```
In [2]: import numpy
        import pandas as pd
        from matplotlib import pyplot
        from sklearn.linear_model import LinearRegression
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import r2_score
```

```
In [4]: stats = pd.read_csv('MLB Stats.csv')
        pitchers = pd.read_csv('2016Pitchers.csv')

        stats = stats[stats['Season'] == 2016]
        stats = stats.drop_duplicates(subset=['playerID'])
```

```
In [5]: filtered_stats = stats.filter(items=['playerID', 'Player Name', 'weigh
        t', 'height','AB','R','H','2B','3B','HR','RBI','SB','BB','salary','PA'
        ])
        BA = filtered_stats['H']/filtered_stats['AB']
        OBP = (filtered_stats['H']+filtered_stats['BB'] + stats["HBP"])/(filter
        ed_stats['AB']+filtered_stats['BB']+stats['HBP']+stats['SF'])
        singles = filtered_stats['H'] - ((filtered_stats['2B'])+(filtered_stats
        ['3B'])+(filtered_stats['HR']))
        SLG = ((singles)+(2*filtered_stats['2B'])+(3*filtered_stats['3B'])+(4*f
        iltered_stats['HR']))/filtered_stats['AB']
        OPS = OBP+SLG
        filtered_stats.insert(15,'BA', BA)
        filtered_stats.insert(16,'OBP', OBP)
        filtered_stats.insert(17,'SLG', SLG)
        filtered_stats.insert(18,'OPS', OPS)
```

```
In [6]: pitchers['Name'] = pitchers['Name'].astype('string')
```

```
In [7]: pitchers["Name"] = pitchers["Name"].map(lambda x: x.rsplit('*',1)[0])
```

```
In [8]: filtered_stats['Player Name'] = filtered_stats['Player Name'].astype('s
tring')
```

In [9]:
```
pitchernames = (pitchers["Name"])
index_names=[]
for name in pitchernames:
    index_names.append(name)


for i in range(550):
    testname = ( (filtered_stats.iloc[i])['Player Name'])
    if testname in index_names:
        name = filtered_stats.iloc[i]['Player Name']
        filtered_stats = filtered_stats[filtered_stats['Player Name'] !
= name]
```

In [10]:
```
filtered_stats = filtered_stats[filtered_stats['AB']>=50]
filtered_stats = filtered_stats[filtered_stats['salary']>=750000]
```

In [11]:
```
filtered_stats.head()
```

Out[11]:

	playerID	Player Name	weight	height	AB	R	H	2B	3B	HR	RBI	SB	BB	sala
81	abreujo02	Jose Abreu	255	75	624	67	183	32	1	25	100	0	47	1166666
117	ackledu01	Dustin Ackley	205	73	61	6	9	0	0	0	4	0	8	3200000
132	adamsma01	Matt Adams	245	75	297	37	74	18	0	16	54	0	25	1650000
412	alonsyo01	Yonder Alonso	230	73	482	52	122	34	0	7	56	3	45	2650000
436	altuvjo01	Jose Altuve	165	66	640	108	216	42	5	24	96	30	60	368750

Introduction

We set out to discover what baseball hitter statistics and attributes might have impact on resulting player salary, specifically for the 2016 MLB season, as this was the most recent season with complete, consistent data in a consolidated format.

This research topic is interesting to explore because it is often unclear what leads a player to financial success. Although the obvious answer seems to be that the better the player, the higher the salary, our goal is to dissect attributes and understand which of them most reliably results in an increase a player’s salary. With our research and analysis, our goal is to understand the criteria that leads to an increase in salary and on what aspects the players should focus on if their goal is to make more money. Thus, we elected to look at which specific, workable skills and attributes to identify which ones are expected to make a player more profitable in the MLB rather than analyze advanced analytic data that is often calculated based upon these standard statistics.

In the data that we used there is a possibility that players that are missing from the data. This would mean that our data could mean some players have performed well and made more money, but are not on the list, but could also mean the opposite. Because the amount of players that aren’t represented is low, this should not have real effects on answering our research question. It is also important to take into account that some of the players might not have played for a whole season due to injury or splitting time between the Major and Minor leagues, and that the data might not take that into account. This could affect the attributes influence on salary if players’s statistics decrease when they are not actually playing, but this should be negligible. These, however, are minor issues and the data (particularly after filtering) still should be comprised of reliable, workable statistics capable of yielding accurate results.

After our analysis of the data, we came to discover that four batting statistics stood out as relatively accurate predictors of MLB player salary. These statistics were hits, home runs, walks, and runs batted in (RBIs). While the correlations of these statistics were not particularly strong for linear regresssions, our evaluations of significance yielded results that allowed us to be quite confident in our results. This is because each of these statistics proved they would be extreme outliers from random pairs of salary and each respective statistic in terms of both correlation coefficient and slope, with not a single correlation being greater than the ones we found. As a result, we know that the correlations we discovered were more than a result of random chance.

With our joint model indicating that a positive correlatoin does exist between hits and home runs as well as hits and RBIs, we would reccommend that hitters focus on two things to increase their salary: their hitting power and their "batting eye," or ability to distinguish balls from strikes and swing at pitches accordingly. By doing this, we confidently believe MLB hitters stand a reasonable chance to earn a higher paying contract.

Data Description

- **What are the observations (rows) and the attributes (columns)?** The observations, or rows, are MLB players. Specifically, we are looking at MLB hitters in the 2016 season. The attributes, or columns, consist of several statistics and features of those players, including their salaries and attributes that may have an effect on their salaries (which we are setting out to discover). These include baseball statistics such as RBI as well as personal features such as height and weight. Here is a full list of the attributes we are looking at:
    - playerID
    - Player Name
    - Weight
    - Height
    - AB (at bats)
    - R (runs)
    - H (hits)
    - 2B (doubles)
    - 3B (triples)
    - HR (home runs)
    - RBI (runs batted in)
    - SB (stolen bases)
    - BB (base on balls, a.k.a. walks)
    - Salary
    - PA (plate appearance)
    - BA (batting average)

    - OBP (on base percentage)
    - SLG (slugging percentage)
    - OPS (on-base plus slugging)

- **Why was this dataset created?** The original dataset was created to keep track of player information over the years, for all teams and all positions. This includes not just statistics, but other player features typically looked at in the sports world as well, such as physical attributes. Player data and statistics, in baseball as well as other sports, is typically kept track of to compare players over the years, as well as to each other, to understand the game and how it is evolving. This data is used for analytics throughout the season. We filtered our dataset to contain only hitters in the 2016 season, because this season is one of the most recent with available, plentiful data, and attributes of pitchers that quantify them as "good" and may be linked to a higher salary are very different than that of hitters. We also added some statistics by calculation with other columns. This filtered dataset was created for a more comprehensible analysis of player attributes and salary.

- **Who funded the creation of the dataset?** The datasets used in the creation of our final dataset for analysis were funded by the MLB, which tracks and publishes player data and distributes it to a variety of sources, including posting it on their website. Player and team statistics are public knowledge that are tracked and published throughout each season. Although some of this data was retrieved from sources with no relation to the MLB, the creation of this data was funded by the MLB itself.

- **What processes might have influenced what data was observed and recorded and what was not?** There are many processes that could have influenced what data was seen or was not. For example, due to injuries happening during the season many players have to stop playing for a period of time to allow themselves to recover - this in turn goes as unseen data as their performance cannot be evaluated as a full season to compare with other players. Similarly, there are drug tests that take place during the season such as steroid testing and players who do not pass those tests are suspended from the season which prevents their data from being seen. Short absences like these will impact a player's cumulative statistics for a season. Additionally, cases of ejections can also affect a player’s statistics, as some games may appear with partial or no statistics. However, significant absences from injury, suspension, or ejection do not comprise the majority of the data, meaning that most of what is observed is actually from players who manage to play for a significant majority of the season.

- **What preprocessing was done, and how did the data come to be in the form that you are using?** Baseball-reference.com and similar websites hold a lot of data on player, team and manager statistics. To obtain this data, they have references where they contact the official MLB website for their database and can then process it to obtain the finalized data they need. Most of the data actually is also available for public use as MLB games are watched and recorded by fans, reporters and statisticians alike, hence statistics on these sites will agree with the data the MLB website puts forward. Of course, to preprocess the data from the websites we obtained our info from, we picked out only the relevant records. Since we have 2 csv files we will be using, we will also process those 2 files into 1 single dataframe for easy usage using the pandas library. We did this by limiting the dataset to observations where the Season column is 2016, and then filtering out unwanted columns. We also iterated through the list of player names and checked if they were in the pitchers dataset. If they were, we removed those as well, leaving us with only MLB hitters in the 2016 season. We also removed observations with AB (at bats) less than 50; these are players who batted or players limited times due to injury, playing most of season in the minor league, or other extenuating circumstances. This will remove data that may skew our results. We chose the number 50 based on web searches that pointed to this being a reasonable number.

- **If people are involved, were they aware of the data collection and if so, what purpose did they expect the data to be used for?** All players involved in data collection are well aware that their playing statistics and salaries are not only being collected, but are also generally public knowledge. In fact, one can assume that they expected this data to be used for purposes like these, analyses of their playing ability and monetary value as employees of a business.

- **Where can your raw source data be found, if applicable? Provide a link to the raw data (hosted in a Cornell Google Drive or Cornell Box).** Our raw data was obtained from two sources, baseball-reference.com (a privately sourced data compiling site that gets data directly from MLB) and data.world (an open source data posting platform; data was posted by a user who got statistics from MLB). The datasets can be found at https://drive.google.com/drive/folders/17tmbkDN8lQ_Zw-Yflys3fJclquH2EWyt?usp=sharing

Data Analysis

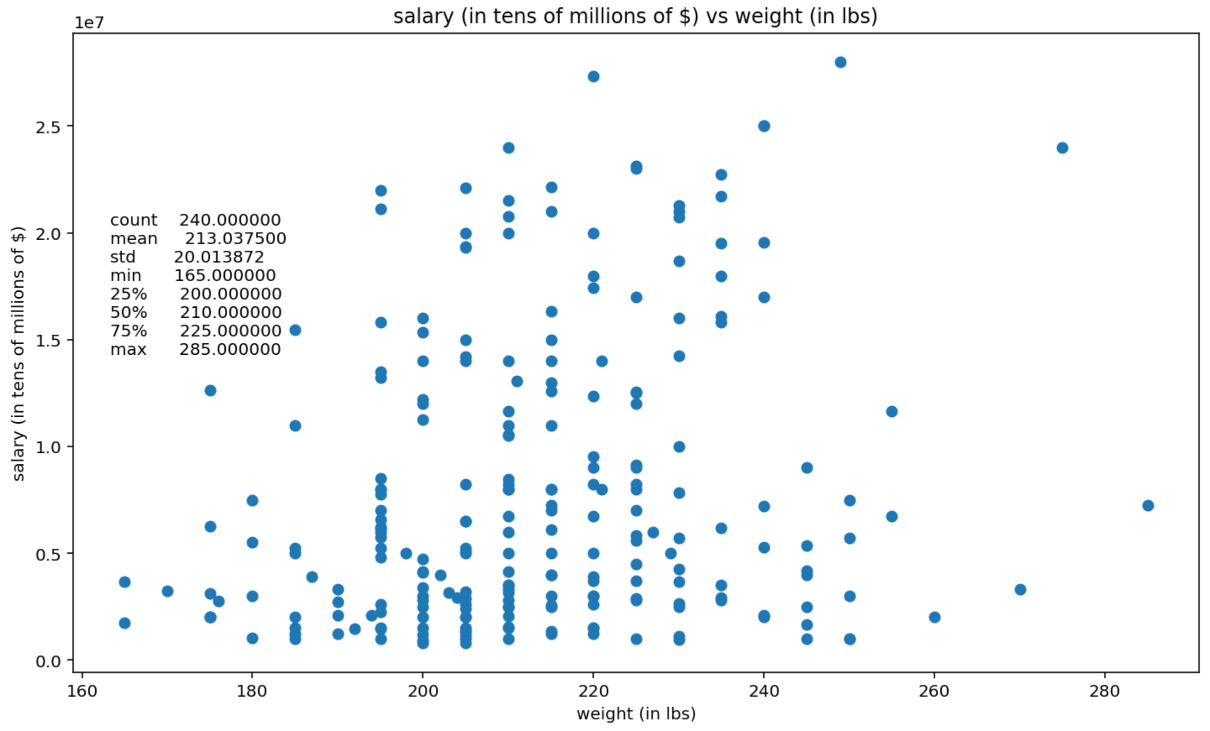
In [69]:
```
#Kartikay
```

In [70]:
```
pyplot.scatter(x=filtered_stats['weight'],y=filtered_stats['salary'])

pearson_corr = filtered_stats[['weight','salary']].corr(method='pearso
n')
spearman_corr = filtered_stats[['weight','salary']].corr(method='spearm
an')
kendall_corr = filtered_stats[['weight','salary']].corr(method='kendal
l')

print("pearson correlation coefficient of weight and salary is ",pearso
n_corr.iloc[:,0][1])
print("spearman correlation coefficient of weight and salary is ",spear
man_corr.iloc[:,0][1])
print("kendall correlation coefficient of weight and salary is ",kendal
l_corr.iloc[:,0][1] )


pyplot.figtext(0.15,0.5, filtered_stats['weight'].describe().to_string
())
pyplot.xlabel("weight (in lbs)")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs weight (in lbs)")
pyplot.show()
```

```
pearson correlation coefficient of weight and salary is  0.214709337545
74414
spearman correlation coefficient of weight and salary is  0.21819876476
937777
```

```
kendall correlation coefficient of weight and salary is  0.152790293649
63823
```

Out[70]:



```
count    240.000000
mean     213.037500
std       20.013872
min      165.000000
25%      200.000000
50%      210.000000
75%      225.000000
max      285.000000
```

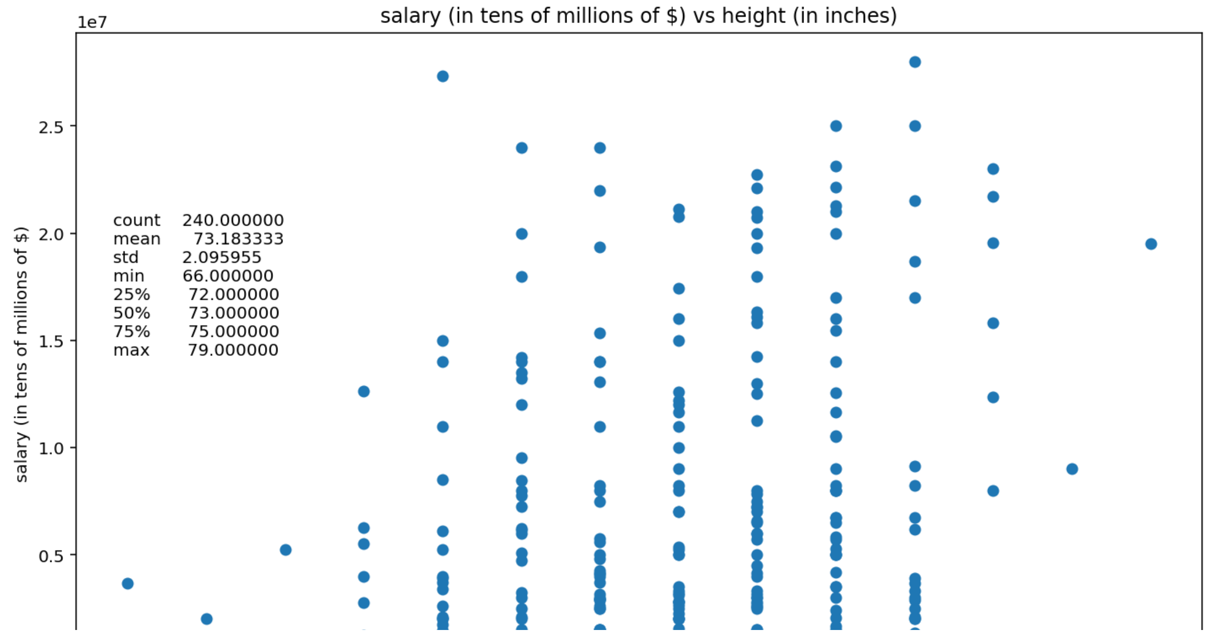
In [71]:

```
pyplot.scatter(x=filtered_stats['height'],y=filtered_stats['salary'])

pearson_corr = filtered_stats[['height','salary']].corr(method='pearso
n')
spearman_corr = filtered_stats[['height','salary']].corr(method='spearm
an')
kendall_corr = filtered_stats[['height','salary']].corr(method='kendal
l')
```

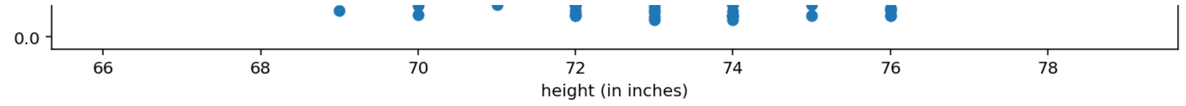
```
print("pearson correlation coefficient of height and salary is ",pearso
n_corr.iloc[:,0][1])
print("spearman correlation coefficient of height and salary is ",spear
man_corr.iloc[:,0][1])
print("kendall correlation coefficient of height and salary is ",kendal
l_corr.iloc[:,0][1] )


pyplot.figtext(0.15,0.5, filtered_stats['height'].describe().to_string
())
pyplot.xlabel("height (in inches)")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs height (in inches)")
pyplot.show()
```

```
pearson correlation coefficient of height and salary is  0.184177789788
91414
spearman correlation coefficient of height and salary is  0.13643952162
339595
kendall correlation coefficient of height and salary is  0.097198592807
60197
```

Out[71]:



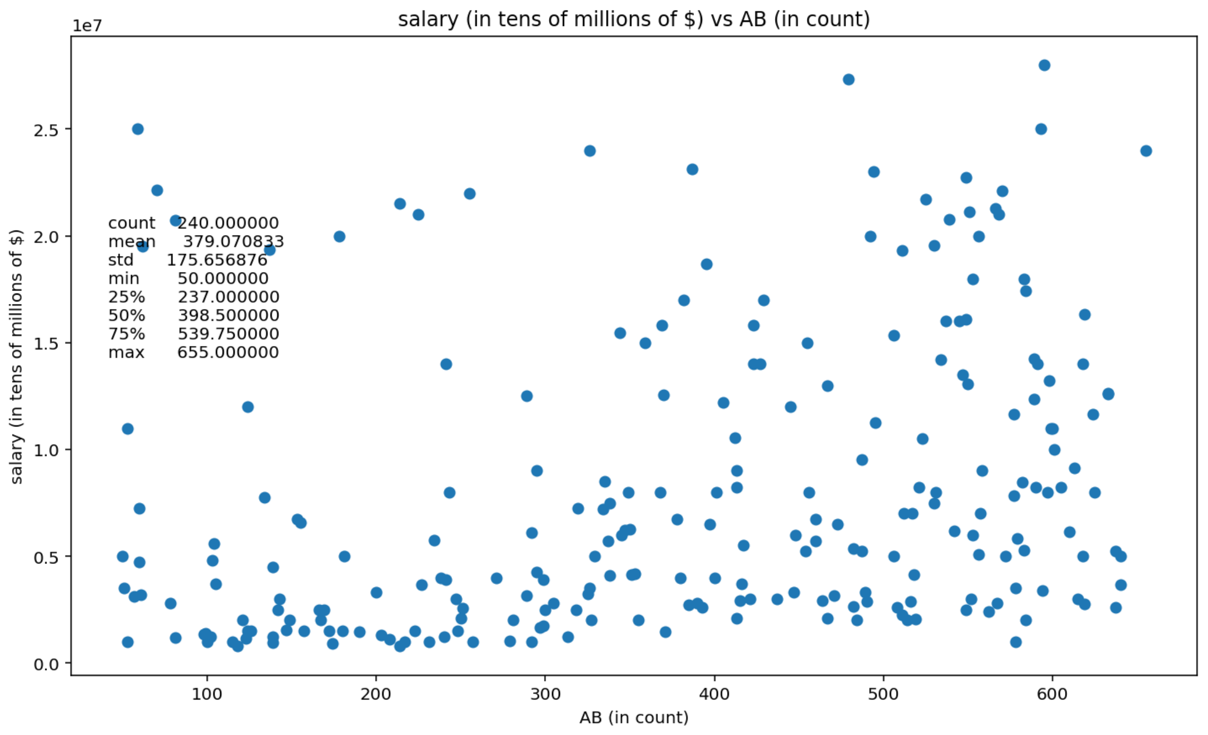
```
count    240.000000
mean      73.183333
std        2.095955
min       66.000000
25%       72.000000
50%       73.000000
75%       75.000000
max       79.000000
```



In [72]:
```
pyplot.scatter(x=filtered_stats['AB'],y=filtered_stats['salary'])

pearson_corr = filtered_stats[['AB','salary']].corr(method='pearson')
spearman_corr = filtered_stats[['AB','salary']].corr(method='spearman')
kendall_corr = filtered_stats[['AB','salary']].corr(method='kendall')

print("pearson correlation coefficient of AB and salary is ",pearson_co
rr.iloc[:,0][1])
print("spearman correlation coefficient of AB and salary is ",spearman_
corr.iloc[:,0][1])
print("kendall correlation coefficient of AB and salary is ",kendall_co
rr.iloc[:,0][1] )

pyplot.figtext(0.15,0.5, filtered_stats['AB'].describe().to_string())
pyplot.xlabel("AB (in count)")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs AB (in count)")
pyplot.show()
```

```
pearson correlation coefficient of AB and salary is  0.2985232817638316
spearman correlation coefficient of AB and salary is  0.397229732337768
7
kendall correlation coefficient of AB and salary is  0.2771983095115121
4
```

Out[72]:

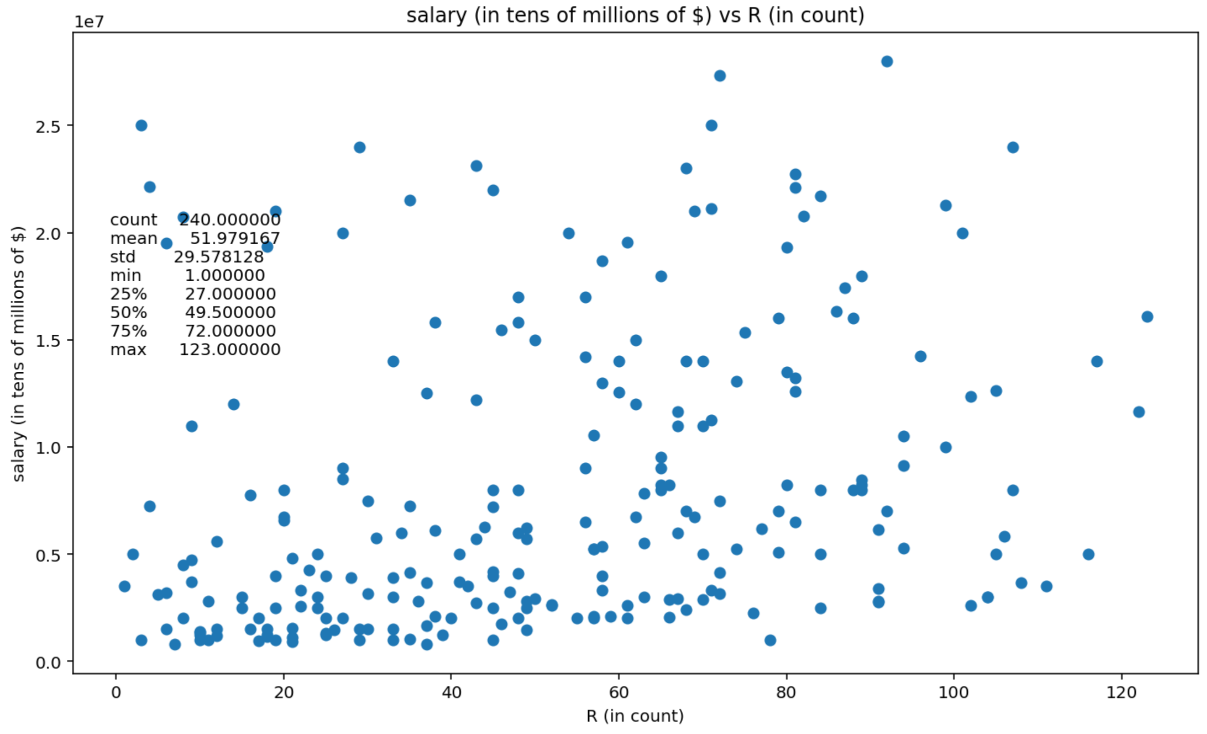


```
count    240.000000
mean     379.070833
std      175.656876
min       50.000000
25%      237.000000
50%      398.500000
75%      539.750000
max      655.000000
```

In [73]:
```
pyplot.scatter(x=filtered_stats['R'],y=filtered_stats['salary'])

pearson_corr = filtered_stats[['R','salary']].corr(method='pearson')
spearman_corr = filtered_stats[['R','salary']].corr(method='spearman')
kendall_corr = filtered_stats[['R','salary']].corr(method='kendall')

print("pearson correlation coefficient of R and salary is ",pearson_cor
r.iloc[:,0][1])
print("spearman correlation coefficient of R and salary is ",spearman_c
orr.iloc[:,0][1])
print("kendall correlation coefficient of R and salary is ",kendall_cor
r.iloc[:,0][1] )


pyplot.figtext(0.15,0.5, filtered_stats['R'].describe().to_string())
pyplot.xlabel("R (in count)")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs R (in count)")
pyplot.show()
```

```
pearson correlation coefficient of R and salary is  0.296222080505239
spearman correlation coefficient of R and salary is  0.3947394754694804
kendall correlation coefficient of R and salary is  0.2750919847266515
```

Out[73]:



```
count    240.000000
mean      51.979167
std       29.578128
min        1.000000
25%       27.000000
50%       49.500000
75%       72.000000
max      123.000000
```

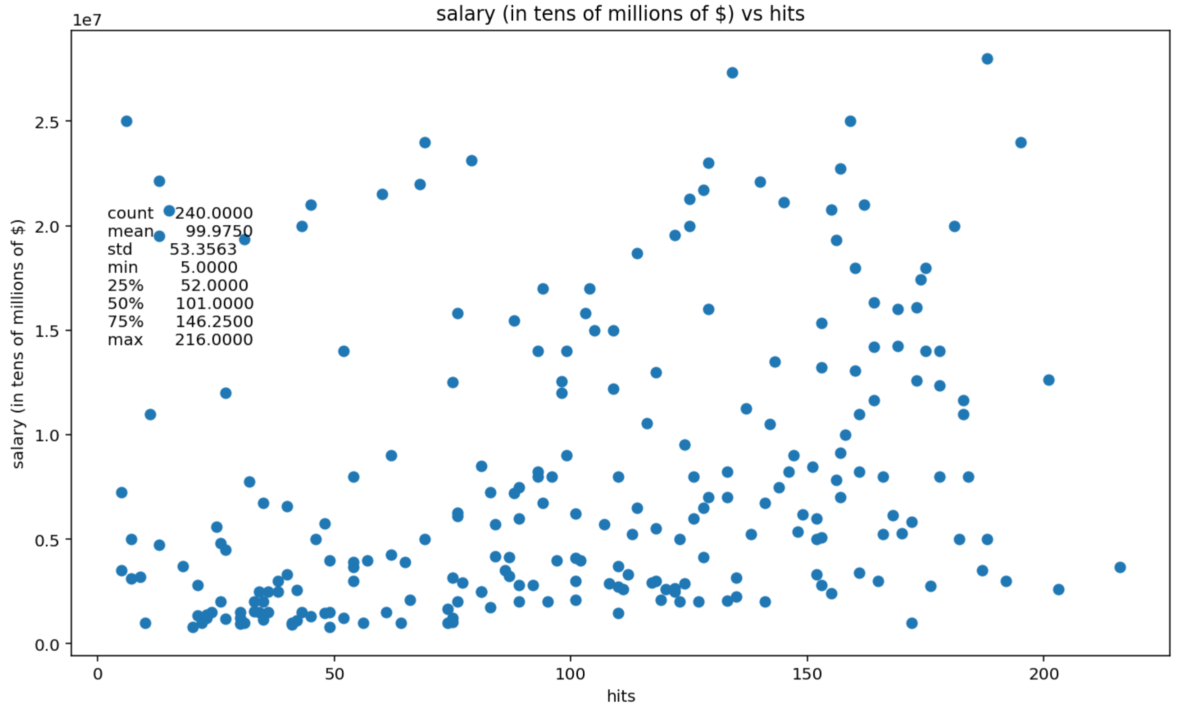
In [74]:

```
# Jacob
pyplot.scatter(x=filtered_stats['H'],y=filtered_stats['salary'])

pearson_corr = filtered_stats[['H','salary']].corr(method='pearson').il
```

```
oc[:,0][1]
spearman_corr = filtered_stats[['H','salary']].corr(method='spearman').
iloc[:,0][1]
kendall_corr = filtered_stats[['H','salary']].corr(method='kendall').il
oc[:,0][1]

print("Pearson correlation coefficient of hits and salary is", pearson_
corr)
print("Spearman correlation coefficient of hits and salary is", spearma
n_corr)
print("Kendall correlation coefficient of hits and salary is", kendall_
corr)

pyplot.figtext(0.15,0.5, filtered_stats['H'].describe().to_string())
pyplot.xlabel("hits")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs hits")
pyplot.show()
```

```
Pearson correlation coefficient of hits and salary is 0.2942717851361
791
Spearman correlation coefficient of hits and salary is 0.393735472239
49845
Kendall correlation coefficient of hits and salary is 0.2765460521274
193
```

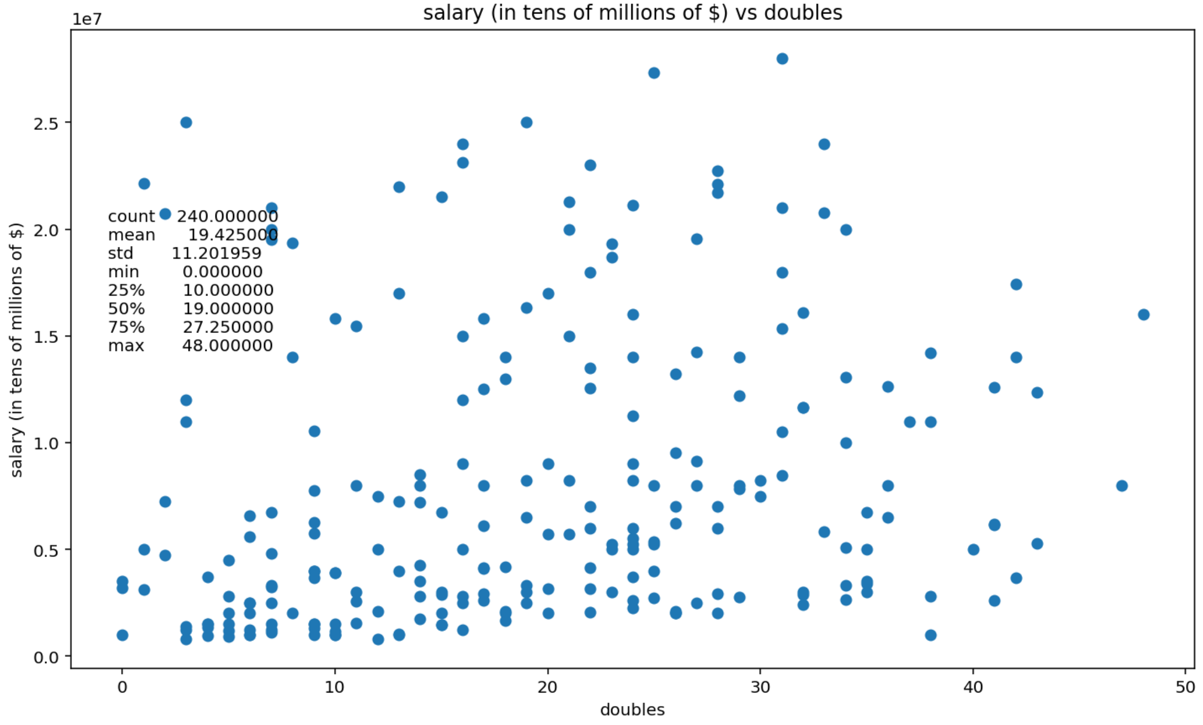
Out[74]:



In [75]:
```
pyplot.scatter(x=filtered_stats['2B'],y=filtered_stats['salary'])

pearson_corr = filtered_stats[['2B','salary']].corr(method='pearson').i
loc[:,0][1]
spearman_corr = filtered_stats[['2B','salary']].corr(method='spearman')
.iloc[:,0][1]
kendall_corr = filtered_stats[['2B','salary']].corr(method='kendall').i
loc[:,0][1]

print("Pearson correlation coefficient of doubles and salary is", pears
on_corr)
print("Spearman correlation coefficient of doubles and salary is", spea
rman_corr)
print("Kendall correlation coefficient of doubles and salary is", kenda
ll_corr)

pyplot.figtext(0.15,0.5, filtered_stats['2B'].describe().to_string())
pyplot.xlabel("doubles")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs doubles")
pyplot.show()
```

```
Pearson correlation coefficient of doubles and salary is 0.228125343551
4241
Spearman correlation coefficient of doubles and salary is 0.33828911514
073
Kendall correlation coefficient of doubles and salary is 0.235787429323
63975
```

Out[75]:



```
count    240.000000
mean      19.425000
std       11.201959
min        0.000000
25%       10.000000
50%       19.000000
75%       27.250000
max       48.000000
```

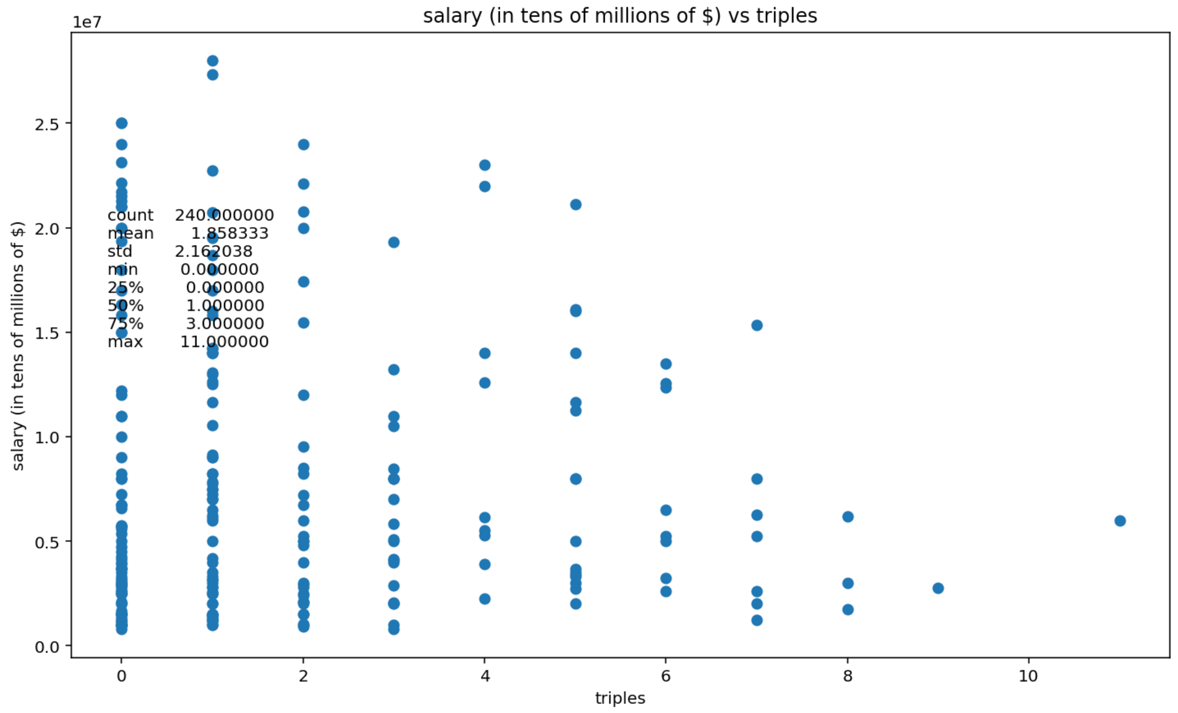
In [76]:

```
pyplot.scatter(x=filtered_stats['3B'],y=filtered_stats['salary'])
```

```
pearson_corr = filtered_stats[['3B','salary']].corr(method='pearson').i
loc[:,0][1]
spearman_corr = filtered_stats[['3B','salary']].corr(method='spearman')
.iloc[:,0][1]
kendall_corr = filtered_stats[['3B','salary']].corr(method='kendall').i
loc[:,0][1]

print("Pearson correlation coefficient of triples and salary is", pears
on_corr)
print("Spearman correlation coefficient of triples and salary is", spea
rman_corr)
print("Kendall correlation coefficient of triples and salary is", kenda
ll_corr)

pyplot.figtext(0.15,0.5, filtered_stats['3B'].describe().to_string())
pyplot.xlabel("triples")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs triples")
pyplot.show()
```

```
Pearson correlation coefficient of triples and salary is -0.069410073
4390679
Spearman correlation coefficient of triples and salary is 0.014180462
748458224
Kendall correlation coefficient of triples and salary is 0.0074759951
12260108
```

Out[76]:

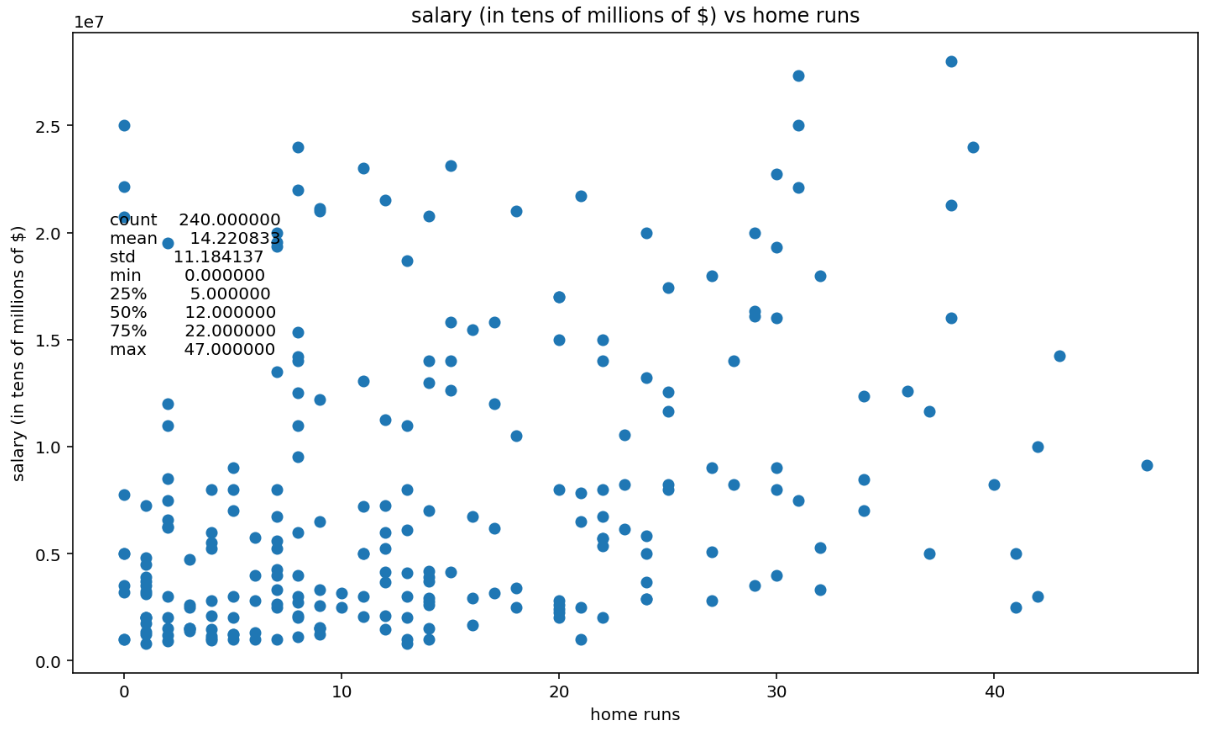


```
count    240.000000
mean       1.858333
std        2.162038
min        0.000000
25%        0.000000
50%        1.000000
75%        3.000000
max       11.000000
```

In [77]:
```
pyplot.scatter(x=filtered_stats['HR'],y=filtered_stats['salary'])

pearson_corr = filtered_stats[['HR','salary']].corr(method='pearson').i
loc[:,0][1]
spearman_corr = filtered_stats[['HR','salary']].corr(method='spearman')
.iloc[:,0][1]
kendall_corr = filtered_stats[['HR','salary']].corr(method='kendall').i
loc[:,0][1]

print("Pearson correlation coefficient of home runs and salary is", pea
rson_corr)
print("Spearman correlation coefficient of home runs and salary is", sp
earman_corr)
print("Kendall correlation coefficient of home runs and salary is", ken
dall_corr)

pyplot.figtext(0.15,0.5, filtered_stats['HR'].describe().to_string())
pyplot.xlabel("home runs")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs home runs")
pyplot.show()
```

```
Pearson correlation coefficient of home runs and salary is 0.3275551220
1437054
Spearman correlation coefficient of home runs and salary is 0.375123053
79838934
Kendall correlation coefficient of home runs and salary is 0.2600739623
874983
```

Out[77]:

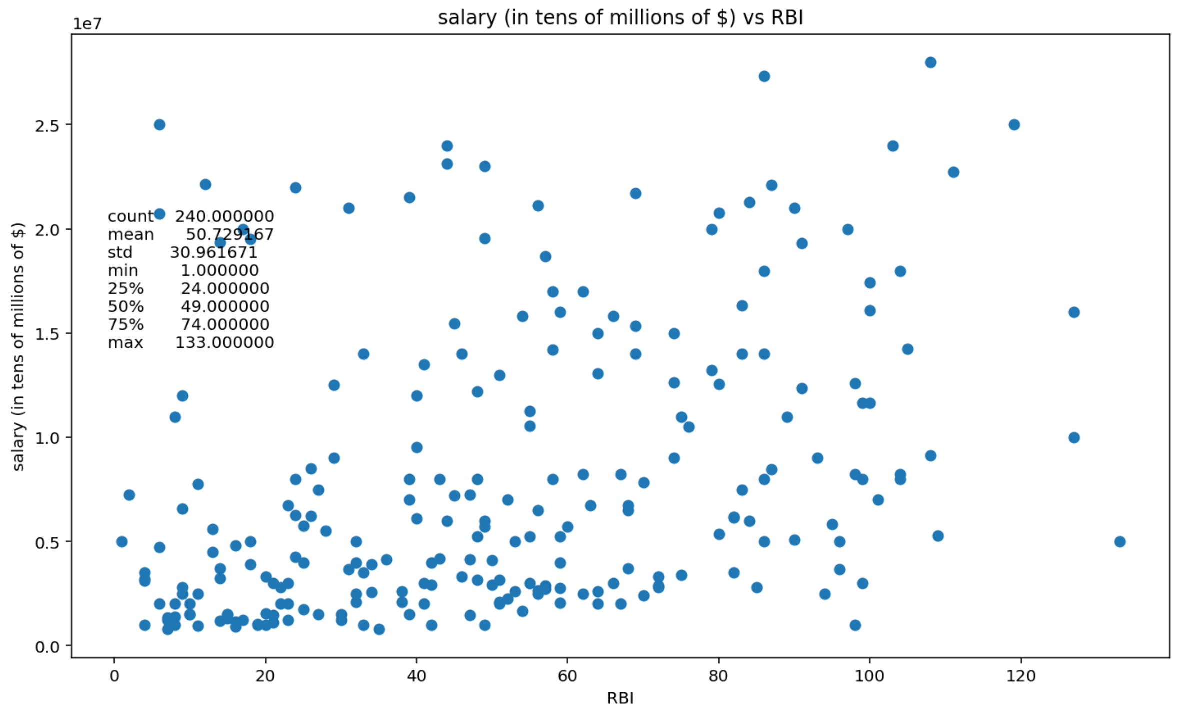


In [78]:

```
pearson_corr = filtered_stats[['RBI','salary']].corr(method='pearson')
spearman_corr = filtered_stats[['RBI','salary']].corr(method='spearman'
)
kendall_corr = filtered_stats[['RBI','salary']].corr(method='kendall')

print("pearson correlation coefficient of RBIs and salary is ",pearson_
corr.iloc[:,0][1])
print("spearman correlation coefficient of RBIs and salary is ",spearma
n_corr.iloc[:,0][1])
print("kendall correlation coefficient of RBIs and salary is ",kendall_
corr.iloc[:,0][1] )

pyplot.scatter(x=filtered_stats['RBI'],y=filtered_stats['salary'])
pyplot.figtext(0.15,0.5, filtered_stats['RBI'].describe().to_string())
pyplot.xlabel("RBI")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs RBI")
pyplot.show()
```

```
pearson correlation coefficient of RBIs and salary is  0.367325796837
4213
spearman correlation coefficient of RBIs and salary is  0.43437303798
82009
kendall correlation coefficient of RBIs and salary is  0.303639528086
41237
```

Out[78]:

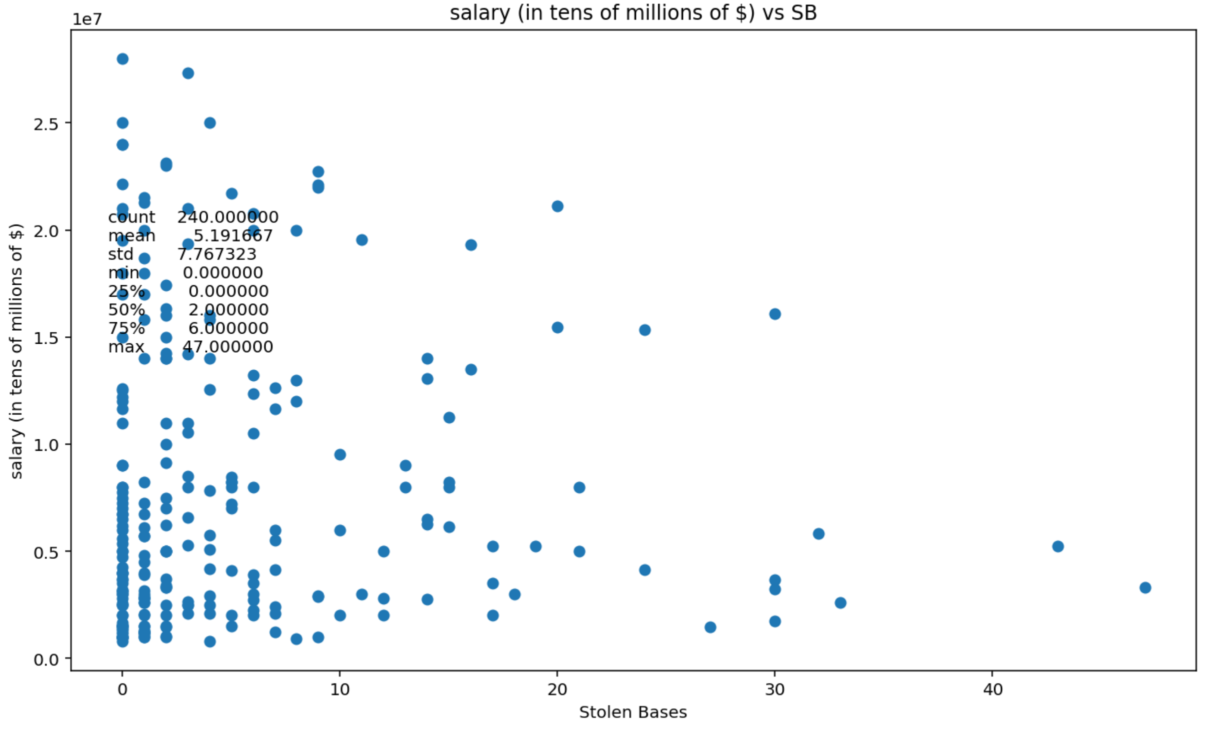


```
count    240.000000
mean      50.729167
std       30.961671
min        1.000000
25%       24.000000
50%       49.000000
75%       74.000000
max      133.000000
```

In [79]:
```
pearson_corr = filtered_stats[['SB','salary']].corr(method='pearson')
spearman_corr = filtered_stats[['SB','salary']].corr(method='spearman')
kendall_corr = filtered_stats[['SB','salary']].corr(method='kendall')

print("pearson correlation coefficient of stolen bases and salary is ",
pearson_corr.iloc[:,0][1])
print("spearman correlation coefficient of stolen bases and salary is "
,spearman_corr.iloc[:,0][1])
print("kendall correlation coefficient of stolen bases and salary is ",
kendall_corr.iloc[:,0][1] )

pyplot.scatter(x=filtered_stats['SB'],y=filtered_stats['salary'])
pyplot.figtext(0.15,0.5, filtered_stats['SB'].describe().to_string())
pyplot.xlabel("Stolen Bases")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs SB")
pyplot.show()
```

```
pearson correlation coefficient of stolen bases and salary is  -0.01865
4414631359834
spearman correlation coefficient of stolen bases and salary is  0.10265
131270934304
kendall correlation coefficient of stolen bases and salary is  0.070261
35732279039
```

Out[79]:



```
count    240.000000
mean       5.191667
std        7.767323
min        0.000000
25%        0.000000
50%        2.000000
75%        6.000000
max       47.000000
```

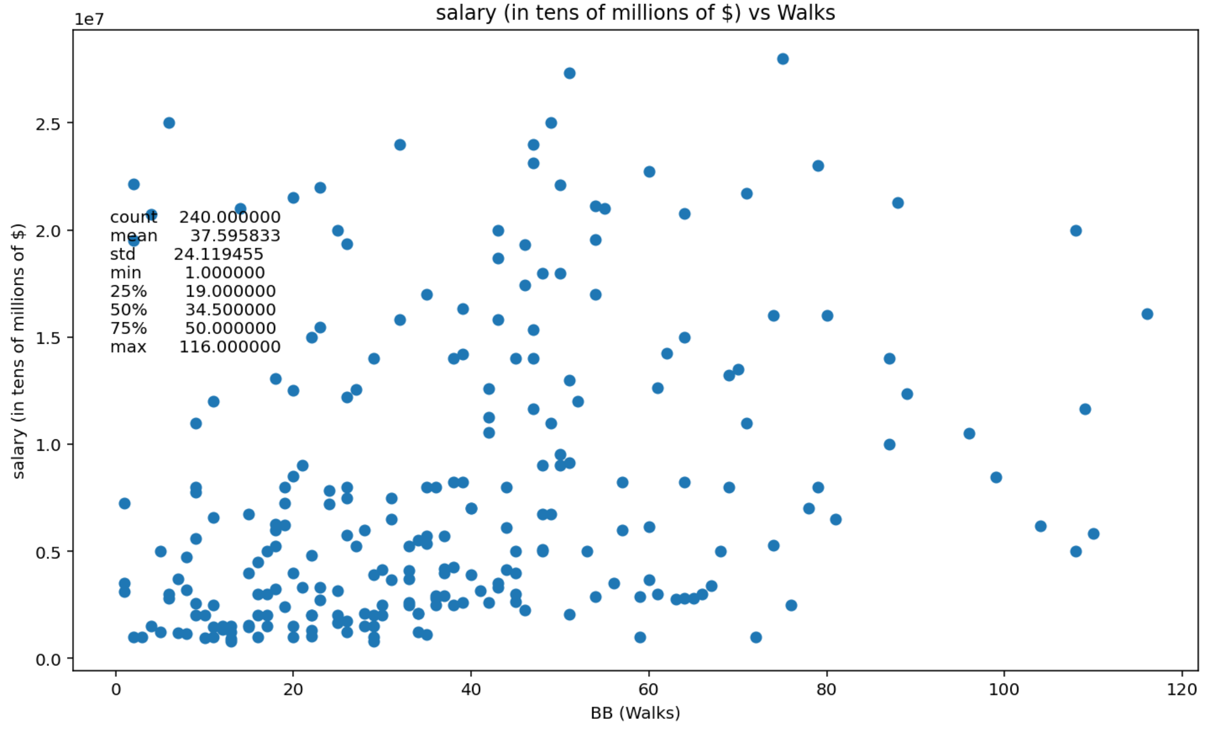
In [80]:

```
pearson_corr = filtered_stats[['BB','salary']].corr(method='pearson')
spearman_corr = filtered_stats[['BB','salary']].corr(method='spearman')
kendall_corr = filtered_stats[['BB','salary']].corr(method='kendall')

print("pearson correlation coefficient of walks and salary is ",pearson
_corr.iloc[:,0][1])
```

```
print("spearman correlation coefficient of walks and salary is ",spearm
an_corr.iloc[:,0][1])
print("kendall correlation coefficient of walks and salary is ",kendall
_corr.iloc[:,0][1] )

pyplot.scatter(x=filtered_stats['BB'],y=filtered_stats['salary'])
pyplot.figtext(0.15,0.5, filtered_stats['BB'].describe().to_string())
pyplot.xlabel("BB (Walks)")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs Walks")
pyplot.show()
```

```
pearson correlation coefficient of walks and salary is  0.3053818459139
6983
spearman correlation coefficient of walks and salary is  0.390066158793
10563
kendall correlation coefficient of walks and salary is  0.2771102894023
0503
```

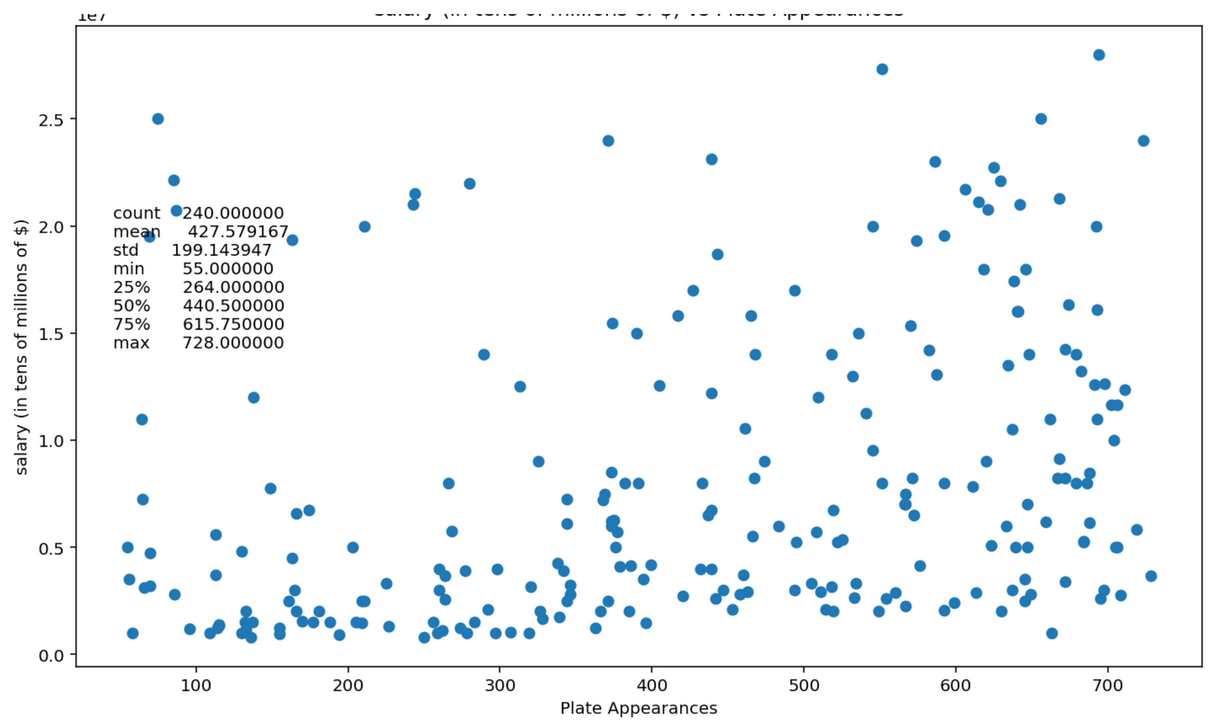
Out[80]:



```
In [81]: pearson_corr = filtered_stats[['PA','salary']].corr(method='pearson')
         spearman_corr = filtered_stats[['PA','salary']].corr(method='spearman')
         kendall_corr = filtered_stats[['PA','salary']].corr(method='kendall')

         print("pearson correlation coefficient of Plate Appearances and salary
          is ",pearson_corr.iloc[:,0][1])
         print("spearman correlation coefficient of Plate Appearances and salary
          is ",spearman_corr.iloc[:,0][1])
         print("kendall correlation coefficient of Plate Appearances and salary
          is ",kendall_corr.iloc[:,0][1] )

         pyplot.scatter(x=filtered_stats['PA'],y=filtered_stats['salary'])
         pyplot.figtext(0.15,0.5, filtered_stats['PA'].describe().to_string())
         pyplot.xlabel("Plate Appearances")
         pyplot.ylabel("salary (in tens of millions of $)")
         pyplot.title("salary (in tens of millions of $) vs Plate Appearances")
         pyplot.show()
```

```
pearson correlation coefficient of Plate Appearances and salary is  0.3
068198969793094
spearman correlation coefficient of Plate Appearances and salary is  0.
4031873110213127
kendall correlation coefficient of Plate Appearances and salary is  0.2
8186173854608887
```

Out[81]:

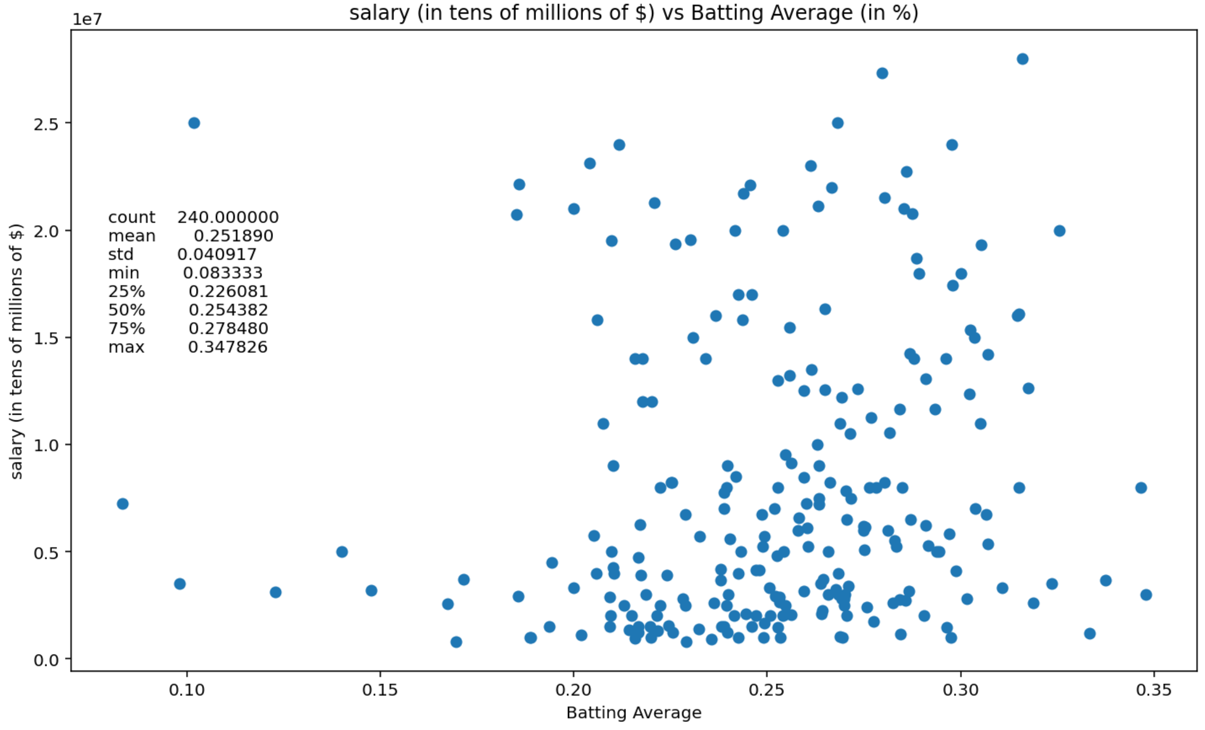


```
count   240.000000
mean    427.579167
std     199.143947
min      55.000000
25%     264.000000
50%     440.500000
75%     615.750000
max     728.000000
```

In [83]:
```
pearson_corr = filtered_stats[['BA','salary']].corr(method='pearson')
spearman_corr = filtered_stats[['BA','salary']].corr(method='spearman')
kendall_corr = filtered_stats[['BA','salary']].corr(method='kendall')

print("pearson correlation coefficient of Batting Average and salary is
 ",pearson_corr.iloc[:,0][1])
print("spearman correlation coefficient of Batting Average and salary i
s ",spearman_corr.iloc[:,0][1])
print("kendall correlation coefficient of Batting Average and salary is
 ",kendall_corr.iloc[:,0][1] )

pyplot.scatter(x=filtered_stats['BA'],y=filtered_stats['salary'])
pyplot.figtext(0.15,0.5, filtered_stats['BA'].describe().to_string())
pyplot.xlabel("Batting Average")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs Batting Average (in
 %)")
pyplot.show()
```

```
pearson correlation coefficient of Batting Average and salary is  0.124
2815669886531
spearman correlation coefficient of Batting Average and salary is  0.21
72590794095625
kendall correlation coefficient of Batting Average and salary is  0.151
2833672263775
```

Out[83]:



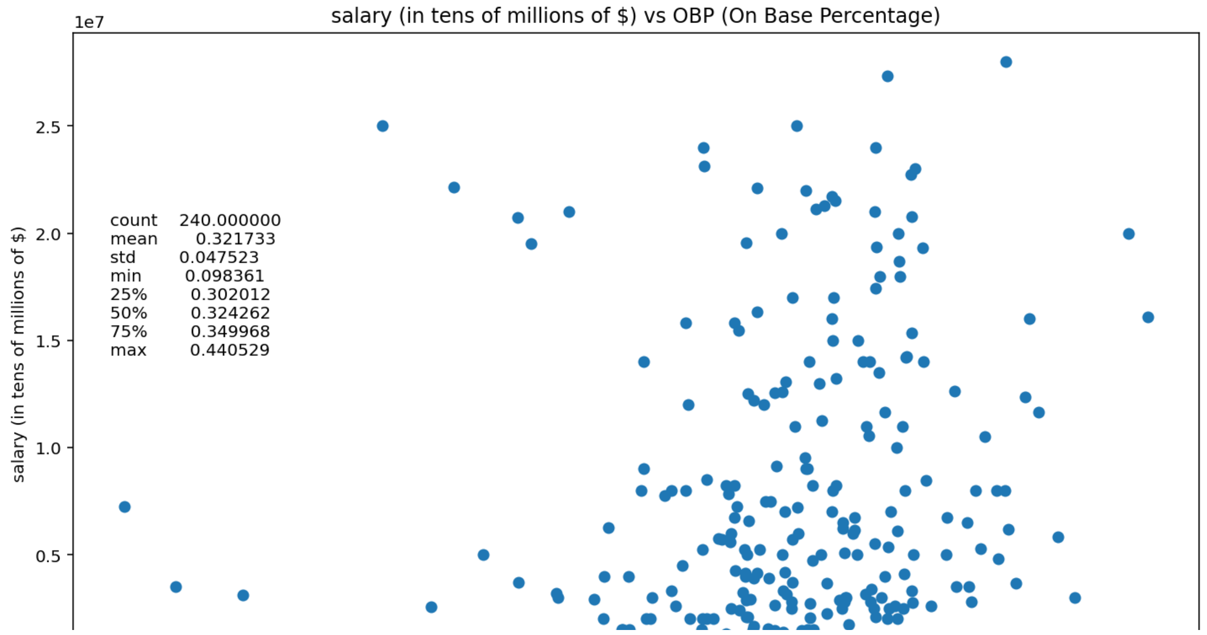
```
count    240.000000
mean       0.251890
std        0.040917
min        0.083333
25%        0.226081
50%        0.254382
75%        0.278480
max        0.347826
```

In [84]:
```
pearson_corr = filtered_stats[['OBP','salary']].corr(method='pearson')
spearman_corr = filtered_stats[['OBP','salary']].corr(method='spearman'
)
kendall_corr = filtered_stats[['OBP','salary']].corr(method='kendall')
```

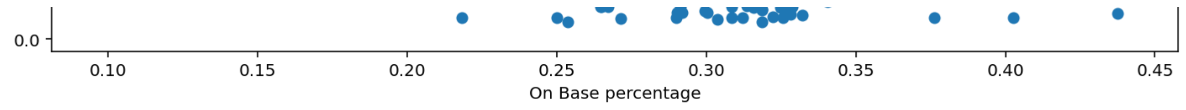
```
print("pearson correlation coefficient of OBP and salary is ",pearson_c
orr.iloc[:,0][1])
print("spearman correlation coefficient of OBP and salary is ",spearman
_corr.iloc[:,0][1])
print("kendall correlation coefficient of OBP and salary is ",kendall_c
orr.iloc[:,0][1] )


pyplot.scatter(x=filtered_stats['OBP'],y=filtered_stats['salary'])
pyplot.figtext(0.15,0.5, filtered_stats['OBP'].describe().to_string())
pyplot.xlabel("On Base percentage")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs OBP (On Base Percent
age)")
pyplot.show()
```

```
pearson correlation coefficient of OBP and salary is  0.126497737122895
47
spearman correlation coefficient of OBP and salary is  0.21405396038253
924
kendall correlation coefficient of OBP and salary is  0.145644025171995
```

Out[84]:



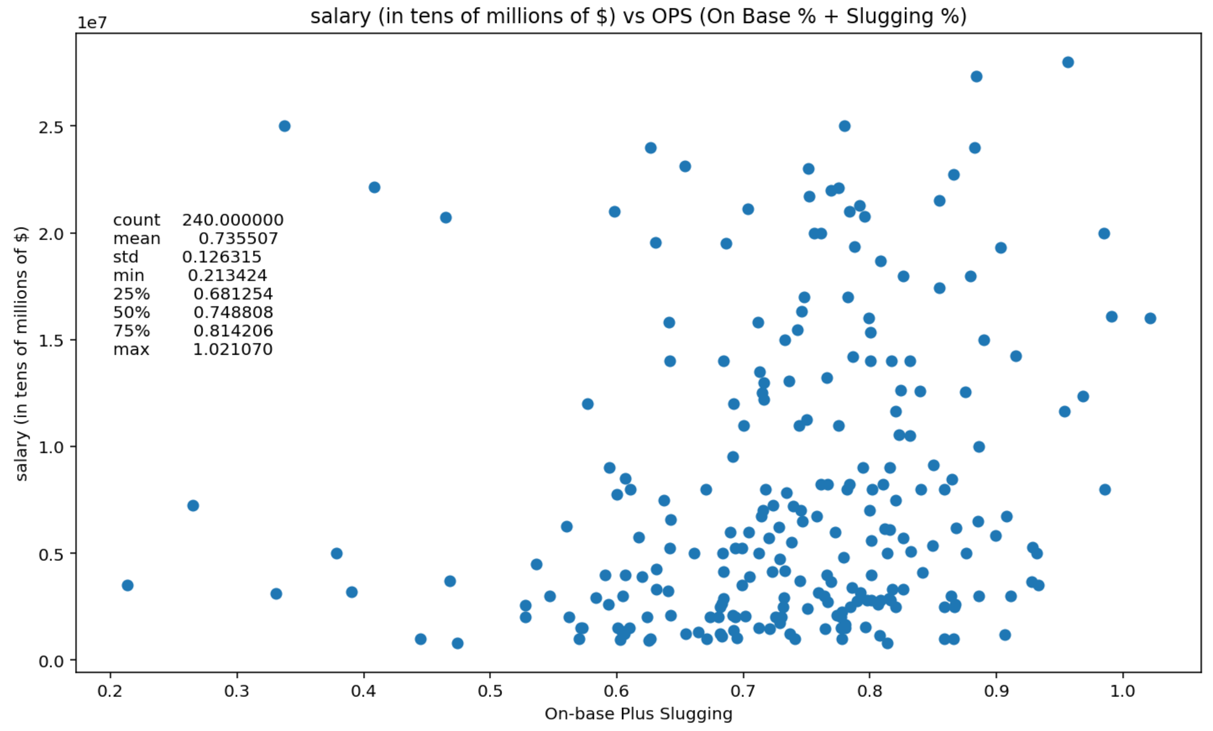
```
count    240.000000
mean       0.321733
std        0.047523
min        0.098361
25%        0.302012
50%        0.324262
75%        0.349968
max        0.440529
```



In [85]:
```
pearson_corr = filtered_stats[['OPS','salary']].corr(method='pearson')
spearman_corr = filtered_stats[['OPS','salary']].corr(method='spearman'
)
kendall_corr = filtered_stats[['OPS','salary']].corr(method='kendall')

print("pearson correlation coefficient of OPS and salary is ",pearson_c
orr.iloc[:,0][1])
print("spearman correlation coefficient of OPS and salary is ",spearman
_corr.iloc[:,0][1])
print("kendall correlation coefficient of OPS and salary is ",kendall_c
orr.iloc[:,0][1] )

pyplot.scatter(x=filtered_stats['OPS'],y=filtered_stats['salary'])
pyplot.figtext(0.15,0.5, filtered_stats['OPS'].describe().to_string())
pyplot.xlabel("On-base Plus Slugging")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs OPS (On Base % + Slu
gging %)")
pyplot.show()
```

```
pearson correlation coefficient of OPS and salary is  0.175617803265755
63
spearman correlation coefficient of OPS and salary is  0.24363365732920
886
kendall correlation coefficient of OPS and salary is  0.168237392791761
11
```

Out[85]:

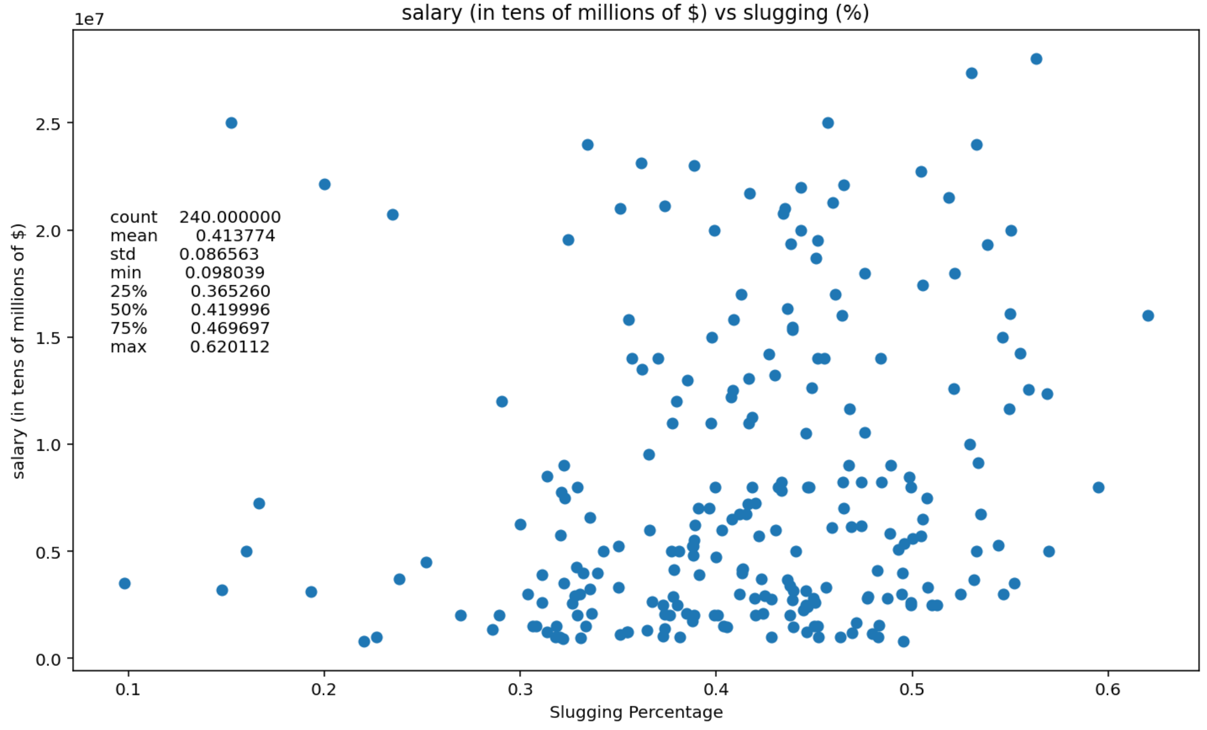


```
count    240.000000
mean       0.735507
std        0.126315
min        0.213424
25%        0.681254
50%        0.748808
75%        0.814206
max        1.021070
```

In [86]:
```
pearson_corr = filtered_stats[['SLG','salary']].corr(method='pearson')
spearman_corr = filtered_stats[['SLG','salary']].corr(method='spearman'
)
kendall_corr = filtered_stats[['SLG','salary']].corr(method='kendall')

print("pearson correlation coefficient of SLG and salary is ",pearson_c
orr.iloc[:,0][1])
print("spearman correlation coefficient of SLG and salary is ",spearman
_corr.iloc[:,0][1])
print("kendall correlation coefficient of SLG and salary is ",kendall_c
orr.iloc[:,0][1] )


pyplot.scatter(x=filtered_stats['SLG'],y=filtered_stats['salary'])
pyplot.figtext(0.15,0.5, filtered_stats['SLG'].describe().to_string())
pyplot.xlabel("Slugging Percentage")
pyplot.ylabel("salary (in tens of millions of $)")
pyplot.title("salary (in tens of millions of $) vs slugging (%)")
pyplot.show()
```

```
pearson correlation coefficient of SLG and salary is  0.186818364056228
04
spearman correlation coefficient of SLG and salary is  0.23965575475143
397
kendall correlation coefficient of SLG and salary is  0.164704688567037
5
```

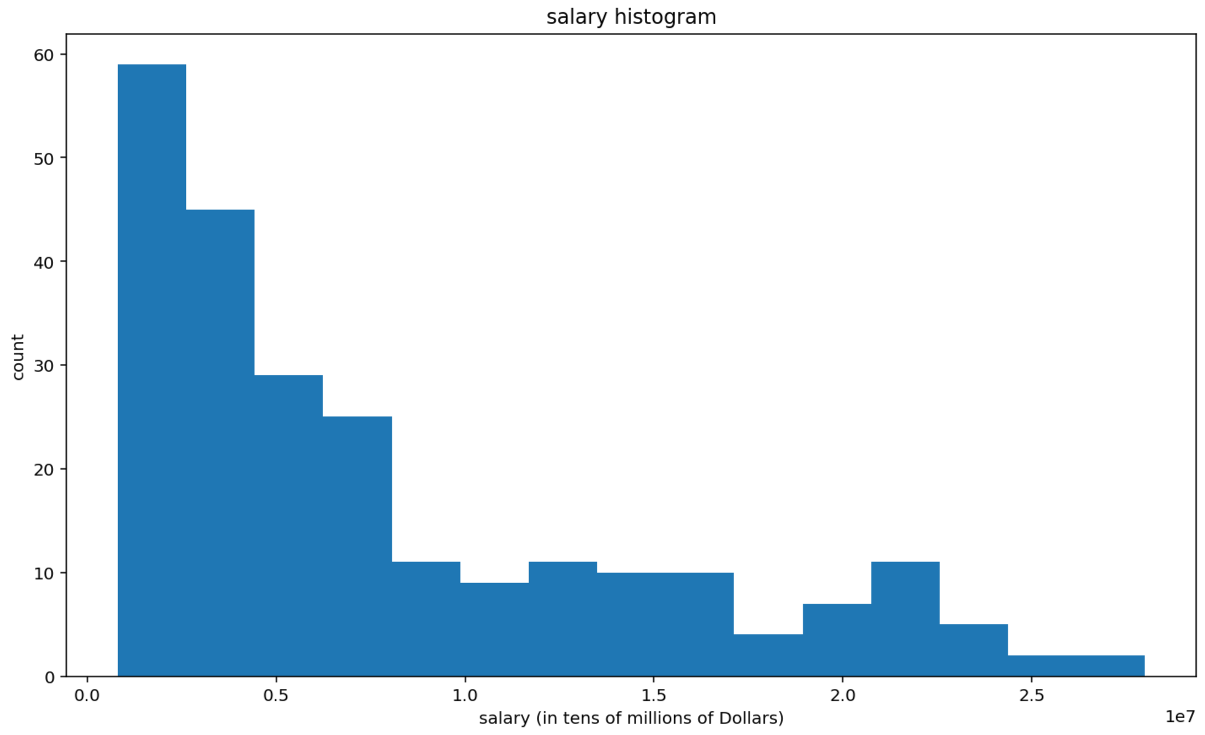
Out[86]:



```
count    240.000000
mean       0.413774
std        0.086563
min        0.098039
25%        0.365260
50%        0.419996
75%        0.469697
max        0.620112
```

In [87]:
```
pyplot.hist(filtered_stats['salary'], bins=15)
pyplot.title('salary histogram')
pyplot.xlabel('salary (in tens of millions of Dollars)')
pyplot.ylabel('count')
pyplot.show()
```

Out[87]:



In [66]:
```
def simple_regression(n):
    model = LinearRegression().fit(filtered_stats[[n]], filtered_stats
[['salary']])
    print("Linear regression slope for " + n + ":", model.coef_[0][0])
    print("Linear regression intercept for " + n + ":", model.intercept
_[0])
    print("Linear regression score for " + n + ":", model.score(filtere
d_stats[[n]], filtered_stats[['salary']]))
```

```
In [67]: simple_regression('RBI')
```

```
Linear regression slope for RBI: 79867.17527891042
Linear regression intercept for RBI: 3828717.8374136067
Linear regression score for RBI: 0.134928241022247
```

```
In [68]: simple_regression('HR')
```

```
Linear regression slope for HR: 197161.996905589
Linear regression intercept for HR: 5076505.185671769
Linear regression score for HR: 0.10729235795784921
```

```
In [69]: simple_regression('BB')
```

```
Linear regression slope for BB: 85234.81437586324
Linear regression intercept for BB: 4675839.207860775
Linear regression score for BB: 0.09325807181382373
```

```
In [70]: simple_regression('H')
```

```
Linear regression slope for H: 37128.22668418655
Linear regression intercept for H: 4168418.620581783
Linear regression score for H: 0.08659588352723402
```

```
In [108]: multiple_model_all = LinearRegression().fit(filtered_stats[['RBI', 'HR'
          , 'BB', 'H']], filtered_stats[['salary']])
          print("Multiple regression all variables slopes:", multiple_model_all.c
          oef_[0])
          print("Multiple regression all variables intercept:", multiple_model_al
          l.intercept_[0])
          print("Multiple regression all variables score:", multiple_model_all.sc
          ore(filtered_stats[['RBI', 'HR', 'BB', 'H']], filtered_stats[['salary'
          ]]))
```

```
Multiple regression all variables slopes: [118891.71151326 -53693.52230
346  23622.05989112 -23422.65420427]
Multiple regression all variables intercept: 4066191.0938644363
Multiple regression all variables score: 0.14285335125400866
```

In [51]:
```
logregRBI = LogisticRegression(max_iter=1000)
logregRBI.fit(filtered_stats[['salary']],filtered_stats['RBI'])
print("Logistic Regression score for RBI & Salary", logregRBI.score(fil
tered_stats[['salary']],filtered_stats['RBI']))
```

```
Logistic Regression score for RBI & Salary 0.020833333333333332
```

In [52]:
```
logregHR = LogisticRegression(max_iter=1000)
logregHR.fit(filtered_stats[['salary']],filtered_stats['HR'])
print("Logistic Regression score for HR & Salary", logregHR.score(filte
red_stats[['salary']],filtered_stats['HR']))
```

```
Logistic Regression score for HR & Salary 0.0625
```

In [54]:
```
logregBB = LogisticRegression(max_iter=1000)
logregBB.fit(filtered_stats[['salary']],filtered_stats['BB'])
print("Logistic Regression score for Walks & Salary", logregBB.score(fi
ltered_stats[['salary']],filtered_stats['BB']))
```

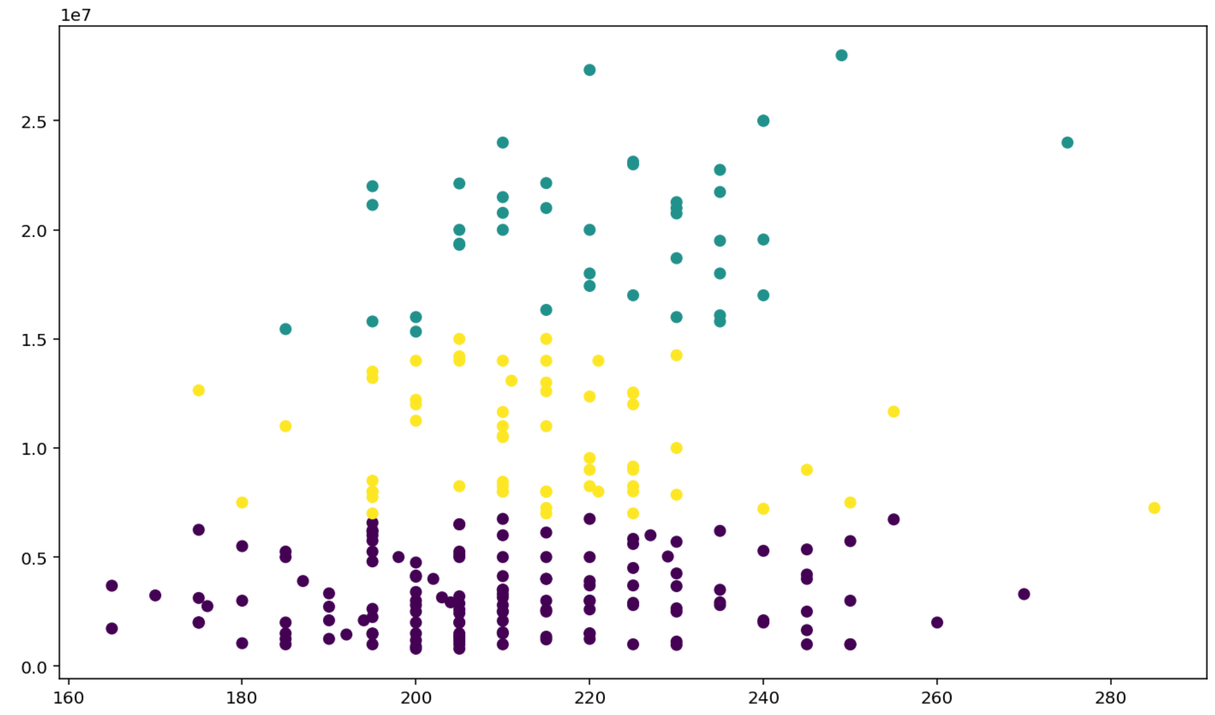
```
Logistic Regression score for Walks & Salary 0.020833333333333332
```

In [55]:
```
logregH = LogisticRegression(max_iter=1000)
logregH.fit(filtered_stats[['salary']],filtered_stats['H'])
print("Logistic Regression score for Hits & Salary", logregH.score(filt
ered_stats[['salary']],filtered_stats['H']))
```

```
Logistic Regression score for Hits & Salary 0.0125
```

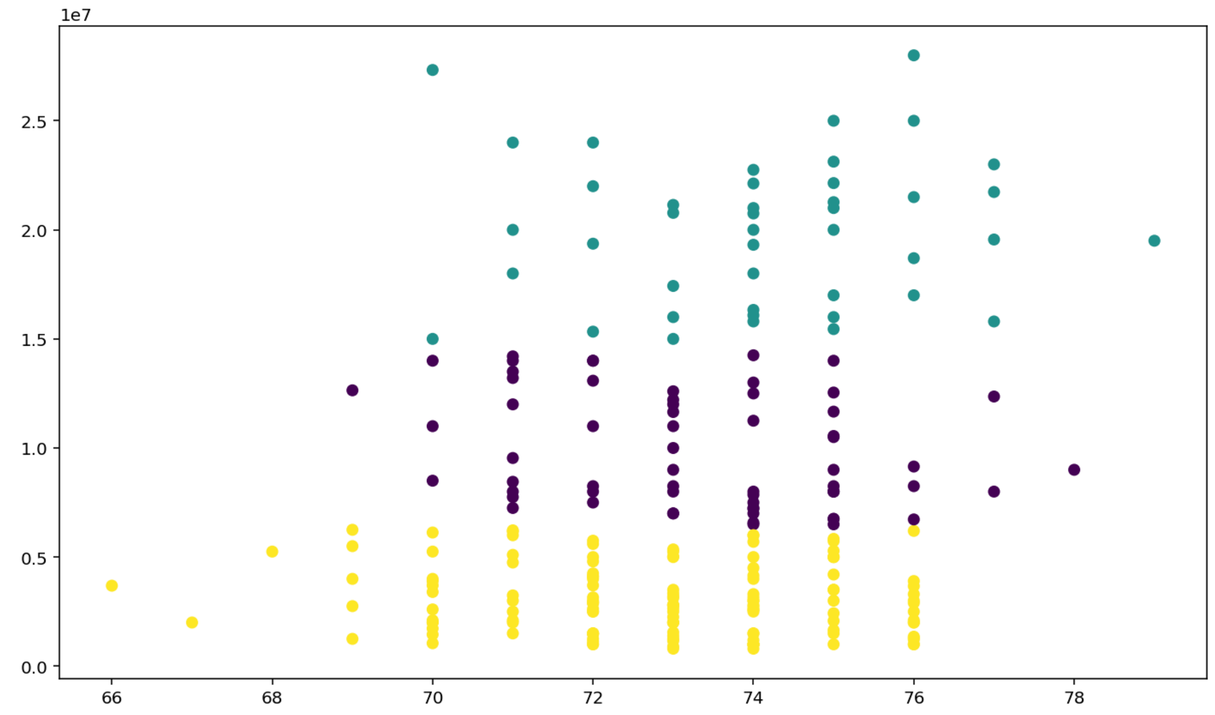
In [60]:
```
#clustering
from sklearn.cluster import KMeans

values = filtered_stats[['weight', 'salary']]
clustering= KMeans(n_clusters=3)
clustering.fit(values)
labels = clustering.labels_

pyplot.scatter(filtered_stats['weight'],filtered_stats['salary'],c=labe
ls,cmap='viridis')
pyplot.show()
```

Out[60]:



In [61]:
```
#clustering
from sklearn.cluster import KMeans

values = filtered_stats[['height', 'salary']]
clustering= KMeans(n_clusters=3)
clustering.fit(values)
labels = clustering.labels_

pyplot.scatter(filtered_stats['height'],filtered_stats['salary'],c=labe
ls,cmap='viridis')
pyplot.show()
```

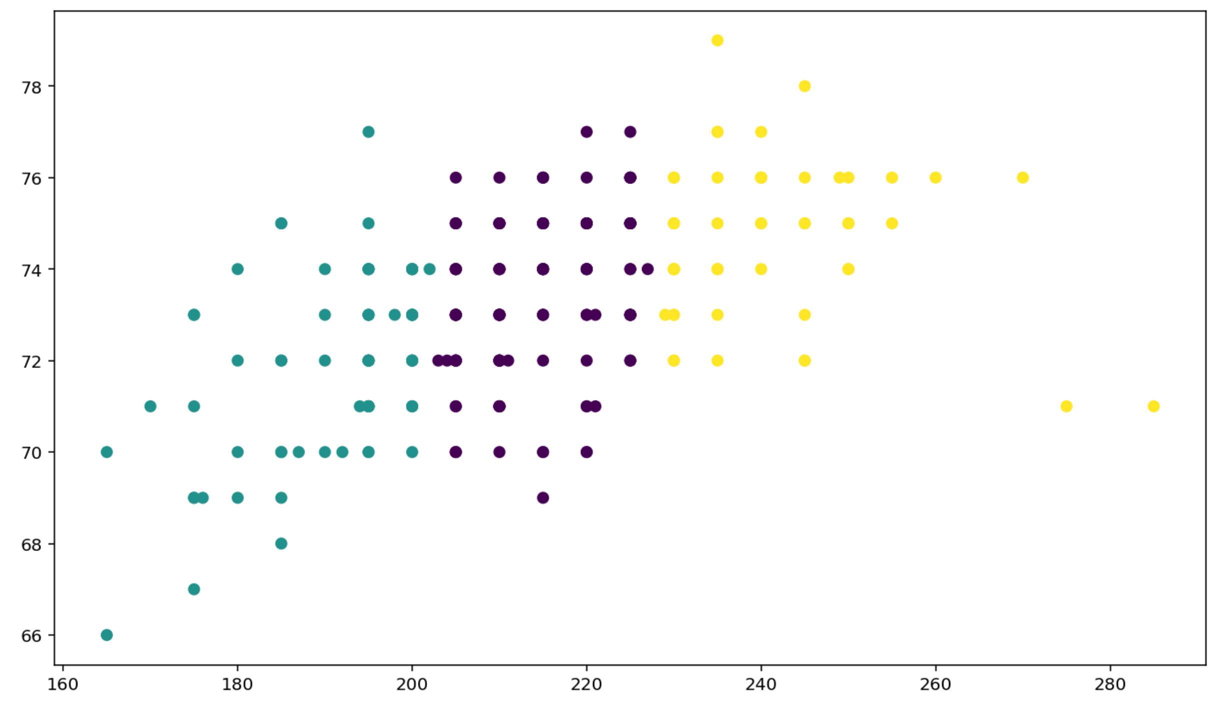
Out[61]:



In [62]:

```
#clustering
from sklearn.cluster import KMeans

values = filtered_stats[['weight', 'height']]
clustering= KMeans(n_clusters=3)
clustering.fit(values)
labels = clustering.labels_

pyplot.scatter(filtered_stats['weight'],filtered_stats['height'],c=labe
ls,cmap='viridis')
pyplot.show()
```

Out[62]:



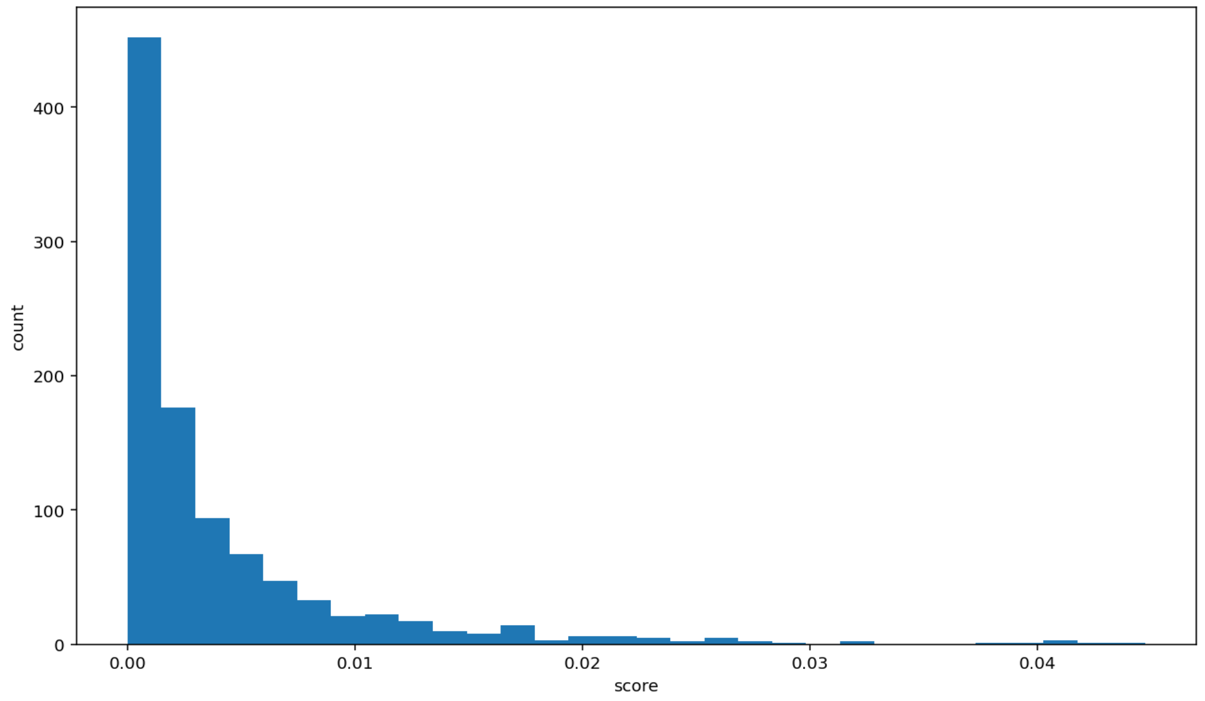
Evaluation of Significance

In [22]:
```
def permuted_salary():
    return numpy.random.permutation(filtered_stats['salary'].copy())
```

In [46]:
```
scores = numpy.zeros(1000)
for i in range(1000):
    salaries = permuted_salary()
    model = LinearRegression().fit(filtered_stats[['RBI']], salaries)
    scores[i] = model.score(filtered_stats[['RBI']], salaries)

pyplot.hist(scores, bins=30)
pyplot.xlabel('score')
pyplot.ylabel('count')
pyplot.show()
```

```
count = 0
for i in range(len(scores)):
    if scores[i] > 0.134928241022247:
        count += 1
print("Count of scores greater than real regression score for RBI:", co
unt)
```

Out[46]:



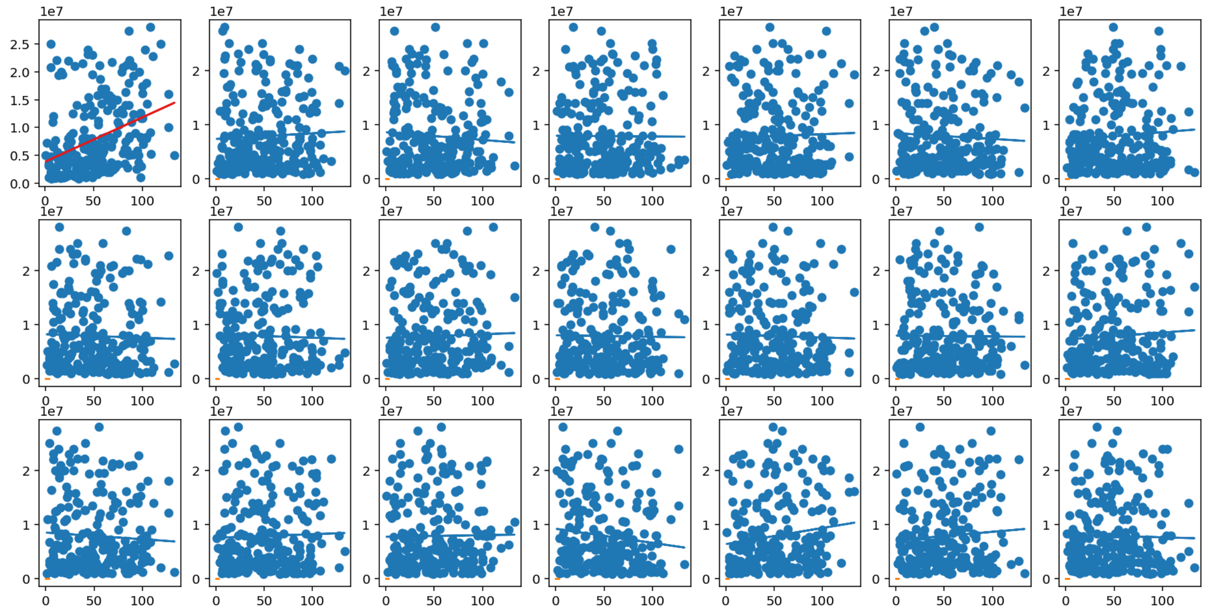
```
Count of scores greater than real regression score for RBI: 0
```

In [59]:

```
print("Permuted scatterplots for RBI:")
palette = pyplot.get_cmap("Set1")
pyplot.figure(figsize=(16,8))
for panel in range(1,22):
    pyplot.subplot(3,7,panel)

    if panel == 1:
        pyplot.scatter(filtered_stats['RBI'], filtered_stats['salary'])
        pyplot.plot(filtered_stats['RBI'], RBI_model.predict(filtered_s
tats[['RBI']]), color=palette(0))

    else:
        salaries = permuted_salary()
        permuted = LinearRegression().fit(filtered_stats[['RBI']], sala
ries)
        permuted_slope = permuted.coef_[0]
        permuted_predicted = permuted.predict(filtered_stats[['RBI']])
        pyplot.scatter(filtered_stats['RBI'], salaries)
        color = palette(1)
        if permuted_slope > RBI_model.coef_[0]:
            color = palette(0)
        pyplot.plot(filtered_stats['RBI'], permuted_predicted, color)

pyplot.show()
```

```
Permuted scatterplots for RBI:
```

Out[59]:

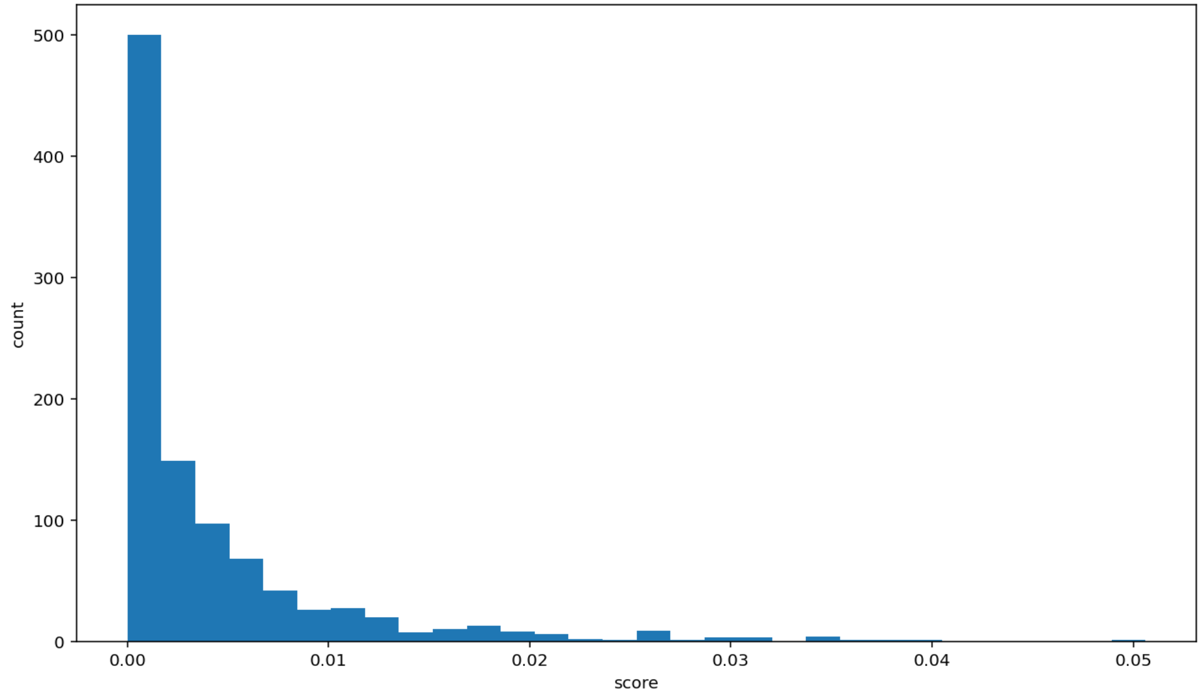


In [47]:

```
scores = numpy.zeros(1000)
```

```
for i in range(1000):
    salaries = permuted_salary()
    model = LinearRegression().fit(filtered_stats[['HR']], salaries)
    scores[i] = model.score(filtered_stats[['HR']], salaries)

pyplot.hist(scores, bins=30)
pyplot.xlabel('score')
pyplot.ylabel('count')
pyplot.show()

count = 0
for i in range(len(scores)):
    if scores[i] > 0.10729235795784921:
        count += 1
print("Count of scores greater than real regression score for HR:", cou
nt)
```

Out[47]:

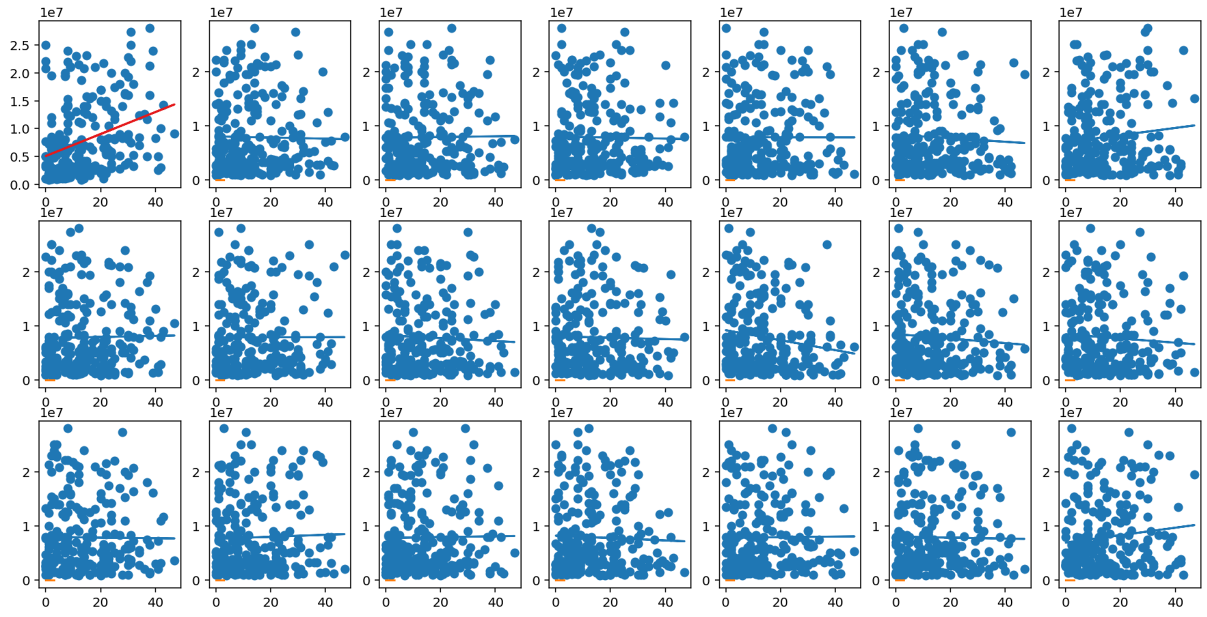


```
Count of scores greater than real regression score for HR: 0
```

In [57]:
```
print("Permuted scatterplots for home runs:")
palette = pyplot.get_cmap("Set1")
pyplot.figure(figsize=(16,8))
for panel in range(1,22):
    pyplot.subplot(3,7,panel)

    if panel == 1:
        pyplot.scatter(filtered_stats['HR'], filtered_stats['salary'])
        pyplot.plot(filtered_stats['HR'], HR_model.predict(filtered_sta
ts[['HR']]), color=palette(0))

    else:
        salaries = permuted_salary()
        permuted = LinearRegression().fit(filtered_stats[['HR']], salar
ies)
        permuted_slope = permuted.coef_[0]
        permuted_predicted = permuted.predict(filtered_stats[['HR']])
        pyplot.scatter(filtered_stats['HR'], salaries)
        color = palette(1)
        if permuted_slope > HR_model.coef_[0]:
            color = palette(0)
        pyplot.plot(filtered_stats['HR'], permuted_predicted, color)

pyplot.show()
```

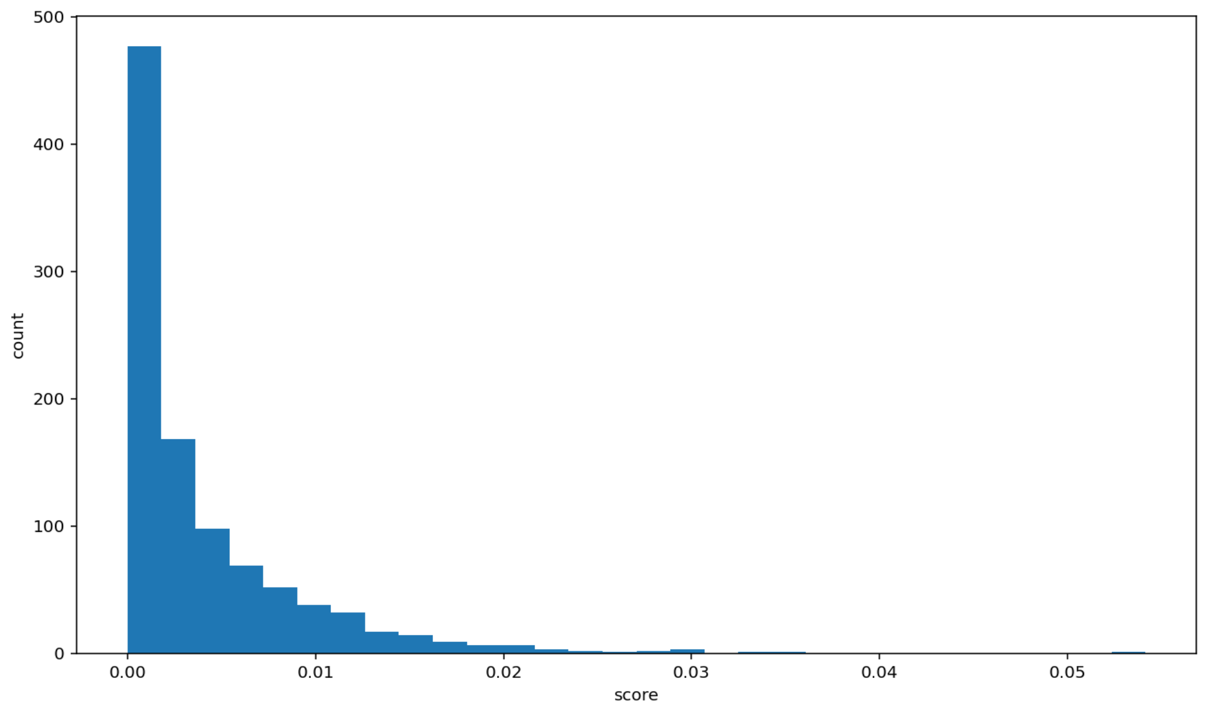
```
Permuted scatterplots for home runs:
```

Out[57]:



```
In [48]: scores = numpy.zeros(1000)
         for i in range(1000):
             salaries = permuted_salary()
             model = LinearRegression().fit(filtered_stats[['BB']], salaries)
             scores[i] = model.score(filtered_stats[['BB']], salaries)

         pyplot.hist(scores, bins=30)
         pyplot.xlabel('score')
         pyplot.ylabel('count')
         pyplot.show()

         count = 0
         for i in range(len(scores)):
             if scores[i] > 0.09325807181382373:
                 count += 1
         print("Count of scores greater than real regression score for walks:",
         count)
```

Out[48]:



```
Count of scores greater than real regression score for walks: 0
```

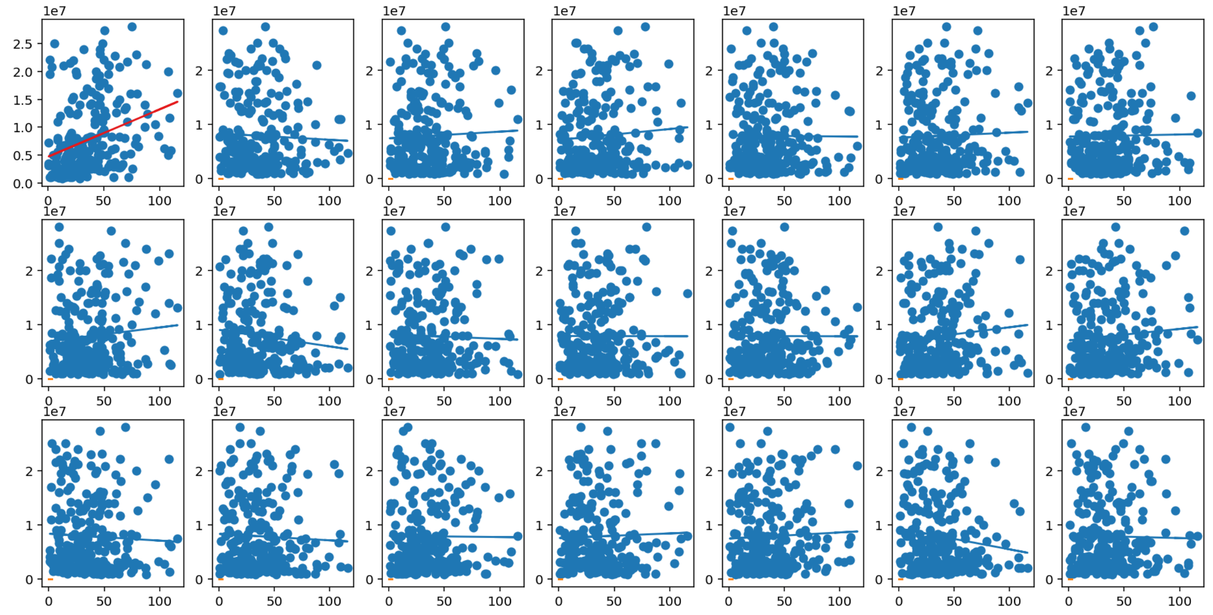
In [56]:

```
print("Permuted scatterplots for walks:")
palette = pyplot.get_cmap("Set1")
pyplot.figure(figsize=(16,8))
for panel in range(1,22):
    pyplot.subplot(3,7,panel)

    if panel == 1:
        pyplot.scatter(filtered_stats['BB'], filtered_stats['salary'])
        pyplot.plot(filtered_stats['BB'], BB_model.predict(filtered_sta
ts[['BB']]), color=palette(0))

    else:
        salaries = permuted_salary()
        permuted = LinearRegression().fit(filtered_stats[['BB']], salar
ies)
```

```
        permuted_slope = permuted.coef_[0]
        permuted_predicted = permuted.predict(filtered_stats[['BB']])
        pyplot.scatter(filtered_stats['BB'], salaries)
        color = palette(1)
        if permuted_slope > BB_model.coef_[0]:
            color = palette(0)
        pyplot.plot(filtered_stats['BB'], permuted_predicted, color)

pyplot.show()
```

```
Permuted scatterplots for walks:
```

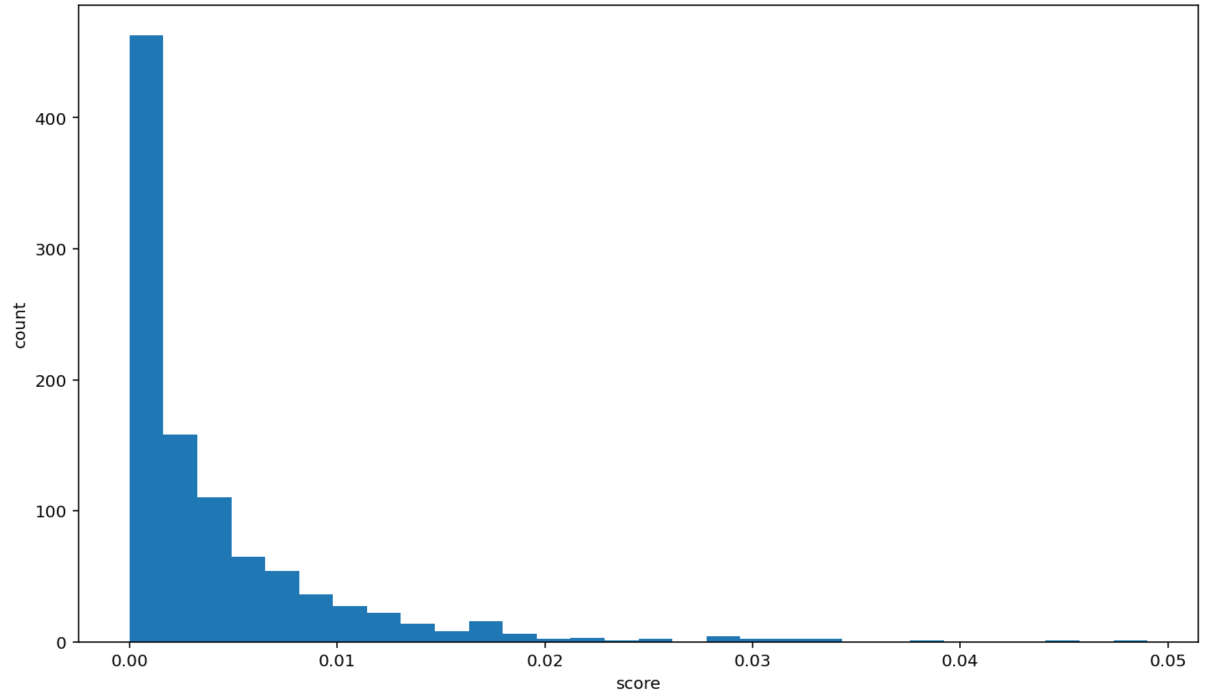
Out[56]:



In [49]:

```
scores = numpy.zeros(1000)
for i in range(1000):
    salaries = permuted_salary()
    model = LinearRegression().fit(filtered_stats[['H']], salaries)
    scores[i] = model.score(filtered_stats[['H']], salaries)

pyplot.hist(scores, bins=30)
pyplot.xlabel('score')
pyplot.ylabel('count')
pyplot.show()

count = 0
for i in range(len(scores)):
    if scores[i] > 0.08659588352723402:
        count += 1
print("Count of scores greater than real regression score for hits:", c
ount)
```

Out[49]:



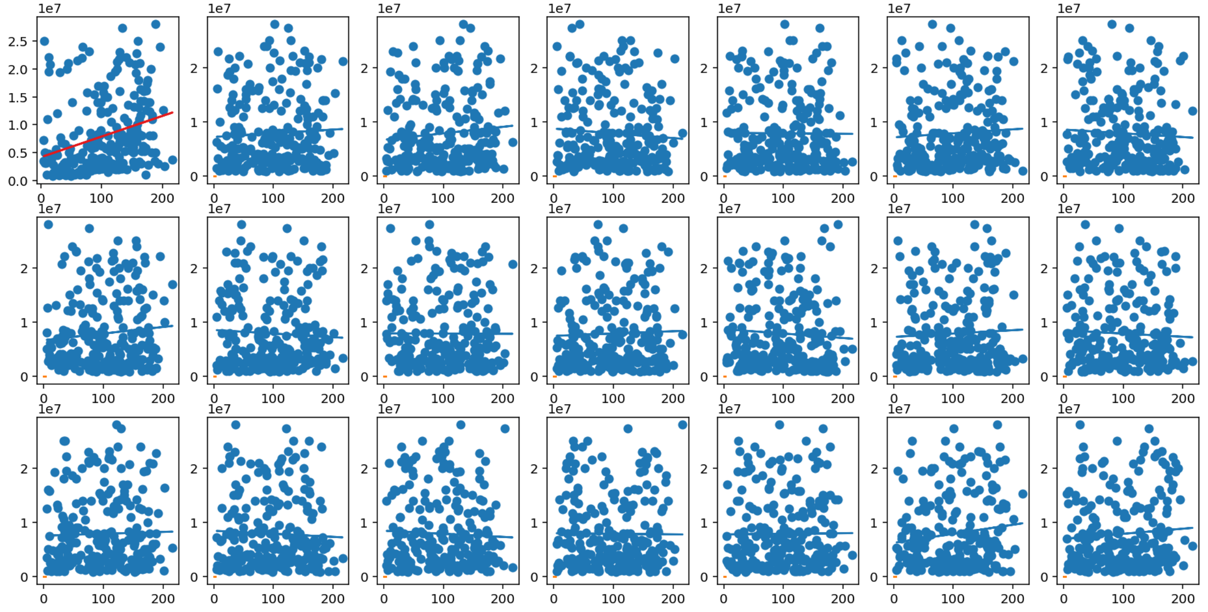
```
Count of scores greater than real regression score for hits: 0
```

In [53]:

```
print("Permuted scatterplots for hits:")
palette = pyplot.get_cmap("Set1")
pyplot.figure(figsize=(16,8))
for panel in range(1,22):
    pyplot.subplot(3,7,panel)
    if panel == 1:
        pyplot.scatter(filtered_stats['H'], filtered_stats['salary'])
        pyplot.plot(filtered_stats['H'], H_model.predict(filtered_stats
[['H']]), color=palette(0))

    else:
        salaries = permuted_salary()
        permuted = LinearRegression().fit(filtered_stats[['H']], salari
es)
        permuted_slope = permuted.coef_[0]
        permuted_predicted = permuted.predict(filtered_stats[['H']])
        pyplot.scatter(filtered_stats['H'], salaries)
        color = palette(1)
        if permuted_slope > H_model.coef_[0]:
            color = palette(0)
        pyplot.plot(filtered_stats['H'], permuted_predicted, color)

pyplot.show()
```

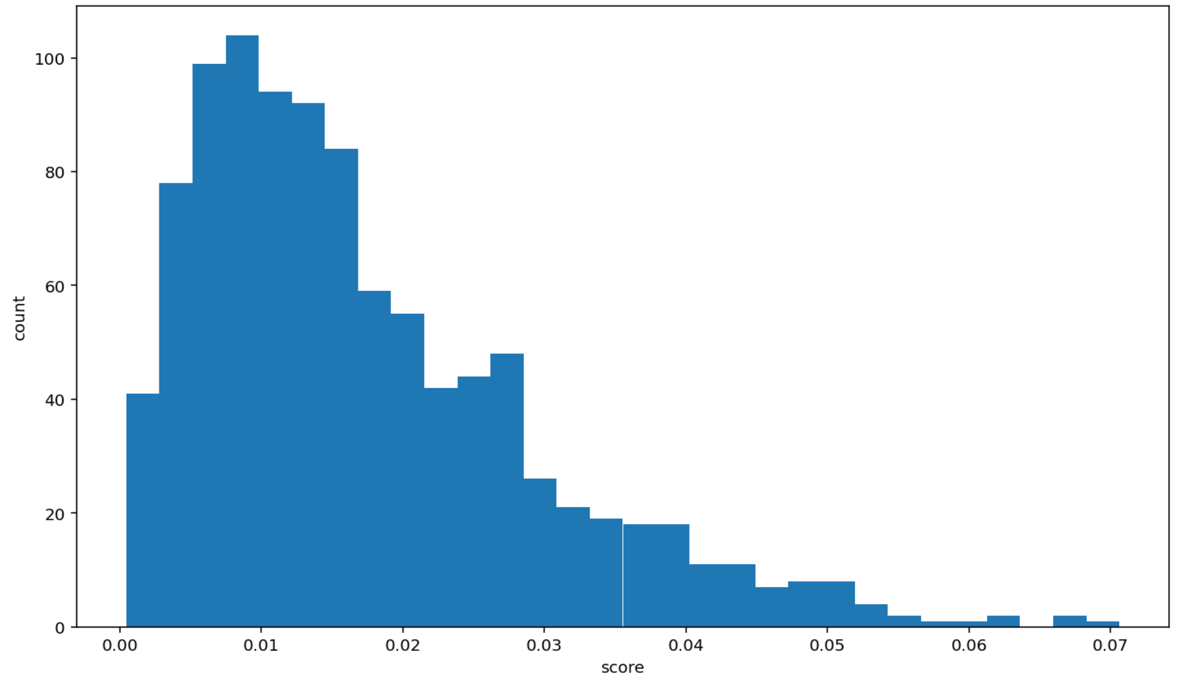
```
Permuted scatterplots for hits:
```

Out[53]:



In [50]:
```
scores = numpy.zeros(1000)
for i in range(1000):
    salaries = permuted_salary()
    model = LinearRegression().fit(filtered_stats[['RBI', 'HR', 'BB',
'H']], salaries)
    scores[i] = model.score(filtered_stats[['RBI', 'HR', 'BB', 'H']], s
alaries)

pyplot.hist(scores, bins=30)
pyplot.xlabel('score')
pyplot.ylabel('count')
pyplot.show()

count = 0
for i in range(len(scores)):
    if scores[i] > 0.14285335125400866:
        count += 1
print("Count of scores greater than real multiple regression score for
 all variables:", count)
```

Out[50]:



```
Count of scores greater than real multiple regression score for all v
ariables: 0
```

Conclusion

Over the course of our data analysis, we had many interesting observations. First, as shown the graphs we made scatterplots to see which attributes affect the salary the most. We measured the 3 correlation coefficients we have learnt about (Spearman, Pearson, and kendall) and we plotted the graphs out to see the patterns. From the results, we saw during this first stage of using summary functions we noticed that HR, RBI, BB and H seemed to have the most impact on a MLB batter's salary as the coefficient values were high and there appeared to be some sort of dependency shown through the scatter plots. The salary histogram after these scatter plots showed how the salaries varies in terms of frequency. As you can see most salaries earned by MLB hitters is only between 0.1 to 0.75 tens of millions of dollars following a exponentially decay based plot. If you notice from the earlier scatter plots RBI (runs batted in) had the best correlation values against salary indicating that it affects the salary the strongest most positively. This hypothesis was also supported by the linear regression modelling where RBI had the highest linear reg score against salary showing that the relationship can to a good extent be modeled in a linear fashion. This makes sense theoretically, because the more runs you are contributing in each of your innings increases the chances of your team winning by a lot and is taken seriously by coaches as it is a great indicator of performance. Note however, the linear regression for all the models is still very weak as it is close to zero so there is uncertainty in reaching such a conclusion of linear dependence. We also tried out logistic regression models for these 4 best attributes to see how they impact the salary if measured through a logistic function. The surprising results in these models was that HR (homeruns) seemed to fit the salary in terms of a logistic function indicating that as homeruns seem to get as big as possible then salary tapers off. However, the score of the regression model is still very low so there is uncertainty in reaching such a conclusion. The reason this might have happened is because not enough data might be there for high salary and high homerun hitters. For physical attributes such as height and weight we also used k-means clustering to run models against them with respect to salary as well as we found there to be 3 main clusters of MLB hitters. If you notice there is an optimal cluster of weight and heights that results in the highest salary and the other 2 clusters for from less optimal to least optimal. We notice that there is an ideal weight and height that makes the best MLB hitter physcial specimen which means they would perform the best in the MLB. This makes sense because genetics is very important in sports in general and the optimal weight of 220 pounds and optimal height of 74 inches means the player is tall and so they have good reach with the bat and they are muscular enough to provide enough drive into the ball. The count of scores greater than real regression scores for the 4 variables show that the regression models are not random because if you see the multiple permuted plots of salary all lines are blue which indicate none of the scores are widely different enough to make them red which makes us confident of our current models. This explains the evaluation of signficance as it shows that we are confident of our linear models. The scores for our logistic models were very low so we are not so confident of them.

Source Code

Not applicable; all of our code is contained within this notebook file.

Acknowledgements

We would like to acknowledge the following orgnizations for both the use of their Python modules as well as the supporting reference documentation they've made available on these websites:

- https://numpy.org/
- https://pandas.pydata.org/
- https://matplotlib.org/
- https://scikit-learn.org/

We would also like to acknowledge the following websites for providing us with the data we needed to create this project:

- https://data.world/makeovermonday/2019w19/workspace/file?filename=MLB+Stats.csv
- https://www.baseball-reference.com/leagues/MLB/2016-standard-pitching.shtml

```
In [0]:
```