In [1]:
```
from datascience import *
import numpy as np
import pandas as pd

%matplotlib inline
import matplotlib.pyplot as plots
plots.style.use('fivethirtyeight')
```

In [2]:
```
pwd
```

Out[2]: 'C:\\Users\\guophie\\SAIL Application'

In [3]:
```
cbb: Table = Table.read_table('cbb.csv')
cbb
```

Out[3]:

TEAM	CONF	G	W	ADJOE	ADJDE	BARTHAG	EFG_O	EFG_D	TOR	TORD	ORB	DRB	FTR	FTRD	2P_O	2P_D	3P_O	3P_D	ADJ_T	WAB	POSTSEASON	SEED	YEAR
North Carolina	ACC	40	33	123.3	94.9	0.9531	52.6	48.1	15.4	18.2	40.7	30	32.3	30.4	53.9	44.6	32.7	36.2	71.7	8.6	2ND	1	2016
Wisconsin	B10	40	36	129.1	93.6	0.9758	54.8	47.7	12.4	15.8	32.1	23.7	36.2	22.4	54.8	44.7	36.5	37.5	59.3	11.3	2ND	1	2015
Michigan	B10	40	33	114.4	90.4	0.9375	53.9	47.7	14	19.5	25.5	24.9	30.7	30	54.7	46.8	35.2	33.2	65.9	6.9	2ND	3	2018
Texas Tech	B12	38	31	115.2	85.2	0.9696	53.5	43	17.7	22.8	27.4	28.7	32.9	36.6	52.8	41.9	36.5	29.7	67.5	7	2ND	3	2019
Gonzaga	WCC	39	37	117.8	86.3	0.9728	56.6	41.1	16.2	17.1	30	26.2	39	26.9	56.3	40	38.2	29	71.5	7.7	2ND	1	2017
Kentucky	SEC	40	29	117.2	96.2	0.9062	49.9	46	18.1	16.1	42	29.7	51.8	36.8	50	44.9	33.2	32.2	65.9	3.9	2ND	8	2014
Michigan	B10	38	30	121.5	93.7	0.9522	54.6	48	14.6	18.7	32.5	29.4	28.4	22.7	53.4	47.6	37.9	32.6	64.8	6.2	2ND	4	2013
Duke	ACC	39	35	125.2	90.6	0.9764	56.6	46.5	16.3	18.6	35.8	30.2	39.8	23.9	55.9	46.3	38.7	31.4	66.4	10.7	Champions	1	2015
Virginia	ACC	38	35	123	89.9	0.9736	55.2	44.7	14.7	17.5	30.4	25.4	29.1	26.3	52.5	45.7	39.5	28.9	60.7	11.1	Champions	1	2019
North Carolina	ACC	39	33	121	91.5	0.9615	51.7	48.1	16.2	18.6	41.3	25	34.3	31.6	51	46.3	35.5	33.9	72.8	8.4	Champions	1	2017

... (2445 rows omitted)

Cleaning the *cbb* dataset to include only 2019 March Madness teams, in order of ascending *SEED*.

In [4]:
```
cbb_2019 = cbb.where("YEAR", 2019)
```

In [5]:
```
mm_2019 = cbb_2019.where("SEED", are.strictly_between(0, 17)).sort("SEED")
```

Converting the *POSTSEASON* str column into an int column named *RANK.*

In [6]:
```
rank_column = make_array()
for i in np.arange(mm_2019.num_rows):
    if mm_2019.column("POSTSEASON")[i] == "Champions":
        rank_column = np.append(rank_column, 1)
    if mm_2019.column("POSTSEASON")[i] == "2ND":
        rank_column = np.append(rank_column, 2)
    if mm_2019.column("POSTSEASON")[i] == "F4":
        rank_column = np.append(rank_column, 4)
    if mm_2019.column("POSTSEASON")[i] == "E8":
        rank_column = np.append(rank_column, 8)
    if mm_2019.column("POSTSEASON")[i] == "S16":
        rank_column = np.append(rank_column, 16)
    if mm_2019.column("POSTSEASON")[i] == "R32":
        rank_column = np.append(rank_column, 32)
    if mm_2019.column("POSTSEASON")[i] == "R64":
        rank_column = np.append(rank_column, 64)
    if mm_2019.column("POSTSEASON")[i] == "R68":
        rank_column = np.append(rank_column, 68)
```

In [7]:
```
mm_2019 = mm_2019.drop("POSTSEASON").with_column("RANK", rank_column)
mm_2019
```

Out[7]:

TEAM	CONF	G	W	ADJOE	ADJDE	BARTHAG	EFG_O	EFG_D	TOR	TORD	ORB	DRB	FTR	FTRD	2P_O	2P_D	3P_O	3P_D	ADJ_T	WAB	SEED	YEAR	RANK
Virginia	ACC	38	35	123	89.9	0.9736	55.2	44.7	14.7	17.5	30.4	25.4	29.1	26.3	52.5	45.7	39.5	28.9	60.7	11.1	1	2019	1
Duke	ACC	38	32	118.9	89.2	0.9646	53.6	45	17.5	19.4	35.6	29.5	33.2	24	58	45	30.8	29.9	73.6	11.2	1	2019	8
Gonzaga	WCC	37	33	123.4	89.9	0.9744	59	44.2	14.9	19	31.5	26.8	35.3	25.9	61.4	43.4	36.3	30.4	72	7	1	2019	8
North Carolina	ACC	36	29	120.1	91.4	0.9582	52.9	48.9	17.2	18.3	35.3	22.8	30.2	28.4	52.1	47.9	36.2	33.5	76	10	1	2019	16
Kentucky	SEC	37	30	117.5	89.8	0.9568	53	46.6	18.6	17.9	36.8	25.5	41.9	26.8	52.9	43.6	35.4	34.3	66.9	8.8	2	2019	8
Michigan St.	B10	39	32	119.9	91	0.9597	55.2	43.9	18.5	14.9	33.9	26.4	33.6	27.5	54.3	41.9	37.8	31.6	68.6	10.7	2	2019	4
Michigan	B10	37	30	114.6	85.6	0.9665	51.6	44.1	13.9	18	24.7	24.8	27.5	24.1	51.8	44.3	34.2	29.1	65.9	9.2	2	2019	16
Tennessee	SEC	36	31	122.8	95.2	0.9488	55.3	48.1	15.8	18	31.6	30.2	33.3	34.9	55.4	44.7	36.7	35.4	68.8	9.9	2	2019	16
Texas Tech	B12	38	31	115.2	85.2	0.9696	53.5	43	17.7	22.8	27.4	28.7	32.9	36.6	52.8	41.9	36.5	29.7	67.5	7	3	2019	2
Purdue	B10	36	26	122.8	94.3	0.9539	53.6	49	15.8	18.6	34.5	27	29.9	31.7	51.5	47.2	37.4	34.2	67	6.1	3	2019	8

... (58 rows omitted)

Converting the *SEED* column into a column named *Standard SEED* that can be compared with *RANK*.

In [8]:
```
# Limiting the accepted values for Standard Seed to be 1, 2, 4, 8, 16, 32, 64, and 68.

mm_2019 = mm_2019.with_column("Standard SEED", mm_2019.sort("RANK").column("RANK")).drop("SEED")
mm_2019
```

Out[8]:

TEAM	CONF	G	W	ADJOE	ADJDE	BARTHAG	EFG_O	EFG_D	TOR	TORD	ORB	DRB	FTR	FTRD	2P_O	2P_D	3P_O	3P_D	ADJ_T	WAB	YEAR	RANK	Standard SEED
Virginia	ACC	38	35	123	89.9	0.9736	55.2	44.7	14.7	17.5	30.4	25.4	29.1	26.3	52.5	45.7	39.5	28.9	60.7	11.1	2019	1	1
Duke	ACC	38	32	118.9	89.2	0.9646	53.6	45	17.5	19.4	35.6	29.5	33.2	24	58	45	30.8	29.9	73.6	11.2	2019	8	2
Gonzaga	WCC	37	33	123.4	89.9	0.9744	59	44.2	14.9	19	31.5	26.8	35.3	25.9	61.4	43.4	36.3	30.4	72	7	2019	8	4
North Carolina	ACC	36	29	120.1	91.4	0.9582	52.9	48.9	17.2	18.3	35.3	22.8	30.2	28.4	52.1	47.9	36.2	33.5	76	10	2019	16	4
Kentucky	SEC	37	30	117.5	89.8	0.9568	53	46.6	18.6	17.9	36.8	25.5	41.9	26.8	52.9	43.6	35.4	34.3	66.9	8.8	2019	8	8
Michigan St.	B10	39	32	119.9	91	0.9597	55.2	43.9	18.5	14.9	33.9	26.4	33.6	27.5	54.3	41.9	37.8	31.6	68.6	10.7	2019	4	8
Michigan	B10	37	30	114.6	85.6	0.9665	51.6	44.1	13.9	18	24.7	24.8	27.5	24.1	51.8	44.3	34.2	29.1	65.9	9.2	2019	16	8
Tennessee	SEC	36	31	122.8	95.2	0.9488	55.3	48.1	15.8	18	31.6	30.2	33.3	34.9	55.4	44.7	36.7	35.4	68.8	9.9	2019	16	8
Texas Tech	B12	38	31	115.2	85.2	0.9696	53.5	43	17.7	22.8	27.4	28.7	32.9	36.6	52.8	41.9	36.5	29.7	67.5	7	2019	2	16
Purdue	B10	36	26	122.8	94.3	0.9539	53.6	49	15.8	18.6	34.5	27	29.9	31.7	51.5	47.2	37.4	34.2	67	6.1	2019	8	16

... (58 rows omitted)

Linear Regression and determing correlation between *Standard SEED* and *ADJOE.*

Adjusted Offensive Efficiency (ADJOE) = Points Scored per 100 possessions, or trips down the floor with the basketball, against an average Division I opponent.

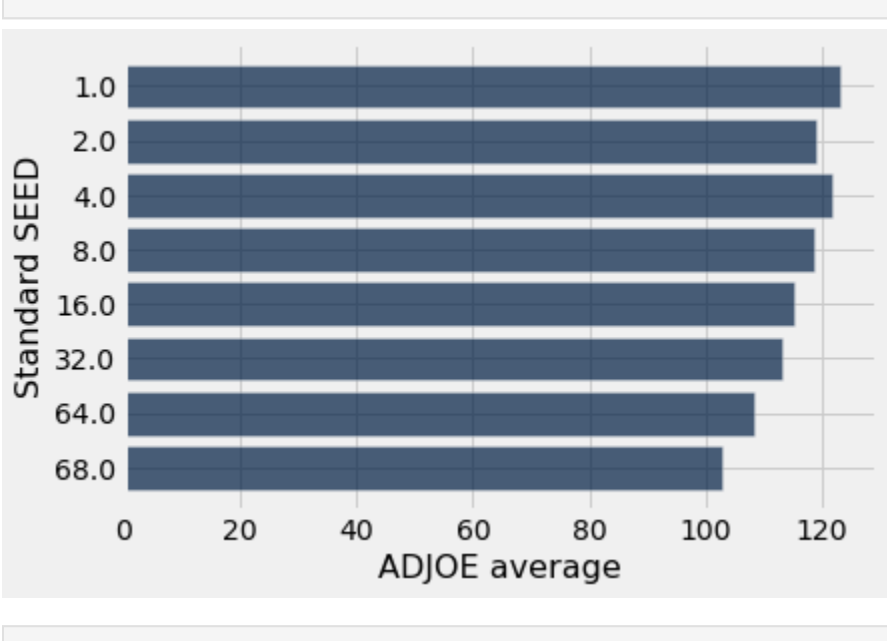
In [9]:
```
def standard_units(x):
    """Converts an array of numbers to standard units"""
    return (x - np.average(x))/np.std(x)
```

In [10]:
```
def correlation(t, x, y):
    """t is a table, x and y are column labels"""
    x_in_standard_units = standard_units(t.column(x))
    y_in_standard_units = standard_units(t.column(y))
    return np.average(x_in_standard_units * y_in_standard_units)
```

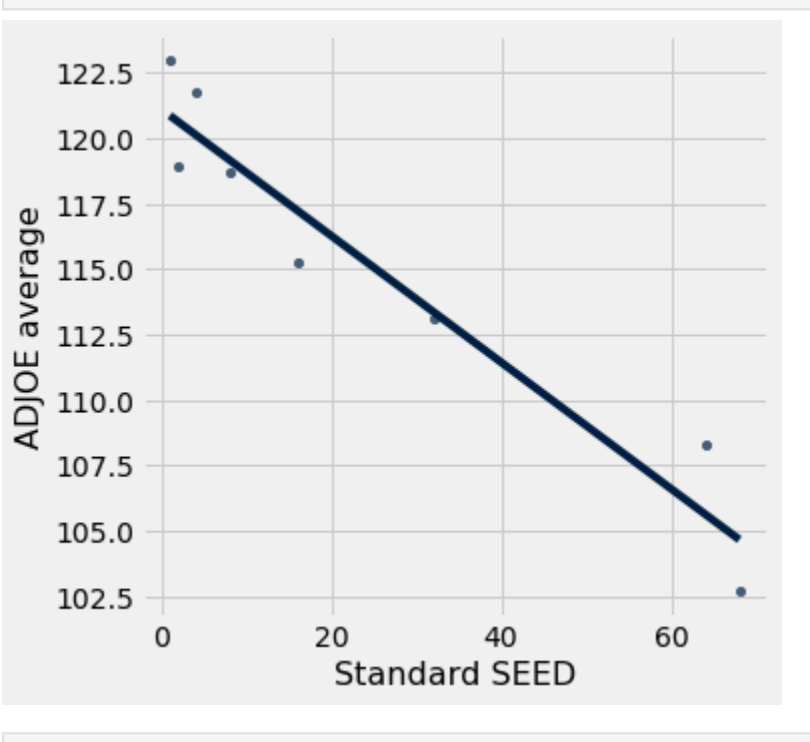
In [11]:
```
def slope(t, x, y):
    r = correlation(t, x, y)
    y_sd = np.std(t.column(y))
    x_sd = np.std(t.column(x))
    return r * y_sd / x_sd

def intercept(t, x, y):
    x_mean = np.mean(t.column(x))
    y_mean = np.mean(t.column(y))
    return y_mean - slope(t, x, y)*x_mean
```

In [12]:
```
def prediction_at(t, x, y, x_value):
    '''
    t - table
    x - label of x column
    y - label of y column
    x_value - the x value for which we want to predict y
    '''
    return slope(t, x, y) * x_value + intercept(t, x, y)
```

In [13]:
```
st_seed_ADJOE = mm_2019.group("Standard SEED", np.average).select("Standard SEED", "ADJOE average")
st_seed_ADJOE.barh("Standard SEED", "ADJOE average")
```



In [14]:
```
st_seed_ADJOE.scatter("Standard SEED", "ADJOE average", fit_line = True)
```



In [15]:
```
correlation(st_seed_ADJOE, "Standard SEED", "ADJOE average")
```

Out[15]: -0.9616538709786773

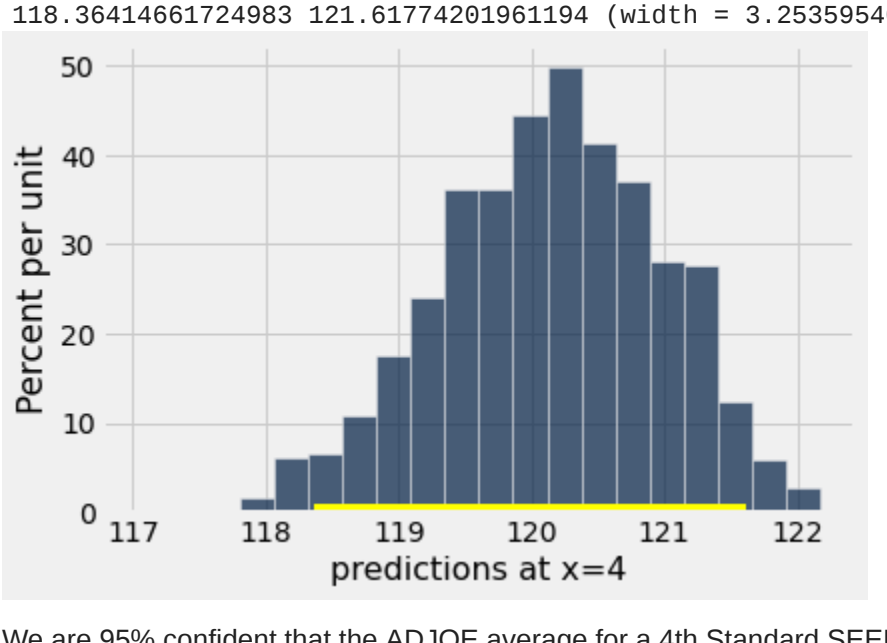
There is a strong negative association between Standard SEED and ADJOE average.

Boostrap Prediction and confidence intervals for *Standard SEED* and *ADJOE average*.

In [16]:
```
def bootstrap_prediction(t, x, y, new_x, repetitions=1000):

    # Bootstrap the scatter, predict, collect
    predictions = make_array()

    for i in np.arange(repetitions):
        bootstrap_sample = t.sample()
        prediction = prediction_at(bootstrap_sample, x, y, new_x)
        predictions = np.append(predictions, prediction)

    # Find the ends of the approximate 95% prediction interval
    left = percentile(2.5, predictions)
    right = percentile(97.5, predictions)

    # Display results
    Table().with_column('Prediction', predictions).hist(bins=20)
    plots.xlabel('predictions at x='+str(new_x))
    plots.plot([left, right], [0, 0], color='yellow', lw=8);
    print('Approximate 95%-confidence interval for height of true line:')
    print(left, right, '(width =', right - left, ')')
```

In [17]:
```
bootstrap_prediction(st_seed_ADJOE, "Standard SEED", "ADJOE average", 4)
```

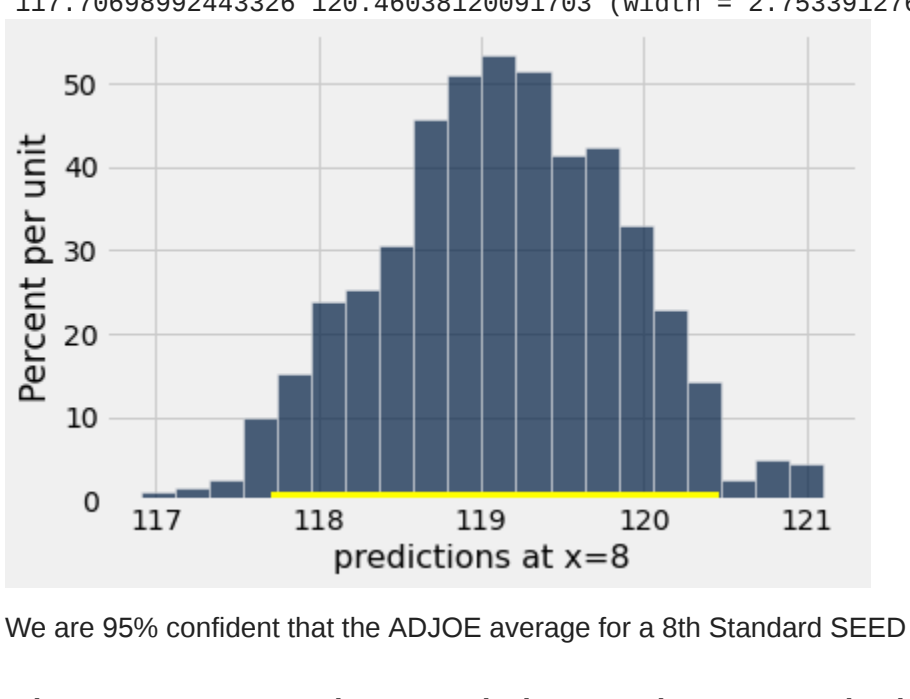
```
Approximate 95%-confidence interval for height of true line:
118.36414661724983 121.61774201961194 (width = 3.2535954023621088 )
```



We are 95% confident that the ADJOE average for a 4th Standard SEED team is between 118.36 and 121.62.

In [18]:
```
bootstrap_prediction(st_seed_ADJOE, "Standard SEED", "ADJOE average", 8)
```

```
Approximate 95%-confidence interval for height of true line:
117.70698992443326 120.46038120091703 (width = 2.753391276483768 )
```

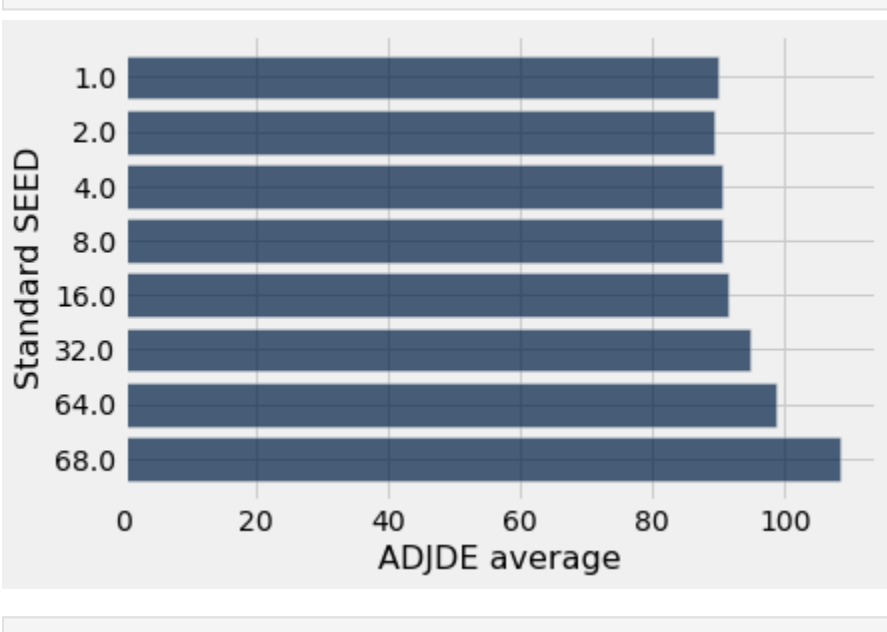


We are 95% confident that the ADJOE average for a 8th Standard SEED team is between 117.71 and 120.46.

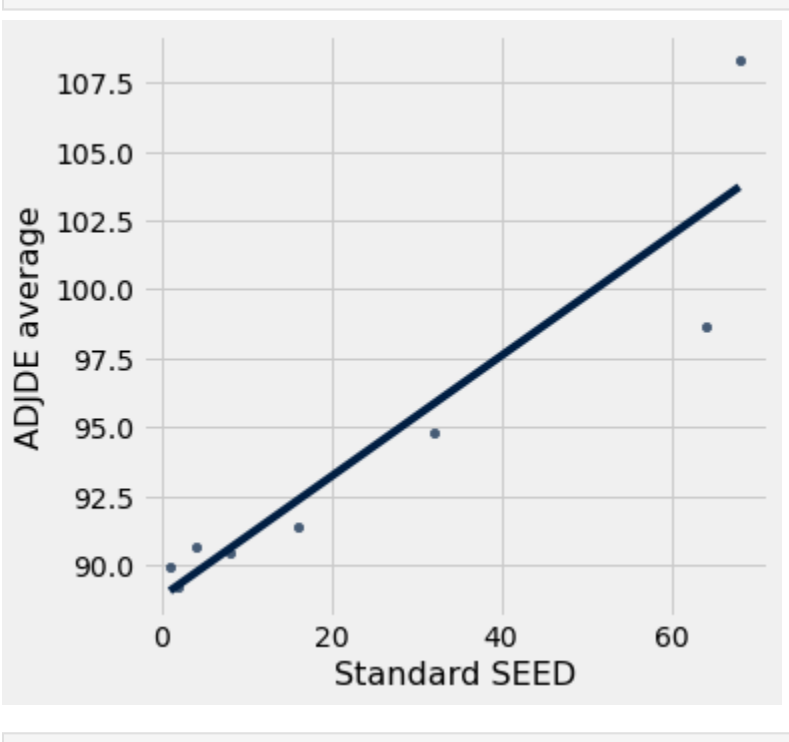
Linear Regression and determing correlation between *Standard SEED* and *ADJDE average.*

Adjusted Defensive Efficiency (ADJDE) = Points Allowed per 100 possessions against an average Division I opponent.

In [19]:
```
st_seed_ADJDE = mm_2019.group("Standard SEED", np.average).select("Standard SEED", "ADJDE average")
st_seed_ADJDE.barh("Standard SEED", "ADJDE average")
```



In [20]:
```
st_seed_ADJDE.scatter("Standard SEED", "ADJDE average", fit_line = True)
```



In [21]:
```
correlation(st_seed_ADJDE, "Standard SEED", "ADJDE average")
```

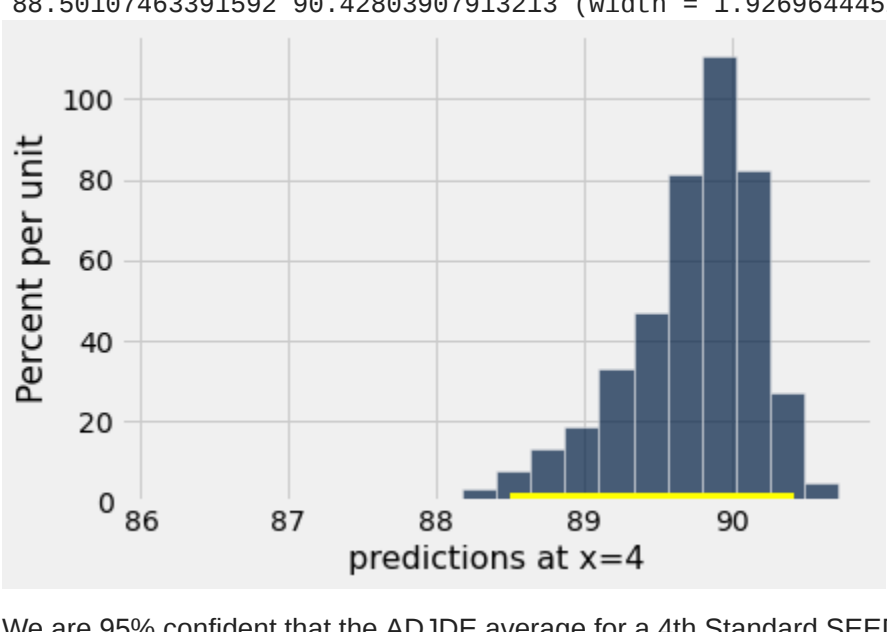
Out[21]: 0.9261184488219529

There is a strong positive association between Standard SEED and ADJDE average.

Boostrap Prediction and confidence intervals for *Standard SEED* and *ADJDE average*.

In [22]:
```
bootstrap_prediction(st_seed_ADJDE, "Standard SEED", "ADJDE average", 4)
```

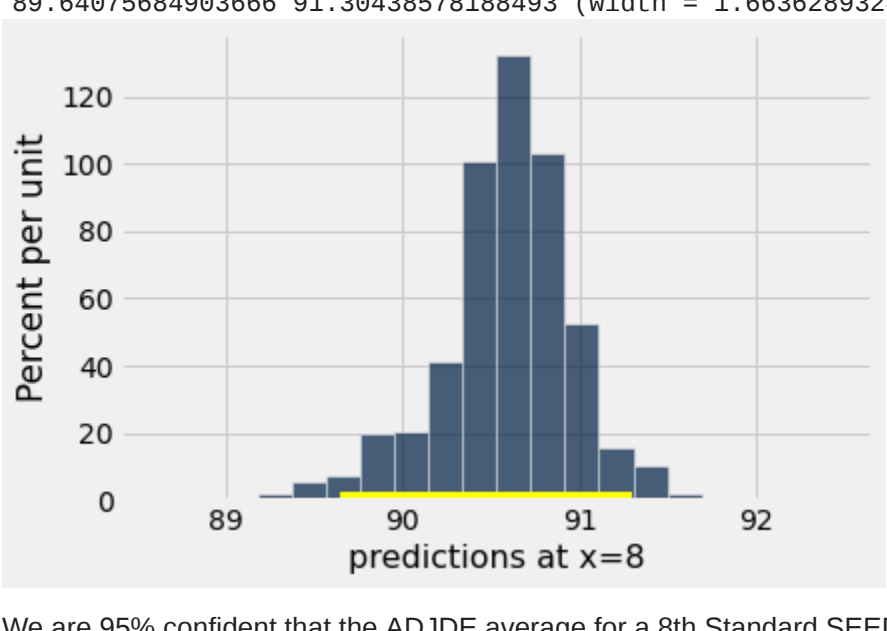
```
Approximate 95%-confidence interval for height of true line:
88.50107463391592 90.42803907913213 (width = 1.926964445216214 )
```



We are 95% confident that the ADJDE average for a 4th Standard SEED team is between 88.50 and 90.43.

In [33]:
```
bootstrap_prediction(st_seed_ADJDE, "Standard SEED", "ADJDE average", 8)
```

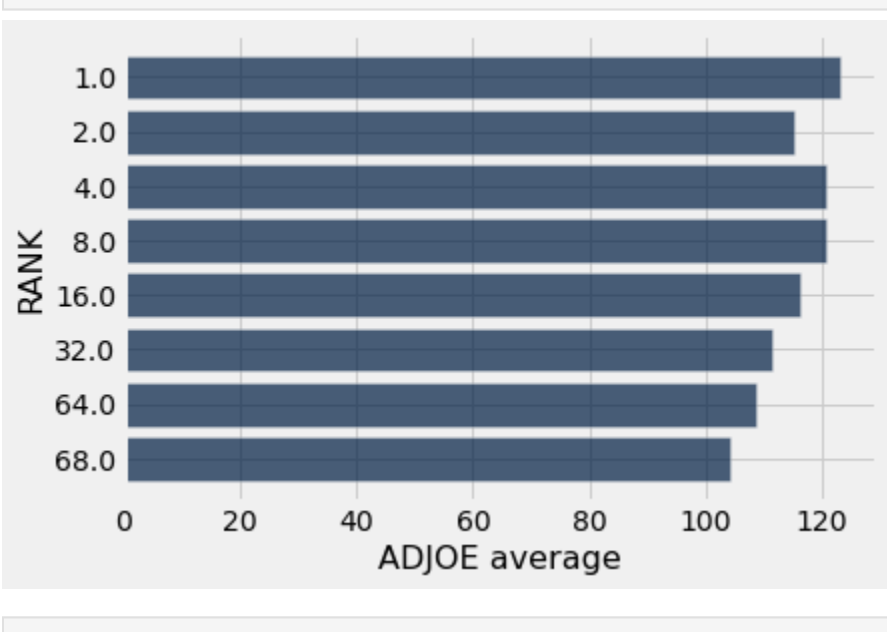
```
Approximate 95%-confidence interval for height of true line:
89.64075684903666 91.30438578188493 (width = 1.663628932848269 )
```



We are 95% confident that the ADJDE average for a 8th Standard SEED team is between 89.64 and 91.30.

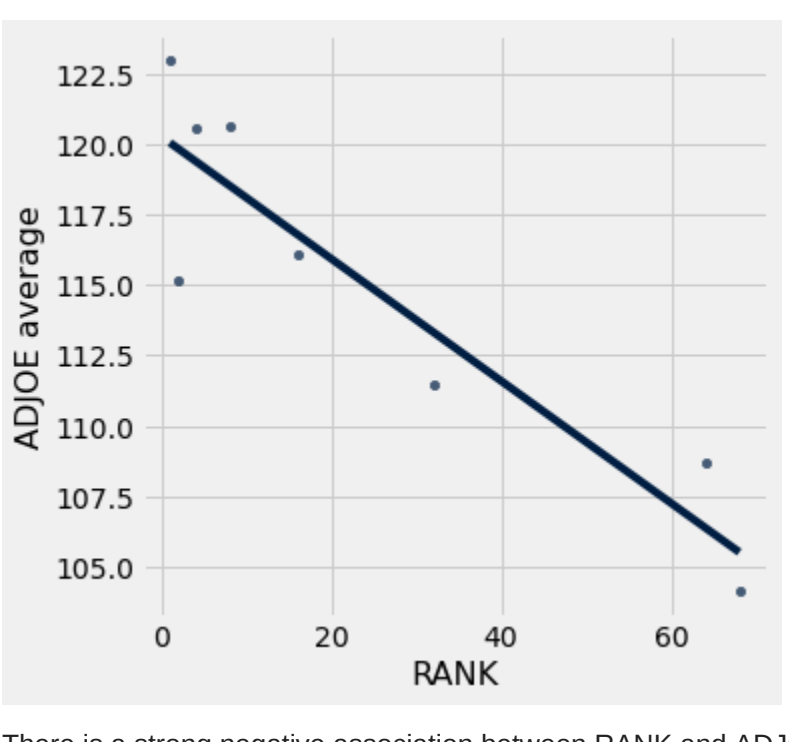
Linear Regression and determing correlation between *RANK* and *ADJOE average.*

In [24]:
```
rank_ADJOE = mm_2019.group("RANK", np.average).select("RANK", "ADJOE average")
rank_ADJOE.barh("RANK", "ADJOE average")
```



In [25]:
```
rank_ADJOE.scatter("RANK", "ADJOE average", fit_line = True)
correlation(rank_ADJOE, "RANK", "ADJOE average")
```

Out[25]: -0.9176629845352036

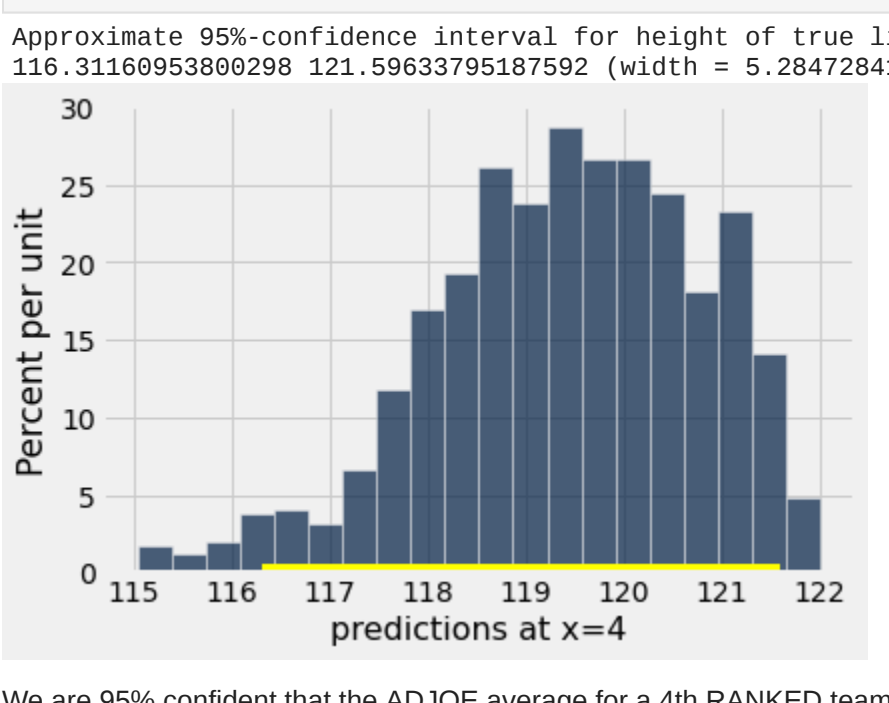


There is a strong negative association between RANK and ADJOE average.

Boostrap Prediction and confidence intervals for *RANK* and *ADJOE average*.

In [26]:
```
bootstrap_prediction(rank_ADJOE, "RANK", "ADJOE average", 4)
```

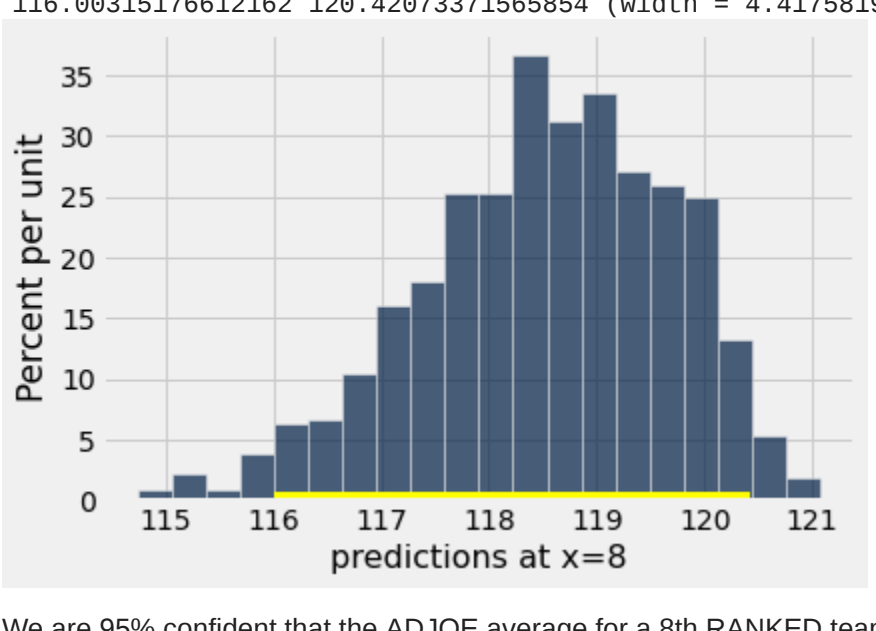
```
Approximate 95%-confidence interval for height of true line:
116.31160953800298 121.59633795187592 (width = 5.284728413872941 )
```



We are 95% confident that the ADJOE average for a 4th RANKED team is between 116.31 and 121.60.

In [27]:
```
bootstrap_prediction(rank_ADJOE, "RANK", "ADJOE average", 8)
```

```
Approximate 95%-confidence interval for height of true line:
116.00315176612162 120.42073371565854 (width = 4.417581949536924 )
```

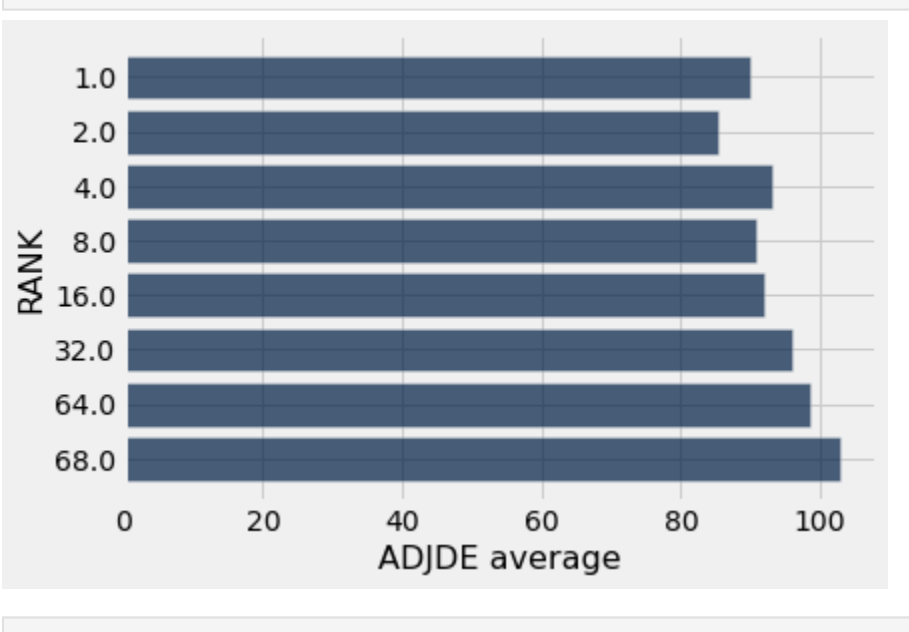


We are 95% confident that the ADJOE average for a 8th RANKED team is between 116.00 and 120.42.

Conclusion: Through the correlation coefficient, we find that there is a strong association between *Standard SEED* and *ADJOE average* (-0.9616538709786773) and *RANK* and *ADJOE average* (-0.9176629845352036). Additionally, from our bootstraping prediction, we find that the 95% confidence interval showing where *ADJOE average* is located for a 4th *Standard SEED* team [118.36, 121.62] experiences much overlap with the 95% confidence interval showing where *ADJOE average* is located for a 4th RANKED team [116.31, 121.60]. We also find that the 95% confidence interval showing where *ADJOE average* is located for a 8th *Standard SEED* team [117.71, 120.46] experiences much overlap with the 95% confidence interval showing where *ADJOE average* is located for a 8th RANKED team [116.00, 120.42]. In both the scenarios, the NCAA Seed (converted to *Standard SEED*), which is heavily correlated with ADJOE average, is effective at predicting a team's *RANK*.

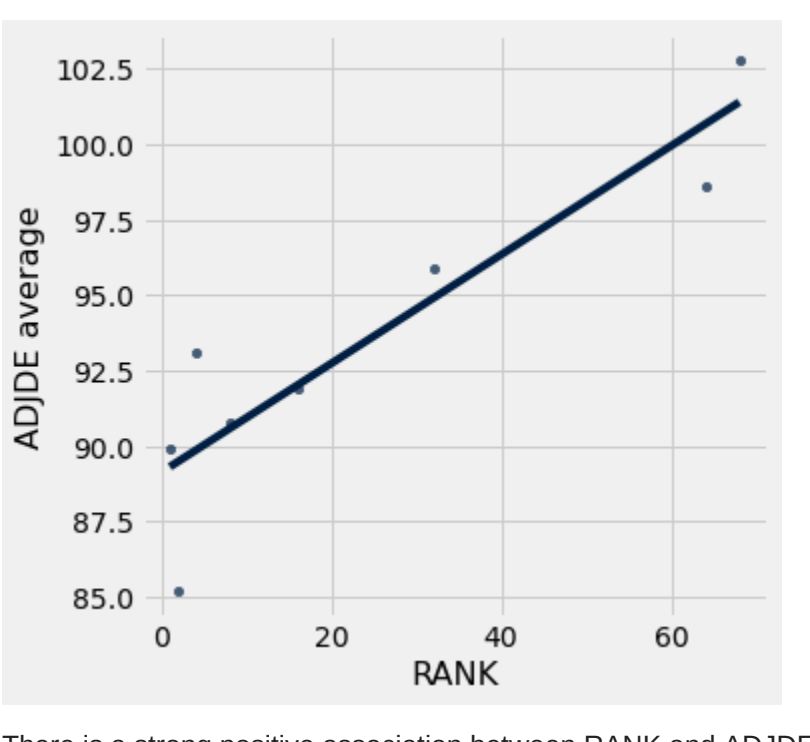
Linear Regression and determing correlation between *RANK* and *ADJDE average.*

In [28]:
```
rank_ADJDE = mm_2019.group("RANK", np.average).select("RANK", "ADJDE average")
rank_ADJDE.barh("RANK", "ADJDE average")
```



In [29]:
```
rank_ADJDE.scatter("RANK", "ADJDE average", fit_line = True)
correlation(rank_ADJDE, "RANK", "ADJDE average")
```

Out[29]: 0.9087902125187143

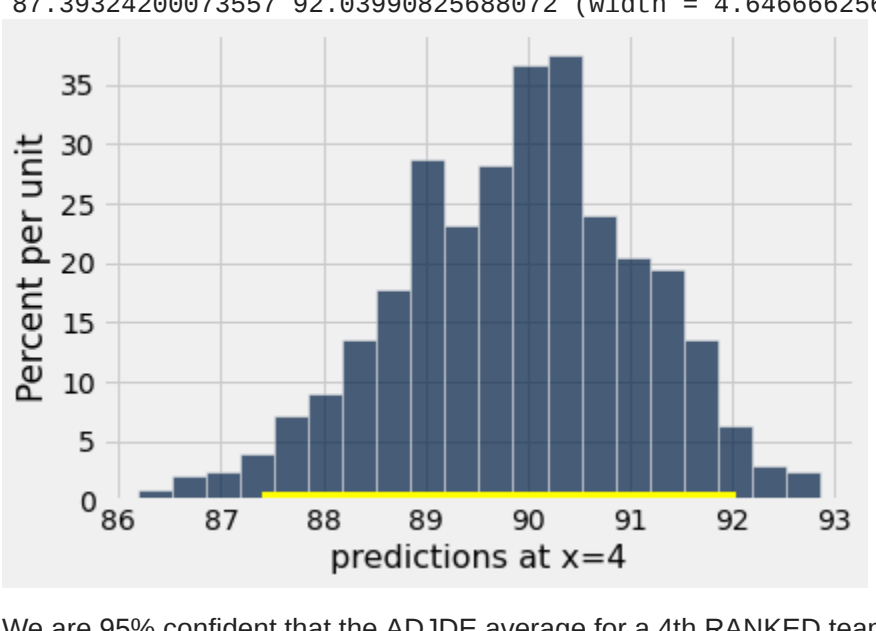


There is a strong positive association between RANK and ADJDE average.

Boostrap Prediction and confidence intervals for *RANK* and *ADJDE average*.

In [30]:
```
bootstrap_prediction(rank_ADJDE, "RANK", "ADJDE average", 4)
```

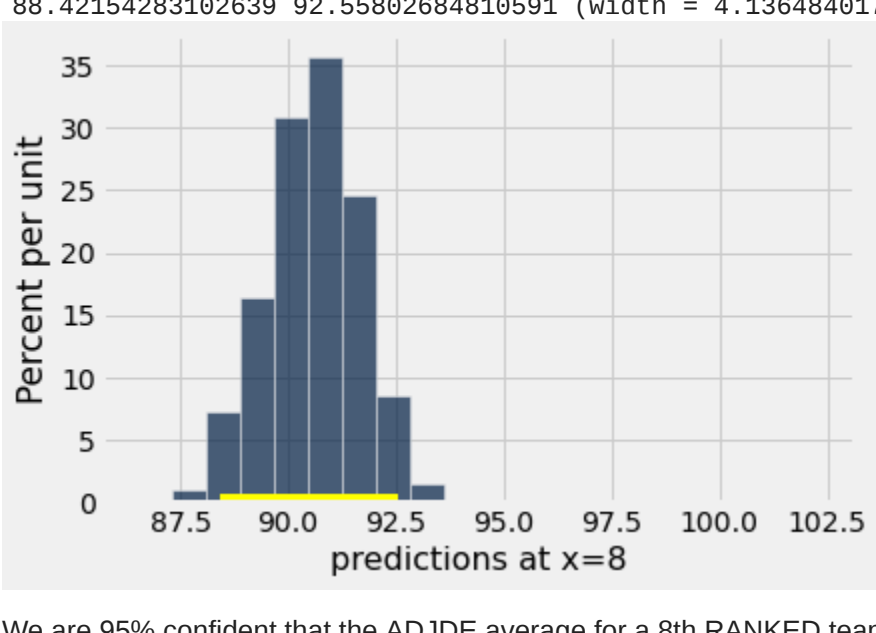
```
Approximate 95%-confidence interval for height of true line:
87.39324200073557 92.03990825688072 (width = 4.646666256145153 )
```



We are 95% confident that the ADJDE average for a 4th RANKED team is between 87.39 and 92.04.

In [31]:
```
bootstrap_prediction(rank_ADJDE, "RANK", "ADJDE average", 8)
```

```
Approximate 95%-confidence interval for height of true line:
88.42154283102639 92.55802684810591 (width = 4.136484017079525 )
```



We are 95% confident that the ADJDE average for a 8th RANKED team is between 88.42 and 92.56.

Conclusion: Through the correlation coefficient, we find that there is a strong association between *Standard SEED* and *ADJDE average* (0.9261184488219529) and *RANK* and *ADJDE average* (0.9087902125187143). Additionally, from our bootstraping prediction, we find that the 95% confidence interval showing where *ADJDE average* is located for a 4th *Standard SEED* team [88.50, 90.43] experiences much overlap with the 95% confidence interval showing where *ADJDE average* is located for a 4th RANKED team [87.39, 92.04]. We also find that the 95% confidence interval showing where *ADJDE average* is located for a 8th *Standard SEED* team [89.64, 91.30] experiences much overlap with the 95% confidence interval showing where *ADJDE average* is located for a 8th RANKED team [88.42, 92.56]. In both the scenarios, the NCAA Seed (converted to *Standard SEED*), which is heavily correlated with ADJDE average, is effective at predicting a team's *RANK*.

Creating an algorithm to predict which *RANK* a team will place into.

Will this algorithm be more effective at predicting a team's *RANK* than *Standard SEED*?

In [34]:
```
# Predicting the rank of a team using ADJOE and ADJDE seperately.
ADJOE_sorted = mm_2019.sort("ADJOE", descending = True).column("TEAM")
ADJDE_sorted = mm_2019.sort("ADJDE").column("TEAM")

new_ADJOErank_column = make_array()
new_ADJDErank_column = make_array()

for i in np.arange(mm_2019.num_rows):
        new_ADJOErank_column = np.append(new_ADJOErank_column, i+1)
for i in np.arange(mm_2019.num_rows):
        new_ADJDErank_column = np.append(new_ADJDErank_column, i+1)

new_ADJOE_table = mm_2019.sort("ADJOE", descending = True).with_column("New Rank", new_ADJOErank_column).select("TEAM", "RANK", "New Rank")
new_ADJDE_table = mm_2019.sort("ADJDE").with_column("New Rank", new_ADJDErank_column).select("TEAM", "RANK", "New Rank")
```

In [35]:
```
new_ADJOE_table.show(4)
```

TEAM	RANK	New Rank
Gonzaga	8	1
Virginia	1	2
Tennessee	16	3
Purdue	8	4

... (64 rows omitted)

In [36]:
```
new_ADJDE_table.show(4)
```

TEAM	RANK	New Rank
Texas Tech	2	1
Michigan	16	2
VCU	64	3
Kansas St.	64	4

... (64 rows omitted)

In [37]:
```
# Predicting the rank of a team using the average of their New Ranks, based on ADJOE and ADJDE.
raw_predicted_rank = make_array()

for i in new_ADJOE_table.column("TEAM"):
    raw_predicted_rank = np.append(raw_predicted_rank, np.mean(
        [new_ADJOE_table.where("TEAM", i).column("New Rank")[0], new_ADJDE_table.where("TEAM", i).column("New Rank")[0]]))
```

In [38]:
```
rank_prediction_table = new_ADJOE_table.with_column("Raw Predicted Rank", raw_predicted_rank).drop("New Rank").sort("Raw Predicted Rank")
rank_prediction_table
```

Out[38]:

TEAM	RANK	Raw Predicted Rank
Gonzaga	8	5
Virginia	1	5
Duke	8	7
Texas Tech	2	9
North Carolina	16	9.5
Michigan St.	4	9.5
Kentucky	8	9.5
Michigan	16	10.5
Houston	16	13
Purdue	8	15

... (58 rows omitted)

In [39]:
```
# Limiting the accepted values for Predicted Rank to be 1, 2, 4, 8, 16, 32, 64, and 68.
predicted_rank = make_array()
x: int = 0

for i in np.arange(rank_prediction_table.num_rows):
    if i == 0:
        predicted_rank = np.append(predicted_rank, 1)
    if i == 1:
        predicted_rank = np.append(predicted_rank, 2)
    if x >= 2 and x < 4:
        predicted_rank = np.append(predicted_rank, 4)
    if x >= 4 and x < 8:
        predicted_rank = np.append(predicted_rank, 8)
    if x >= 8 and x < 16:
        predicted_rank = np.append(predicted_rank, 16)
    if x >= 16 and x < 32:
        predicted_rank = np.append(predicted_rank, 32)
    if x >= 32 and x < 64:
        predicted_rank = np.append(predicted_rank, 64)
    if x >= 64 and x < 68:
        predicted_rank = np.append(predicted_rank, 68)
    x = x + 1
```

In [40]:
```
prediction_table = rank_prediction_table.with_column("Predicted Rank", predicted_rank).drop("Raw Predicted Rank")
prediction_table
```

Out[40]:

TEAM	RANK	Predicted Rank
Gonzaga	8	1
Virginia	1	2
Duke	8	4
Texas Tech	2	4
North Carolina	16	8
Michigan St.	4	8
Kentucky	8	8
Michigan	16	8
Houston	16	16
Purdue	8	16

... (58 rows omitted)

In [41]:
```
# Limiting the accepted values for SEED to be 1, 2, 4, 8, 16, 32, 64, and 68.
NCAA_prediction_table = mm_2019.select("TEAM", "RANK", "Standard SEED")
NCAA_prediction_table
```

Out[41]:

TEAM	RANK	Standard SEED
Virginia	1	1
Duke	8	2
Gonzaga	8	4
North Carolina	16	4
Kentucky	8	8
Michigan St.	4	8
Michigan	16	8
Tennessee	16	8
Texas Tech	2	16
Purdue	8	16

... (58 rows omitted)

In [42]:
```
def distance(point1, point2):
    """Returns the distance between point1 and point2
    where each argument is an array
    consisting of the coordinates of the point"""
    return np.sqrt(np.sum((point1 - point2)**2))
```

In [43]:
```
distance(prediction_table.column("Predicted Rank"), prediction_table.column("RANK"))
```

Out[43]: 144.79640879524604

In [44]:
```
distance(NCAA_prediction_table.column("Standard SEED"), NCAA_prediction_table.column("RANK"))
```

Out[44]: 169.82343772283025

In [45]:
```
distance(prediction_table.column("Predicted Rank"), prediction_table.column("RANK")) / distance(NCAA_prediction_table.column("Standard SEED"), NCAA
```

Out[45]: 0.8526291231459407

In [46]:
```
def check_matchingvalues(tbl, col1, col2):
    x: int = 0
    for i in np.arange(tbl.num_rows):
        if tbl.column(col1)[i] == tbl.column(col2)[i]:
            x = x + 1
        else:
            x = x
    return x
```

In [47]:
```
check_matchingvalues(prediction_table, "Predicted Rank", "RANK")
```

Out[47]: 34

In [48]:
```
check_matchingvalues(NCAA_prediction_table, "Standard SEED", "RANK")
```

Out[48]: 30

Conclusion: Using the algorithm I created to calculate *Predicted Rank*, I found that the sum of the distances between *Predicted Rank* and *RANK* for every team in prediction_table (144.79640879524604) was less than the sum of the distances between *Standard SEED* and *RANK* for every team in NCAA_prediction_table (169.82343772283025). In addition, the matching values between *Predicted Rank* and *RANK* exceeded the matching values between *Standard SEED* and *RANK* by 4 values. Thus, the algorithm shown above to calculate *Predicted Rank* based on the average of a team's ADJOE and ADJDE rankings was slightly more accurate at predicting a team's *RANK* than the official NCAA seed.