MSAS Tutorial Sequence -Module Five-

WRITTEN BY ADVAY MUCHOOR

GAME TIME

Goals of Today

1. Data cleaning
2. More row operations
3. Getting information from your data
   1. Column information operations
   2. Data Visualizations
4. Ideas for the last tutorial?

Cleaning Data

WHY DO WE NEED TO DO IT?

Values Out of Range

```
# Create a copy of passing (need to use the copy function in order to create a new dataframe)
best_ratio = passing_stats.copy(deep=True)
# If we do not use the copy function, changing our new table will also change our original (which we don't want!!!)

best_ratio["TD Ratio"] = round(best_ratio["TD"] / best_ratio["Int"], 2)
best_ratio = best_ratio.sort_values("TD Ratio", ascending = False)
best_ratio.head()
```

	Rk	Player	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk	TD Ratio
105	106	Albert Wilson	MIA	26	wr	7	3	1	1	100.0	52	1	0	52.0	52.0	7.4	158.3	99.7	0	inf
74	75	Mohamed Sanu	ATL	29	WR	16	16	1	2	50.0	5	1	0	2.5	5.0	0.3	95.8	32.3	0	inf
84	85	Kevin Byard	TEN	25	FS	16	16	1	1	100.0	66	1	0	66.0	66.0	4.1	158.3	NaN	0	inf
81	82	Chris Boswell	PIT	27	K	15	0	1	1	100.0	2	1	0	2.0	2.0	0.1	118.7	NaN	0	inf
77	78	Danny Amendola	MIA	33	WR	15	15	1	1	100.0	28	1	0	28.0	28.0	1.9	158.3	100.0	0	inf

Invalid/Unwanted Data

```
best_completions = passing_stats.sort_values("Cmp%", ascending = False)
best_completions.head()
```

	Rk	Player	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
105	106	Albert Wilson	MIA	26	wr	7	3	1	1	100.0	52	1	0	52.0	52.0	7.4	158.3	99.7	0
94	95	Sam Koch	BAL	36	P	16	0	1	1	100.0	21	0	0	21.0	21.0	1.3	118.7	11.8	0
70	71	Odell Beckham	NYG	26	WR	12	12	2	2	100.0	106	2	0	53.0	53.0	8.8	158.3	NaN	0
71	72	Julian Edelman	NWE	32	WR	12	12	2	2	100.0	43	0	0	21.5	21.5	3.6	118.7	94.3	0
76	77	Nelson Agholor	PHI	25	WR	16	16	1	1	100.0	15	0	0	15.0	15.0	0.9	118.7	100.0	0

Filtering

- Most of these cleaning operations can be taken care of simply by using inequalities
  - We learned this last module
    - Check if the position is a quarterback
    - Set a threshold for the number of passes thrown
    - Set a threshold for the number of games thrown/started
- We can exclude these rows from our dataset simply because we do not need them
- But what about the rows where values are undefined?

Missing values

In [41]:

```
passing_stats = pd.read_excel("nfl_stats.xlsx")
passing_stats
```

85	86	Tarik Cohen*+	CHI	23	rb/wr	16	7	1	1	100.0	1	1	0	1.0	1.0	0.1	118.7	100.0	0
86	87	Logan Cooke	JAX	23	P	16	0	1	1	100.0	4	0	0	4.0	4.0	0.3	83.3	NaN	0
87	88	Eric Ebron*	IND	25	te/wr	16	8	0	1	0.0	0	0	0	0.0	NaN	0.0	39.6	0.0	0
88	89	Bruce Ellington	2TM	27	NaN	7	3	0	1	0.0	0	0	0	0.0	NaN	0.0	39.6	NaN	0
89	90	Larry Fitzgerald	ARI	35	WR	16	16	1	1	100.0	32	1	0	32.0	32.0	2.0	158.3	100.0	0
90	91	Dontrell Hilliard	CLE	23	NaN	11	0	0	1	0.0	0	0	1	0.0	NaN	0.0	0.0	0.0	0
91	92	DeAndre Hopkins*+	HOU	26	WR	16	16	0	1	0.0	0	0	0	0.0	NaN	0.0	39.6	2.9	0
92	93	Darius Jennings	TEN	26	NaN	16	0	1	1	100.0	21	0	0	21.0	21.0	1.3	118.7	99.9	0
93	94	Zay Jones	BUF	23	WR	16	15	0	1	0.0	0	0	0	0.0	NaN	0.0	39.6	0.0	0
94	95	Sam Koch	BAL	36	P	16	0	1	1	100.0	21	0	0	21.0	21.0	1.3	118.7	11.8	0
95	96	Christian McCaffrey	CAR	22	RB	16	16	1	1	100.0	50	1	0	50.0	50.0	3.1	158.3	100.0	0
96	97	Anthony Miller	CHI	24	wr	15	4	1	1	100.0	8	0	0	8.0	8.0	0.5	100.0	81.2	0
97	98	Matt Prater	DET	34	K	16	0	1	1	100.0	8	1	0	8.0	8.0	0.5	139.6	NaN	0
98	99	Emmanuel Sanders	DEN	31	WR	12	12	1	1	100.0	28	1	0	28.0	28.0	2.3	158.3	100.0	0

“NaN” Values

- An “NaN” value is a value that is undefined
- Stands for “Not a Number”
  - Usually placeholders for cells that were left empty in the original DataFrame
  - Can also be used to represent division by zero in some cases
- Why do we care about NaN values?
  - Disruptive to our DataFrame
  - Messes up operations on a series we want to do

Why are NaN values so bad???

```
# fails because NaN values exist
all_qbs = passing_stats[passing_stats["Pos"].str.contains("qb")]
```

```
---------------------------------------------------------------------------
--
ValueError                                Traceback (most recent call las
t)
<ipython-input-94-47b7ff64c0cb> in <module>
      1 # fails because NaN values exist
----> 2 all_qbs = passing_stats[passing_stats["Pos"].str.contains("qb")]

~\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, k
ey)
   2915
   2916         # Do we have a (boolean) 1d indexer?
-> 2917         if com.is_bool_indexer(key):
   2918             return self._getitem_bool_array(key)
   2919

~\Anaconda3\lib\site-packages\pandas\core\common.py in is_bool_indexer(ke
y)
    122             if not lib.is_bool_array(key):
    123                 if isna(key).any():
--> 124                     raise ValueError(na_msg)
    125                 return False
    126             return True

ValueError: cannot index with vector containing NA / NaN values
```

Locate rows with NaN values

```
# Find the rows that have missing values
# Allows you to get a better feel for the types of errors that are ocurring
nan_rows = passing_stats[passing_stats.isnull().any(axis=1)]
nan_rows.head(10)
```

	Rk	Player	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
50	51	DeShone Kizer	GNB	22	NaN	3	0	20	42	47.6	187	0	2	4.5	9.4	62.3	40.5	25.2	4
55	56	Mike Glennon	ARI	29	NaN	2	0	15	21	71.4	174	1	0	8.3	11.6	87.0	112.0	77.7	1
56	57	Matt Cassel	DET	36	NaN	2	0	7	17	41.2	59	0	1	3.5	8.4	29.5	26.3	8.8	1
57	58	Joshua Dobbs	PIT	23	NaN	5	0	6	12	50.0	43	0	1	3.6	7.2	8.6	24.0	58.0	0
59	60	Matt Schaub	ATL	37	NaN	3	0	5	7	71.4	20	0	0	2.9	4.0	6.7	74.1	60.6	0
60	61	Robert Griffin	BAL	28	NaN	3	0	2	6	33.3	21	0	0	3.5	10.5	7.0	44.4	2.1	0
61	62	Kyle Lauletta	NYG	23	NaN	2	0	0	5	0.0	0	0	1	0.0	NaN	0.0	0.0	0.1	0
62	63	Jacoby Brissett	IND	26	NaN	4	0	2	4	50.0	2	0	0	0.5	1.0	0.5	56.2	100.0	0
63	64	Johnny Hekker	LAR	28	P	16	0	2	4	50.0	19	0	0	4.8	9.5	1.2	63.5	NaN	0
64	65	Geno Smith	LAC	28	NaN	5	0	1	4	25.0	8	0	0	2.0	8.0	1.6	39.6	0.7	1

Fill all NaN values

```
# Fill all NaN values with the same value
# Look at the 50th row (as seen in the previous cell) to see the effective changes
fill_zeros = passing_stats.fillna(0)
fill_zeros.loc[[50]]
```

	Rk	Player	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
50	51	DeShone Kizer	GNB	22	0	3	0	20	42	47.6	187	0	2	4.5	9.4	62.3	40.5	25.2	4

- When would filling all values with the same value be useful
  - If the entire table is comprised of strings, we can fill an NaN cell with an empty string
  - If the entire table is filled with integers, we can fill with a -1 and only check values that are positive
- Other than this, there aren’t many use cases
  - It is better to be more specific in the ways you want to deal with NaN values
  - How can we do this?

Option #1 – Delete all rows with NaN values

```
# remove all rows with NaN values
removed_NaN = passing_stats[~passing_stats.isnull().any(axis=1)]
removed_NaN.head()
```

	Rk	Player	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
0	1	Ben Roethlisberger	PIT	36	QB	16	16	452	675	67.0	5129	34	16	7.6	11.3	320.6	96.5	71.0	24
1	2	Andrew Luck*	IND	29	QB	16	16	430	639	67.3	4593	39	15	7.2	10.7	287.1	98.7	69.4	18
2	3	Matt Ryan	ATL	33	QB	16	16	422	608	69.4	4924	35	7	8.1	11.7	307.8	108.1	68.5	42
3	4	Kirk Cousins	MIN	30	QB	16	16	425	606	70.1	4298	30	10	7.1	10.1	268.6	99.7	58.2	40
4	5	Aaron Rodgers*	GNB	35	QB	16	16	372	597	62.3	4442	25	2	7.4	11.9	277.6	97.6	54.4	49

```
removed_NaN.shape
```

(69, 19)

Make sure to restart the index!

```
removed_NaN.loc[50:60]
```

	Rk	Player	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
51	52	Mark Sanchez	WAS	32	qb	2	1	19	35	54.3	138	0	3	3.9	7.3	69.0	28.0	4.4	7
52	53	Kyle Allen	CAR	22	qb	2	1	20	31	64.5	266	2	0	8.6	13.3	133.0	113.1	96.4	0
53	54	Matt Barkley	BUF	28	qb	1	1	15	25	60.0	232	2	0	9.3	15.5	232.0	117.4	83.4	1
54	55	Teddy Bridgewater	NOR	26	qb	5	1	14	23	60.9	118	1	1	5.1	8.4	23.6	70.6	39.8	2
58	59	Taysom Hill	NOR	28	te/wr	16	4	3	7	42.9	64	0	1	9.1	21.3	4.0	36.3	41.1	1

Make sure to restart the index!

```
removed_NaN = removed_NaN.reset_index(drop=True)
removed_NaN.loc[50:60]
```

	Rk	Player	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
50	52	Mark Sanchez	WAS	32	qb	2	1	19	35	54.3	138	0	3	3.9	7.3	69.0	28.0	4.4	7
51	53	Kyle Allen	CAR	22	qb	2	1	20	31	64.5	266	2	0	8.6	13.3	133.0	113.1	96.4	0
52	54	Matt Barkley	BUF	28	qb	1	1	15	25	60.0	232	2	0	9.3	15.5	232.0	117.4	83.4	1
53	55	Teddy Bridgewater	NOR	26	qb	5	1	14	23	60.9	118	1	1	5.1	8.4	23.6	70.6	39.8	2
54	59	Taysom Hill	NOR	28	te/wr	16	4	3	7	42.9	64	0	1	9.1	21.3	4.0	36.3	41.1	1
55	68	Derrick Henry	TEN	24	RB	16	12	2	3	66.7	14	0	0	4.7	7.0	0.9	77.1	59.3	0
56	72	Julian Edelman	NWE	32	WR	12	12	2	2	100.0	43	0	0	21.5	21.5	3.6	118.7	94.3	0
57	74	Jarvis Landry*	CLE	26	WR	16	14	1	2	50.0	63	0	0	31.5	63.0	3.9	95.8	100.0	0
58	75	Mohamed Sanu	ATL	29	WR	16	16	1	2	50.0	5	1	0	2.5	5.0	0.3	95.8	32.3	0
59	77	Nelson Agholor	PHI	25	WR	16	16	1	1	100.0	15	0	0	15.0	15.0	0.9	118.7	100.0	0
60	78	Danny Amendola	MIA	33	WR	15	15	1	1	100.0	28	1	0	28.0	28.0	1.9	158.3	100.0	0

Option #2, Fill each column differently

- The first step in this is to determine which columns have NaN values. This is done pretty easily using the following function

```
# Find out which columns have NaN values
passing_stats.columns[passing_stats.isna().any()].tolist()
```

```
['Pos', 'Y/C', 'QBR']
```

Option #2, Fill each column differently

```
# Set up a dictionary to hold the deafult values to fill NaN values
defaults = {
    "Pos" : "NA",
    "Y/C" : 0,
    "QBR" : 0
}

# Fill these values in the table
cleaned = passing_stats.fillna(defaults)
cleaned.loc[[50]]
```

	Rk	Player	Tm	Age	Pos	G	GS	Cmp	Att	Cmp%	Yds	TD	Int	Y/A	Y/C	Y/G	Rate	QBR	Sk
50	51	DeShone Kizer	GNB	22	NA	3	0	20	42	47.6	187	0	2	4.5	9.4	62.3	40.5	25.2	4

Final tips for data cleaning

- Always be sure to clean your data before doing any analysis
- Be mindful of the values you are removing from the table
  - Make sure you are aware of the consequences of removing or changing values
  - Excluding values that you should not be excluding can have adverse effects on your final analysis

Getting information from the table

- The next section of this module will focus on trends and visualizations
  - Not designed to be exhaustive, but rather point you in the right direction
- A lot of analysis is designed for you to determine what is important and what isn't
- Think about your goals before you start cleaning and visualizing!

Let’s transition to a new dataset

- Every game log of the 2018 season for every player
- Open up a new jupyter notebook and read in the new file

```
import pandas as pd
```

```
df = pd.read_excel("nba_stats.xlsx")
```

```
df.head()
```

	Name	Player_Game_Number	Date	Age	Team	Location	Opponent	Result	GS	TP	...	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	Plus
0	Álex Abrines	1	2018-10-16	25-076	OKC	Away	GSW	L (-8)	0	23:28	...	0	2	2	0	0	0	0	2	8	
1	Álex Abrines	2	2018-10-19	25-079	OKC	Away	LAC	L (-16)	0	32:06	...	1	1	2	1	1	0	2	0	10	
2	Álex Abrines	3	2018-10-21	25-081	OKC	Home	SAC	L (-11)	0	5:20	...	0	1	1	2	0	0	0	0	0	
3	Álex Abrines	4	2018-10-25	25-085	OKC	Home	BOS	L (-6)	0	18:33	...	0	1	1	1	0	0	0	2	6	
4	Álex Abrines	5	2018-10-28	25-088	OKC	Home	PHO	W (+7)	0	24:20	...	0	1	1	0	2	0	0	2	2	

Let’s understand the dataset

- What do the columns mean?
- What are the types of each column?
- What is the dataset telling me?
- What possible trends could I visualize?

Get information about a column

```
df["AST"].max()
```

```
24
```

```
df["AST"].describe()
```

```
count    26101.000000
mean         2.317268
std          2.525151
min          0.000000
25%          0.000000
50%          2.000000
75%          3.000000
max         24.000000
Name: AST, dtype: float64
```

```
df[df["AST"] == df["AST"].max()]
```

	Name	Player_Game_Number	Date	Age	Team	Location	Opponent	Result	GS	TP	...	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PT
24962	Russell Westbrook	33	2019-01-10	30-059	OKC	Away	SAS	L (-7)	1	49:35	...	2	11	13	24	2	0	3	4	2

1 rows × 29 columns

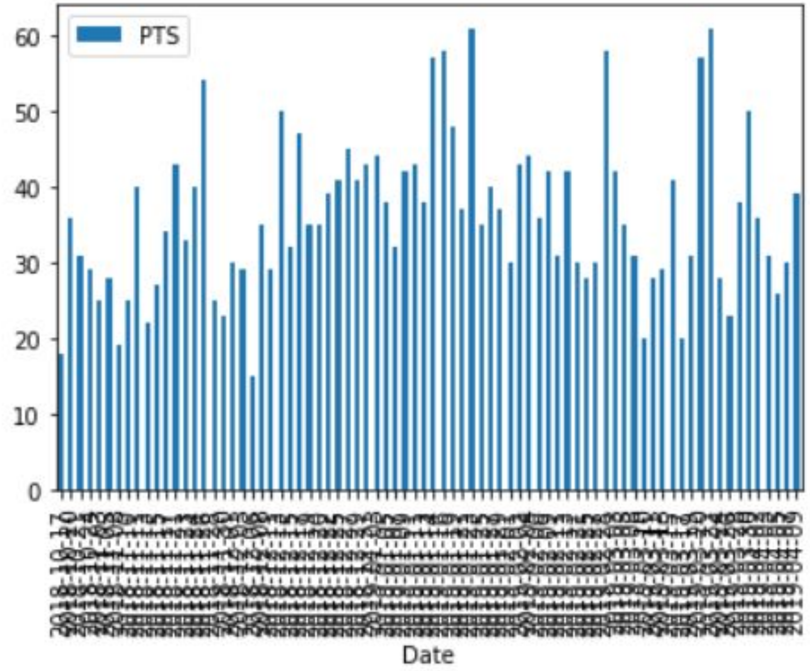
Plotting

- At the top of your jupyter notebook, we will need to import a new library that allows you to plot data
  - import matplotlib.pyplot as plt
- This is not included in the pandas library
- Why are data visualizations important?
- Good tutorial: http://queirozf.com/entries/pandas-dataframe-plot-examples-with-matplotlib-pyplot

Bar Graphs

- Provide an x axis, y axis, and the type of graph you want to see
  - In this case, the kind is “bar”
- What are bar graphs good for?
  - This example might not be the best idea for a bar graph
    - How can we make it better?

```python
# This plot shows how many points he scored in each game
# It looks super congested!
plt = harden.plot(x="Date", y="PTS", kind="bar")
```



Bar Graph Part 2, Creating a smaller range

- Looking at every date in the season was too much. How about we narrow down our range to be smaller?
- Let’s look at the first two weeks of December (arbitrary)
- Is there a way we can easily take our DataFrame to slice out this region?
  - Because the dates are stored as strings, it makes it difficult to compare using inequalities
  - How do we know if a date is in that range?
- We know the dates that we want are in the format: YYYY-MM-DD
  - We also know we want the dates of 2018-12-01 to 2018-12-14
  - How can we do this? For loop!

What is missing here?

```
# Let's look at a specific range of dates (first two weeks of decem
# Date format in the table is YYYY-MM-DD
dates = []
for day in range(14):
    date = "2018-12-" + str(day)
    dates.append(date)
print(dates)
```

```
['2018-12-0', '2018-12-1', '2018-12-2', '2018-12-3', '2018-12-4',
'2018-12-5', '2018-12-6', '2018-12-7', '2018-12-8', '2018-12-9',
'2018-12-10', '2018-12-11', '2018-12-12', '2018-12-13']
```

Fixed Code

```
# Lets fix the code in the previous cell
dates = []
for day in range(14):
    if day < 10:
       str_day = "0" + str(day + 1)
    else:
        str_day = str(day + 1)
    date = "2018-12-" + str_day
    dates.append(date)
print(dates)
```

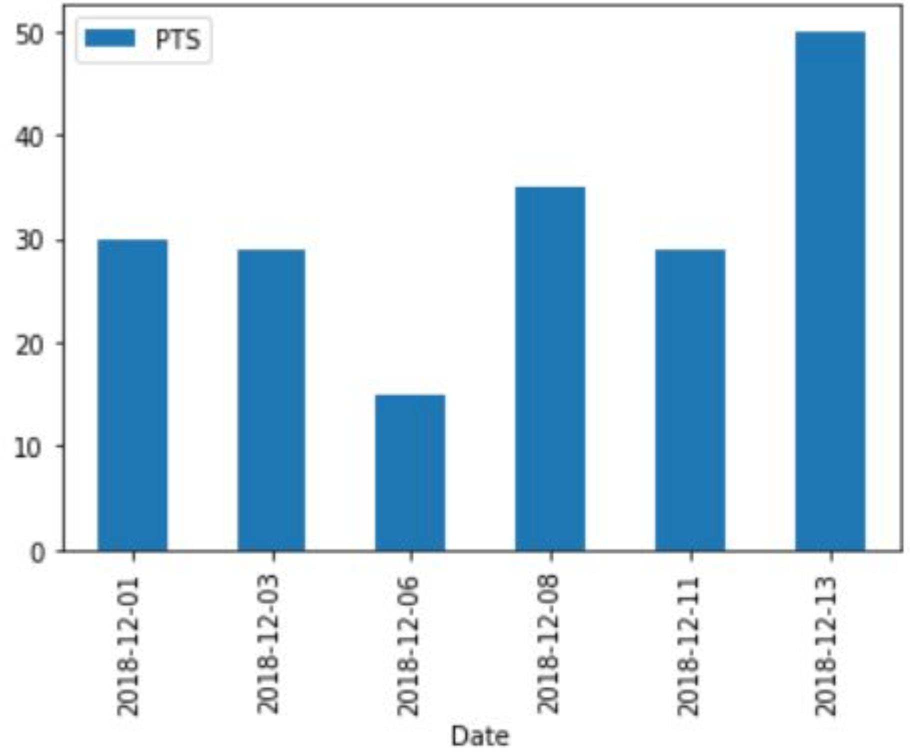
```
['2018-12-01', '2018-12-02', '2018-12-03', '2018-12-04', '2018-12-05', '2018-12-06', '2018-12-07', '2018-12-08', '2018-12-0
9', '2018-12-010', '2018-12-11', '2018-12-12', '2018-12-13', '2018-12-14']
```

Fixed Bar Graph

```
# Lets replot our histogram
small_harden = harden[harden["Date"].isin(dates)]
small_harden
```

	Name	Player_Game_Number	Date	Age	Team	Location	Opponent	Result	GS	TP
18	James Harden	19	2018-12-01	29-097	HOU	Home	CHI	W (+16)	1	30:22
19	James Harden	20	2018-12-03	29-099	HOU	Away	MIN	L (-12)	1	37:01
20	James Harden	21	2018-12-06	29-102	HOU	Away	UTA	L (-27)	1	27:58
21	James Harden	22	2018-12-08	29-104	HOU	Away	DAL	L (-3)	1	36:39
22	James Harden	23	2018-12-11	29-107	HOU	Home	POR	W (+7)	1	32:28
23	James Harden	24	2018-12-13	29-109	HOU	Home	LAL	W (+15)	1	35:28

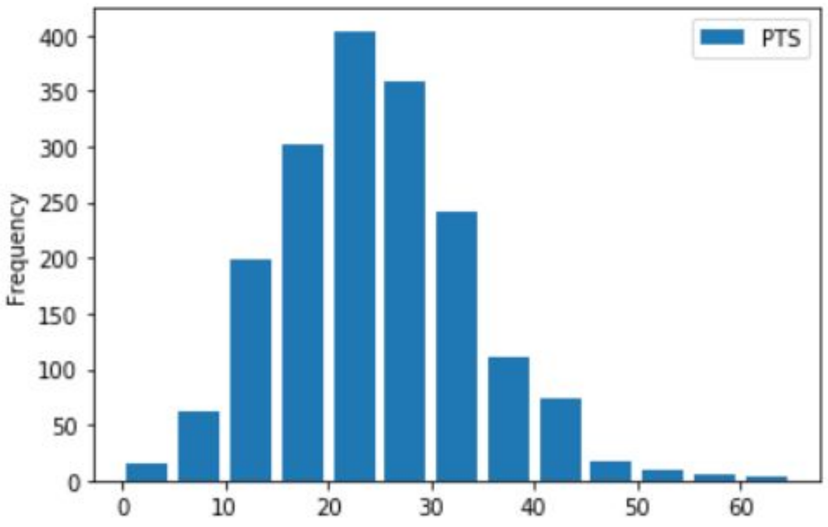
```
plt = small_harden.plot(x="Date", y="PTS", kind="bar")
```



Histograms

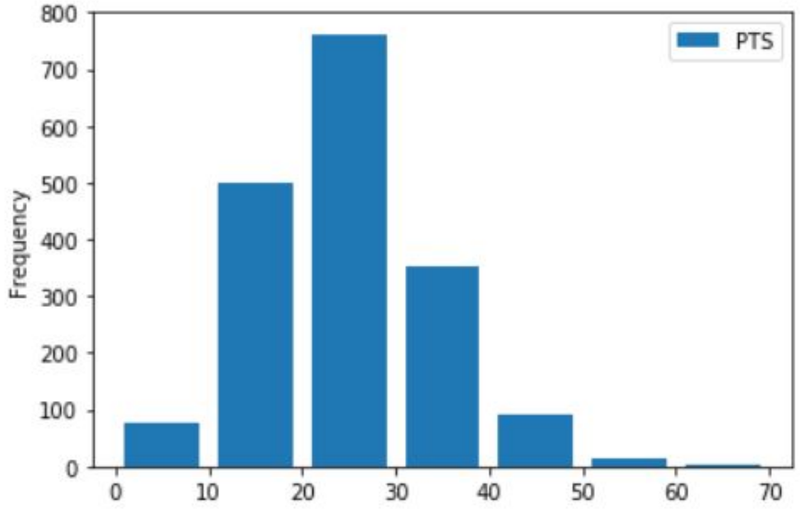
- Histograms look similar to bar graphs but show groupings of data instead of individual values
  - When creating a histogram, determine the column you want to analyze and the “bins” you want to use
- What are histograms good for?
  - Determining which values are the most common
  - Examining distributions

```
plt = allstars_df[["PTS"]].plot(
    kind="hist",
    bins = [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65],
    rwidth=0.8
)
```



Same Histogram, different bin size!

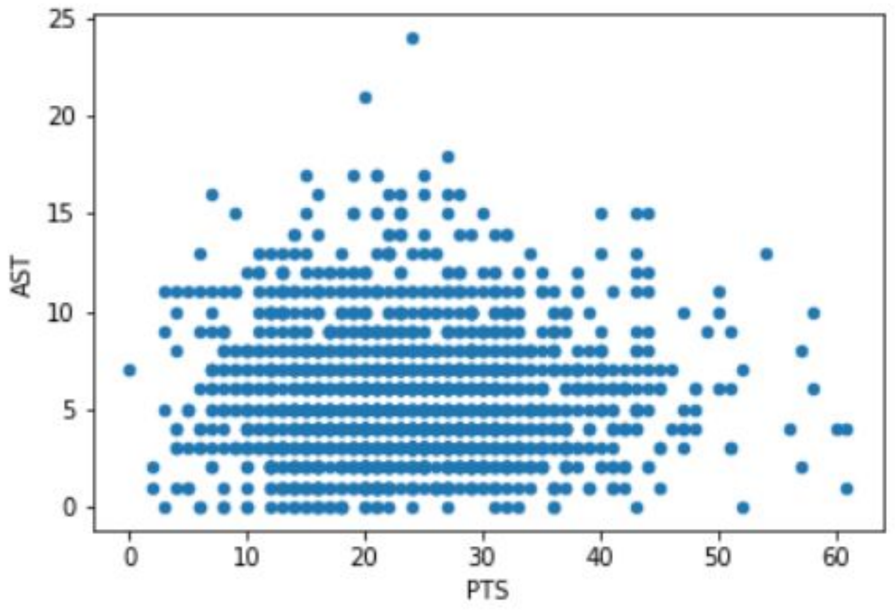
```python
# Histogram displying frequency of scoring grouped in bins of 10
plt = allstars_df[["PTS"]].plot(
    kind="hist",
    bins = [0, 10, 20, 30, 40, 50, 60, 70],
    rwidth=0.8
)
```



Scatter Plots

- Scatter plots attempt to display the relationship between two variables
  - Why is this useful?
- Provide two variables you want to see and plot!

```
# Scatter plot between Points and Assists
plt = allstars_df.plot(x="PTS", y = "AST", kind="scatter")
```



Further applications for you to explore

- Improving your visualizations
  - Adding titles, axis labels
  - Using different colors
  - Exploring different types of visualizations
- Linear regression
  - Too long for this tutorial
- Graphing multiple trends on the same plot
  - Use different colors to display multiple trends

Any Questions?

What do you want to see in the last module?

Options

- Webscraping live demo
- Linear regression tutorial
- More visualization help
- General live coding demo
- Any other options?