Basketball Salary Analytics:

An analysis of NBA & WNBA player salaries and their predictors

Allie Ridgway, John Hazelton, Jake Kolessar, and Julie Crowe

AGENDA

- → Project Management
- → Data Retrieval & Cleanup
- → Experimental Design
- → Exploratory Data Analysis
- → Model Building
- → Visualizations
- → Findings & Conclusions
- → Questions

Scrum Project Management

- ➔ Defined user stories to align with project goals
- ➔ Maintained a backlog of tasks to be completed
- ➔ Met regularly as a group to discuss progress and areas of concern
- ➔ Collaborated on tasks in order to efficiently and effectively meet deadlines

Web Scraping

```python
URL = 'https://www.spotrac.com/wnba/rankings/average/'
page = requests.get(URL).text
soup = BeautifulSoup(page, 'lxml')

names= []
for name in soup.find_all('a', class_= 'team-name'):
  names.append(name.get_text())

teams=[]
for team in soup.find_all('div', class_= 'rank-position'):
  teams.append(team.get_text())

avg_salaries=[]
for salaries in soup.find_all('span', class_= 'info'):
  avg_salaries.append(salaries.get_text())

pos= []
pos2= []
for p in soup.find_all('td', class_= 'center small'):
  pos.append(p.get_text())

for i in pos:
    j = i.replace(' ','').replace('\n', '')
    pos2.append(j)
pos2[136] = 'F'

age= []
for v in pos2:
    if len(v) % 2 == 0:
        age.append(v)

positions= []
for v in pos2:
    if len(v) % 2 != 0:
        positions.append(v)

wnba= 'WNBA'
```

```python
url = 'https://www.basketball-reference.com/leagues/NBA_2021_per_game.html'
html_doc = requests.get(url)

#parse the html from site:
parsed_html = BeautifulSoup(html_doc.content, 'html.parser')

#extract specific table we are interested in (per-game stats for each player):
table = parsed_html.find(id='per_game_stats')

##Header:
#Locate the table header, extract header values, and store in list:
table_header = table.find('thead') #html 'thead' element contains all header-related data
header_elements = table_header.find_all('th') #store all 'th' (header) elements from 'thead' element

headers = [] #initialize empty list to later store headers
for header in header_elements:
    item = header.get_text().strip() #extract each header value (text)
    headers.append(item) #append each header value to list
headers


##Body:
#Locate table body, extract data values, and store in list:
table_body = table.find('tbody') #html 'tbody' element contains all body-related data
body_rows = table_body.find_all('tr') #store all 'tr' (row) elements from 'tbody' element

rows = [] #initialize empty list to later store data rows
for row in body_rows:
    row_header = row.find('th').get_text() #extract the row's header (season)
    items = row.find_all('td') #extract data values from row
    row = [row_header] #initialize list  w/ row header
    for item in items: #iterate through the values of the row and store each in row list
        row.append(item.get_text())
    rows.append(row)
```

Our Data – NBA & WNBA Player Stats and Salaries

- Web Scraping & Pre-processing:
  - Scraped data separately for the leagues
    - Pulled from 2 sites (spotrac for salaries, basketballreference for stats)
  - Removed NA’s & duplicate entries (players that switched teams)
- Merging:
  - Removed accent marks from player names
  - Adjusted column names to match b/w datasets
- Data Types:
  - Numerical & string data:
    - Player, league, positon, points, asists, etc.

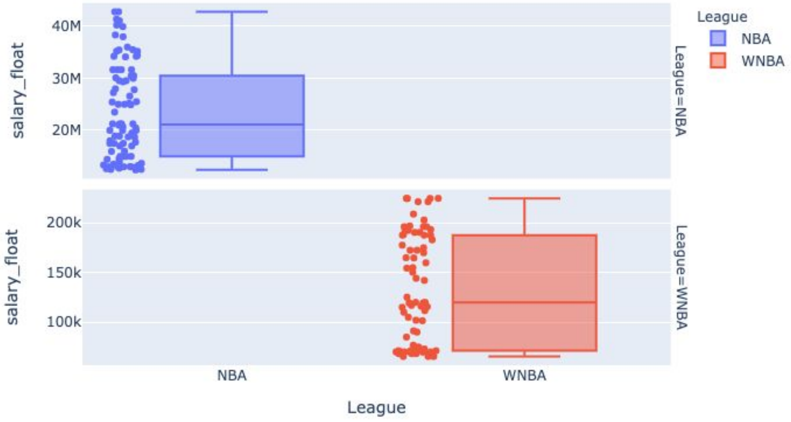
	Player	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	...	PTS	League	Team	Position	Age	Avg_Salary	salary_float	salary_ratio	GS%	Salary_Rank
0	James Harden	42	42	37.1	8.0	17.2	.463	2.8	7.8	.358	...	25.2	NBA	BKN	G	31	$42,782,880	42782880.0	0.00578	1.000000	1
1	John Wall	40	40	32.2	7.3	18.2	.404	2.0	6.2	.317	...	20.6	NBA	HOU	G	30	$42,782,880	42782880.0	0.00578	1.000000	2
2	Russell Westbrook	54	54	35.7	8.3	19.0	.437	1.3	4.1	.311	...	21.8	NBA	WAS	G	31	$41,358,814	41358814.0	0.00559	1.000000	3
3	Kevin Durant	25	22	32.5	9.4	17.3	.544	2.5	5.4	.467	...	27.5	NBA	BKN	F	32	$41,063,925	41063925.0	0.00555	0.880000	4
4	Stephen Curry	53	53	34.1	10.2	20.9	.489	5.2	12.1	.427	...	31.3	NBA	GSW	G	32	$40,231,758	40231758.0	0.00544	1.000000	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
158	Lexie Brown	17	13	374	2.2	6.5	0.342	0.8	3.1	0.269	...	6.4	WNBA	CHI	G	26	$70,040	70040.0	0.00117	0.764706	159
161	Seimone Augustus	21	0	332	2.6	5.2	0.491	0.6	1	0.545	...	5.9	WNBA	LA	G	36	$70,040	70040.0	0.00117	0.000000	162
162	Nia Coffey	15	1	230	1.1	2.5	0.421	0.5	1.4	0.333	...	2.7	WNBA	LA	F	25	$70,040	70040.0	0.00117	0.066667	163
163	Bria Holmes	18	4	291	1.9	5.4	0.357	0.6	1.7	0.333	...	4.9	WNBA	LA	G	27	$70,040	70040.0	0.00117	0.222222	164
164	Theresa Plaisance	13	0	90	0.8	2.2	0.379	0.4	1.3	0.294	...	2.5	WNBA	WAS	F	28	$70,040	70040.0	0.00117	0.000000	165

158 rows × 34 columns

Experimental Design

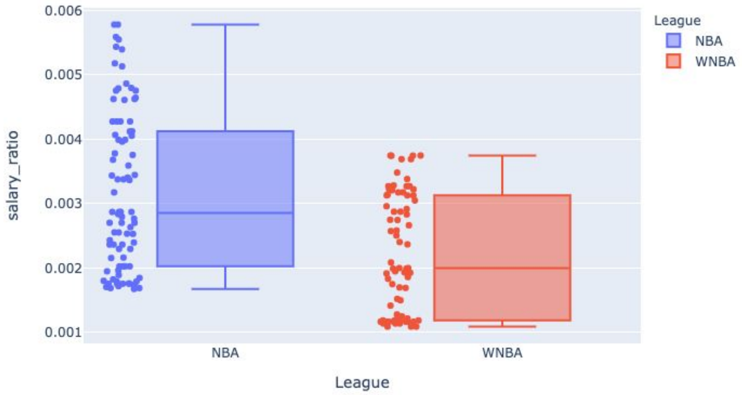
1) Data Retrieval
    - Included only 100 highest paid players from each league
        - Avoid skewness from rookie salaries
2) Driving Question/Thesis
    - Is there a legitimate difference in wages between the NBA & WNBA?
        - Contributors to salary valued differently b/w leagues?
        - Difference in salary trends w/ age b/w leagues?
        - Overvalued & undervalued players?
3) Exploratory Data Analysis
    - Scale salaries based on league revenue
    - Compare leagues on scatter plots
    - Look for statistical difference b/w leagues (wage gap)
4) Model Generation
    - Use multiple linear regression to predict salaries based on player stats (for each league)
    - Compare models
5) Visualization & Conclusions
    - Wage gap?
    - Stats valued differently between leagues?
    - Most undervalued, overvalued players?

Exploratory Data Analysis



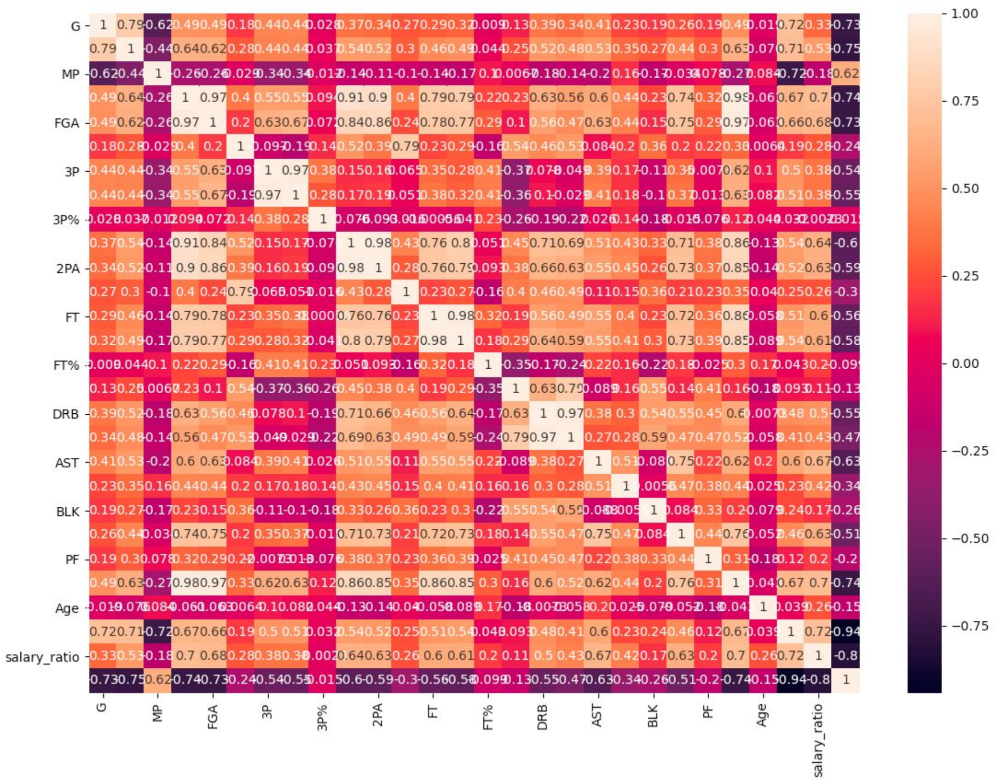
Views of Salaries vs League

Exploratory Data Analysis, cont.



Adjusted Views of Salaries Ratio vs League

Correlation Matrix



Model Building

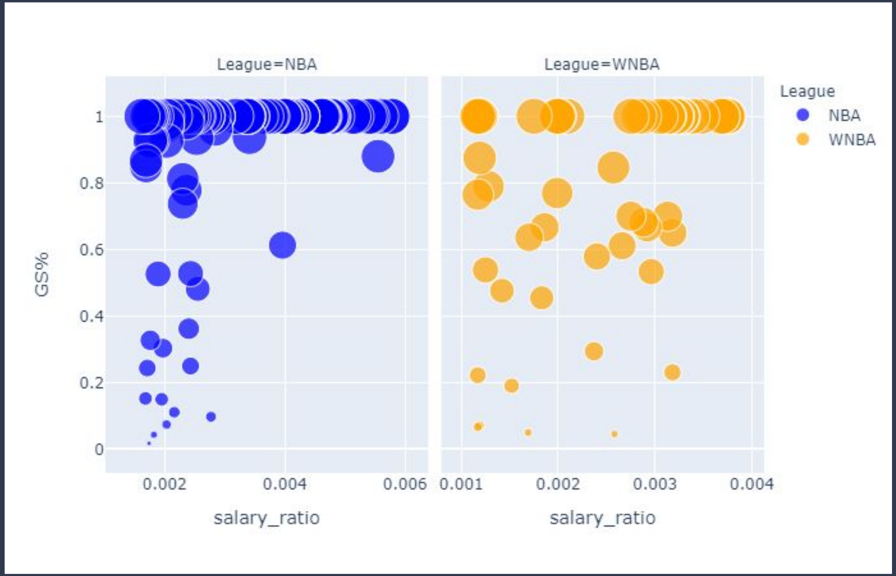
```python
class linearModel:
    def __init__(self, df):
        self.stats = df
        # clean up the dataframe
        columns_to_drop = ['League', 'Team', 'Position', 'Age', 'Avg_Salary', 'salary_ratio', 'Salary_Rank']
        self.stats = self.stats.dropna()
        self.stats = self.stats.drop(columns_to_drop, axis=1)
        # seperate response and predictors
        self.salary = self.stats.iloc[:, -1]
        self.stats = self.stats.iloc[:, 4:-1]
        # Split the sample data into train and test data
        self.X_train, self.X_test, self.Y_train, self.Y_test = train_test_split(self.stats,
                                                                              self.salary,
                                                                              test_size=0.2, random_state=4)
        # fit the linear model
        self.model = LinearRegression().fit(self.X_train, self.Y_train)

    def feature_selection(self, direction=None):
        sfs = SequentialFeatureSelector(self.model, n_features_to_select=3, direction=direction)
        self.sfs = sfs.fit(self.X_train, self.Y_train)
        # filter the columns by the selected features
        self.X_train_columns = self.X_train.columns[sfs.get_support()]
        self.X_test_columns = self.X_test.columns[sfs.get_support()]
        self.X_train = self.X_train[self.X_train_columns]
        self.X_test = self.X_test[self.X_test_columns]
        # fit the linear model
        self.model = LinearRegression().fit(self.X_train, self.Y_train)
```

- Our team utilized the sklearn module to build a statistical model with OOP
- It features powerful tool for predictive data analysis
- We split our data into test and training data
- Fit a Linear Regression model
- Performed Forward and Backward selection techniques
- Calculated the R-squared values
- Selected the 3 most significant predictors

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            salary_float   R-squared:                       0.526
Model:                             OLS   Adj. R-squared:                  0.506
Method:                  Least Squares   F-statistic:                     26.23
Date:                 Tue, 27 Apr 2021   Prob (F-statistic):           1.57e-11
Time:                         20:09:33   Log-Likelihood:                -1278.3
No. Observations:                   75   AIC:                             2565.
Df Residuals:                       71   BIC:                             2574.
Df Model:                            3
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       4.744e+06   2.37e+06      1.998      0.050    9696.710    9.48e+06
AST         1.862e+06   3.96e+05      4.701      0.000    1.07e+06    2.65e+06
BLK         2.973e+06   1.36e+06      2.186      0.032    2.62e+05    5.68e+06
PTS         5.289e+05    1.4e+05      3.791      0.000    2.51e+05    8.07e+05
==============================================================================
```

Games Started (%) vs. Salary Ratio



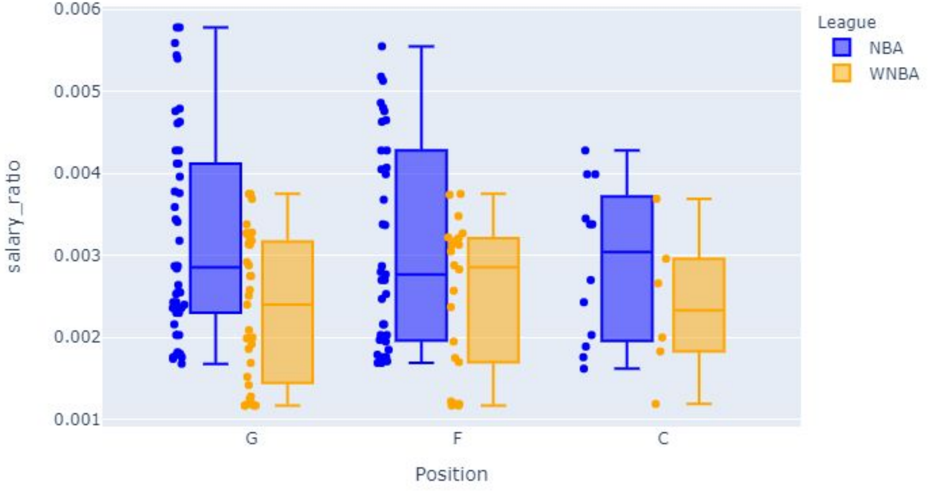
Code Sample:

```
fig = px.scatter(
        df, x='salary_ratio', y='GS%',
        color='League',
        color_discrete_sequence=["blue",
        "orange"], size='GS%',
        facet_col='League'
        )

fig.update_xaxes(matches=None)

fig.show()
```

Player Position vs. Salary Ratio



Use of the Findings

Who can use this data?

- This type of data analysis can be useful to teams deciding how much to pay players
- Players can look for new offers if they find they are undervalued
- Fans can get angry at management for overpaying players

Questions?