Programming for Data Science [redacted]

Summative Assessment 2

Aggression and Performance within the English Premier League from 2000-2024

Student: *Ben Coake*

[redacted]

Project Plan

The Data

(Cortés, 2024) published a dataset on Kaggle containing match level statistics on every English Premier League match since the 1993-1994 season. This is a compilation of data taken from https://football-data.co.uk, which collates a myriad of football statistics across various years, leagues & countries. At the rawest level of detail, the dataset contains insights including but not limited to:

- General Match information (Date, Season, Teams etc.)
- Match Statistics (Goals, Shots, Corners, Fouls etc)
- Pre-Match betting odds sourced from the online platform Bet365
- Pre-Match market average betting odds
- Match Outcome (Score, Points etc)

As the dataset is compiled at match level, each row entry contains data for two football teams. This has been accounted for during initial data preparation as 'team' is a useful primary key for this dataset; should the data need to be combined in any way. Relating to matters of data quality, there are concerns around the completeness of this dataset. The data is historical so many fields are only partially complete, in particular statistics relating to betting odds. Checking for null values during data preparation will uncover the subset of data appropriate for analysis. However, other match statistics are readily available on the internet and can be validated, easing concerns over data accuracy. Additionally, there are no concerns about timeliness as the dataset is updated on a weekly cadence, with the latest row entry from the 2024-10-06 matchday at the time of analysis.

Project Aim and Objectives

This project aims to explore the data points indicative of team aggression, examining the potential settings in which they occur, and in particular, the impact this has on a team's ability to perform well.

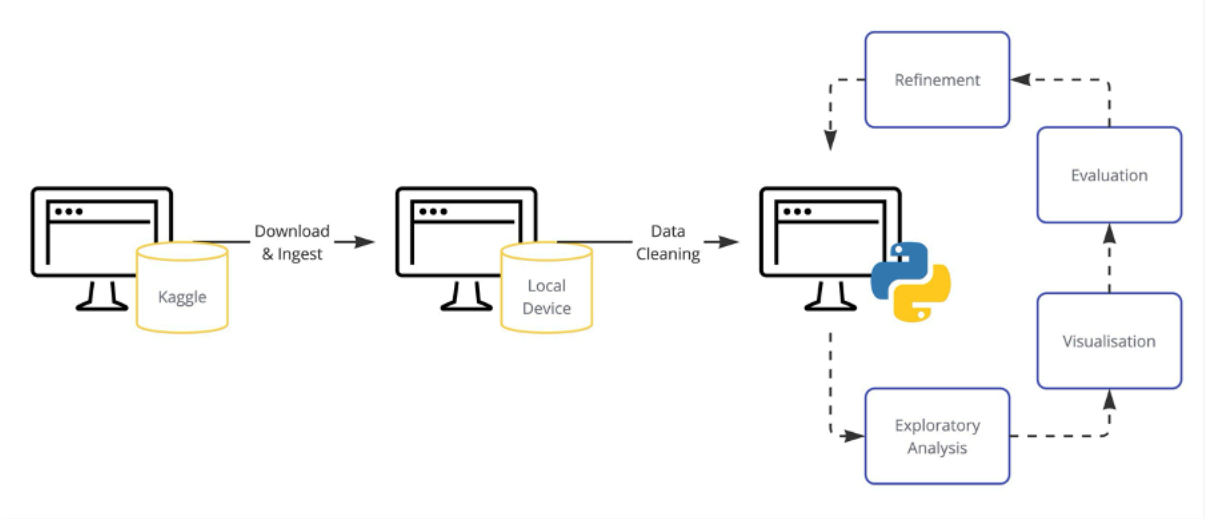
The analysis will be conducted at two levels of detail. To begin with, it will examine each team's performance holistically, building an understanding of the teams that are more or less likely to present as aggressive. This will be followed by a more detailed correlation analysis, which aims to give insight on the relationship between aggression and performance metrics, which could be used as a diagnostic tool for breaking down team performance.

This project has been broken down into 3 research questions:

1. **Do levels of aggression differ when playing home or away?**
   Objective 1: Construct a comparative visualisation of aggression against fixture type.
2. **Is aggression beneficial to performance?**
   Objective 2: Construct a comparative visualisation of aggression against performance
3. **What aggression metrics have a significant impact on team performance?**
   Objective 3: Build a function to run correlation analysis for a given team

System Design

Architecture



As outlined in the architecture diagram above, this project follows the principles of the data analytics lifecycle. The dataset required is downloaded and ingested from the online source Kaggle, then stored on a local device (or Coursera Labs in this instance). Within Jupyter Notebooks, a series of data preparation, analysis, visualisation & evaluation steps will then be conducted, with any refinements made before finalising the analysis.

Processing Modules and Algorithms

- Preparing data for analysis by isolating the appropriate fields, checking and removing `null` content using the `pandas` package.
- Separating and re-combining the data to account for multiple variables per row entry (Home & Away team).
- Introducing the `matplotlib.pyplot` and `seaborn` packages to create visualisations (scatter plots).
- Defining a function using both `seaborn` and `scipy.stats` packages to create a correlation calculation and visualisation tool.

Program Code

Data Preparation

Modules

In [2]:
```
import pandas as pd                         # Dataframe handling
import matplotlib.pyplot as plt             # Data visualisation
from scipy.stats import pearsonr            # To calculate correlation
import seaborn as sns                       # Data visualisation

# Additional module used to remove deprecation warnings from code output
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

Data Cleaning

In [3]:
```
# Ingesting Data
df = pd.read_csv('PremierLeague.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11850 entries, 0 to 11849
Data columns (total 43 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   MatchID                 11850 non-null  object
 1   Season                  11850 non-null  object
 2   MatchWeek               11850 non-null  int64
 3   Date                    11850 non-null  object
 4   Time                    1970 non-null   object
 5   HomeTeam                11850 non-null  object
 6   AwayTeam                11850 non-null  object
 7   FullTimeHomeTeamGoals   11850 non-null  int64
 8   FullTimeAwayTeamGoals   11850 non-null  int64
 9   FullTimeResult          11850 non-null  object
 10  HalfTimeHomeTeamGoals   11090 non-null  float64
 11  HalfTimeAwayTeamGoals   11090 non-null  float64
 12  HalfTimeResult          11090 non-null  object
 13  Referee                 9190 non-null   object
 14  HomeTeamShots           9190 non-null   float64
 15  AwayTeamShots           9190 non-null   float64
 16  HomeTeamShotsOnTarget   9190 non-null   float64
 17  AwayTeamShotsOnTarget   9190 non-null   float64
 18  HomeTeamCorners         9190 non-null   float64
 19  AwayTeamCorners         9190 non-null   float64
 20  HomeTeamFouls           9190 non-null   float64
 21  AwayTeamFouls           9190 non-null   float64
 22  HomeTeamYellowCards     9190 non-null   float64
 23  AwayTeamYellowCards     9190 non-null   float64
 24  HomeTeamRedCards        9190 non-null   float64
 25  AwayTeamRedCards        9190 non-null   float64
 26  B365HomeTeam            8430 non-null   float64
 27  B365Draw                8430 non-null   float64
 28  B365AwayTeam            8430 non-null   float64
 29  B365Over2.5Goals        2973 non-null   float64
 30  B365Under2.5Goals       2973 non-null   float64
 31  MarketMaxHomeTeam       1970 non-null   float64
 32  MarketMaxDraw           1970 non-null   float64
 33  MarketMaxAwayTeam       1970 non-null   float64
 34  MarketAvgHomeTeam       1970 non-null   float64
 35  MarketAvgDraw           1970 non-null   float64
 36  MarketAvgAwayTeam       1970 non-null   float64
 37  MarketMaxOver2.5Goals   1970 non-null   float64
 38  MarketMaxUnder2.5Goals  1970 non-null   float64
 39  MarketAvgOver2.5Goals   1970 non-null   float64
 40  MarketAvgUnder2.5Goals  1970 non-null   float64
 41  HomeTeamPoints          11850 non-null  int64
 42  AwayTeamPoints          11850 non-null  int64
dtypes: float64(29), int64(5), object(9)
memory usage: 3.9+ MB
```

In [4]:

```python
# Dropping columns (by column index) that are not useful for this analysis
# Remove all row entries with null content [.dropna()] in any row (to ensure data completeness)
df_slim = df.drop(df.columns[[4]],axis=1).drop(df.iloc[:,10:14],axis=1).drop(df.iloc[:,26:41],axis=
df_slim.info()

print('\nDate range is between',min(df_slim['Date']),'and',max(df_slim['Date']))

# There is a maximum of 9190 non-null results in the data
# This has complete data from 2000 - 2024
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9190 entries, 2660 to 11849
Data columns (total 23 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   MatchID                9190 non-null   object
 1   Season                 9190 non-null   object
 2   MatchWeek              9190 non-null   int64
 3   Date                   9190 non-null   object
 4   HomeTeam               9190 non-null   object
 5   AwayTeam               9190 non-null   object
 6   FullTimeHomeTeamGoals  9190 non-null   int64
 7   FullTimeAwayTeamGoals  9190 non-null   int64
 8   FullTimeResult         9190 non-null   object
 9   HomeTeamShots          9190 non-null   float64
 10  AwayTeamShots          9190 non-null   float64
 11  HomeTeamShotsOnTarget  9190 non-null   float64
 12  AwayTeamShotsOnTarget  9190 non-null   float64
 13  HomeTeamCorners        9190 non-null   float64
 14  AwayTeamCorners        9190 non-null   float64
 15  HomeTeamFouls          9190 non-null   float64
 16  AwayTeamFouls          9190 non-null   float64
 17  HomeTeamYellowCards    9190 non-null   float64
 18  AwayTeamYellowCards    9190 non-null   float64
 19  HomeTeamRedCards       9190 non-null   float64
 20  AwayTeamRedCards       9190 non-null   float64
 21  HomeTeamPoints         9190 non-null   int64
 22  AwayTeamPoints         9190 non-null   int64
dtypes: float64(12), int64(5), object(6)
memory usage: 1.7+ MB

Date range is between 2000-08-19 and 2024-10-06
```

Objective 1: Do levels of aggression differ when playing home or away?

Code & Visualisation

The code below:

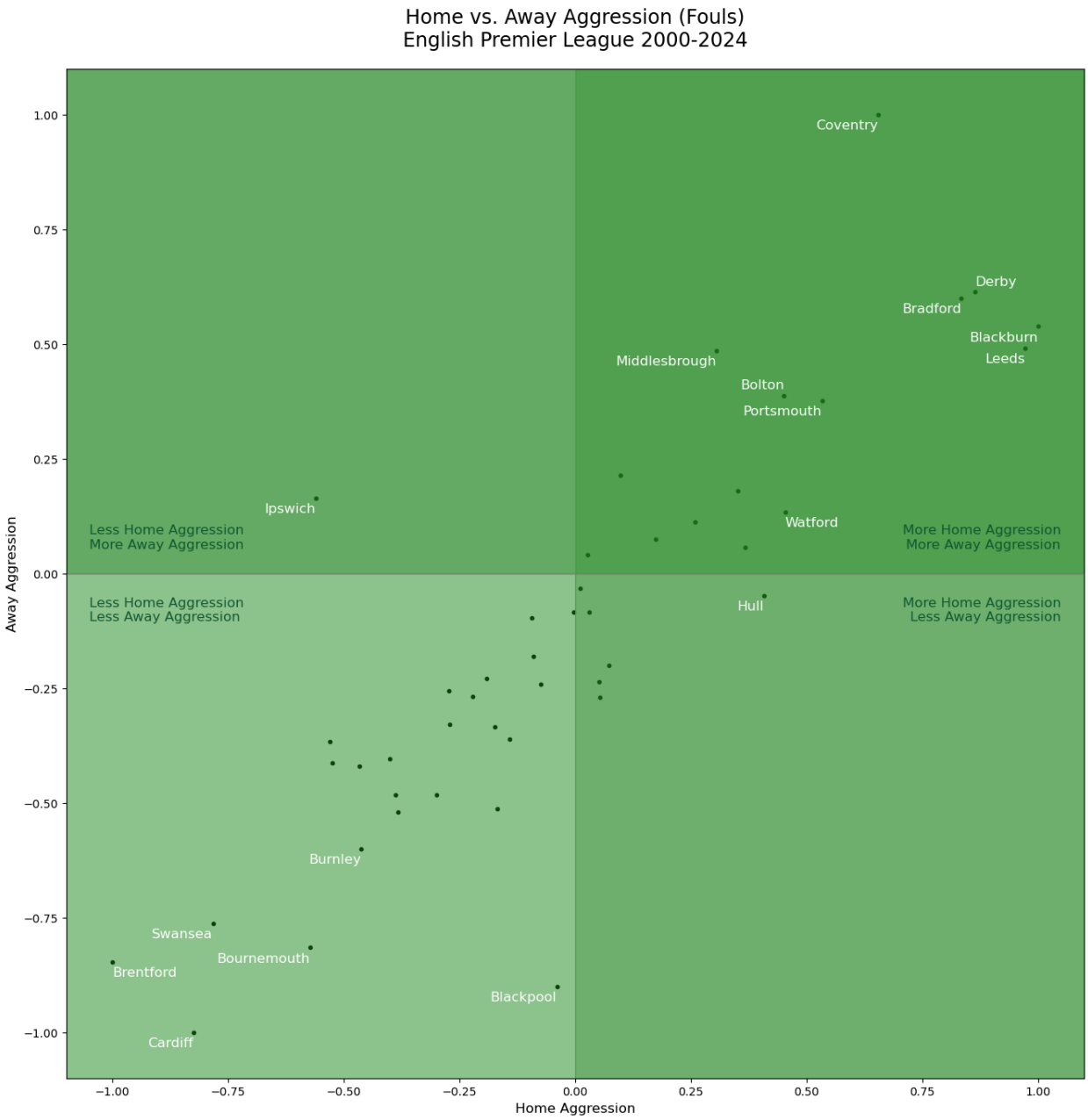
- Splits the `df_slim` dataframe into mean-aggregated home & away team datasets (`h_aggression` and `a_aggression` respectively).
- Normalises the average number of fouls per team. This is a technique used to scale each value into a specified range; in this instance (-1,1), to make results easily comparable.
- Combines aggregated home & away datasets into `ha_aggression`.
- Creates a scatter plot using `matplotlib.pyplot` with labelled quadrants to visualise home & away aggression.

In [5]:
```
# Aggregate each dataset by team, finding the avg red, yellow & fouls per game
h_aggression = df_slim[['HomeTeam',
                        'HomeTeamFouls',
                        'HomeTeamYellowCards',
                        'HomeTeamShots',
                        'HomeTeamShotsOnTarget',
                        'HomeTeamPoints',
                        'HomeTeamRedCards']].groupby('HomeTeam').agg({'HomeTeamFouls':'mean',
                                                                      'HomeTeamYellowCards':'mean',
                                                                      'HomeTeamRedCards':'mean',
                                                                      'HomeTeamShots':'mean',
                                                                      'HomeTeamShotsOnTarget':'mean',
                                                                      'HomeTeamPoints':'mean'})
a_aggression = df_slim[['AwayTeam',
                        'AwayTeamFouls',
                        'AwayTeamYellowCards',
                        'AwayTeamShots',
                        'AwayTeamShotsOnTarget',
                        'AwayTeamPoints',
                        'AwayTeamRedCards']].groupby('AwayTeam').agg({'AwayTeamFouls':'mean',
                                                                      'AwayTeamYellowCards':'mean',
                                                                      'AwayTeamRedCards':'mean',
                                                                      'AwayTeamShots':'mean',
                                                                      'AwayTeamShotsOnTarget':'mean',
                                                                      'AwayTeamPoints':'mean'})
```

In [6]:
```
# There are 3 variables to consider here: The team, Home/Away matches, and fouls.
# The best way to visualise this would be using a scatter plot
# Normalising both home & away fouls between (-1,1) to make each team's aggression easily comparabl
h_aggression['HFoulsNorm'] = 2 * (
    (h_aggression['HomeTeamFouls'] - h_aggression['HomeTeamFouls'].min())/
    (h_aggression['HomeTeamFouls'].max() - h_aggression['HomeTeamFouls'].min())) -1

a_aggression['AFoulsNorm'] = 2 * (
    (a_aggression['AwayTeamFouls'] - a_aggression['AwayTeamFouls'].min())/
    (a_aggression['AwayTeamFouls'].max() - a_aggression['AwayTeamFouls'].min())) -1
```

In [7]:
```
# Both home and away datasets are now comparable and can be easily joined & visualised
ha_aggression = h_aggression.join(a_aggression)
```

In [8]:
```
# Creating visualisation and various formatting elements
fig,ax = plt.subplots(figsize=(16,16))
ax.set_axis_off
plt.scatter(ha_aggression['HFoulsNorm'],ha_aggression['AFoulsNorm'],marker='.',color='black')
ax.axhline(y=0,c='grey',lw=0.75)
ax.axvline(x=0,c='grey',lw=0.75)
ax.set_ylabel('Away Aggression',size='larger')
ax.set_xlabel('Home Aggression',size='larger')
ax.set_title('Home vs. Away Aggression (Fouls)\nEnglish Premier League 2000-2024',pad=20,size='xx-l
ax.set_xlim(-1.1,1.1)
ax.set_ylim(-1.1,1.1)

# Adding team names into the visualisation, manually adjusting certain values where data points ove
# Only allowing visualisation to show extreme values to avoid overlap around (0,0)
for i, team in enumerate(ha_aggression.index):
    if (ha_aggression['HFoulsNorm'].iloc[i] > 0.4 or
        ha_aggression['HFoulsNorm'].iloc[i] < -0.55 or
        ha_aggression['AFoulsNorm'].iloc[i] > 0.25 or
        ha_aggression['AFoulsNorm'].iloc[i] < -0.55):
        if team == 'Derby':
            plt.annotate(team,xy=(ha_aggression['HFoulsNorm'].iloc[i],ha_aggression['AFoulsNorm'].i
                         ha='left',va='bottom',color='white',size='larger')
        elif team == 'Bolton':
            plt.annotate(team,xy=(ha_aggression['HFoulsNorm'].iloc[i],ha_aggression['AFoulsNorm'].i
                         ha='right',va='bottom',color='white',size='larger')
        elif team == 'Watford':
            plt.annotate(team,xy=(ha_aggression['HFoulsNorm'].iloc[i],ha_aggression['AFoulsNorm'].i
                         ha='left',va='top',color='white',size='larger')
        elif team == 'Brentford':
            plt.annotate(team,xy=(ha_aggression['HFoulsNorm'].iloc[i],ha_aggression['AFoulsNorm'].i
                         ha='left',va='top',color='white',size='larger')
        else:
            plt.annotate(team,xy=(ha_aggression['HFoulsNorm'].iloc[i],ha_aggression['AFoulsNorm'].i
                         ha='right',va='top',color='white',size='larger')


# Annotating each quadrant of the scatter plot
ax.text(1.05,0.05,"More Home Aggression\nMore Away Aggression",va='bottom',size='large',ha='right',
ax.text(1.05,-0.05,"More Home Aggression\nLess Away Aggression",va='top',size='large',ha='right',co
ax.text(-1.05,-0.05,"Less Home Aggression\nLess Away Aggression",va='top',size='large',color='#145a
ax.text(-1.05,0.05,"Less Home Aggression\nMore Away Aggression",va='bottom',size='large',color='#14

# Shading quadrants
ax.axvspan(-1.1,1.1,alpha=0.5,color='forestgreen')
ax.axhspan(0,1.1,alpha=0.4,color='forestgreen')
ax.axvspan(0,1.1,alpha=0.3,color='forestgreen');
```



Observations

In general, the majority of teams follow along a linear trend about the line `y=x`, suggesting there is no significant difference in levels of aggression when playing at home compared to playing away. As this data is comparative, the point (0,0) denotes a team which displays average levels of aggression both home and away. Furthermore, over 50% of teams lie within ±0.5 of the centre, which should be expected.

There are, however, some extreme and potentially anomalous results. Ipswich is the only team within the upper left quadrant, displaying less aggression when playing at home, but more when playing away in comparison to other Premier League teams. There are also distinct clusters along the fringe of the linear trend. Swansea, Bournemouth, Brentford & Cardiff lie at the most extreme end of the lower left quandrant (displaying **less** overall aggression regardless of fixture type), whilst Coventry, Derby, Bradford, Leeds & Blackburn lie at the most extreme end of the upper right quadrant (displaying **more** overall aggression regardless of fixture type).

Objective 2: Is aggression beneficial to performance?

Code & Visualisation

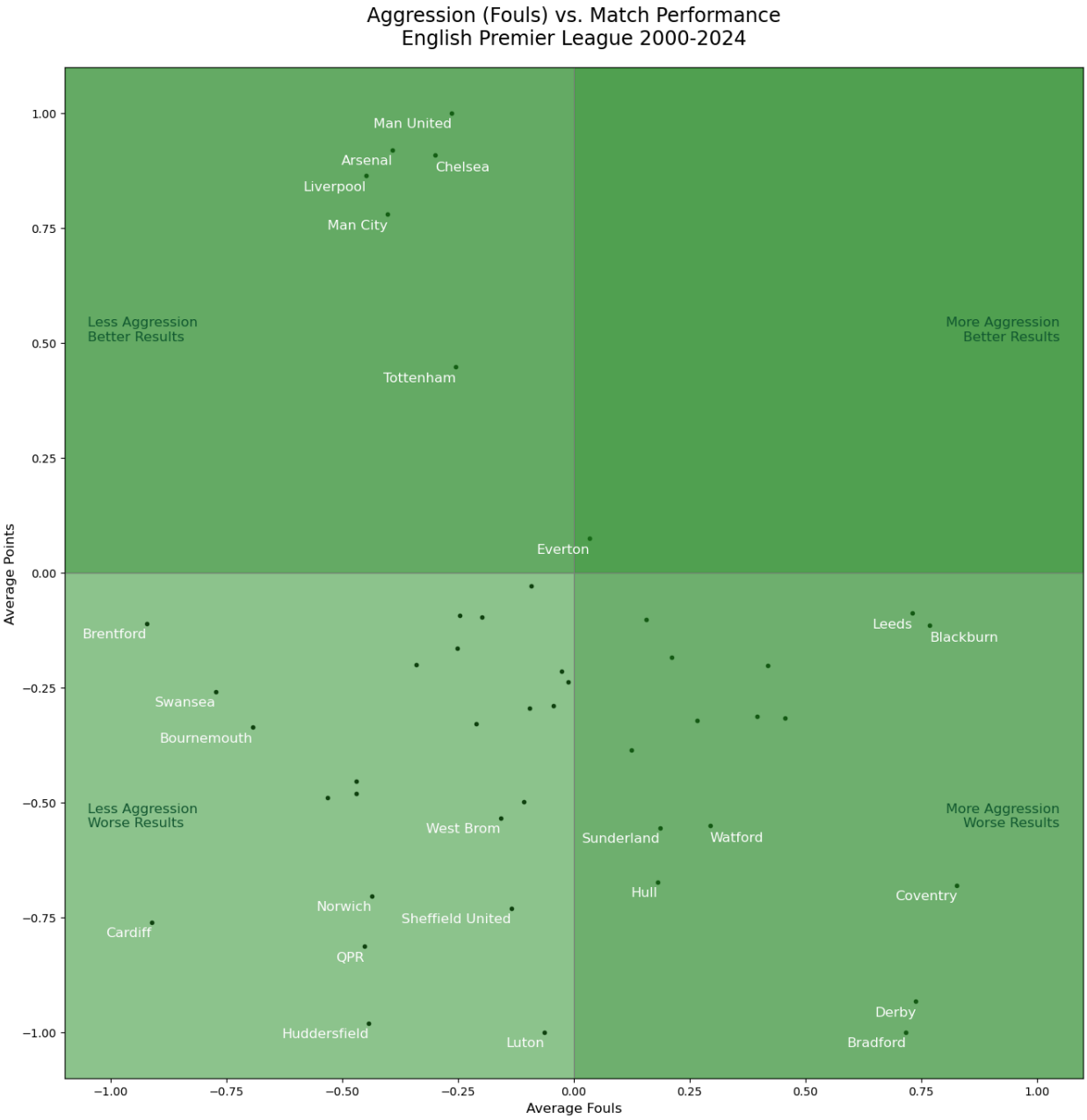
The code below:

- Uses the existing `ha_aggression` dataframe to calculate overall average normalised values for team fouls and points.

- Creates a scatter plot, similar to Objective 1, but visualising aggression (fouls) against points per game.

In [9]:
```
# Taking the average normalised value across both home and away fixtures per team
ha_aggression['AllFoulsNorm'] = (ha_aggression['HFoulsNorm'] + ha_aggression['AFoulsNorm'])/2

# Averaging and normalising the number of points earned per team
ha_aggression['AVGPoints'] = (ha_aggression['HomeTeamPoints'] + ha_aggression['AwayTeamPoints'])/2

ha_aggression['AVGPointsNorm'] = 2 * (
    (ha_aggression['AVGPoints'] - ha_aggression['AVGPoints'].min())/
    (ha_aggression['AVGPoints'].max() - ha_aggression['AVGPoints'].min())) -1
```

In [10]:
```
# Code follows a similar structure to Objective 1
fig,ax = plt.subplots(figsize=(16,16))
ax.set_axis_off
plt.scatter(ha_aggression['AllFoulsNorm'],ha_aggression['AVGPointsNorm'],marker='.',color='black')
ax.axhline(y=0,c='grey',lw=0.75)
ax.axvline(x=0,c='grey',lw=0.75)
ax.set_ylabel('Average Points',size='larger')
ax.set_xlabel('Average Fouls',size='larger')
ax.set_title('Aggression (Fouls) vs. Match Performance\nEnglish Premier League 2000-2024',pad=20,si
ax.set_xlim(-1.1,1.1)
ax.set_ylim(-1.1,1.1)

for i, team in enumerate(ha_aggression.index):
    if (ha_aggression['AVGPointsNorm'].iloc[i] > 0 or
        ha_aggression['AVGPointsNorm'].iloc[i] < -0.5 or
        ha_aggression['AllFoulsNorm'].iloc[i] < -0.6 or
        ha_aggression['AllFoulsNorm'].iloc[i] > 0.6):
        if team in ('Chelsea','Blackburn','Watford'):
            plt.annotate(team,xy=(ha_aggression['AllFoulsNorm'].iloc[i],ha_aggression['AVGPointsNor
                         ha='left',va='bottom',color='white',size='larger')
        else:
            plt.annotate(team,xy=(ha_aggression['AllFoulsNorm'].iloc[i],ha_aggression['AVGPointsNor
                         ha='right',va='top',color='white',size='larger')

ax.text(1.05,0.5,"More Aggression\nBetter Results",va='bottom',size='large',ha='right',color='#145a
ax.text(1.05,-0.5,"More Aggression\nWorse Results",va='top',size='large',ha='right',color='#145a32'
ax.text(-1.05,-0.5,"Less Aggression\nWorse Results",va='top',size='large',color='#145a32')
ax.text(-1.05,0.5,"Less Aggression\nBetter Results",va='bottom',size='large',color='#145a32');
ax.axvspan(-1.1,1.1,alpha=0.5,color='forestgreen')
ax.axhspan(0,1.1,alpha=0.4,color='forestgreen')
ax.axvspan(0,1.1,alpha=0.3,color='forestgreen');
```



Observations

The results above provide an interesting pivot to the plot constructed for Objective 1. The linear aggression trend seen previously is still clear across the lower left & right quadrants of this plot, but changing the independent variable to average match points has surfaced new clusters of extreme results:

- Within the lower left quadrant Swansea, Bournemouth, Brentford & Cardiff remain as the least aggressive teams. These teams also see worse than average performance in addition to lower levels of aggression, suggesting a potential link between aggression and performance.
- Within the lower right quadrant Coventry, Derby, Bradford, Leeds & Blackburn remain as the most aggressive teams. However, this has no positive effect on performance as each team falls under average by points.
- Only 7 teams index above average for performance, with 5 teams **significantly** over indexing. However, these teams are below index for aggression. This suggests an alternative hypothesis that aggression may not be significantly beneficial to team performance, which warrants further investigation.

Objective 3: What aggression metrics have a significant impact on team performance?

Code & Visualisation

The code below:

- Defines a function `correlation` which takes inputs (x,y) for independent and dependent variables, the team name, and fixture type:
  - Checks for errors with inputs.
  - Automatically generates a basic regression plot using `seaborn`.
  - Utilises `scipy.stats` to also calculate and output Pearson's Correlation Coefficient.
  - Runs correlation analysis on select teams, based on results from Objectives 1 & 2.

In [32]:
```
# Creating a function that can analyse correlation
# Requires 4 inputs: x,y which are the 2 aggression factors to correlate, the team & the type of fi
def correlation(x,y,team,fixture):

    xy_list = ['Fouls','YellowCards','Shots','ShotsOnTarget','Points','RedCards']
    boolean = x in xy_list and y in xy_list
    boolean2 = fixture in ['Home','Away']

    while boolean == False:
        # Validation that dependent & indpendent variables are correct
        print('Error: One or more x,y values not recognised. Please enter a valid field from the fo
        print(xy_list)
        break
    else:
        # Validation that team inputted is correct
        while (team in df_slim['HomeTeam'].unique()) == False:
            print('Team not recognised. Please enter a team from the following list:')
            print(df_slim['HomeTeam'].unique())
            break
        else:
            # Validation that fixture inputted is either Home or Away
            while boolean2 == False:
                print('Please either enter "Home" or "Away" as the fixture')
                break
            else:
                if fixture == 'Home':
                    x = 'HomeTeam' + x
                    y = 'HomeTeam' + y
                    all_home_stats = df_slim[['HomeTeam','HomeTeamFouls','HomeTeamYellowCards',
                                              'HomeTeamShots','HomeTeamShotsOnTarget','HomeTeamPoir
                                              'HomeTeamRedCards']].loc[df_slim['HomeTeam'] == team]

                    # Calculating Pearson's correlation coefficient, and evaluating the linear trer
                    coef = round(pearsonr(all_home_stats[x],all_home_stats[y])[0],4)
                    if coef >0:
                        corr_type = 'POSITIVE'
                    else:
                        corr_type = 'NEGATIVE'

                    # Creating a 2x1 plot containing correlation summary and the data points visual
                    fig,ax = plt.subplots(1,2,figsize=(16,9))
                    plt.subplot(2,2,2)
                    sns.regplot(x=all_home_stats[x],y=all_home_stats[y],
                                marker="x",color='seagreen')
                    plt.xlabel(x)
                    plt.ylabel(y)
                    plt.title(f'''{x} vs. {y}''',size='xx-large')

                    plt.subplot(2,2,1)
                    plt.title(f'''Team: {team}''',size='xx-large')
                    plt.axis('off')
                    plt.text(0.5,0.5,
                             f'''Pearson Correlation Coefficient:\n{coef}\n\nLinear Relationship:\r
                             va='center',ha='center',size='x-large')

                else:
                    # Follows the same structure as above
                    x = 'AwayTeam' + x
                    y = 'AwayTeam' + y
                    all_away_stats = df_slim[['AwayTeam','AwayTeamFouls','AwayTeamYellowCards',
                                              'AwayTeamShots','AwayTeamShotsOnTarget','AwayTeamPoir
                                              'AwayTeamRedCards']].loc[df_slim['HomeTeam'] == team]

                    coef = round(pearsonr(all_away_stats[x],all_away_stats[y])[0],4)
                    if coef >0:
                        corr_type = 'POSITIVE'
                    else:
```

```
                    corr_type = 'NEGATIVE'

                fig,ax = plt.subplots(1,2,figsize=(16,9))
                plt.subplot(2,2,2)
                sns.regplot(x=all_away_stats[x],y=all_away_stats[y],
                            marker="x",color='seagreen')
                plt.xlabel(x)
                plt.ylabel(y)
                plt.title(f'''{x} vs. {y}''',size='xx-large')

                plt.subplot(2,2,1)
                plt.title(f'''Team: {team}''',size='xx-large')
                plt.axis('off')
                plt.text(0.5,0.5,
                         f'''Pearson Correlation Coefficient:\n{coef}\n\nLinear Relationship:\r
                         va='center',ha='center',size='x-large')
```

In [33]:
```
# Error checking: Independent & Dependent variables
correlation('Fouls','Bananas','Arsenal','Home')
```

```
Error: One or more x,y values not recognised. Please enter a valid field from the following list:
['Fouls', 'YellowCards', 'Shots', 'ShotsOnTarget', 'Points', 'RedCards']
```

In [34]:
```
# Error checking: Team Name
correlation('ShotsOnTarget','Points','UniversityofLeeds','Home')
```

```
Team not recognised. Please enter a team from the following list:
['Charlton' 'Chelsea' 'Coventry' 'Derby' 'Leeds' 'Leicester' 'Liverpool'
 'Sunderland' 'Tottenham' 'Man United' 'Arsenal' 'Bradford' 'Ipswich'
 'Middlesbrough' 'Everton' 'Man City' 'Newcastle' 'Southampton' 'West Ham'
 'Aston Villa' 'Bolton' 'Blackburn' 'Fulham' 'Birmingham' 'West Brom'
 'Portsmouth' 'Wolves' 'Norwich' 'Crystal Palace' 'Wigan' 'Reading'
 'Sheffield United' 'Watford' 'Hull' 'Stoke' 'Burnley' 'Blackpool' 'QPR'
 'Swansea' 'Cardiff' 'Bournemouth' 'Brighton' 'Huddersfield' 'Brentford'
 "Nott'm Forest" 'Luton']
```

In [35]:
```
# Error checking: Fixture Type
correlation('Shots','Points','Man City','Ethiad Stadium')
```

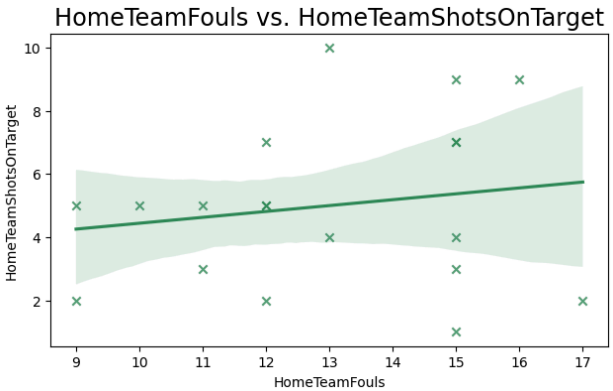
```
Please either enter "Home" or "Away" as the fixture
```

In [37]:
```
correlation('Fouls','ShotsOnTarget','Coventry','Home')
correlation('Fouls','ShotsOnTarget','Coventry','Away')

correlation('Fouls','ShotsOnTarget','Leeds','Home')
correlation('Fouls','ShotsOnTarget','Leeds','Away')
```

Team: Coventry

Pearson Correlation Coefficient:
0.1703

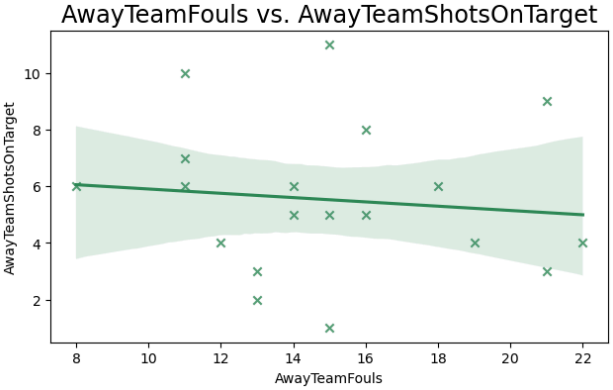
Linear Relationship:
POSITIVE



Team: Coventry

Pearson Correlation Coefficient:
-0.1102

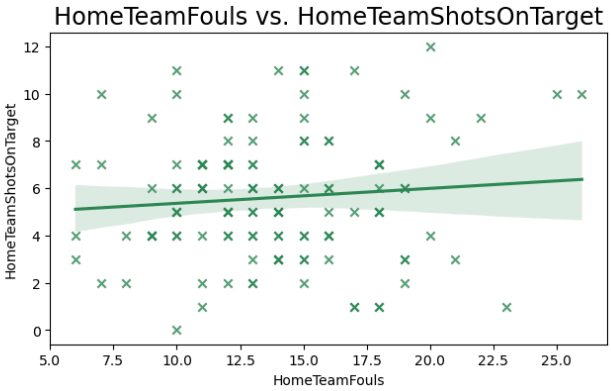
Linear Relationship:
NEGATIVE



Team: Leeds

Pearson Correlation Coefficient:
0.0942

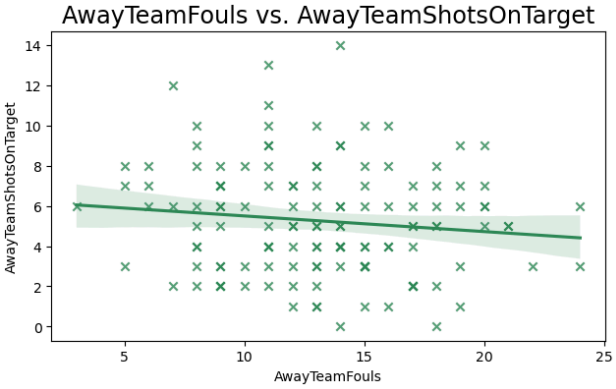
Linear Relationship:
POSITIVE



Team: Leeds

Pearson Correlation Coefficient:
-0.1203

Linear Relationship:
NEGATIVE

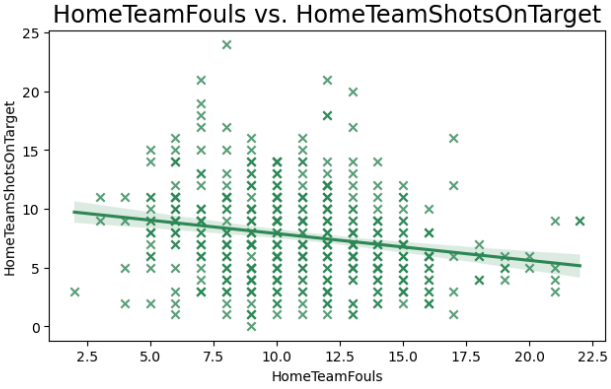


In [31]:
```
correlation('Fouls','ShotsOnTarget','Man United','Home')
correlation('Fouls','ShotsOnTarget','Man United','Away')

correlation('Fouls','ShotsOnTarget','Tottenham','Home')
correlation('Fouls','ShotsOnTarget','Tottenham','Away')
```

Team: Man United

Pearson Correlation Coefficient:
-0.2051

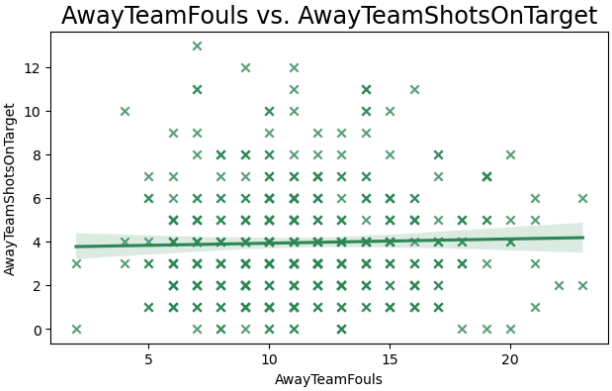
Linear Relationship:
NEGATIVE



Team: Man United

Pearson Correlation Coefficient:
0.0301

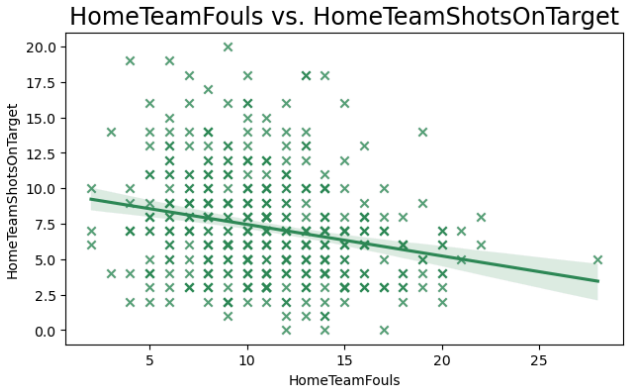
Linear Relationship:
POSITIVE



Team: Tottenham

Pearson Correlation Coefficient:
-0.2377

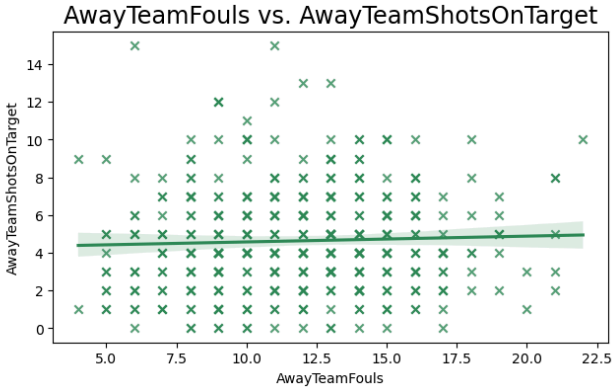
Linear Relationship:
NEGATIVE



Team: Tottenham

Pearson Correlation Coefficient:
0.0397

Linear Relationship:
POSITIVE



Observations

These results provide a useful preamble to regression analysis by focussing on anomalous data points previously identified, and evaluating the insight that correlation shows:

1. The first team explored is Coventry, identified as the most aggressive team overall but with little benefit to performance. However, when breaking down fouls committed against the average number of shots on target, there is some positive relationship uncovered. Pearson's Correlation Coefficient for home fixtures is +0.1703, indicating that more aggression (by fouls committed) creates more goal opportunities. However, knowing that Coventry is an under-performer in the English Premier League, it can be deduced that these results show **relationship without causation**. There is a clear link between aggression and performance, however these results alone are not enough to conclude that higher levels of aggression yield better performance. Leeds is another example of similar behaviour.
2. Man United both over-performs and under-aggresses in the English Premier League, and Pearson's Correlation Coefficient appears somewhat consistent with this statement. For home fixtures, there is a clear negative correlation between aggression and shots on target, whilst there is minimal relationship for away fixtures. Once again this uncovers an example of relationship without causation, as the correlation between aggression and performance is not enough to explain Man United's significant over index in performance. Tottenham is another good example of the same behaviour.

Conclusion

Achievements

Correlation analysis indicates there exists a relationship between aggression and performance, although the type of relationship varies from team to team, highlighting some inconsistencies.

For example: Some teams are comparatively less aggressive and exibit sub-par performance in the Premier League. However, correlation analysis suggests a postive relationship between aggression and performance. Conversely, some teams are comparatively higher performing and less aggressive in the Premier League, however analysis suggests a negative relationship between aggression and performance.

These are conflicting results; as such the lack of causation and context cannot conclude that aggression directly leads to better or worse performance.

Limitations

- Between 2000-2024, teams have participated within the English Premier League for different periods of time, accounting for both promotions and relegations. Therefore, very few teams have the exact same number of matches to compare against, which should be considered when examining any outputs to detect anomalies.
- The primary focus of this analysis is correlation, which aims to identify a relationship between aggression and performance. However, correlation is not a measure of causation, so the outputs cannot be used to deduce the extent for which aggression contributes to performance.
- Some data points are discrete (such as match points - measured as either 0,1 or 3) and should be examined carefully when assessing correlation. As a result, discrete fields relating directly to performance have not been analysed in this project.

Future Work

- In order to investigate causation, this project could be extended to conduct a full regression analysis. Identifying the 'strength' of the relationship between aggression and performance could help identify how significantly one metric impacts the other.
- The dataset could also be expanded to investigate other aggression metrics, such as player ratings, formation, and tackles. Additionally, given that many teams do not consistently compete in the English Premier League, the scope of analysis could be increased across all English divisions of professional football. This could build a clearer picture of the impact of aggression on performance, where each team has played a similar number of matches.

References

Cortés, I. 2024. Premier League [Online]. [Accessed 17 October 2024]. Available from: https://www.kaggle.com/datasets/ajaxianazarenka/premier-league/data