Analysis for the NHS

Utilisation of services and missed appointments

Technical report

Prepared by Lilliana Golob

Contents

Background

The NHS wants to know how to budget and allocate resources. The stakeholders are senior managers (admin) and medical practitioners.

The business problem can be summarised as:

1. Does it need to increase capacity and employ more resources?
2. Can it meet demand with its existing resources?

My hypothesis is the latter is true.

Questions I'd like to answer

1. Which healthcare professionals deliver the most appointments?
2. Which appointment types are most popular?
3. Are there months that are busier?
4. How does duration vary from healthcare professional?
5. Why are people missing appointments? Is it because of professional type or time between booking?

Considerations

The data is inconsistent because of how it's captured at practice level.

- Count of appointments is an estimate.
- Recording appointment duration varies by practice.
- Online consults may be logged as face-to-face.
- Not all home visits are logged.

Data wrangling

Data validation

There are no missing values in the three main dataframes **ad**, **ar**, and **nc**. 167 are missing in **tweets**, but we can get #hashtags from *tweet_full_text*.

There is one common column across **ad**, **ar**, and **nc** (*icb_ons_code*), but no unique primary key so I won't merge them.

Functions

I created functions for data validation, IQR and outliers. I used a separate cell for .head( ) because the output is easier to read.

```
def review_data(df):
    """
    Review the data in a dataframe returning datatypes,
    missing values, descriptive statistics,
    duplicates and unique values.
    """

    print("COLUMN NAMES AND DATA TYPES")
    info_df = df.info()
    print("")

    print("MISSING VALUES")
    missing_df = df.isnull().sum()
    print(missing_df)
    print("")

    print("DESCRIPTIVE STATS")
    desc_stats = df.describe()
    print(desc_stats)
    print("")

    print("COUNT OF DUPLICATES")
    dup_count = df.duplicated().sum
    print(dup_count)
    print("")

    print("COUNT UNIQUE")
    unique_count = df.nunique()
    print(unique_count)
    print("")

    return info_df, missing_df, desc_stats, dup_count, unique_count
```

```
def calculate_IQR(df):
    """
    Calculates IQR and number of outliers with a threshold of 1.5,
    for 'count_of_appointments' column in specified dataframe df,
    and prints the IQR and len of outliers.
    """

    Q1 = Q1 = df['count_of_appointments'].quantile(0.25)
    Q3 = df['count_of_appointments'].quantile(0.75)
    IQR = Q3 - Q1

    # identify outliers
    threshold = 1.5
    outliers = df[(df['count_of_appointments'] < Q1 - threshold * IQR) | (df['count_of_appointments'] > Q3 + thresho

    return print(f"IQR: {(IQR)}"), print(f"Number of outliers: {len(outliers)}")
```

Descriptive stats

Looking at IQR, we have outliers; but the quantity compared to the count is high. The metadata mentioned inconsistent data capture so I'll leave them and recommend investigation.

count_of_appointments	**ad**	**ar**	**nc**
count	137,793	596,821	817,394
std	1,546	5,856	1,084
mean	1,219	1,244	362
min	1	1	1
max	15,400	211,265	16,590
Q1 (25%)	194	7	7
Q2 (50%)	696	47	25
Q3 (75%)	1,621	308	128
IQR	1,427	301	121
Quantity of outliers	9,291	97,348	147,958

Data cleaning

Date

I changed *appointment_date* in **ad** to datetime. The range for *appointment_date* is consistent across dataframes (1 Dec 2021 to 30 Jun 2022), but inconsistent for *appointment_month*.

```python
# Find min and max for appointment_date across all dataframes

print(f"ad min appointment date: {ad['appointment_date'].min()}")
print(f"ad max appointment date: {ad['appointment_date'].max()}")
print("")
print(f"nc min appointment date: {ad['appointment_date'].min()}")
print(f"nc max appointment date: {nc['appointment_date'].max()}")
```

```
ad min appointment date: 2021-12-01 00:00:00
ad max appointment date: 2022-06-30 00:00:00

nc min appointment date: 2021-12-01 00:00:00
nc max appointment date: 2022-06-30 00:00:00
```

```python
# Comparing range of dates in appointment_month

print(f"ar min appointment month: {ar['appointment_month'].min()}")
print(f"ar max appointment month: {ar['appointment_month'].max()}")
print("")
print(f"nc min appointment month: {nc['appointment_month'].min()}")
print(f"nc max appointment month: {nc['appointment_month'].max()}")
```

```
ar min appointment month: 2020-01
ar max appointment month: 2022-06

nc min appointment month: 2021-08
nc max appointment month: 2022-06
```

Duplicates

There are 21,604 duplicates in **ar** (3.6%). The statss vary slightly if we remove them:

- mean: 1244.6 vs 1290.8
- std: 5856.8 vs 5960.8
- IQR1: 7 vs 8
- IRQ3: 308 vs 332

The metadata mentioned some cleaning, so I'm keeping them and prefer to further investigate with the stakeholders.

Exploratory analysis

Initial exploration looked primarily at records by categories.

Locations

There are 106 locations, 42 boards and 7 regions. NHS North West London has the most appointments (6,976,986) and is one of five with the most records. Kent and Medway is in the top five for number of records and appointments.

For this report we'll focus analysis across all locations.

```
appts_location = ad.groupby(['sub_icb_location_name'])['count_of_appointments'].sum().sort_values(ascending = False)
appts_location.head(5)
```

```
sub_icb_location_name
NHS North West London ICB - W2U3Z              6976986
NHS North East London ICB - A3A8R              5341883
NHS Kent and Medway ICB - 91Q                  5209641
NHS Hampshire and Isle Of Wight ICB - D9Y0V    4712737
NHS South East London ICB - 72Q                4360079
Name: count_of_appointments, dtype: int64
```

```
print(f"Number of records by {ad['sub_icb_location_name'].value_counts()}")
```

```
Number of records by sub_icb_location_name
NHS Norfolk and Waveney ICB - 26A                        1484
NHS Kent and Medway ICB - 91Q                            1484
NHS North West London ICB - W2U3Z                        1484
NHS Bedfordshire Luton and Milton Keynes ICB - M1J4Y     1484
NHS Greater Manchester ICB - 14L                         1484
```

Common appointments

Face-to-face with 'Other practice staff' then with GPs are the most common appointments.

Same day are also most common.

```
mode_hcptype = ar.groupby(['hcp_type'])['appointment_mode'].value_counts()
print(f"Number of records by: {mode_hcptype}")
```

```
Number of records by: hcp_type             appointment_mode
GP                   Face-to-Face        71672
                     Telephone           69344
                     Home Visit          36894
                     Video/Online        29713
                     Unknown             18413
Other Practice staff Face-to-Face        72844
                     Telephone           69851
                     Home Visit          49331
                     Video/Online        29356
                     Unknown             20175
Unknown              Unknown             40559
                     Face-to-Face        35894
                     Telephone           27288
                     Home Visit          24969
                     Video/Online          518
Name: count, dtype: int64
```

```
# By number of appointments

time_book_appts = ar.groupby(['time_between_book_and_appointment'])['count_of_appointments'].sum().sort_values(ascen
print(f"Appointments available by {time_book_appts}")
```

```
Appointments available by time_between_book_and_appointment
Same Day                342747171
2 to 7 Days             153794531
8  to 14 Days            86846519
1 Day                    67716097
15  to 21 Days           42710574
22  to 28 Days           25536541
More than 28 Days        23050987
Unknown / Data Quality     402105
Name: count_of_appointments, dtype: int64
```

```
# By number of records

print(f"Number of records by: {ar['time_between_book_and_appointment'].value_counts()}")
```

```
Number of records by: time_between_book_and_appointment
Same Day                95502
2 to 7 Days             92409
1 Day                   88957
8  to 14 Days           82698
15  to 21 Days          73666
22  to 28 Days          68755
More than 28 Days       65147
Unknown / Data Quality  29687
Name: count, dtype: int64
```

Missed appointments

Most missed appointment (DNA) records are face-to-face with 'Other practice staff', the most common appointment type.

```
mode_status_type = ar.groupby(['appointment_status','appointment_mode', 'hcp_type'])['appointment_mode'].value_count
print(f"Number of records by: {mode_status_type}")
```

```
Number of records by: appointment_status  appointment_mode  hcp_type
Attended            Face-to-Face      GP                    24729
                                      Other Practice staff  24972
                                      Unknown               14777
                    Home Visit        GP                    16765
                                      Other Practice staff  20749
                                      Unknown               11094
                    Telephone         GP                    24171
                                      Other Practice staff  24200
                                      Unknown               12425
                    Unknown           GP                     7039
                                      Other Practice staff   7723
                                      Unknown               14764
                    Video/Online      GP                    15164
                                      Other Practice staff  13208
                                      Unknown                 357
DNA                 Face-to-Face      GP                    22879
                                      Other Practice staff  23334
                                      Unknown                7557
                    Home Visit        GP                     6662
                                      Other Practice staff  11097
                                      Unknown                3627
                    Telephone         GP                    21511
                                      Other Practice staff  22052
                                      Unknown                6133
                    Unknown           GP                     5419
                                      Other Practice staff   6165
                                      Unknown               12200
                    Video/Online      GP                     6317
                                      Other Practice staff   8345
                                      Unknown                  62
Unknown             Face-to-Face      GP                    24064
                                      Other Practice staff  24538
                                      Unknown               13560
                    Home Visit        GP                    13467
```

Surprisingly, a longer time between booking and appointment did not result in higher DNA.

Most missed appointments were booked on the same day. Records for 28+ days in advance had the lowest number of missed appointments.

While 28+ days is a lower portion of DNA records, relative to the number of same type of records, it's only 2.6% lower than same day DNA records.

Same day	*28+ days*
Total DNA records = 163,360	Total DNA records = 163,360
Same day total records = 95,502	28+ days total records = 65,147
Same day DNA records = 28,390	28+ days DNA records = 17,638
29.7% of same day records	***27.1% of all 28+ days records***
17.4% of all DNA records	*10.8% of all DNA records*

```
status_time_book = ar.groupby(['appointment_status','time_between_book_and_appointment'])['appointment_status'].valu
print(f"Number of records by: {status_time_book}")
```

```
Number of records by: appointment_status  time_between_book_and_appointment
Attended             1 Day                              34432
                     15  to 21 Days                     28602
                     2 to 7 Days                        34606
                     22  to 28 Days                     26788
                     8  to 14 Days                      31881
                     More than 28 Days                  24689
                     Same Day                           35673
                     Unknown / Data Quality             15466
DNA                  1 Day                              25186
                     15  to 21 Days                     20327
                     2 to 7 Days                        26735
                     22  to 28 Days                     18718
                     8  to 14 Days                      22974
                     More than 28 Days                  17638
                     Same Day                           28390
                     Unknown / Data Quality              3392
Unknown              1 Day                              29339
                     15  to 21 Days                     24737
                     2 to 7 Days                        31068
                     22  to 28 Days                     23249
                     8  to 14 Days                      27843
                     More than 28 Days                  22820
                     Same Day                           31439
                     Unknown / Data Quality             10829
Name: count, dtype: int64
```

```
print(f"Number of records by: {ar['time_between_book_and_appointment'].value_counts()}")
```

```
Number of records by: time_between_book_and_appointment
Same Day                 95502
2 to 7 Days              92409
1 Day                    88957
8  to 14 Days            82698
15  to 21 Days           73666
22  to 28 Days           68755
More than 28 Days        65147
Unknown / Data Quality   29687
Name: count, dtype: int64
```

By date

I created functions for appointments/ records by *service_setting* by location and date.

```python
# Function to find number of reords for specific location and date range

def records_per_location():
    """
    Counts the number of records by 'service_setting',
    for a specific'sub_icb_location_name' (location_name),
    which can be a partial description and ignores case sensitivity,
    for a date range specified by the user (start) (end) that must be entered as yyyy-mm-dd,
    a new smaller dataframe (sr) is created based on nc dataframe,
    the result is held in a new filtered dataframe (srf),
    and displays the number (count) of records for each 'service_setting' grouped by 'sub_icb_loation_name'.
    """

    location = input("Enter the location: ")
    start = input("Enter the start date as yyyy-mm-dd (eg 2021-12-31): ")
    end = input("Enter the end date as yyyy-mm-dd (eg 2021-12-31): ")

    sr = nc[['service_setting', 'appointment_date', 'count_of_appointments', 'sub_icb_location_name']].copy()

    srf = sr[(sr['appointment_date'] >= start) &
             (sr['appointment_date'] <= end) &
             (sr['sub_icb_location_name'].str.contains(location))]

    return print(f"\n"
                 f"Number of records between {start} and {end} by: \n"
                 f"{srf.groupby(['sub_icb_location_name', 'service_setting'])['count_of_appointments'].count()}")
```

```python
# Function to find number of appointments for a location and date range

def appointments_per_location():
    """
    Counts the number of appointments available by 'service_setting',
    by a 'sub_icb_location_name' specified by the user (location),
    which can be a partial description and ignores case sensitivity,
    for a date range specified by the user (start) (end) that must be entered as yyyy-mm-dd,
    a new smaller dataframe (sr) is created based on nc dataframe,
    the result is held in a new filtered dataframe (srf),
    and displays the sum of 'count_of_appointments' for each 'service_setting' grouped by 'sub_icb_loation_name'.
    """

    location = input("Enter the location: ")
    start = input("Enter the start date as yyyy-mm-dd (eg 2021-12-31): ")
    end = input("Enter the end date as yyyy-mm-dd (eg 2021-12-31): ")

    sr = nc[['service_setting', 'appointment_date', 'count_of_appointments', 'sub_icb_location_name']].copy()

    srf = sr[(sr['appointment_date'] >= start) &
             (sr['appointment_date'] <= end) &
             (sr['sub_icb_location_name'].str.contains(location))]

    return print(f"\n"
                 f"Total appointments available between {start} and {end} by: \n"
                 f"{srf.groupby(['sub_icb_location_name', 'service_setting'])['count_of_appointments'].sum()}")
```

```python
# Use function to find number of appointments by service setting

appointments_per_location()
```

```
Enter the location: North West London
Enter the start date as yyyy-mm-dd (eg 2021-12-31): 2022-01-01
Enter the end date as yyyy-mm-dd (eg 2021-12-31): 2022-06-01

Total appointments available between 2022-01-01 and 2022-06-01 by:
sub_icb_location_name              service_setting
NHS North West London ICB - W2U3Z  Extended Access Provision      98159
                                   General Practice             4804239
                                   Other                         152897
                                   Primary Care Network          109840
                                   Unmapped                      391106
Name: count_of_appointments, dtype: int64
```

```python
appointments_per_location()
```

```
Enter the location: Kent
Enter the start date as yyyy-mm-dd (eg 2021-12-31): 2022-01-01
Enter the end date as yyyy-mm-dd (eg 2021-12-31): 2022-06-01

Total appointments available between 2022-01-01 and 2022-06-01 by:
sub_icb_location_name          service_setting
NHS Kent and Medway ICB - 91Q  Extended Access Provision      25841
                               General Practice             3895574
                               Other                          99496
                               Primary Care Network           58191
                               Unmapped                       92520
Name: count_of_appointments, dtype: int64
```

For the top two locations, General Practice has most appointments from Jan to June 2022.

March 2022 has most appointments in **ad** and most most records across all three.

```
# Calculate total appointments per month for ad dataframe
total_appointments_ad = ad.groupby([ad['appointment_date'].dt.year, ad['appointment_date'].dt.month]).agg({'count_
total_appointments_ad = total_appointments_ad.apply(lambda x: x.sort_values(ascending=False))
total_appointments_ad
```

		count_of_appointments
appointment_date	appointment_date	
2022	3	27170002
	5	25343941
	6	23715317
	1	23597196
	2	23351939
2021	12	22853483
2022	4	21948814

Number of records per month for each dataframe

lowest value highest value

appointment_month	ad	ar*	nc
2021-08	*No data*	*19,786*	*69,999*
2021-09	*No data*	*20,441*	*74,922*
2021-10	*No data*	*20,562*	*74,078*
2021-11	*No data*	*20,766*	*77,652*
2021-12	19,507	20,393	72,651
2022-01	19,643	20,225	71,896
2022-02	18,974	20,133	71,769
2022-03	21,236	20,532	82,822
2022-04	19,078	20,073	70,012
2022-05	20,128	20,276	77,425
2022-06	19,227	20,231	74,168

** data starts 2020-01*

Incomplete data

There are a lot of 'unmapped', 'unknown' or 'inconsistently mapped'. For example:

service_setting: 17% are 'Other' and 3.4% (27,419) are 'Unmapped'.

```
# Service setting

print(f"Number of records by {nc['service_setting'].value_counts()}")
```

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

context_type: 'Inconsistently mapped' or 'Unmapped' are 2 of 3 options.

```
# Context type

print(f"Number of records by {nc['context_type'].value_counts()}")
```

```
Number of records by context_type
Care Related Encounter    700481
Inconsistent Mapping       89494
Unmapped                   27419
Name: count, dtype: int64
```

national_category: Most are 'Inconsistent mapping', and 27,419 are 'Unmapped'.

```
# National category

print(f"Number of records by {nc['national_category'].value_counts()}")
```

```
Number of records by national_category
Inconsistent Mapping                                                89494
General Consultation Routine                                        89329
General Consultation Acute                                          84874
Planned Clinics                                                     76429
Clinical Triage                                                     74539
Planned Clinical Procedure                                          59631
Structured Medication Review                                        44467
Service provided by organisation external to the practice           43095
Home Visit                                                          41850
Unplanned Clinical Activity                                         40415
Patient contact during Care Home Round                              28795
Unmapped                                                            27419
Care Home Visit                                                     26644
Social Prescribing Service                                          26492
Care Home Needs Assessment & Personalised Care and Support Planning 23505
Non-contractual chargeable work                                     20896
Walk-in                                                             14179
Group Consultation and Group Education                               5341
Name: count, dtype: int64
```

appointment_status: 'Unknown' is second highest count.

```
# Appointment status

print(f"Count of records by {ar['appointment_status'].value_counts()}")
```

```
Count of records by appointment_status
Attended    232137
Unknown     201324
DNA         163360
Name: count, dtype: int64
```

hcp_type: 22.7% of records are unknown.

```
# Number of records by hcp type

print(f"Total records by: {ar['hcp_type'].value_counts()}")
```

```
Total records by: hcp_type
Other Practice staff    241557
GP                      226036
Unknown                 129228
Name: count, dtype: int64
```

time_between_book_and_appointment: 18.1% of records are 'unknown'.

```
# By number of records

print(f"Number of records by: {ar['time_between_book_and_appointment'].value_counts()}")
```

```
Number of records by: time_between_book_and_appointment
Same Day                  95502
2 to 7 Days               92409
1 Day                     88957
8  to 14 Days             82698
15  to 21 Days            73666
22  to 28 Days            68755
More than 28 Days         65147
Unknown / Data Quality    29687
Name: count, dtype: int64
```

appointment_duration: Unknown/data quality is the highest count of appointments and records.

```
# By number of appointments available

duration_appts = ad.groupby(['actual_duration'])['count_of_appointments'].sum().sort_values(ascending = False)
print(f"Appointments available by {duration_appts}")
```

```
Appointments available by actual_duration
Unknown / Data Quality    40284086
6-10 Minutes              33800815
1-5 Minutes               28600865
11-15 Minutes             25160882
16-20 Minutes             16004247
21-30 Minutes             15026365
31-60 Minutes              9103432
Name: count_of_appointments, dtype: int64
```

```
# By number of records

print(f"Number of records by appointment {ad['actual_duration'].value_counts()}")
```

```
Number of records by appointment actual_duration
Unknown / Data Quality    20161
1-5 Minutes               19909
6-10 Minutes              19902
11-15 Minutes             19738
16-20 Minutes             19534
21-30 Minutes             19452
31-60 Minutes             19097
Name: count, dtype: int64
```

Visualisations

Approach

I created visualisations to help identify patterns, and tell a story to stakeholders.

I used line charts because they're easy to read and ideal for time series. I chose colour blindness for accessibility, and added custom titles and axis labels.

When plots showed overplotting (clusters top/bottom), I reduced the variables to 'zoom in'.

I also created three functions:

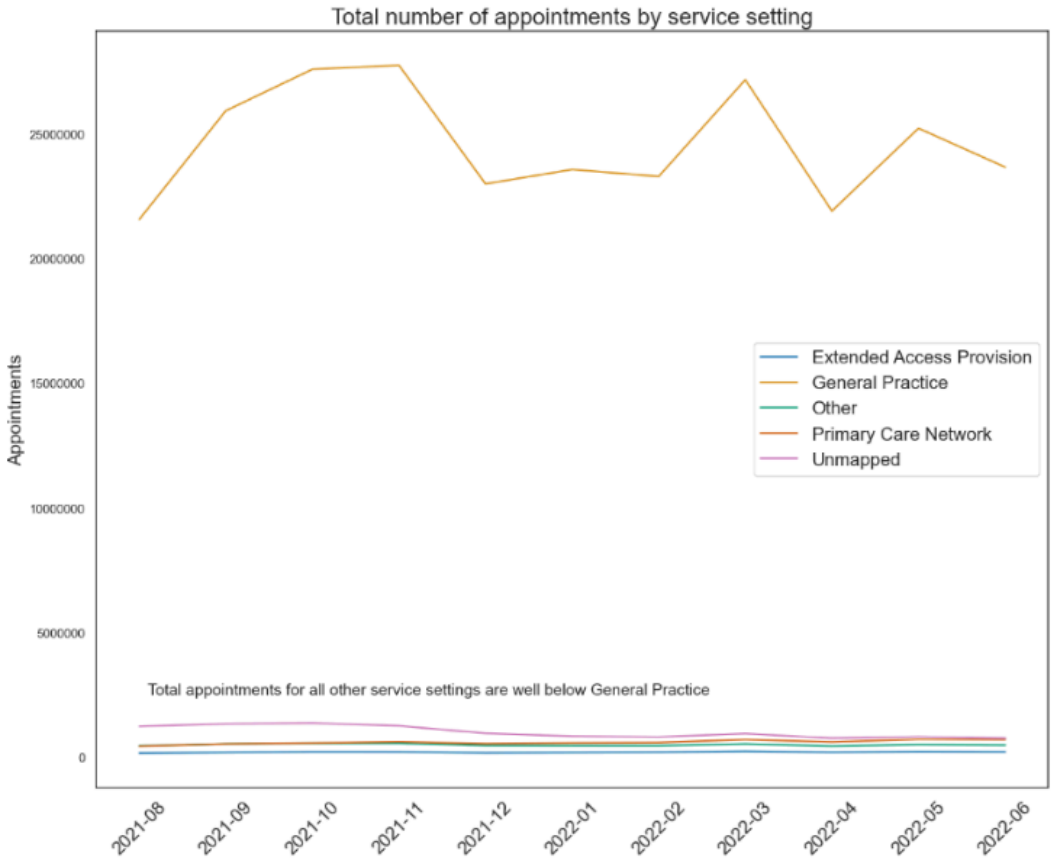
```
# Function to set up a plot

def set_up_plot():
    """
    Sets the plot figure size to 15, 12
    and the style to white
    """

    plt.figure(figsize = (15, 12))
    sns.set_style('white')

    return print("The plot is set up; start making your visualisation.")
```

```
# Function to create lineplot

def create_lineplot(plot_name, data_var, x_var, y_var,
                    hue_var, style_var,
                    plot_title, x_label, y_label, file_name):
    """
    Creates a lineplot grouped by hue and/or line style,
    using colorblind palette,
    displays the lineplot,
    and saves image as png at 300 dpi.
    """

    plot_name = sns.lineplot(x = x_var,
             y = y_var,
             data = data_var,
             hue = hue_var,
             style = style_var,
             palette = 'colorblind',
             linewidth = 3,
             errorbar = None)

    # Customise titles
    plt.title(plot_title, size = 20)
    plt.xlabel(x_label, size = 16)
    plt.ylabel(y_label, size = 16)
    plt.legend(title = '', fontsize = '16')

    # Make sure numbers are in full, rather than scientific notation
    plt.ticklabel_format(style ='plain', axis ='y')

    # Move legend outside of plot
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)

    return plot_name, plt.savefig(file_name, dpi = 300)
```

```python
# Function to create barplot

def create_barplot(plot_name, data_var,
                   x_var, y_var,
                   order_var,
                   plot_title, x_label, y_label, file_name):

    """
    Creates a barplot using colorblind palette,
    with custom title, x and y labels,
    with custom order of the bars as a list,
    displays the barchart and saves image as png at 300 dpi.
    """

    plot_name = sns.barplot(x = x_var,
                            y = y_var,
                            data = data_var,
                            palette = 'colorblind',
                            errorbar = None,
                            order = order_var)

    # Customise titles
    plt.title(plot_title, size = 20)
    plt.xlabel(x_label, size = 16)
    plt.ylabel(y_label, size = 16)
    plt.legend(title = '', fontsize = '16')

    # Make sure numbers are in full, rather than scientific notation
    plt.ticklabel_format(style ='plain', axis ='y')

    # Move legend outside of plot
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)

    return plot_name, plt.savefig(file_name, dpi = 300)
```

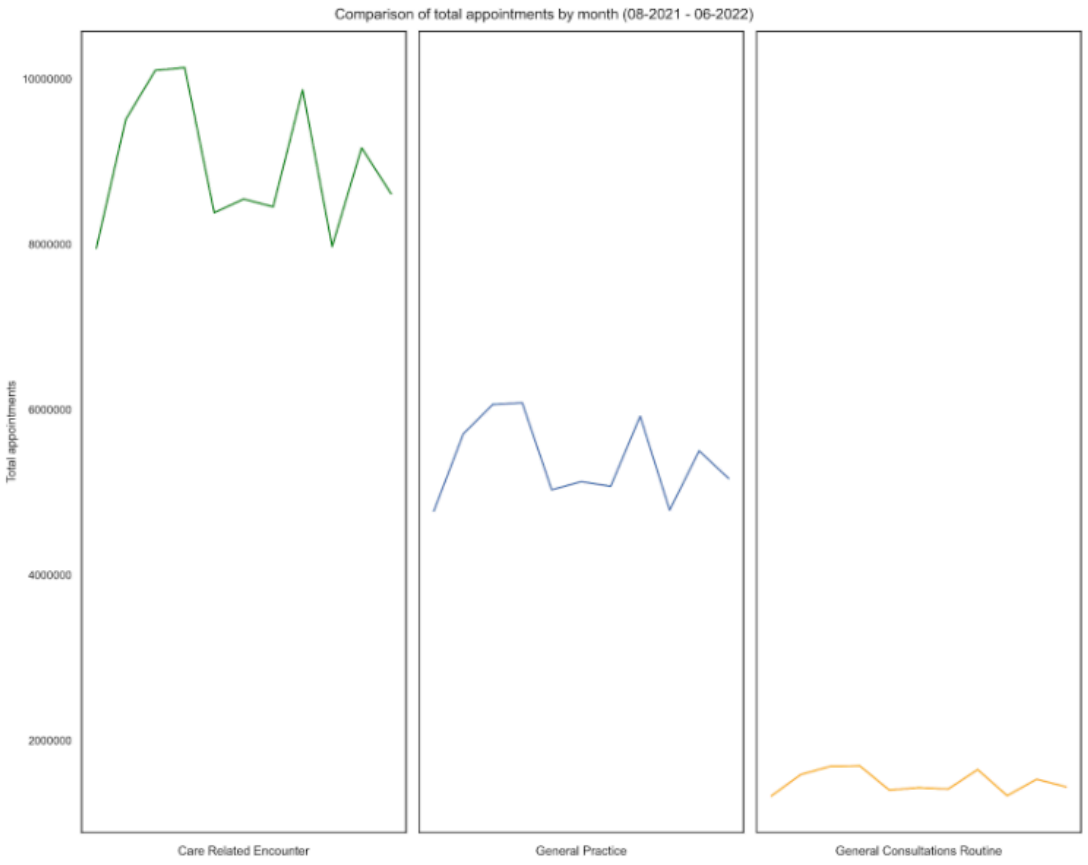
Examples

General Practice is a clear winner



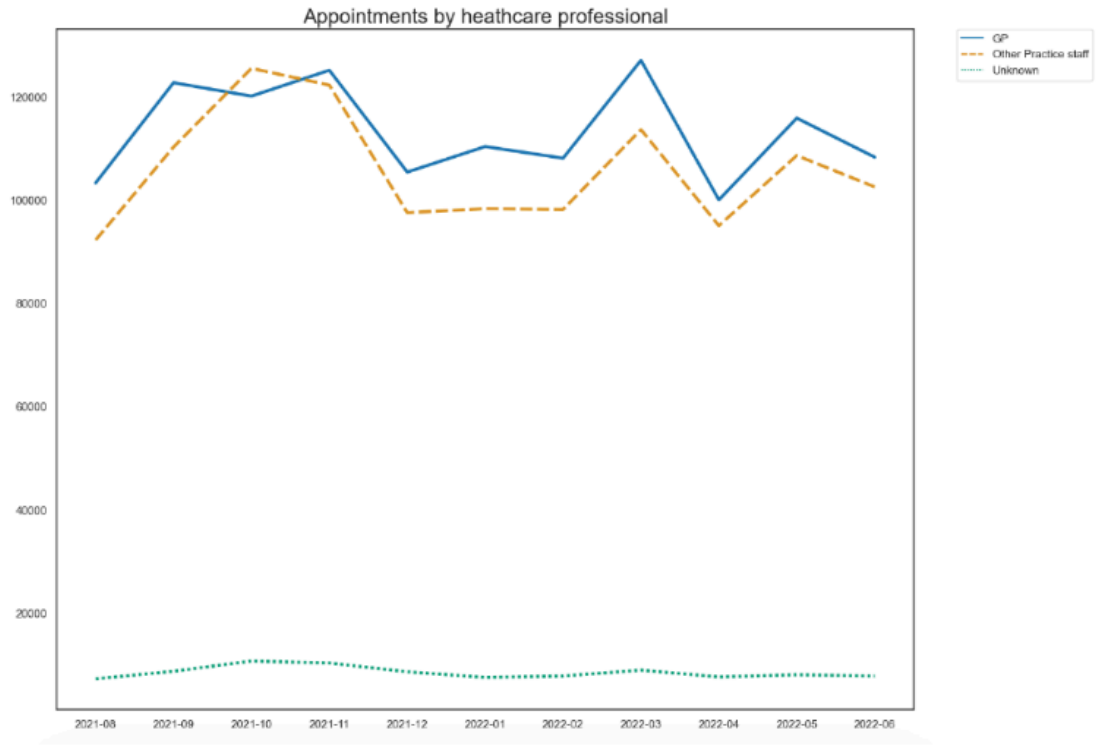
nc_ss_plot.png

Compare the context, service, category over time .com



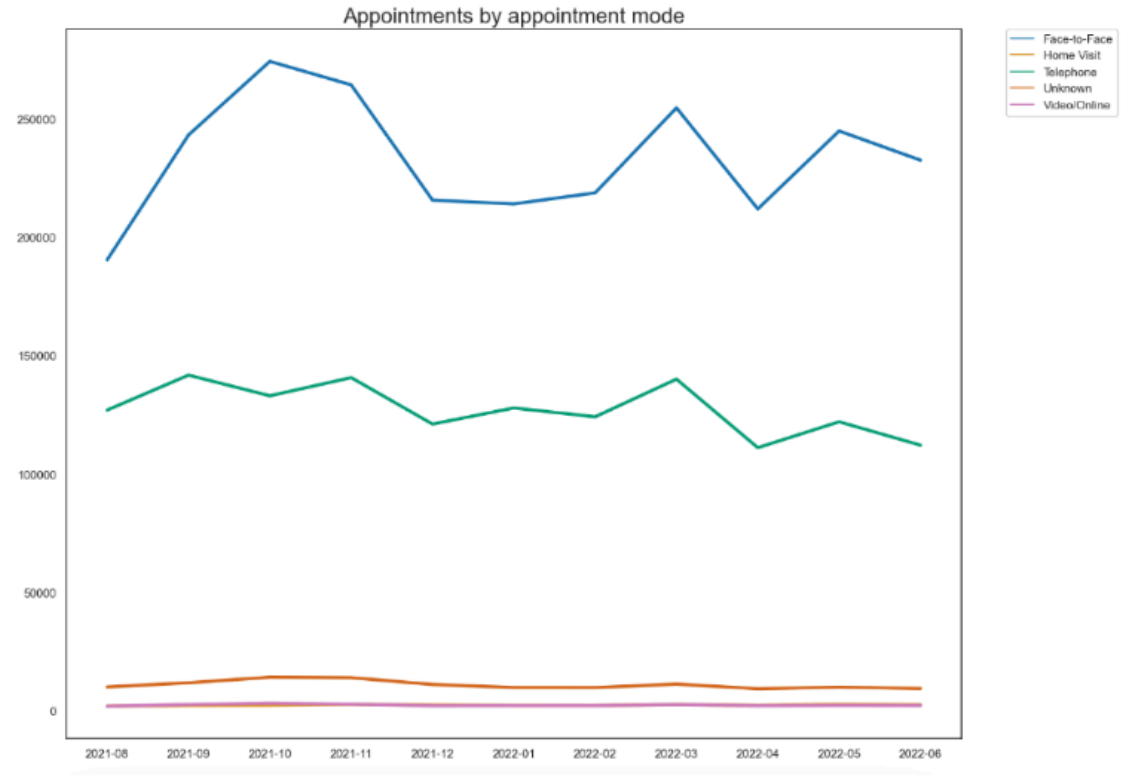
nc_appts_compare.png

Who's in demand?



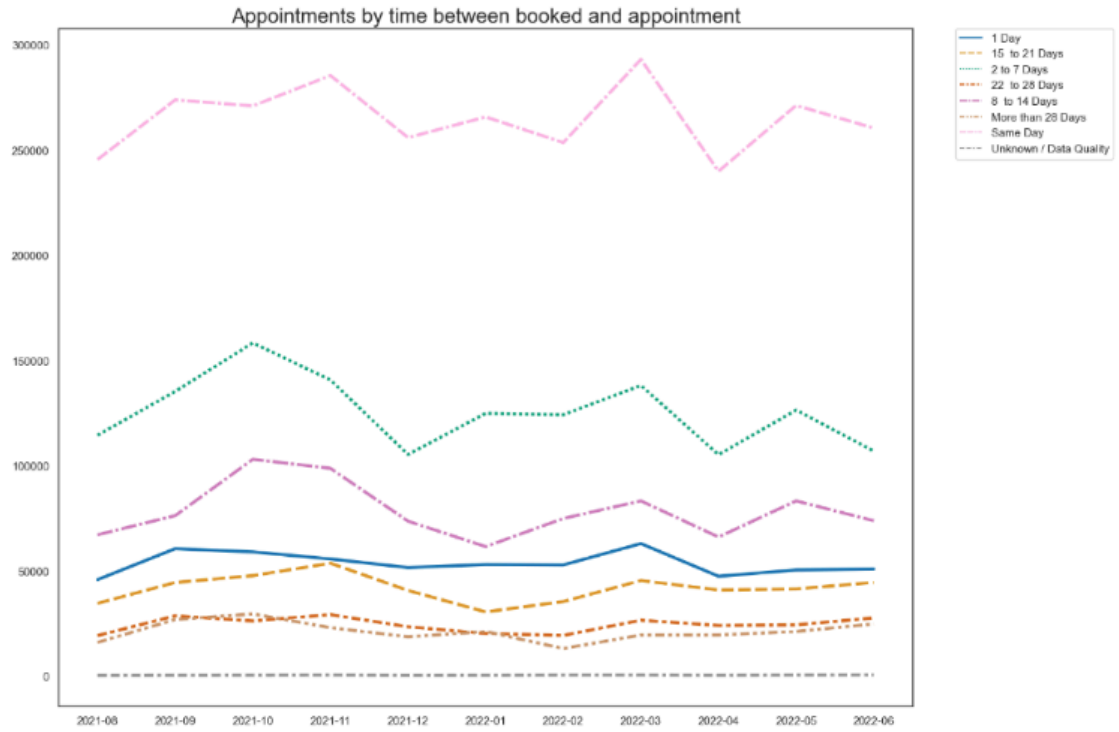
ar_hcp_month.png

Are people too busy in December, January and February for face-to-face appointments?



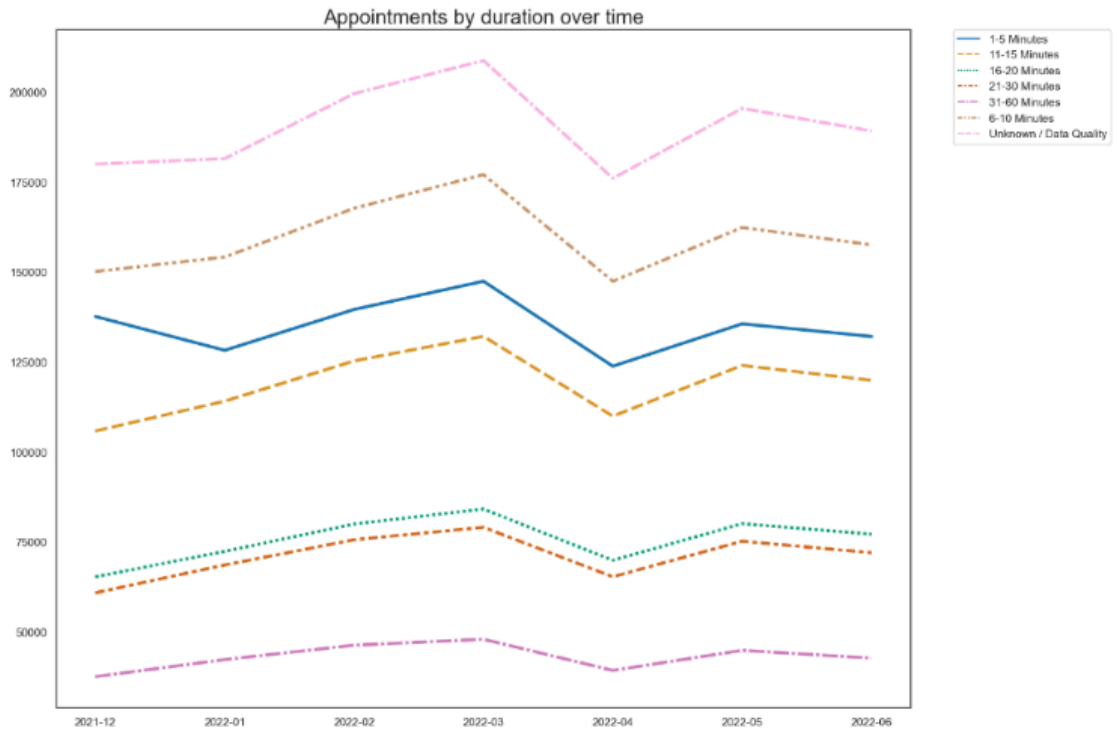
ar_mode.png

No surprise that most appointments are booked on the same day

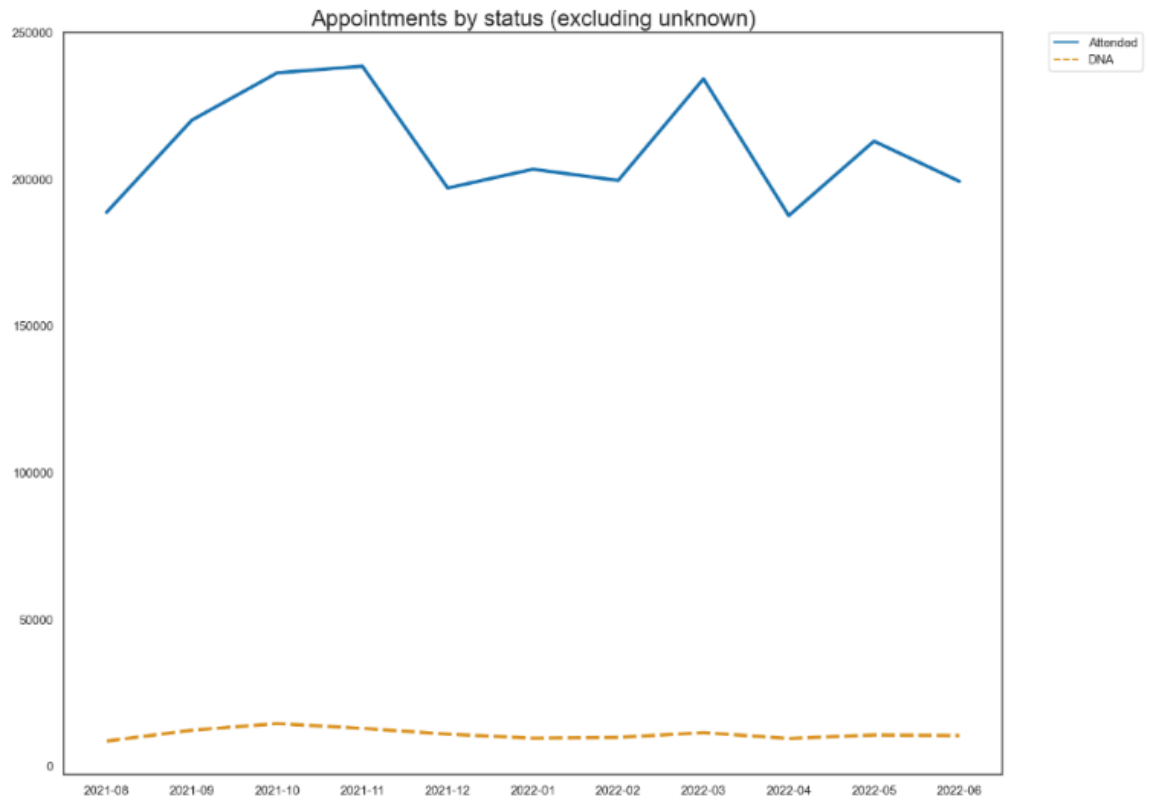


ar_time_book.png

Is 6-10 minutes long enough for an appointment?

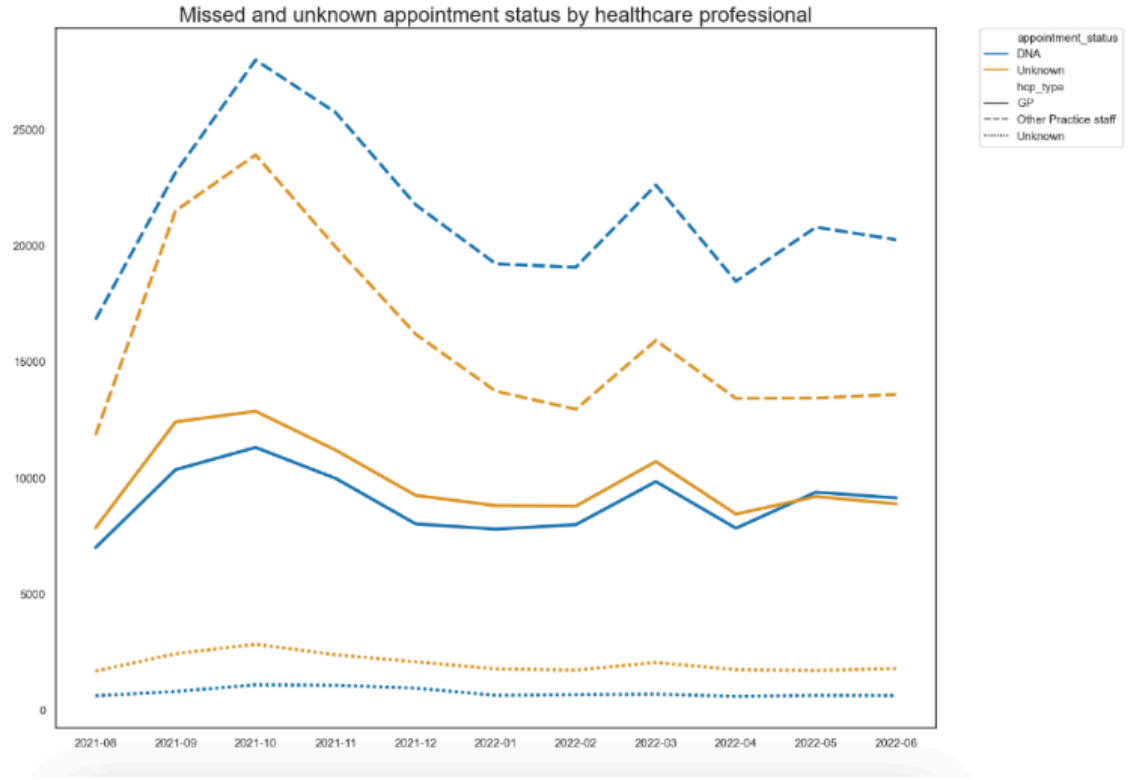


Does the time of year affect DNAs?



ar_status_ex.png

Does healthcare professional type influence appointment status?

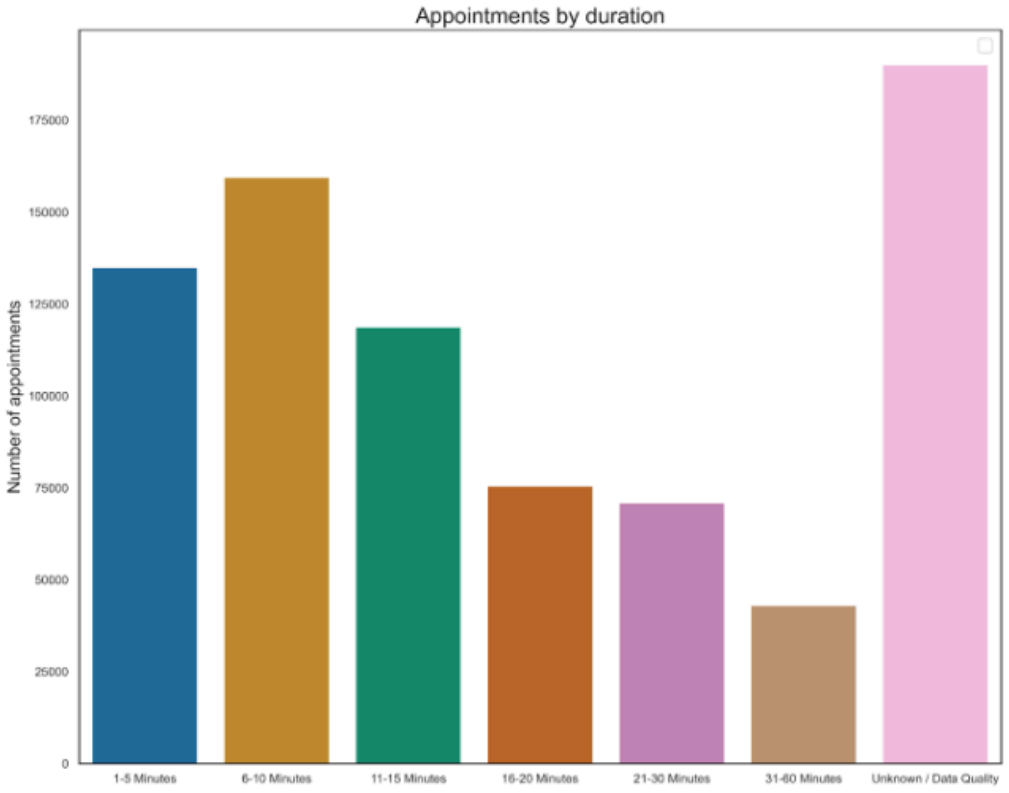


ar_status_hcp_ex.png

Missed appointments: the one time we want the figures to go down

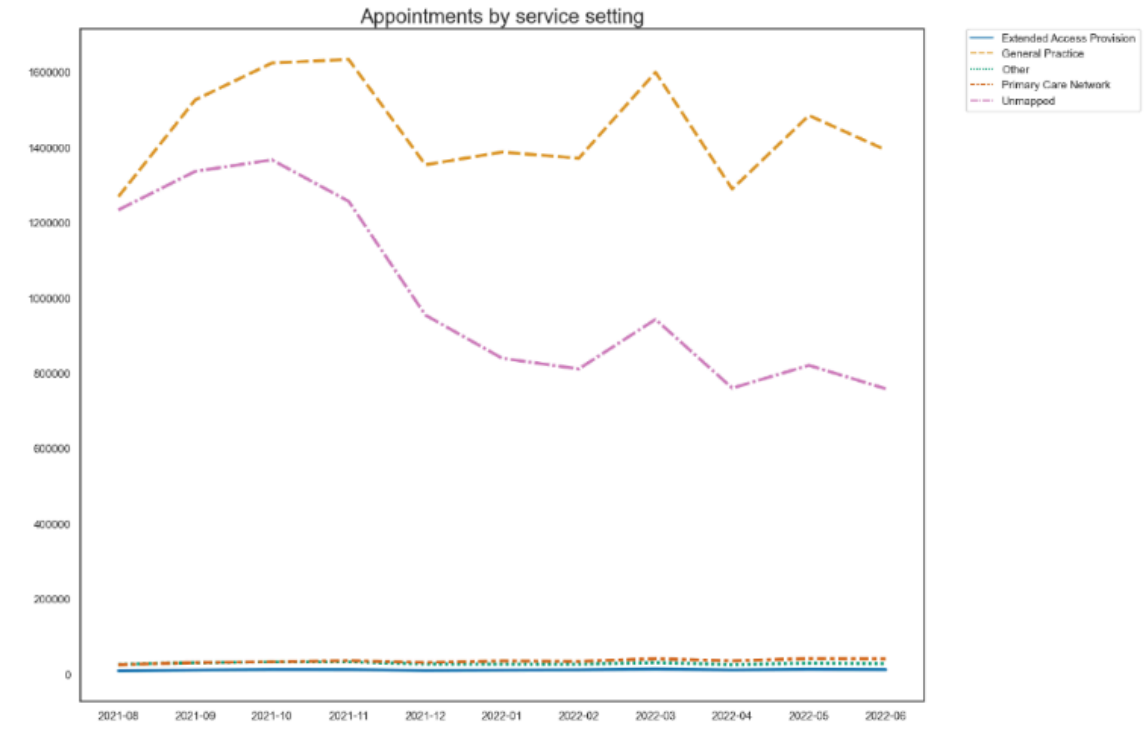
ar_status_time.png

Did you know most appointment durations are unknown?



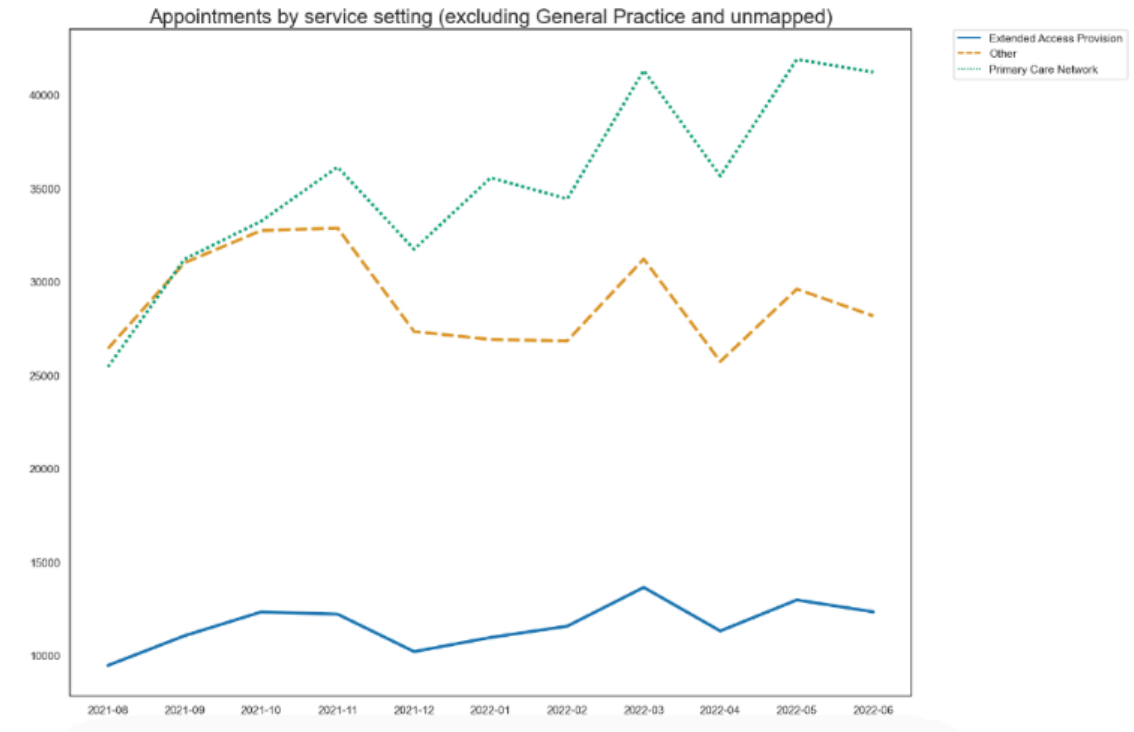
appt_duration_plot.png

Good news! Unmapped are decreasing... but what are all the lines at the bottom?



nc_agg_service.png

I can see clearly now that primary care network appointments are on the up

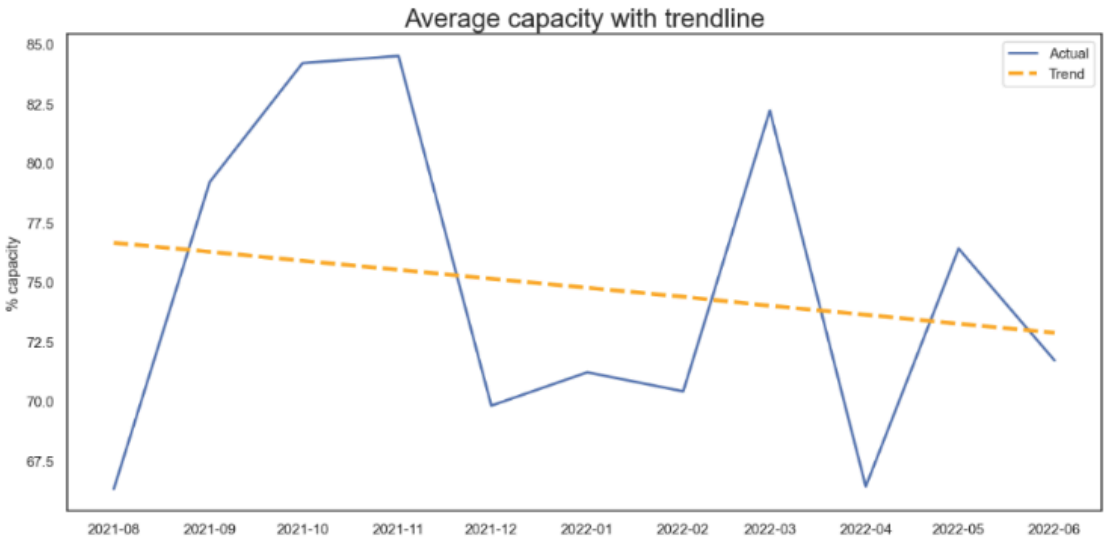


nc_agg_service_exgp_un.png

At your busiest, you were at 84.5 % capacity!

ar_capacity_plot.png

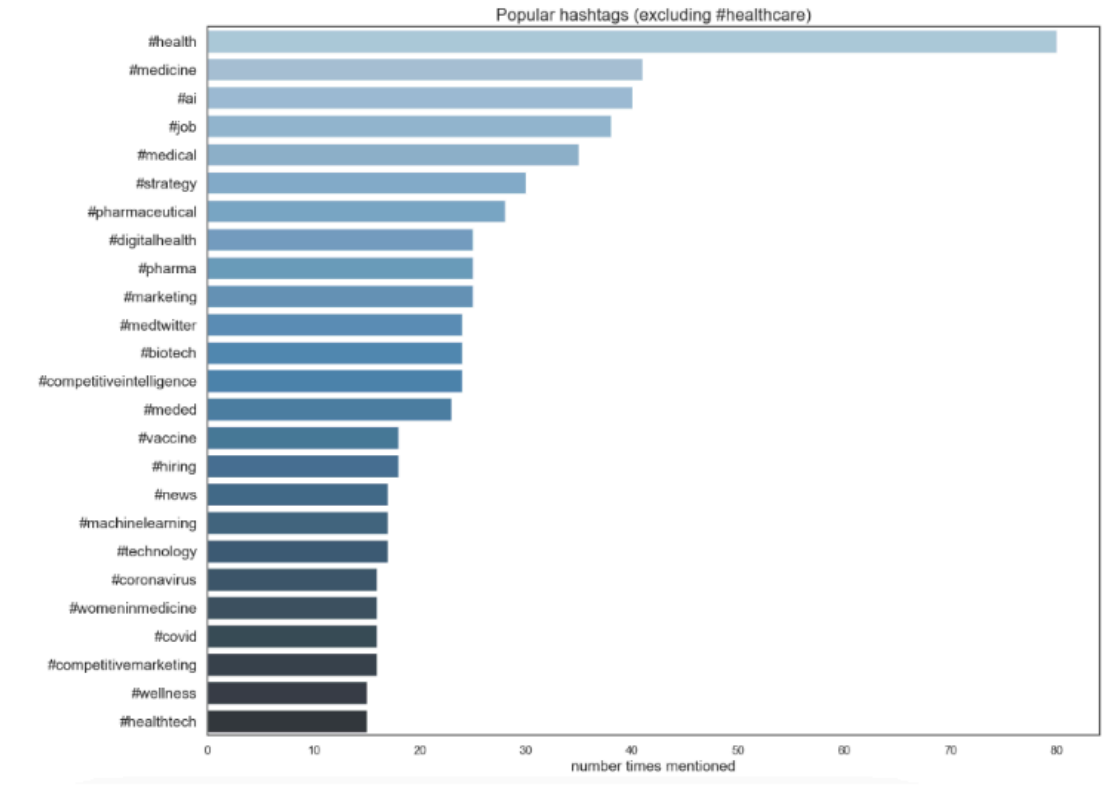
With this downward trend, you've got more capacity



Twitter data

Analysing #hashtags did not give significant insights towards the business problem, except indicate digital technology is popular – perhaps the NHS can use it to minimise DNAs[1].

What are people talking about?



tags_plot_h.png

[1] See Appendix

Insights

The key insights and trends from this analysis are:

Capacity

Ranges from 66.3% to 84.5% with a downward trend – current resources are sufficient. Capacity reduces in winter, most likely because people suffering from flu and other weather related illnesses.

Heathcare professional

Unsurprisingly most appointments are with GPs. 'Other healthcare professionals' appointments follow a similar pattern.

Appointment mode

Most appointments are face-to-face and peak in October and March. There's quite a dip in December to February. Could it be because of Christmas and healther routines (e.g. resolutions, dry January)?

Appointment duration

Excluding unknowns, most appointments are between 6 to 10 minutes long, followed by 1-5 minutes, and then 11-15 minutes.

Time between booked and appointment

There are similar peaks in March and October, apart from appointments booked one day in advance. These are pretty flat over the year with a minor peak in March.

Missed appointments

The trend for DNAs month-on-month looks flat – seasons may not be an influencer. Conversely, when factoring in healthcare professional, most DNAs are with 'Other practice staff'. People appear less likely to miss a GP appointment.

In terms of when booked, most DNAs are for appointments booked 2-7 days and 8-14 days in advance. DNAs are decreasing overall.

Unknown/unmapped

We're only seeing part of the picture. For example, there are more unknown appointment durations than anything else. And with many service settings records unmapped and all other service settings (apart from General Practice) clustered at the bottom, they appear as outliers.

Recommendations

Here are my recommendations:

1. Maintain existing capacity and resources.
2. Reduce missed appointments.
3. Improve data capture and completeness.

Suggested next steps:

Because of the data quality, I'd like to (if possible):

- Examine outliers in *count_of_appointments*.
- Investigate duplicates.
- Expand 'Other practice staff' to include specific roles[2].
- Reduce *national_category* options to improve data quality.
- Reclassify unknowns and unmapped.

[2] See Appendix

Appendix

Common reasons for missed appointments (DNAs)

Two common reasons for missing GP appointments are forgetting the appointment[3], and difficulty cancelling appointments especially if you have to telephone to do so[4]. It can be a challenge to book an appointment (with long durations on hold), so people may be reluctant to go through the hassle again to cancel.

With this in mind, booking, cancelling, and changing appointments needs to be as quick and easy as possible – along with timely reminders. From analysis of Twitter #hashtags, ai, digital health, technology and healthtech were popular indicating people are interested in health-related technology. With this in mind, more could be done with digital technology to maximise appointment availability and increase attendance.

For example: text and email reminders with a quick link to cancel or reschedule (sent two days, and morning of), calendar invitations with reminders, online appointment manager, and an app that allows you to easily make, cancel or reschedule.

Other practice staff

Expand 'other practice staff' to include specific roles. For example:

1. Acupuncturist
2. Chiropodist
3. Dispenser
4. Mental Health (Counsellor, Psychiatric Nurse)
5. Nurse (Practice Nurse, District Nurse)
6. Osteopath
7. Physiotherapist
8. Other Practice Staff (Interpreter/Link Worker, Health Visitor, Other)

We could then look at appointments for these types in more detail and in comparison to GPs. With the intention to redirect appointments from GPs to minimise avoidable appointments[5] and, meet BMA guidelines for safe levels of patient contacts (maximum 25 per day)[6].

It would also be interesting to compare healthcare professional against appointment duration to see who provides the longest. I suspect some of these healthcare professionals (e.g. chiropodist, mental health practioner) will exceed the typical 6-10 minutes duration.

At the moment the required data is across two dataframes (ad and ar) so it would need to combined but I'm not sure if there's a unique primary key to relate them.

---

[3] The changing face of missed appointments
[4] How can we reduce the number of missed GP appointments
[5] 1 in 4 GP appointments are potentially avoidable
[6] BMA guidelines for safe working in general practice

Get in touch

lilliana.golob@gmail.com | 07501 450668