Lecture8.1_GroupBy

February 25, 2025

1 Pandas: `groupby`

- We have seen how to create Pandas Series and DataFrames, and how to read flat files in to Python using Pandas.
- We also learned about indexing, slicing and filtering in Pandas.
- We saw how to identify and deal with missing data.

Today we will look at how to use groupby to find out more about our data.

1.1 groupby

A `groupby` operation involves some combination of splitting the object, applying a function, and combining the results. This can be used to **group large amounts of data** and compute operations on these groups.

This process is sometimes called 'aggregation' because we aggregate over groups in the data.

1.1.1 What is an aggregation function?

It is a function that performs a calculation on multiple values and return a single *summarized* result.

We have seen these as examples in DataFrames and Series, where you applied, for example, the `.sum()` to find the total sum of a specific column: `pl['HTHG'].sum()`

Function	Description
`.sum()`	Adds up all values in a group
`.mean()`	Computes the average (mean) of a group
`.count()`	Counts the number of elements in a group
`.min()`	Finds the smallest value in a group
`.max()`	Finds the largest value in a group
`.median()`	Finds the median of a group
`.std()`	Computes the standard deviation
`.var()`	Computes the variance
`.value_counts()`	Return a Series containing counts of unique values.

1.2 Using `groupby` on a DataFrame

To illustrate the use of groupby, we will use a dataset containing data on all games from the `Premier League` for the 2017/18 and 2018/19 seasons. The main columns we will use are: 1.

Season: Whether the game was played in the 2017/18 or 2018/19 season 2. Date: The date on which the game was played 3. HomeTeam: The team that played at home in the match 4. AwayTeam: The team that played away from home in the match 5. FTHG: Full Time Home Goals 6. FTAG: Full Time Away Goals 7. Referee: The referee name 8. HR: Home Team Red Cards 9. AR: Home Team Red Cards

1.2.1 Example Premier League dataset using GroupBy

Import data

[1]:
```
# Import needed libraries
import numpy as np
import pandas as pd
import os
```

[2]:
```
# change current working library
directory = "C:/Users/cepedazk/Jupyter Notebook/Datasets/"
os.chdir(directory)
```

[3]:
```
# open the csv file using read_csv, no need to use delimitator (parameter 'sep'␣
↪in read_csv)
pl = pd.read_csv("pl_2seasons.csv")

display(pl.iloc[:10]) # show the first ten rows, also you can use pl.head(10)
```

	Season	Date	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	\
0	20172018	11/08/2017	Arsenal	Leicester	4	3	H	2	
1	20172018	12/08/2017	Brighton	Man City	0	2	A	0	
2	20172018	12/08/2017	Chelsea	Burnley	2	3	A	0	
3	20172018	12/08/2017	Crystal Palace	Huddersfield	0	3	A	0	
4	20172018	12/08/2017	Everton	Stoke	1	0	H	1	
5	20172018	12/08/2017	Southampton	Swansea	0	0	D	0	
6	20172018	12/08/2017	Watford	Liverpool	3	3	D	2	
7	20172018	12/08/2017	West Brom	Bournemouth	1	0	H	1	
8	20172018	13/08/2017	Man United	West Ham	4	0	H	1	
9	20172018	13/08/2017	Newcastle	Tottenham	0	2	A	0	

	HTAG	HTR	…	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR
0	2	D	…	10	3	9	12	9	4	0	1	0	0
1	0	D	…	2	4	6	9	3	10	0	2	0	0
2	3	A	…	6	5	16	11	8	5	3	3	2	0
3	2	A	…	4	6	7	19	12	9	1	3	0	0
4	0	H	…	4	1	13	10	6	7	1	1	0	0
5	0	D	…	2	0	10	13	13	0	2	1	0	0
6	1	H	…	4	5	14	8	3	3	0	3	0	0
7	0	H	…	6	2	15	3	8	2	3	1	0	0
8	0	H	…	6	1	19	7	11	1	2	2	0	0
9	0	D	…	3	6	6	10	5	7	1	2	1	0

```
[10 rows x 23 columns]
```

[4]:
```
pl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 760 entries, 0 to 759
Data columns (total 23 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Season    760 non-null    int64
 1   Date      760 non-null    object
 2   HomeTeam  760 non-null    object
 3   AwayTeam  760 non-null    object
 4   FTHG      760 non-null    int64
 5   FTAG      760 non-null    int64
 6   FTR       760 non-null    object
 7   HTHG      760 non-null    int64
 8   HTAG      760 non-null    int64
 9   HTR       760 non-null    object
 10  Referee   760 non-null    object
 11  HS        760 non-null    int64
 12  AS        760 non-null    int64
 13  HST       760 non-null    int64
 14  AST       760 non-null    int64
 15  HF        760 non-null    int64
 16  AF        760 non-null    int64
 17  HC        760 non-null    int64
 18  AC        760 non-null    int64
 19  HY        760 non-null    int64
 20  AY        760 non-null    int64
 21  HR        760 non-null    int64
 22  AR        760 non-null    int64
dtypes: int64(17), object(6)
memory usage: 136.7+ KB
```

[5]:
```
pl.Season.unique()

# Use unique to see the unique values within the dataframe.
# There are two seasons in the dataset
```

[5]:
```
array([20172018, 20182019], dtype=int64)
```

Using `groupby` to group rows in a DataFrame Use `groupby` method to group rows in a DataFrame, based on the values of one or more columns. This will return a `DataFrameGroupBy`object, which you can then use to apply `aggregate` functions, like `sum()`, `mean()`, `count()`, `min()`, and `max()`.

Key points: * It groups data based on unique values in a column. * It does not return a `DataFrame` immediately but a `GroupBy` object. * You need to use functions like `.sum()`, `.mean()`, or `.count()`

to get meaningful results.

Let's apply `groupby` on some columns on dataframe `pl`: * Think of some questions that can be answered using groupby for the `pl` data. * Group by seasons and find out how many half time goals were scored in each season. * Group by full time goals and find the tally for each team. * Group by referee and see how many red cards they gave * We will run some of your suggestions, along with the examples in the following cells.

Group by seasons and find out how many half time goals were scored in each season.

[10]:
```
print(pl.groupby('Season'))

# no dataframe is returned, but an GroupBy object
# you need to apply aggregation functions
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001A43C8C7CD0>
```

[7]:
```
pl.groupby('Season').FTHG

# this will be the same with columns, a groupby is returned, not a series
# again, you need to apply an aggregation function
```

[7]:
```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x000001A43DCCA010>
```

[8]:
```
# Lets now see when we apply an aggregation function
# In this case, lets use get the mean of Full-time home team goals group by␣
 ↪Season

pl.groupby('Season').FTHG.mean()

# We can see in season 17-18, the mean of full-time home team goals was 1.5,
#  whereas in the net session, 1.56
```

[8]:
```
Season
20172018    1.531579
20182019    1.568421
Name: FTHG, dtype: float64
```

[14]:
```
pl.groupby('Season').FTHG.sum()
```

[14]:
```
Season
20172018    582
20182019    596
Name: FTHG, dtype: int64
```

Group by full time goals and find the total score for each team.

[15]:
```
print(pl.groupby('HomeTeam').FTHG)

# Grouping data by Home Teams
```

```
# again, a groupby object is returned.
# Apply aggregation functions.
```

```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x000001A43E255250>
```

[16]:
```
# Get the mean of Full time home team goals for each Home Team
pl.groupby('HomeTeam').FTHG.mean()
```

[16]:
```
HomeTeam
Arsenal           2.526316
Bournemouth       1.473684
Brighton          1.131579
Burnley           1.052632
Cardiff           1.105263
Chelsea           1.815789
Crystal Palace    1.263158
Everton           1.526316
Fulham            1.157895
Huddersfield      0.684211
Leicester         1.289474
Liverpool         2.631579
Man City          3.105263
Man United        1.868421
Newcastle         1.184211
Southampton       1.236842
Stoke             1.052632
Swansea           0.894737
Tottenham         1.947368
Watford           1.394737
West Brom         1.105263
West Ham          1.473684
Wolves            1.473684
Name: FTHG, dtype: float64
```

[17]:
```
# Get the sum of Full time home team goals for each Home Team
pl.groupby('HomeTeam').FTHG.sum()
```

[17]:
```
HomeTeam
Arsenal           96
Bournemouth       56
Brighton          43
Burnley           40
Cardiff           21
Chelsea           69
Crystal Palace    48
Everton           58
Fulham            22
Huddersfield      26
```

```
Leicester          49
Liverpool         100
Man City          118
Man United         71
Newcastle          45
Southampton        47
Stoke              20
Swansea            17
Tottenham          74
Watford            53
West Brom          21
West Ham           56
Wolves             28
Name: FTHG, dtype: int64
```

- You can also group by multiple columns by adding a list of the name of the columns on `groupby` method.
- This is referred to as `hierarchical grouping`.
- Example below applies a group by on two columns: `Season` and `HomeTeam`, and then applies `mean()` function on Full Time Home Team Goals.

[18]:
```
print(pl.groupby(['Season', 'HomeTeam']).FTHG.mean())
```

```
Season    HomeTeam
20172018  Arsenal           2.842105
          Bournemouth       1.368421
          Brighton          1.263158
          Burnley           0.842105
          Chelsea           1.578947
          Crystal Palace    1.526316
          Everton           1.473684
          Huddersfield      0.842105
          Leicester         1.315789
          Liverpool         2.368421
          Man City          3.210526
          Man United        2.000000
          Newcastle         1.105263
          Southampton       1.052632
          Stoke             1.052632
          Swansea           0.894737
          Tottenham         2.105263
          Watford           1.421053
          West Brom         1.105263
          West Ham          1.263158
20182019  Arsenal           2.210526
          Bournemouth       1.578947
          Brighton          1.000000
          Burnley           1.263158
```

```
Cardiff           1.105263
Chelsea           2.052632
Crystal Palace    1.000000
Everton           1.578947
Fulham            1.157895
Huddersfield      0.526316
Leicester         1.263158
Liverpool         2.894737
Man City          3.000000
Man United        1.736842
Newcastle         1.263158
Southampton       1.421053
Tottenham         1.789474
Watford           1.368421
West Ham          1.684211
Wolves            1.473684
Name: FTHG, dtype: float64
```

[19]:
```
pl.groupby('Season').FTHG.sum()

# You can see grouping by `Season` and grouping by `Season and TeamHome`
#   have different structures when applying .sum() on Full-time home team goals.
```

[19]:
```
Season
20172018    582
20182019    596
Name: FTHG, dtype: int64
```

[20]:
```
# Another example of hierarchical grouping (also refer to multi-level grouping)
# Get the mean of Half-time away team goals group by Season and Away team
pl.groupby(['Season','AwayTeam']).HTAG.mean()
```

[20]:
```
Season    AwayTeam
20172018  Arsenal           0.368421
          Bournemouth       0.473684
          Brighton          0.421053
          Burnley           0.473684
          Chelsea           0.789474
          Crystal Palace    0.210526
          Everton           0.421053
          Huddersfield      0.263158
          Leicester         0.789474
          Liverpool         1.000000
          Man City          1.263158
          Man United        0.631579
          Newcastle         0.421053
          Southampton       0.368421
          Stoke             0.421053
```

```
            Swansea           0.210526
            Tottenham         0.684211
            Watford           0.368421
            West Brom         0.210526
            West Ham          0.526316
20182019    Arsenal           0.736842
            Bournemouth       0.684211
            Brighton          0.368421
            Burnley           0.526316
            Cardiff           0.157895
            Chelsea           0.578947
            Crystal Palace    0.631579
            Everton           0.578947
            Fulham            0.315789
            Huddersfield      0.368421
            Leicester         0.263158
            Liverpool         0.631579
            Man City          1.105263
            Man United        1.000000
            Newcastle         0.684211
            Southampton       0.421053
            Tottenham         0.947368
            Watford           0.578947
            West Ham          0.526316
            Wolves            0.368421
Name: HTAG, dtype: float64
```

[21]:
```
pl.groupby(['Season','HomeTeam']).FTHG.sum()
```

[21]:
```
Season    HomeTeam
20172018  Arsenal           54
          Bournemouth       26
          Brighton          24
          Burnley           16
          Chelsea           30
          Crystal Palace    29
          Everton           28
          Huddersfield      16
          Leicester         25
          Liverpool         45
          Man City          61
          Man United        38
          Newcastle         21
          Southampton       20
          Stoke             20
          Swansea           17
          Tottenham         40
```

```
            Watford           27
            West Brom         21
            West Ham          24
20182019    Arsenal           42
            Bournemouth       30
            Brighton          19
            Burnley           24
            Cardiff           21
            Chelsea           39
            Crystal Palace    19
            Everton           30
            Fulham            22
            Huddersfield      10
            Leicester         24
            Liverpool         55
            Man City          57
            Man United        33
            Newcastle         24
            Southampton       27
            Tottenham         34
            Watford           26
            West Ham          32
            Wolves            28
Name: FTHG, dtype: int64
```

[22]:
```
pl.groupby(['Season', 'HomeTeam']).FTHG.max()
```

[22]:
```
Season    HomeTeam
20172018  Arsenal           5
          Bournemouth       4
          Brighton          4
          Burnley           2
          Chelsea           5
          Crystal Palace    5
          Everton           4
          Huddersfield      4
          Leicester         3
          Liverpool         5
          Man City          7
          Man United        4
          Newcastle         3
          Southampton       4
          Stoke             3
          Swansea           4
          Tottenham         5
          Watford           4
          West Brom         2
```

```
           West Ham          3
20182019   Arsenal           5
           Bournemouth       4
           Brighton          3
           Burnley           4
           Cardiff           4
           Chelsea           5
           Crystal Palace    5
           Everton           4
           Fulham            4
           Huddersfield      1
           Leicester         3
           Liverpool         5
           Man City          6
           Man United        4
           Newcastle         3
           Southampton       3
           Tottenham         5
           Watford           4
           West Ham          4
           Wolves            4
Name: FTHG, dtype: int64
```

[23]:
```
pl.groupby(['Season', 'HomeTeam']).HR.sum()
```

[23]:
```
Season    HomeTeam
20172018  Arsenal           1
          Bournemouth       1
          Brighton          1
          Burnley           0
          Chelsea           3
          Crystal Palace    0
          Everton           1
          Huddersfield      2
          Leicester         2
          Liverpool         0
          Man City          1
          Man United        0
          Newcastle         2
          Southampton       0
          Stoke             1
          Swansea           0
          Tottenham         0
          Watford           2
          West Brom         0
          West Ham          0
20182019  Arsenal           0
```

```
Bournemouth       1
Brighton          2
Burnley           0
Cardiff           0
Chelsea           0
Crystal Palace    1
Everton           0
Fulham            1
Huddersfield      3
Leicester         2
Liverpool         1
Man City          0
Man United        1
Newcastle         1
Southampton       2
Tottenham         0
Watford           3
West Ham          0
Wolves            0
Name: HR, dtype: int64
```

Group by referee and see how many red cards they gave

[24]:
```
# group by referee name and then get the sum of red card on the away team
pl.groupby(['Referee']).AR.sum()
```

[24]:
```
Referee
A Madley        0
A Marriner      3
A Taylor        1
C Kavanagh      3
C Pawson        6
D Coote         0
G Scott         2
J Moss          4
K Friend        1
L Mason         2
L Probert       4
M Atkinson      3
M Dean         10
M Jones         1
M Oliver        7
N Swarbrick     0
P Tierney       0
R East          1
R Madley        1
S Attwell       2
S Hooper        0
```

```
Name: AR, dtype: int64
```

[25]:
```
pl.groupby(['Season', 'Referee'])[['HR', 'AR']].sum()
```

[25]:

Season	Referee	HR	AR
20172018	A Marriner	2	2
	A Taylor	1	0
	C Kavanagh	0	2
	C Pawson	4	0
	D Coote	0	0
	G Scott	1	1
	J Moss	1	2
	K Friend	0	0
	L Mason	0	2
	L Probert	1	1
	M Atkinson	3	2
	M Dean	0	3
	M Jones	0	1
	M Oliver	2	3
	N Swarbrick	0	0
	P Tierney	0	0
	R East	1	1
	R Madley	1	1
	S Attwell	0	1
	S Hooper	0	0
20182019	A Madley	0	0
	A Marriner	2	1
	A Taylor	0	1
	C Kavanagh	1	1
	C Pawson	1	6
	D Coote	0	0
	G Scott	0	1
	J Moss	3	2
	K Friend	2	1
	L Mason	1	0
	L Probert	1	3
	M Atkinson	0	1
	M Dean	3	7
	M Oliver	3	4
	P Tierney	1	0
	R East	0	0
	S Attwell	0	1
	S Hooper	0	0

Use `agg()` for using multiple aggregation functions Use `agg()` to run multiple functions simultaneously on a `groupby` object. Remove the `.` and `()` and just send the name of the aggrega-

tion function. For instance, for `.sum()`, you just indicate `'sum'` with single or double quotes `""`. Send multiple functions using a `list`.

For some reason mean does not work in the example below, you have to specify np.mean. Apparently this is a bug of Pandas!

[27]:
```
import numpy as np

display(pl.groupby(['Season','HomeTeam'])[['FTHG','FTAG']].agg(['sum', 'max','count', 'mean']))

# this also works
# pl.groupby(['Season','HomeTeam']).FTHG.agg([sum, max, len, np.mean]).head()
```

		FTHG				FTAG			
		sum	max	count	mean	sum	max	count	mean
Season	HomeTeam								
20172018	Arsenal	54	5	19	2.842105	20	3	19	1.052632
	Bournemouth	26	4	19	1.368421	30	4	19	1.578947
	Brighton	24	4	19	1.263158	25	5	19	1.315789
	Burnley	16	2	19	0.842105	17	3	19	0.894737
	Chelsea	30	5	19	1.578947	16	3	19	0.842105
	Crystal Palace	29	5	19	1.526316	27	3	19	1.421053
	Everton	28	4	19	1.473684	22	5	19	1.157895
	Huddersfield	16	4	19	0.842105	25	4	19	1.315789
	Leicester	25	3	19	1.315789	22	3	19	1.157895
	Liverpool	45	5	19	2.368421	10	3	19	0.526316
	Man City	61	7	19	3.210526	14	3	19	0.736842
	Man United	38	4	19	2.000000	9	2	19	0.473684
	Newcastle	21	3	19	1.105263	17	3	19	0.894737
	Southampton	20	4	19	1.052632	26	4	19	1.368421
	Stoke	20	3	19	1.052632	30	4	19	1.578947
	Swansea	17	4	19	0.894737	24	4	19	1.263158
	Tottenham	40	5	19	2.105263	16	4	19	0.842105
	Watford	27	4	19	1.421053	31	6	19	1.631579
	West Brom	21	2	19	1.105263	29	4	19	1.526316
	West Ham	24	3	19	1.263158	26	4	19	1.368421
20182019	Arsenal	42	5	19	2.210526	16	3	19	0.842105
	Bournemouth	30	4	19	1.578947	25	4	19	1.315789
	Brighton	19	3	19	1.000000	28	5	19	1.473684
	Burnley	24	4	19	1.263158	32	5	19	1.684211
	Cardiff	21	4	19	1.105263	38	5	19	2.000000
	Chelsea	39	5	19	2.052632	12	2	19	0.631579
	Crystal Palace	19	5	19	1.000000	23	3	19	1.210526
	Everton	30	4	19	1.578947	21	6	19	1.105263
	Fulham	22	4	19	1.157895	36	5	19	1.894737
	Huddersfield	10	1	19	0.526316	31	4	19	1.631579
	Leicester	24	3	19	1.263158	20	4	19	1.052632

Liverpool	55	5	19	2.894737	10	3	19	0.526316
Man City	57	6	19	3.000000	12	3	19	0.631579
Man United	33	4	19	1.736842	25	3	19	1.315789
Newcastle	24	3	19	1.263158	25	3	19	1.315789
Southampton	27	3	19	1.421053	30	3	19	1.578947
Tottenham	34	5	19	1.789474	16	3	19	0.842105
Watford	26	4	19	1.368421	28	4	19	1.473684
West Ham	32	4	19	1.684211	27	4	19	1.421053
Wolves	28	4	19	1.473684	21	3	19	1.105263

Use `apply()` for specifying functions yourself

- We use `.apply()` to apply a function specified by you.
- We will learn more about writing functions in tomorrow's class.
- Side note:
  - A **lambda function** is a *small* anonymous function that is defined using the keyword `lambda:`
  - The syntax of a lambda function is: `lambda arguments: expression`
  - `arguments` are the input parameters that the function will accept.
  - `expression`, the operation that the function will perform.
- What does the following function do?

[42]:
```
# where
# df is my argument
# and after ":" is my expression
pl.groupby(['Season','HomeTeam']).apply(lambda df: df.FTHG.iloc[0])
```

[42]:
```
Season    HomeTeam
20172018  Arsenal           4
          Bournemouth       0
          Brighton          0
          Burnley           0
          Chelsea           2
          Crystal Palace    0
          Everton           1
          Huddersfield      1
          Leicester         2
          Liverpool         1
          Man City          1
          Man United        4
          Newcastle         0
          Southampton       0
          Stoke             1
          Swansea           0
          Tottenham         1
          Watford           3
          West Brom         1
```

```
          West Ham          2
20182019  Arsenal           0
          Bournemouth       2
          Brighton          3
          Burnley           1
          Cardiff           0
          Chelsea           3
          Crystal Palace    0
          Everton           2
          Fulham            0
          Huddersfield      0
          Leicester         2
          Liverpool         4
          Man City          6
          Man United        2
          Newcastle         1
          Southampton       0
          Tottenham         3
          Watford           2
          West Ham          1
          Wolves            2
dtype: int64
```

Create a dataframe with the following information and store it in variable `dataEmployee`:

Department	Employee	Salary	Experience (Unit:Years)
HR	Alice	50000	5
HR	Bob	55000	6
HR	Alan	75000	10
IT	Charlie	70000	7
IT	Annie	62050	8
IT	Pearse	70000	7
IT	Diana	72000	8
IT	Sam	70000	7
IT	Daniel	52891	3
Finance	Eve	65005	6
Finance	Frank	68000	7
Marketing	Grace	60000	5
Marketing	Sarah	62000	6
Marketing	Brain	45500	2
Marketing	Helen	32150	1

- Group the data by the `Department` column and count how many employees are in each department.
- Find the total salary paid to employees in each department.
- Compute the average years of experience for employees in each department.
- Find the highest and lowest salary and experience in each department.

- Use `.agg()` to calculate the `mean` salary and `max` experience for each department.
- Group by both `Department` and `Experience` and find the average salary.
- Create a new column that shows each employee's salary as a percentage of the department's total salary.
- Use `.apply()` with `groupby()` to return the name of the employee with the highest salary in each department.