Import Merged NBA Game Data

In [ ]:
```
import pandas as pd
NBA_Games=pd.read_csv("../../Data/Week 2/NBA_Games.csv")
NBA_Games.head()
```

Explore the dataset

Qualitative (Categorical) vs. Quantitative (Numerical) Data

-- To assess the variable type in Python, we use the "dtypes" command.

- object: qualitative variable -- variables that are not in numerical form
- int64: quantitative & discrete -- integer
- float64: quantitative & continuous -- real numbers that may contain decimal points

In [ ]:
```
NBA_Games.dtypes
```

In data analysis, we often convert categorical variable into dummy variable, if the observation belongs to the specified category, the dummy variable indicating the category would equal to 1, otherwise it equals to 0.

Convert a categorical variable to a dummy variable

The variable "WL" only carries two values, win or lose. We will create dummy variables to capture the categories.

We can use the "pd.get_dummies" function to convert a categorical variable to dummy variable. This function will also omit any missing value.

In [ ]:
```
dummy=pd.get_dummies(NBA_Games, columns=['WL'])
```

In [ ]:
```
dummy.columns
```

Notice that two variables are created, WL_L and WL_W. WL_L=1 if the team lost and WL_L=0 if the team won. The original variable WL is deleted.

We can attach the "WL_W" dummy variable back to our NBA_Games dataset using the pd.concat function.

- axis=1 specifies that we are adding a column to the dataset;
- axis=0 indicates adding rows

In [ ]:
```
NBA_Games=pd.concat([NBA_Games, dummy['WL_W']], axis=1)
NBA_Games.head()
```

Rename "WL_W" to "WIN"

In [ ]:
```
NBA_Games.rename(columns={'WL_W':'WIN'}, inplace=True)
NBA_Games.head()
```

Working with date variable

In sports, we often have to work with date and time data.

In [ ]:
```
NBA_Games['GAME_DATE'].dtype
```

The date variable is originally stored as an object. In this case, each date is treated equally without ordering.

We can use the "pd.*to*datetime()" command to convert the object variable to a date variable.

In [ ]:
```
import datetime
NBA_Games['GAME_DATE']=pd.to_datetime(NBA_Games['GAME_DATE'])
NBA_Games['GAME_DATE'].head()
```

Descriptive and Summary Analyses

Summarize numerical data

We can use the "describe()" command to calculate summary statistics. This will return basic summary statistics for all the numerical variables which include the total number of observations (count), the average, standard deviation, min and max, median, and the first and third quartiles of the values of the variable.

In [ ]:
```
NBA_Games.describe()
```

We can also add non-numerical variable into this summary statistics table by adding the argument "include='all'". You will see that for non-numerical variables, it provides the number of distinct values in the variable and the most frequently appeared value of the variable as well as its frequency. For date variable, in addition to providing the most frequent date appeared in the dataset, it also summarizes the start and end dates of the dataset.

In [ ]:
```
NBA_Games.describe(include='all')
```

We can summarize a single variable by specifying the variable.

In [ ]:
```
NBA_Games['PTS'].describe()
```

We can also calculate individual statistics by using the mean(), median(), std().

- Calculate mean of a numerical variable

In [ ]:
```
NBA_Games['FGM'].mean()
```

- Calculate median of a numerical variable

In [ ]:
```
NBA_Games['FGM'].median()
```

- Calculate standard deviation of a numerical variable

In [ ]:
```
NBA_Games['FGM'].std()
```

Self Test

1. Find the mean of field goals attempted;
2. Find the median of 3-point field goals made;
3. Find the standard deviation of the number of rebounds

In [ ]:
```
#Your Code Here
```

We can also calculate the summary statistics of a variable based on another variable, usually based on a different categorical variable.

- Calculate means by groups using "groupby" command.

In [ ]:
```
NBA_Games.groupby(['WL']).mean()
```

- Calculate the mean of a single (points in this example) variable by group.

In [ ]:
```
NBA_Games.groupby(['WL'])['PTS'].mean()
```

Summarize date variable

- We can find some basic statistics of the date variable. The describe() function returns the number of unique value of the date variable, the most frequently appeared date, i.e., the date with most number of games, and the first and the last dates.

In [ ]:
```
NBA_Games['GAME_DATE'].describe()
```

Visualizing data

Histogram

-- We can visualize the distribution of a variable using a histogram.

In [ ]:
```
NBA_Games.hist(column='PTS')
```

We can specify the number of bins in a histogram; different numbers of bins may give us slightly different graphs.

In [ ]:
```
NBA_Games.hist(column='PTS', bins=20)
```

For visual appeal, sometimes it may be helpful to add space between bins.

For example, we can narrow the bin to 0.9 width.

In [ ]:
```
NBA_Games.hist(column='PTS', bins=20, rwidth=0.9)
```

Save edited dataset

In [ ]:
```
NBA_Games.to_csv("../../Data/Week 2/NBA_Games2.csv", index=False)
```

In [ ]:
```

```