northamptonshire ICB
E54000060	NHS Nottingham and Nottinghamshire ICB
E54000061	NHS South Yorkshire ICB
E54000062	NHS Black Country ICB

A.5 The Multi-function Aggregation Function

User Inputs

When the function is called, non-technical users are prompted to input various choices as to how they would like the df grouped, filtered, and sorted:

```
# Multi-functional sum_appointment function.

def sum_appointments_multi():
    new_df_name = input("Please name the new df: ")
    from_df_name = input("From which df (nc, ar, or ad): ")
    group_by1 = input("Input column to groupby? ")
    group_by2_option = input("Would you like a 2nd groupby? y/n: ")
    group_by2 = None
    if group_by2_option.lower() == 'y':
        group_by2 = input("Input 2nd column to groupby?? ")
    location_filter = input("Filter by location? y/n: ")
    location = None
    date_filter = input("Choose specific dates? y/n: ")
    start_date = None
    end_date = None
    sort = None
```

The user input for their selected df (string value) is converted to the df name.

```
    # Convert user input string to dataframe name.

    if from_df_name == 'nc':
        from_df = nc
    elif from_df_name == 'ar':
        from_df = ar
    elif from_df_name == 'ad':
        from_df = ad
    else:
        raise ValueError("Invalid DataFrame name")
```

Location and date selection

If the user selects location and date filtering, the following code is executed:

```
""" There are 4 combinations of user input. Both location and dates, location but no dates,
dates but no loction and neither dates nor location """

# If user chooses date filter - Convert start_date and end_date to datetime objects

# Filter for location and dates

if date_filter.lower() == 'y' and location_filter.lower() == 'y':
    if from_df_name == 'ar':
        location = input("Enter location code: ")
        start_date = input("Start month (YYYY-MM): ")
        end_date = input("End month (YYYY-MM): ")
 # Convert to datetime
        start_date = pd.to_datetime(start_date)
        end_date = pd.to_datetime(end_date)
        # Filter
        filtered_df = from_df[(from_df['icb_ons_code'] == location) &
                              (pd.to_datetime(from_df['appointment_month']) >= start_date) &
                              (pd.to_datetime(from_df['appointment_month']) <= end_date)]
    else:
        location = input("Enter location code: ")
        start_date = input("Start date (YYYY-MM-DD): ")
        end_date = input("End date (YYYY-MM-DD): ")
        # Convert to datetime
        start_date = pd.to_datetime(start_date)
        end_date = pd.to_datetime(end_date)
        # Filter
        filtered_df = from_df[(from_df['icb_ons_code'] == location) &
                      (from_df['appointment_date'] >= start_date) &
                      (from_df['appointment_date'] <= end_date)]
```

If the user selects date filtering only, the following code is executed:

```
# Filter for just dates

elif date_filter.lower() == 'y' and location_filter.lower() == 'n':
    if from_df_name == 'ar':
        start_date = input("Start month (YYYY-MM): ")
        end_date = input("End month (YYYY-MM): ")
        # Convert to datetime
        start_date = pd.to_datetime(start_date)
        end_date = pd.to_datetime(end_date)
        # Filter
        filtered_df = from_df[
            (pd.to_datetime(from_df['appointment_month']) >= start_date) &
            (pd.to_datetime(from_df['appointment_month']) <= end_date)]
    else:
        start_date = input("Start date (YYYY-MM-DD): ")
        end_date = input("End date (YYYY-MM-DD): ")
        # Convert to datetime
        start_date = pd.to_datetime(start_date)
        end_date = pd.to_datetime(end_date)
        # Filter
        filtered_df = from_df[(from_df['appointment_date'] >= start_date) &
                      (from_df['appointment_date'] <= end_date)]
```

If the user selects only location filtering or no filter, the following codes executes.

```
# Filter for just location

elif date_filter.lower() == 'n' and location_filter.lower() == 'y':
    location = input("Enter location code: ")
    filtered_df = from_df[from_df['icb_ons_code'] == location]

# General groupby no filters

else:
    filtered_df = from_df
```

Once the request has exited the filter section the df is grouped by either 2 groups or 1 as requested by the user.

```
# Group and sum appointments based on user-defined columns

if group_by2_option.lower() == 'y':
    grouped_df = filtered_df.groupby([group_by1, group_by2])['count_of_appointments'].sum().reset_index()
else:
    grouped_df = filtered_df.groupby(group_by1)['count_of_appointments'].sum().reset_index()
```

The user can select to sort either into top 5, to sort normally or not to sort (.head(10)). Finally, the grouped df is returned.

```
# If user selects sort data.

sort = input("Sort data? y/n: ")

if sort.lower() == 'y':
    grouped_df.sort_values(by=['count_of_appointments'], inplace=True, ascending=False)
    globals()[new_df_name] = grouped_df # Enables the new df to be used outside of the function.
    top_5 = input("Top 5? y/n: ")

# Output according to user's choice of Top 5 or full data

    if top_5.lower() == "y":
        print("Number of records: ",grouped_df[group_by1].nunique())
        return grouped_df.head()
    else:
        print("Number of records: ",grouped_df[group_by1].nunique())
        return grouped_df

# If sorting isn't selected, the output is in the order of occurrence in the selected dataframe.

else:
    globals()[new_df_name] = grouped_df # Enables the new df to be used outside of the function.
    print("Number of records: ",grouped_df[group_by1].nunique())
    return grouped_df.head(10)
```

Note 1: The use of globals allows the newly created dataframe (named by the user) to be used outside the function.

Note 2: User inputs have been suppressed in the notebook to allow the technical user (marker) to run the code without having to input the parameters manually each time.