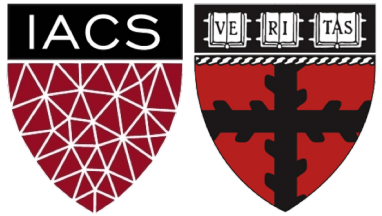
# Lecture 4: EDA and PANDAS

Exploring, investigating, and reconfiguring data.

## Harvard IACS

CS109A

Pavlos Protopapas, Kevin Rader, and Chris Tanner



# ANNOUNCEMENTS

- **Homework 1** has been released. **Due Sept 16 (Wed) @ 11:59pm**
- **Study Break** right after lecture (at **10:15am**)
- **Standard Sections** will be:
  - today at **1:30pm-2:45pm** and
  - **Monday 8:30pm-9:45pm**
- After lecture, please update your Zoom to the latest version

# Learning Objectives

- Understand the importance of exploring and investigating your data
- Feel comfortable using PANDAS to inspect your data
- Be able to perform advanced PANDAS operations

# Agenda

EDA

Advanced PANDAS

# Why is performing exploratory data analysis (EDA) important?

## EDA helps you:

- Ensure your data is as expected/valid/appropriate for the task
- Provides insights into a dataset
- Extract/determine important variables/attributes/features
- Detect outliers and anomalies
- Test underlying assumptions
- Make informed decisions in developing models

## **Approach**:

- EDA is an approach/philosophy **not** just a set of tools or techniques.
- Explore **global** properties: use histograms, scatter plots, and aggregation functions to summarize the data
- Explore **group** properties: group like-items together to compare subsets of the data (are the comparison results reasonable/expected?)
- This approach can be done at any time and any stage of the data science process

**Example**:

- Let's say that we are interested in the English Premier League (football/soccer) and want to build a model to predict a player's market value.

Question

Does age affect one's market value?

# **Example**: Get the data

| name | club | age | position | market value |
|---|---|---|---|---|
| Alexis Sanchez | Arsenal | 28 | LW | 65 |
| Mesut Ozil | Arsenal | 28 | AM | 50 |
| Petr Cech | Arsenal | 35 | GK | 7 |
| Theo Walcott | Arsenal | 28 | RW | 20 |
| Laurent Koscielny | Arsenal | 31 | CB | 22 |

from www.transfermarkt.us

# **Example**: Explore the data

| name | club | age | position | market value |
|---|---|---|---|---|
| Alexis Sanchez | Arsenal | 28 | LW | 65 |
| Mesut Ozil | Arsenal | 28 | AM | 50 |
| | | | GK | 7 |
| | | | RW | 20 |
| | | | CB | 22 |

- Credible/Trustworthy?
- Possibly subjective market values?
- Sampled data

from www.transfermarkt.us

# **Example**: Explore the data

| name | club | age | position | market value |
|---|---|---|---|---|
| Alexis Sanchez | Arsenal | 28 | LW | 65 |
| Mesut Ozil | Arsenal | 28 | AM | 50 |
| Petr Cech | Arsenal | 35 | GK | 7 |
| Theo Walcott | Arsenal | 28 | RW | 20 |
| Laurent Koscielny | Arsenal | 31 | CB | 22 |

from www.transfermarkt.us

# **Example**: Explore the data

| name | club | age | position | market value |
|---|---|---|---|---|
| Alexis Sanchez | Arsenal | 28 | LW | 65 |
| Mesut Ozil | Arsenal | 28 | AM | 50 |
| Petr Cech | Arsenal | 35 | GK | 7 |
| Theo Walcott | Arsenal | 28 | RW | 20 |
| Laurent Koscielr | | | | 22 |

Does it contain the necessary information?

w.transfermarkt.us

# **Example**: Explore the data

| name | club | age | position | market value |
|---|---|---|---|---|
| Alexis Sanchez | Arsenal | 28 | LW | 65 |
| Mesut Ozil | Arsenal | 28 | AM | 50 |
| Petr Cech | Arsenal | 35 | GK | 7 |
| Theo Walcott | Arsenal | 28 | RW | 20 |
| Laurent Koscielny | Arsenal | 31 | CB | 22 |

Missing data? Imputation needed?

# **Example**: Explore the data

| name | club | age | position | market value |
|---|---|---|---|---|
| Alexis Sanchez | Arsenal | 28 | LW | 65 |
| Mesut Ozil | Arsenal | 28 | AM | 50 |
| Petr Cech | Arsenal | 35 | GK | 7 |
| Theo Walcott | Arsenal | 28 | RW | 20 |
| Laurent Koscielny | Arsenal | 31 | CB | 22 |

Are the data types okay (`df.dtypes`)? Should be casted?

us

# **Example**: Explore the data

| name | club | age | position | market value |
|---|---|---|---|---|
| Alexis Sanchez | Arsenal | 28 | LW | 65 |
| Mesut Ozil | Arsenal | 28 | AM | 50 |
| Petr Cech | Arsenal | 35 | GK | 7 |
| Theo Walcott | Arsenal | 28 | RW | 20 |
| Laurent Koscielny | Arsenal | 31 | CB | 22 |

Are the values reasonable? `DataFrame.describe()` …

# Example: Explore the data

| | age | page_views | fpl_value | fpl_points | market_value |
|---|---|---|---|---|---|
| **count** | 461.000000 | 461.000000 | 461.000000 | 461.000000 | 461.000000 |
| **mean** | 26.804772 | 763.776573 | 5.447939 | 57.314534 | 11.012039 |
| **std** | 3.961892 | 931.805757 | 1.346695 | 53.113811 | 12.257403 |
| **min** | 17.000000 | 3.000000 | 4.000000 | 0.000000 | 0.050000 |
| **25%** | 24.000000 | 220.000000 | 4.500000 | 5.000000 | 3.000000 |
| **50%** | 27.000000 | 460.000000 | 5.000000 | 51.000000 | 7.000000 |
| **75%** | 30.000000 | 896.000000 | 5.500000 | 94.000000 | 15.000000 |
| **max** | 38.000000 | 7664.000000 | 12.500000 | 264.000000 | 75.000000 |

Are the values reasonable? `DataFrame.describe()` …

# Example: Explore the data

| | age | page_views | fpl_value | fpl_points | market_value |
|---|---|---|---|---|---|
| count | 461.0000 | | | | |
| mean | 26.8047 | | | | |
| std | 3.961892 | 931.805757 | 1.346695 | 55.113611 | 257403 |
| min | 17.000000 | 3.000000 | 4.000000 | 0.000000 | 0.050000 |
| 25% | 24.000000 | 220.000000 | 4.500000 | 5.000000 | 3.000000 |
| 50% | 27.000000 | 460.000000 | 5.000000 | 51.000000 | 7.000000 |
| 75% | 30.000000 | 896.000000 | 5.500000 | 94.000000 | 15.000000 |
| max | 38.000000 | 7664.000000 | 12.500000 | 264.000000 | 75.000000 |

This seems abnormally low. Is it correct? Who is this?

Are the values reasonable? `DataFrame.describe()` …

# Example: Explore the data

| | age | page_views | fpl_value | fpl_points | market_value |
|---|---|---|---|---|---|
| cou | | | | | 461.000000 |
| me | | | | | 11.012039 |
| s | | | | | 12.257403 |
| min | 17.000000 | 3.000000 | 4.000000 | 0.000000 | 0.050000 |
| 25% | 24.000000 | 220.000000 | 4.500000 | 5.000000 | 3.000000 |
| 50% | 27.000000 | 460.000000 | 5.000000 | 51.000000 | 7.000000 |
| 75% | 30.000000 | 896.000000 | 5.500000 | 94.000000 | 15.000000 |
| max | 38.000000 | 7664.000000 | 12.500000 | 264.000000 | 75.000000 |

This also seems suspicious. Is it correct? Who is this?

Are the values reasonable? `DataFrame.describe()` …

## Inspecting suspicious data

This accounts for both extreme values that we noticed. But, is this data **truly accurate?** It's worth validating online, elsewhere.

```
import pandas as pd
df = pd.read_csv("epl.csv")
df.iloc[df['market_value'].idxmin()]
```

```
name                Eduardo Carvalho
club                         Chelsea
age                               34
position                          LW
position_cat                       1
market_value                    0.05
page_views                       467
fpl_value                          5
fpl_sel                        0.10%
fpl_points                         0
region                             2
nationality                 Portugal
new_foreign                        0
age_cat                            6
club_id                            5
big_club                           1
new_signing                        1
Name: 109, dtype: object
```

# Example: Explore the data

| | age | page_views | fpl_value | fpl_points | market_value |
|---|---|---|---|---|---|
| coun | | | | | |
| mea | | | | | |
| st | | | | | |
| mi | | | | | |
| 25% | 24.000000 | 220.000000 | 4.500000 | 5.000000 | .000000 |
| 50% | 27.000000 | 460.000000 | 5.000000 | 51.000000 | 7.000000 |
| 75% | 30.000000 | 896.000000 | 5.500000 | 94.000000 | 15.000000 |
| max | 38.000000 | 7664.000000 | 12.500000 | 264.000000 | 75.000000 |

**What is going on here?!** Is someone actually paid that much more than others? And someone scores that much more?

from www.transfermarkt.us

```
df.loc[df['market_value'] >= 15].sort_values(by='market_value', ascending=False).head(15)
```

| | name | club | age | position | position_cat | market_value | page_views | fpl_value | fpl_se | fpl_points |
|---|---|---|---|---|---|---|---|---|---|---|
| **92** | Eden Hazard | Chelsea | 26 | LW | 1 | 75.0 | 4220 | 10.5 | 2.30% | 224 |
| **263** | Paul Pogba | Manchester+United | 24 | CM | 2 | 75.0 | 7435 | 8.0 | 19.50% | 115 |
| **0** | Alexis Sanchez | Arsenal | 28 | LW | 1 | 65.0 | 4329 | 12.0 | 17.10% | 264 |
| **241** | Sergio Aguero | Manchester+City | 29 | CF | 1 | 65.0 | 4046 | 11.5 | 9.70% | 175 |
| **240** | Kevin De Bruyne | Manchester+City | 26 | AM | 1 | 65.0 | 2252 | 10.0 | 17.50% | 199 |
| **377** | Harry Kane | Tottenham | 23 | CF | 1 | 60.0 | 4161 | 12.5 | 35.10% | 224 |
| **104** | N%27Golo Kante | Chelsea | 26 | DM | 2 | 50.0 | 4042 | 5.0 | 13.80% | 83 |
| **1** | Mesut Ozil | Arsenal | 28 | AM | 1 | 50.0 | 4395 | 9.5 | 5.60% | 167 |
| **260** | Romelu Lukaku | Manchester+United | 24 | CF | 1 | 50.0 | 3727 | 11.5 | 45.00% | 221 |
| **93** | Diego Costa | Chelsea | 28 | CF | 1 | 50.0 | 4454 | 10.0 | 3.00% | 196 |
| **214** | Philippe Coutinho | Liverpool | 25 | AM | 1 | 45.0 | 2958 | 9.0 | 30.80% | 171 |
| **242** | Raheem Sterling | Manchester+City | 22 | LW | 1 | 45.0 | 2074 | 8.0 | 3.80% | 149 |
| **376** | Dele Alli | Tottenham | 21 | CM | 2 | 45.0 | 4626 | 9.5 | 38.60% | 225 |
| **98** | Thibaut Courtois | Chelsea | 25 | GK | 4 | 40.0 | 1260 | 5.5 | 18.50% | 141 |
| **215** | Sadio Mane | Liverpool | 25 | LW | 1 | 40.0 | 3219 | 9.5 | 5.30% | 156 |

# Example: Explore the data

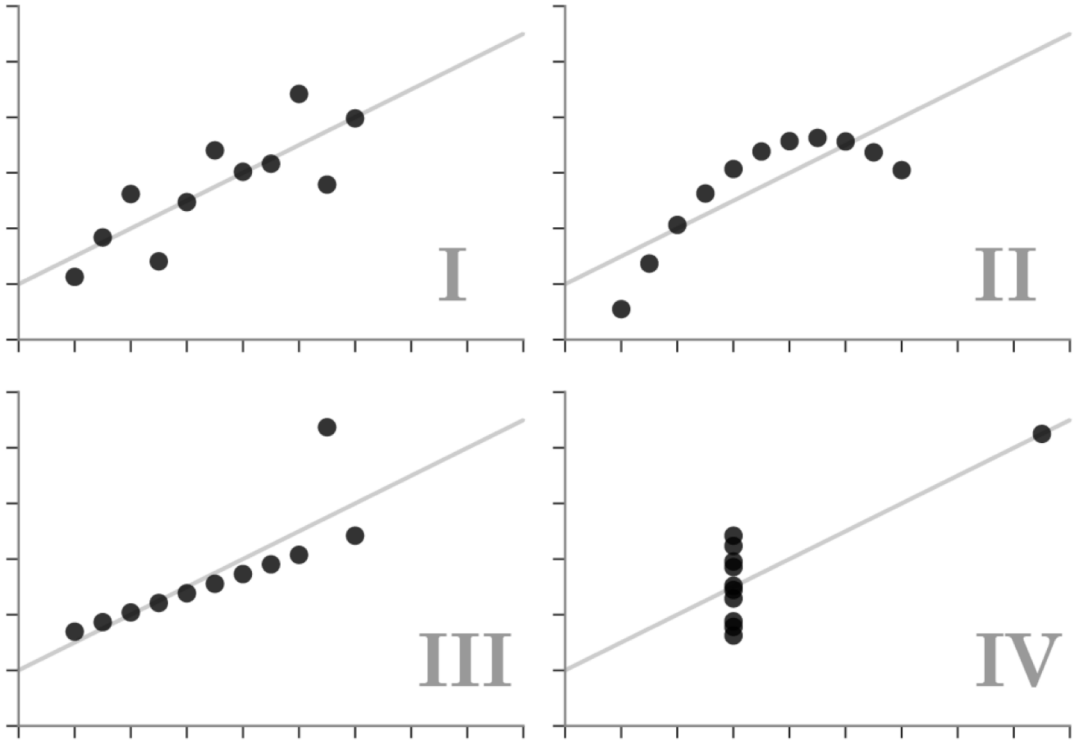
| | age | page_views | fpl_value | fpl_points | market_value |
|---|---|---|---|---|---|
| **count** | 461.000000 | 461.000000 | 461.000000 | 461.000000 | 461.000000 |
| **mean** | 26.804772 | 763.776573 | 5.447939 | 57.314534 | 11.012039 |
| **std** | 3.961892 | 931.805757 | 1.346695 | 53.113811 | 12.257403 |
| **min** | 17.000000 | 3.000000 | 4.000000 | 0.000000 | 0.050000 |
| **25%** | 24.000000 | 220.000000 | 4.500000 | 5.000000 | 3.000000 |
| **50%** | 27.000000 | 460.000000 | 5.000000 | 51.000000 | 7.000000 |
| **75%** | 30.000000 | 896.000000 | 5.500000 | 94.000000 | 15.000000 |
| **max** | 38.000000 | 7664.000000 | 12.500000 | 264.000000 | 75.000000 |

Summary statistics can only reveal so much

# Visualization

## Anscombe's Quartet

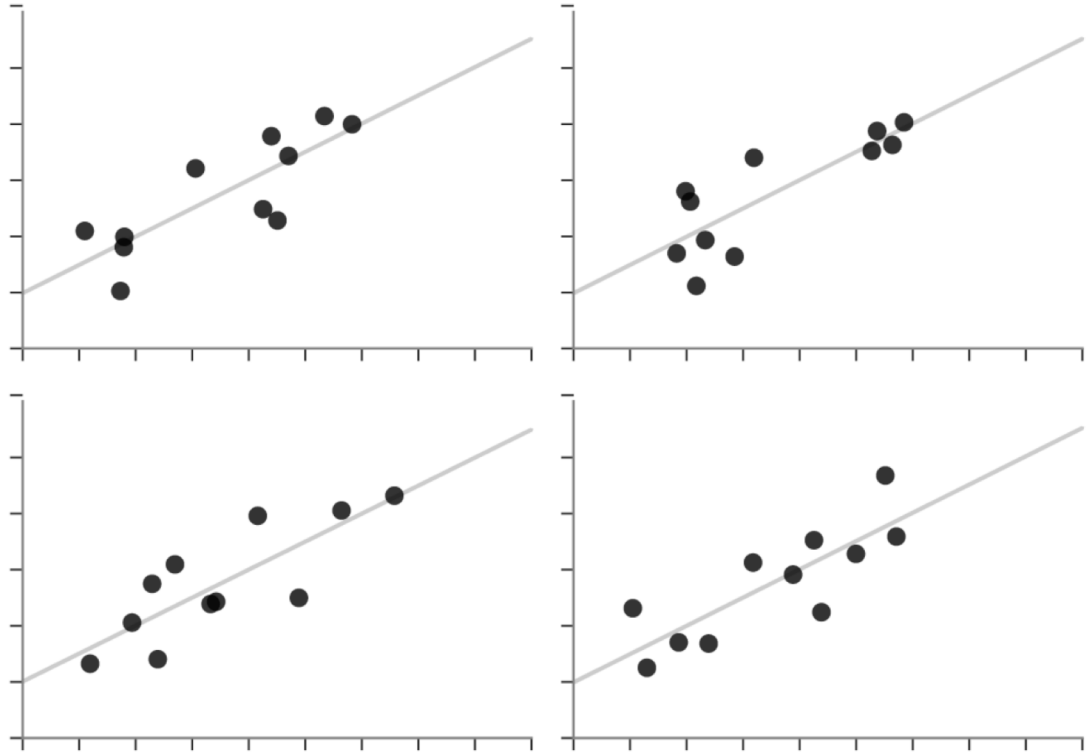
Each dataset has the same summary statistics (mean, standard deviation, correlation), and the datasets are *clearly different*, and *visually distinct*.




**Same stats** do not imply same graphs

## Unstructured Quartet

Each dataset here also has the same summary statistics. However, they are not *clearly different* or *visually distinct*.

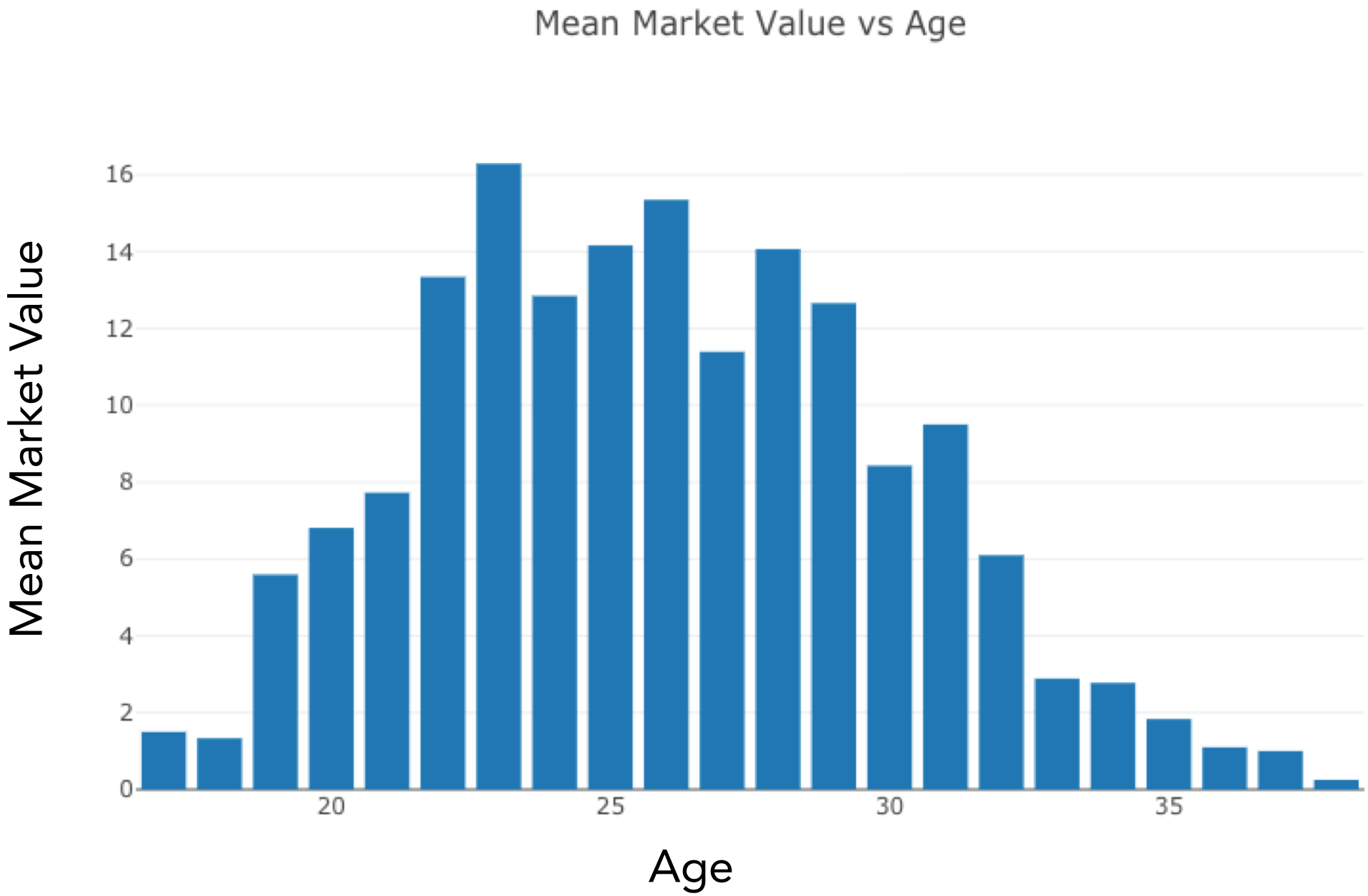


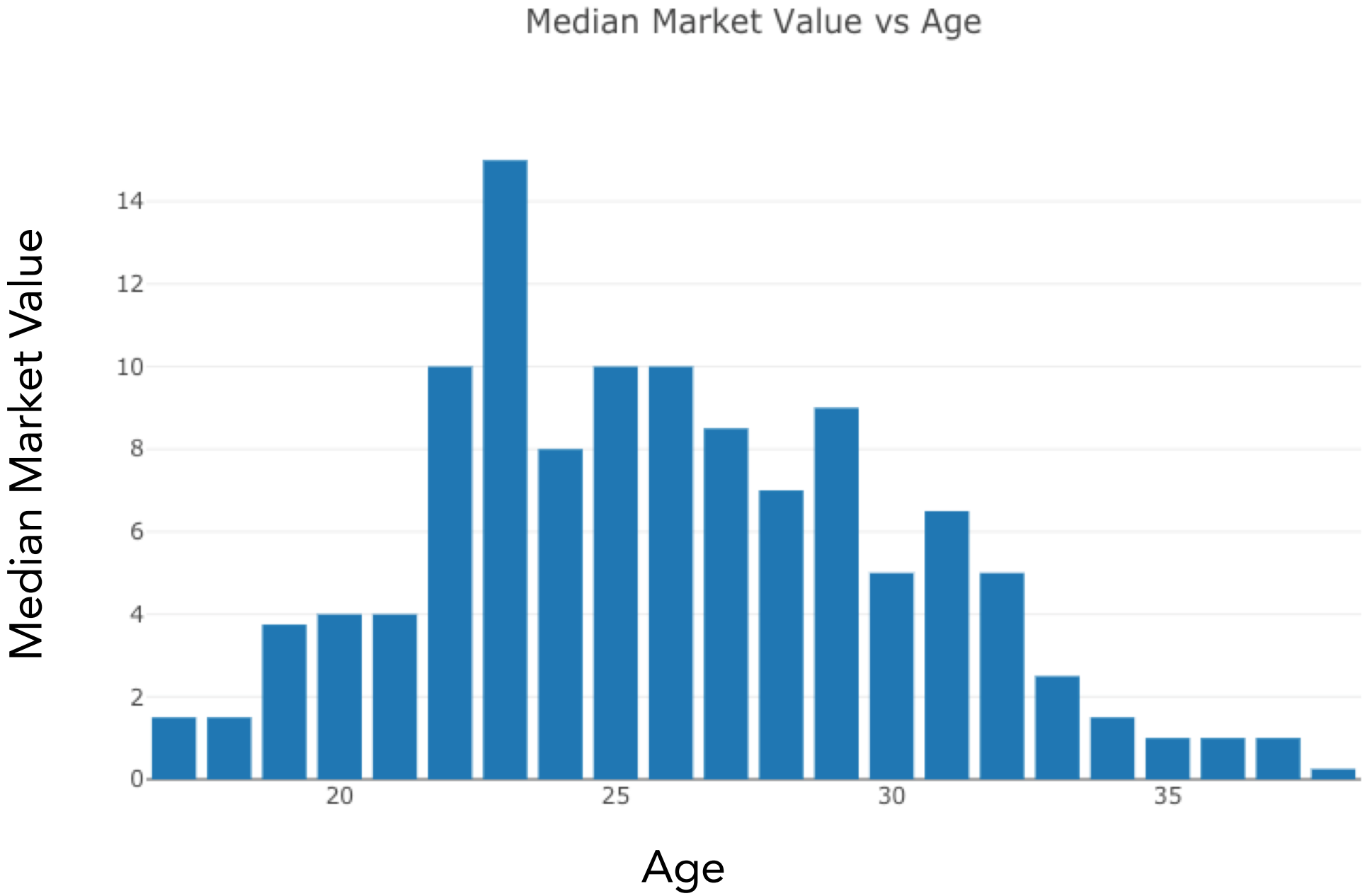
**Same graphs** do not imply same stats

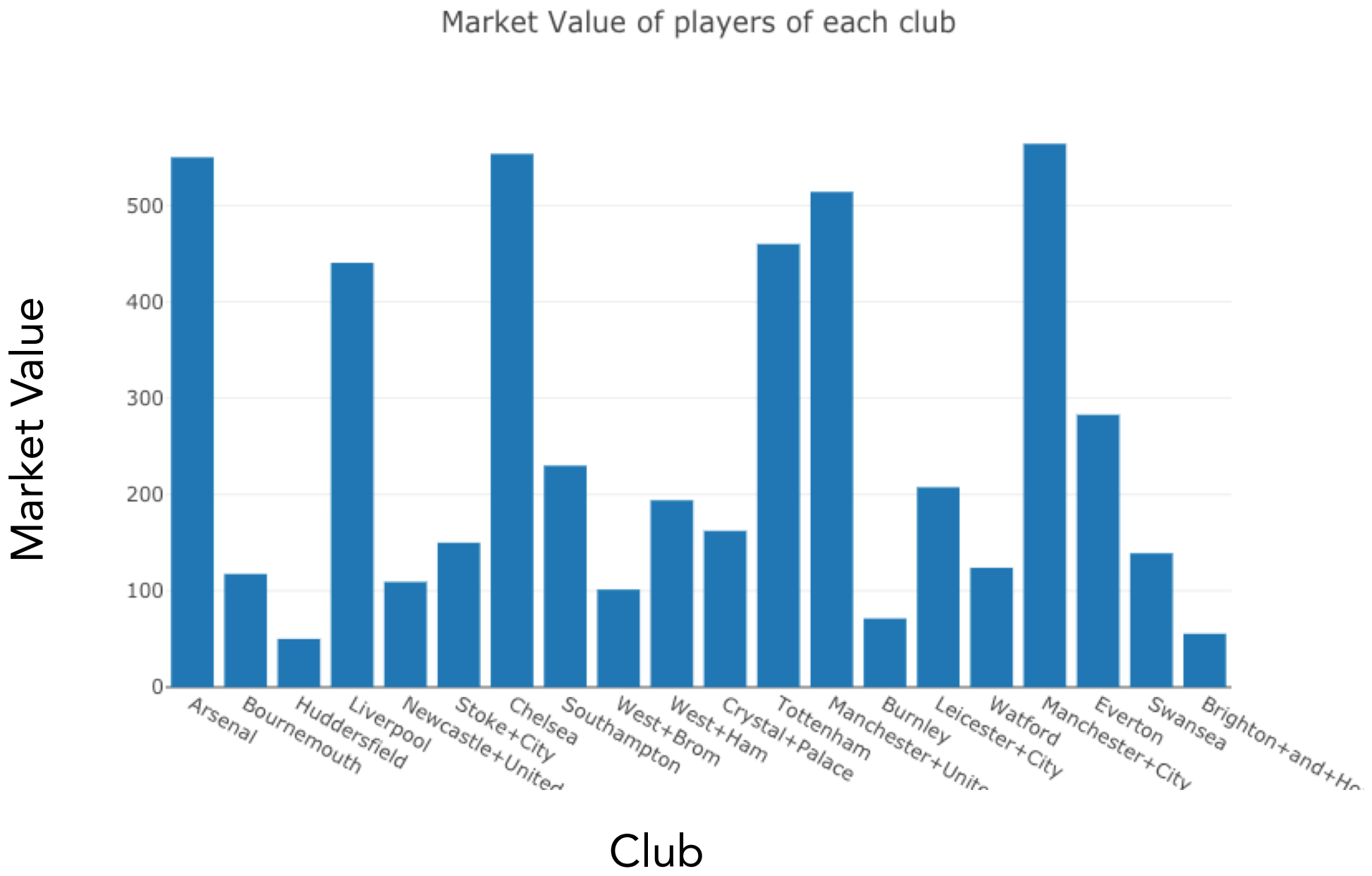
# Visualization

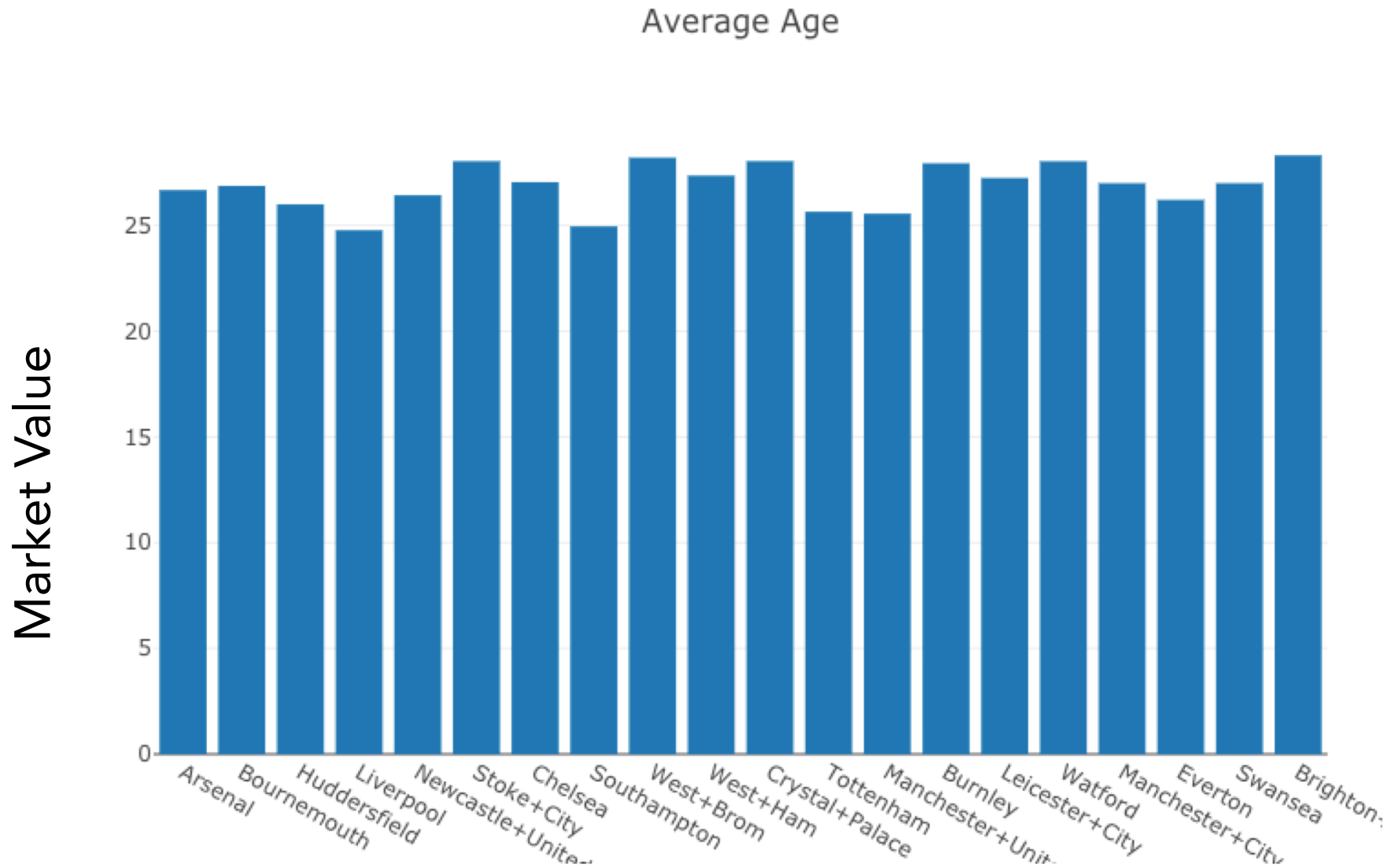
Visualization is incredibly important, both for EDA and for communicating your results to others.

Visualization packages will be used throughout the semester.


Mean Market Value vs Age
Mean Market Value
16
14
12
10
8
6
4
2
0
20
25
30
35
Age


Median Market Value vs Age
Median Market Value
14
12
10
8
6
4
2
0
20
25
30
35
Age


Market Value of players of each club
Market Value
500
400
300
200
100
0
Arsenal
Bournemouth
Huddersfield
Liverpool
Newcastle+United
Stoke+City
Chelsea
Southampton
West+Brom
West+Ham
Crystal+Palace
Tottenham
Manchester+Unite
Burnley
Leicester+City
Watford
Manchester+City
Everton
Swansea
Brighton+and+Ho
Club


Average Age
Market Value
25
20
15
10
5
0
Arsenal
Bournemouth
Huddersfield
Liverpool
Newcastle+Unite
Stoke+City
Chelsea
Southampton
West+Brom
West+Ham
Crystal+Palace
Tottenham
Manchester+Unit
Burnley
Leicester+City
Watford
Manchester+City
Everton
Swansea
Brighton
Club

# Useful PANDAS functions

- .read_csv() # loads a .csv file

**Accessing/processing:**

- df[“column_name”]
  - .max(), .min(),
  - .idxmax(), .idxmin()
- <dataframe> <conditional>
- .loc[] – label-based accessing
- .iloc[] – index-based accessing
- .sort_values()
- .isnull(), .notnull()
- .dropna()
- .any()
- .values() E.g., df[‘column’].values()
  - (df[‘name’] == “Chris”).any()
- [0:3] *# grab the first 3 rows of the DataFrame*

**Grouping/Splitting/Aggregating:**

- .groupby(),
- .get_groups()
- .drop()
- .merge(), .concat() .aggregate()
- .append()
- .sum() .median() .mean()

**High-level viewing:**

- .head() – first N observations
- .tail() – last N observations
- .describe() – statistics of the quantitative data
- .dtypes – the data types of the columns
- .columns – names of the columns
- .shape – the # of (rows, columns)

# Exercise 1 time!

# Useful PANDAS functions

- .read_csv() # loads a .csv file

**Accessing/processing:**

- df[“column_name”]
  - .max(), .min(),
  - .idxmax(), .idxmin()
- <dataframe> <conditional>
- .loc[] – label-based accessing
- .iloc[] – index-based accessing
- .sort_values()
- .isnull(), .notnull()
- .dropna()
- .any()
- .values() E.g., df[‘column’].values()
  - (df[‘name’] == “Chris”).any()
- [0:3] *# grab the first 3 rows of the DataFrame*

**Grouping/Splitting/Aggregating:**

- .groupby(),
- .get_groups()
- .drop()
- .merge(), .concat() .aggregate()
- .append()
- .sum() .median() .mean()

**High-level viewing:**

- .head() – first N observations
- .tail() – last N observations
- .describe() – statistics of the quantitative data
- .dtypes – the data types of the columns
- .columns – names of the columns
- .shape – the # of (rows, columns)

# Useful PANDAS functions

- .read_csv() # loads a .csv file

**Accessing/processing:**

- df[“column_name”]
  - .max(), .min(),
  - .idxmax(), .idxmin()
- <dataframe> <conditional>
- .loc[] – label-based accessing
- .iloc[] – index-based accessing
- .sort_values()
- .isnull(), .notnull()
- .dropna()
- .any()
- .values() E.g., df[‘column’].values()
  - (df[‘name’] == “Chris”).any()
- [0:3] *# grab the first 3 rows of the DataFrame*

**Grouping/Splitting/Aggregating:**

```
df.loc[(df['Height'] >= 172)]
```

```
df.loc[(df['Height'] >= 172) & (df['Height'] <= 240)]
```

```
df.loc[(df['Height'] < 150) | (df['Height'] > 180)]
```

- .head() – first N observations
- .tail() – last N observations
- .describe() – statistics of the quantitative data
- .dtypes – the data types of the columns
- .columns – names of the columns
- .shape – the # of (rows, columns)

# Useful PANDAS functions

- .read_csv() # loa

**Accessing/processi**

- df["column_nar
  - .max(), .mir
  - .idxmax(), .
- <dataframe> <
- .loc[] – label-ba
- .iloc[] – index-b
- .sort_values()
- .isnull(), .notnu
- .dropna()
- .any()
- .values() E.g., df[
  - (df['name'] ==
- [0:3] *# grab the fi*

**Grouping/Splitting/Aggregating:**

- groupby()

```
>>> df
      name         toy       born
0    Alfred         NaN        NaT
1    Batman   Batmobile 1940-04-25
2  Catwoman    Bullwhip        NaT
```

Drop the rows where at least one element is missing.

```
>>> df.dropna()
     name         toy       born
1  Batman   Batmobile 1940-04-25
```

Drop the columns where at least one element is missing.

```
>>> df.dropna(axis='columns')
      name
0    Alfred
1    Batman
2  Catwoman
```

ative data

mns

# Useful PANDAS functions

- .read_csv() # loads a .csv file

**Accessing/processing:**

- df[“column_name”]
  - .max(), .min(),
  - .idxmax(), .i
- <dataframe> <
- .loc[] – label-ba
- .iloc[] – index-b
- .sort_values()
- .isnull(), .notnu
- .dropna()
- .any()
- .values() E.g., df[
  - (df[‘name’] ==
- [0:3] *# grab the first 3 rows of the DataFrame*

**Grouping/Splitting/Aggregating:**

- .groupby(),
- .get_groups()
- .drop()
- .merge(), .concat() .aggregate()
- .append()

“Do any of my values equal True?”

DataFrame → Series

Series → Boolean

ative data

mns

.shape the # of (rows, columns)

## You can keep stacking operations,

## since it's just Python functions

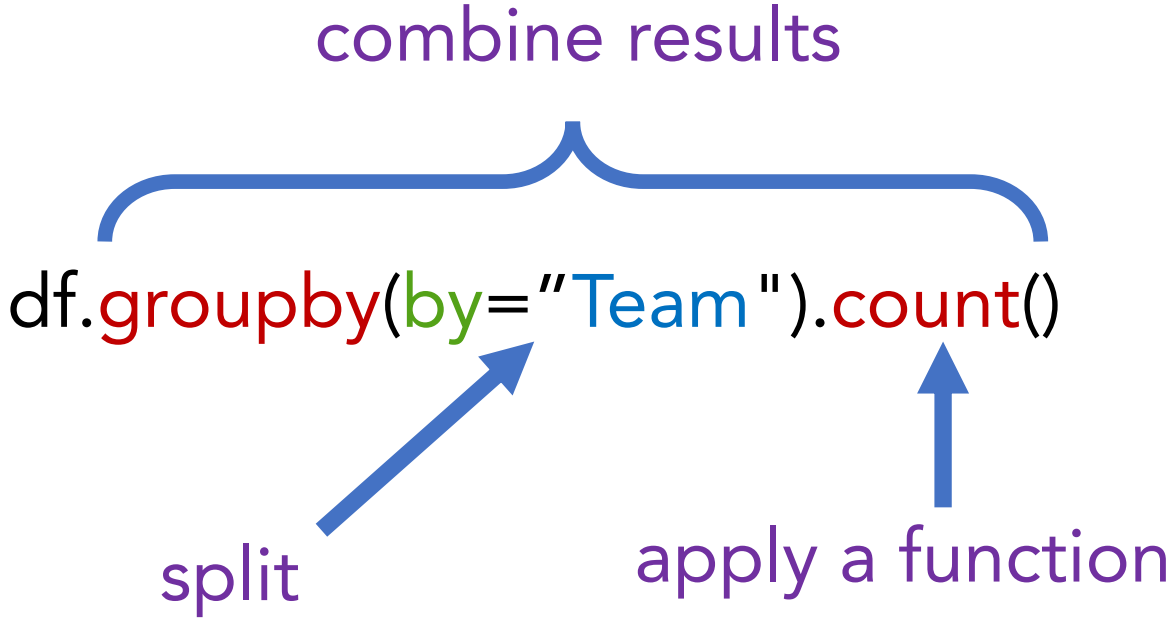
```
df.sort_values(by=["Popularity"], ascending=False).dropna().head(30)[['Energy', ', 'Genre']].groupby(by="Genre").median()
```

```
df.sort_values(by=["Popularity"], ascending=False).dropna().head(30)[['Energy', ', 'Genre']].groupby(by="Genre").median()
```

| Genre | Energy |
|---|---|
| atl hip hop | 59.0 |
| big room | 72.0 |
| brostep | 70.5 |
| canadian hip hop | 46.0 |
| country rap | 59.0 |
| dance pop | 68.0 |
| dfw rap | 56.5 |
| edm | 69.0 |
| electropop | 44.0 |
| escape room | 62.0 |
| latin | 78.0 |
| pop | 68.0 |
| pop house | 74.0 |
| r&b en espanol | 69.0 |
| reggaeton | 66.5 |
| reggaeton flow | 81.0 |
| trap music | 64.0 |

```
df.sort_values(by=["Popularity"], ascending=False).dropna().head(30)[['Energy', ', 'Genre']].groupby(by="Genre").median().
. sort_values(by =["Energy"])
```

| Genre | Energy |
|---|---|
| electropop | 44.0 |
| canadian hip hop | 46.0 |
| dfw rap | 56.5 |
| country rap | 59.0 |
| atl hip hop | 59.0 |
| escape room | 62.0 |
| trap music | 64.0 |
| reggaeton | 66.5 |
| dance pop | 68.0 |
| pop | 68.0 |
| edm | 69.0 |
| r&b en espanol | 69.0 |
| brostep | 70.5 |
| big room | 72.0 |
| pop house | 74.0 |
| latin | 78.0 |
| reggaeton flow | 81.0 |




"A groupby operation involves some combination of **splitting** the object, **applying** a function, and **combining** the results."

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html

# Exercise 2 time!

# Modelling!

- kNN Regression
- Linear Regression
- Multi- and Polynomial Regression
- Model selection and cross validation
- Inference: Bootstrapping and Confidence Intervals
- Regularization
- Visualization
- Classification: kNN and Logistic Regression
- Missing data
- PCA
- Trees
- Boosting
- Neural Networks!