In [27]:
```
from __future__ import print_function
import time
import cfbd
from cfbd.rest import ApiException
from pprint import pprint
import pprint as pp
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from adjustText import adjust_text
import numpy as np
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
from scipy.stats import linregress
```

In [28]:
```
year=2021
season_type = 'regular'
team = 'Oregon'
```

In [29]:
```
configuration = cfbd.Configuration()
configuration.api_key['Authorization'] = 'mtU85YjFNgHylNwj/RowQ+Tffwg65fhiZ
configuration.api_key_prefix['Authorization'] = 'Bearer'
```

In [30]:
```
api_instance = cfbd.MetricsApi(cfbd.ApiClient(configuration))
```

In [31]:
```
# create an instance of the API class
#api_instance = cfbd.PlayerGamePPA(cfbd.ApiClient(configuration))
season = 2021 # int | Year filter

team = 'Oregon' # str | Team filter (optional)
#conference = 'conference_example' # str | Conference filter (optional)
exclude_garbage_time = True # bool | Filter to remove garbage time plays fr
season_type = 'regular' # str | Season type filter (regular or postseason)

api_instance = cfbd.StatsApi(cfbd.ApiClient(configuration))

try:
    # Team stat categories
    api_response = api_instance.get_stat_categories()
    pprint(api_response)
except ApiException as e:
    print("Exception when calling StatsApi->get_stat_categories: %s\n" % e)
```

```
['completionAttempts',
 'defensiveTDs',
 'extraPoints',
 'fieldGoalPct',
 'fieldGoals',
 'firstDowns',
 'fourthDownEff',
 'fumblesLost',
 'fumblesRecovered',
 'interceptions',
 'interceptionTDs',
 'interceptionYards',
 'kickingPoints',
 'kickReturns',
 'kickReturnTDs',
 'kickReturnYards',
 'netPassingYards',
 'passesDeflected',
 'passesIntercepted',
 'passingTDs',
 'possessionTime',
 'puntReturns',
 'puntReturnTDs',
 'puntReturnYards',
 'qbHurries',
 'rushingAttempts',
 'rushingTDs',
 'rushingYards',
 'sacks',
 'tackles',
 'tacklesForLoss',
 'thirdDownEff',
 'totalFumbles',
 'totalPenaltiesYards',
 'totalYards',
 'turnovers',
 'yardsPerPass',
 'yardsPerRushAttempt']
```

PAC 12 Comparison

In [44]:
```
### Returning production ####
configuration = cfbd.Configuration()
configuration.api_key['Authorization'] = 'mtU85YjFNgHylNwj/RowQ+Tffwg65fhiZ
configuration.api_key_prefix['Authorization'] = 'Bearer'
api_instance = cfbd.PlaysApi(cfbd.ApiClient(configuration))
```

In [45]:
```
year = 2021
season_type = 'regular'
week = 1

try:
    # Play by play data
    api_response = api_instance.get_plays(year, week, season_type=season_ty
    pprint(api_response)
except ApiException as e:
    print("Exception when calling PlaysApi->get_plays: %s\n" % e)
```

```
[{'away': 'Alabama',
 'clock': {'minutes': 15, 'seconds': 0},
 'defense': 'Alabama',
 'defense_conference': 'SEC',
 'defense_score': 0,
 'defense_timeouts': 3,
 'distance': 10,
 'down': 1,
 'drive_id': 4012819421,
 'drive_number': 1,
 'game_id': 401281942,
 'home': 'Miami',
 'id': 401281942101849902,
 'offense': 'Miami',
 'offense_conference': 'ACC',
 'offense_score': 0,
 'offense_timeouts': 3,
 'period': 1,
 'play_number': 1,
```

In [46]:
```
pbp = pd.DataFrame.from_records([d.to_dict() for d in api_response])
```

In [47]:
```
year = 2021
season_type = 'regular'
week = 2

try:
    # Play by play data
    api_response = api_instance.get_plays(year, week, season_type=season_ty
    pprint(api_response)
except ApiException as e:
    print("Exception when calling PlaysApi->get_plays: %s\n" % e)
```

```
[{'away': 'Mercer',
 'clock': {'minutes': 15, 'seconds': 0},
 'defense': 'Alabama',
 'defense_conference': 'SEC',
 'defense_score': 0,
 'defense_timeouts': 3,
 'distance': 10,
 'down': 1,
 'drive_id': 4012820561,
 'drive_number': 1,
 'game_id': 401282056,
 'home': 'Alabama',
 'id': 401282056101849902,
 'offense': 'Mercer',
 'offense_conference': None,
 'offense_score': 0,
 'offense_timeouts': 3,
 'period': 1,
 'play_number': 1,
```

In [48]:
```
pbp1 = pd.DataFrame.from_records([d.to_dict() for d in api_response])
```

In [49]:
```
year = 2021
season_type = 'regular'
week = 3

try:
    # Play by play data
    api_response = api_instance.get_plays(year, week, season_type=season_ty
    pprint(api_response)
except ApiException as e:
    print("Exception when calling PlaysApi->get_plays: %s\n" % e)
```

```
[{'away': 'Georgia Southern',
 'clock': {'minutes': 15, 'seconds': 0},
 'defense': 'Georgia Southern',
 'defense_conference': 'Sun Belt',
 'defense_score': 0,
 'defense_timeouts': 3,
 'distance': 10,
 'down': 1,
 'drive_id': 4012820691,
 'drive_number': 1,
 'game_id': 401282069,
 'home': 'Arkansas',
 'id': 401282069101849902,
 'offense': 'Arkansas',
 'offense_conference': 'SEC',
 'offense_score': 0,
 'offense_timeouts': 3,
 'period': 1,
 'play_number': 1,
```

In [50]:
```
pbp2 = pd.DataFrame.from_records([d.to_dict() for d in api_response])
```

In [51]:
```
year = 2021
season_type = 'regular'
week = 4

try:
    # Play by play data
    api_response = api_instance.get_plays(year, week, season_type=season_ty
    pprint(api_response)
except ApiException as e:
    print("Exception when calling PlaysApi->get_plays: %s\n" % e)
```

```
[{'away': 'Southern Mississippi',
 'clock': {'minutes': 14, 'seconds': 46},
 'defense': 'Alabama',
 'defense_conference': 'SEC',
 'defense_score': 7,
 'defense_timeouts': 3,
 'distance': 10,
 'down': 1,
 'drive_id': 4012820811,
 'drive_number': 1,
 'game_id': 401282081,
 'home': 'Alabama',
 'id': 401282081101855301,
 'offense': 'Southern Mississippi',
 'offense_conference': 'Conference USA',
 'offense_score': 0,
 'offense_timeouts': 3,
 'period': 1,
 'play_number': 1,
```

In [52]:
```
pbp3 = pd.DataFrame.from_records([d.to_dict() for d in api_response])
```

In [53]:
```
frames = [pbp,pbp1,pbp2,pbp3]
```

In [54]:
```
result = pd.concat(frames)
```

In [55]:
```
result.columns
```

Out[55]:
```
Index(['id', 'drive_id', 'game_id', 'drive_number', 'play_number', 'offen
se',
       'offense_conference', 'offense_score', 'defense', 'home', 'away',
       'defense_conference', 'defense_score', 'period', 'clock',
       'offense_timeouts', 'defense_timeouts', 'yard_line', 'yards_to_goa
l',
       'down', 'distance', 'yards_gained', 'scoring', 'play_type', 'play_
text',
       'ppa', 'wallclock'],
      dtype='object')
```

In [56]:
```
pp.pprint(result.play_type.unique().tolist())
```

```
['Kickoff',
 'Rush',
 'Pass Incompletion',
 'Pass Reception',
 'Passing Touchdown',
 'Penalty',
 'Punt',
 'Field Goal Good',
 'End Period',
 'Sack',
 'Fumble Recovery (Own)',
 'Timeout',
 'End of Half',
 'Pass Interception Return',
 'Rushing Touchdown',
 'End of Game',
 'Kickoff Return (Offense)',
 'Field Goal Missed',
 'Blocked Punt',
 'Interception',
 'End of Regulation',
 'Safety',
 'Interception Return Touchdown',
 'Fumble Recovery (Opponent)',
 'Blocked Field Goal',
 'Missed Field Goal Return',
 'Fumble Return Touchdown',
 'Blocked Punt Touchdown',
 'Blocked Field Goal Touchdown',
 'Kickoff Return Touchdown',
 'placeholder',
 'Punt Return Touchdown',
 'Defensive 2pt Conversion',
 'Uncategorized']
```

In [57]:
```
off_play_types = ['Pass Reception',
 'Rush',
 'Sack',
 'Pass Incompletion',
 'Fumble Recovery (Opponent)',
 'Passing Touchdown',
 'Rushing Touchdown',
 'Pass Interception Return',
 'Fumble Recovery (Own)',
 'Interception Return Touchdown',
 'Fumble Return Touchdown']

off_plays = result[result['play_type'].isin(off_play_types)].copy()
excl_plays = result[~result['play_type'].isin(off_play_types)].copy()
```

In [58]:
```
# Add flag for run or pass
off_plays.loc[(off_plays['play_type'].str.contains('Pass')), 'pass'] = 1
```

In [59]:
```
off_plays.loc[(off_plays['play_type'].str.contains('Interception')), 'pass'
```

In [60]:
```
off_plays.loc[(off_plays['play_type'].str.contains('Sack')), 'pass'] = 1
```

In [61]:
```
off_plays.loc[(off_plays['play_type'].str.contains('Rush')), 'rush'] = 1
```

In [62]:
```
off_plays.loc[(off_plays['play_type'].str.contains('Interception')), 'yards
```

In [63]:
```
flags_na = {'rush':0, 'pass':0}
```

In [64]:
```
off_plays.fillna(value=flags_na, inplace=True)
```

In [65]:
```
plays = off_plays.groupby('offense').agg({'yards_gained':'count'})
plays.rename(columns={'yards_gained':'Plays'}, inplace=True)
```

In [66]:
```
attempts = off_plays[(off_plays['pass'] == 1) & (off_plays['play_type'] !=
ypa = attempts.groupby('offense').agg({'yards_gained':'mean'})
ypa.rename(columns={'yards_gained':'YPA'}, inplace=True)
```

In [67]:
```
rushes = off_plays[(off_plays['rush'] == 1)]
ypc = rushes.groupby('offense').agg({'yards_gained':'mean'})
ypc.rename(columns={'yards_gained':'YPC'}, inplace=True)
```

In [68]:
```
offense = plays.merge(ypa, left_index= True,right_index= True).merge(ypc, l
offense = offense[offense['Plays'] > 50].copy()

#50 given season
```

In [69]:
```
def calcs(x):
      names = {'Plays': x['yards_gained'].count(),
               'YPA': x[(x['pass'] == 1) & (x['play_type']!= 'Sack')]['yard
               'YPC' :x[x['rush'] == 1]['yards_gained'].mean()
                }
      return pd.Series(names)

offense =  off_plays.groupby('offense').apply(calcs)

# Filter out FCS Games

offense = offense[offense['Plays'] > 50].copy()
```

In [70]:
```
off=offense.sort_values(by='YPA', ascending=False)
```

In [71]:
```
off['YPA_rank']=off['YPA'].rank(ascending=False)
```

In [72]:
```
off['YPC_rank']=off['YPC'].rank(ascending=False)
```

In [73]:
```
#pd.set_option("max_rows",None)
off_df=pd.DataFrame(data=off,columns=['Plays','YPA','YPC','YPA_rank','YPC_r
```

In [74]:
```
off_df['team'] = off_df.index
```

In [75]:
```
teams = ['Fresno State','Ohio State','Stony Brook','Arizona','Stanford','Ca
o_top_df = off_df[off_df.team.isin(teams)]
```

In [76]:
```
o_top_df = o_top_df.round({'Plays':0,'YPA':1,'YPC':1,'YPA_rank':0,'YPC_rank
```

In [77]:
```
o_top_df = o_top_df.drop(columns=['team'])
```

In [78]:
```
o_top_df['Plays'] = o_top_df['Plays'].astype(int)
```

In [79]:
```
o_top_df['YPA_rank'] = o_top_df['YPA_rank'].astype(int)
```

In [80]:
```
o_top_df['YPC_rank'] = o_top_df['YPC_rank'].astype(int)
```

In [81]:
```
o_top_df
```

Out[81]:

offense	Plays	YPA	YPC	YPA_rank	YPC_rank
Ohio State	259	10.5	7.3	10	5
Fresno State	373	9.8	4.6	17	127
UCLA	258	9.5	6.0	25	33
Oregon State	268	8.8	6.2	38	28
Stanford	216	8.4	7.0	50	9
California	254	8.1	6.7	60	18
Oregon	269	7.8	5.9	69	38
Washington	275	7.4	4.6	92	125
Washington State	253	6.7	5.5	112	55
Stony Brook	62	6.3	3.8	123	166
Arizona	294	6.2	4.5	130	133
Utah	246	5.9	6.6	140	21
Colorado	229	4.2	5.0	185	82

In [82]:
```
pac_off_result = result[result['offense_conference'] == 'Pac-12']
```

In [83]:
```
pac_def_result = result[result['defense_conference'] == 'Pac-12']
```

In [84]:
```
pac_off_result = pac_off_result.groupby('offense')['ppa'].mean()
```

In [85]:
```
pac_def_result = pac_def_result.groupby('defense')['ppa'].mean()
```

In [86]:
```
columns = ['ppa']
pac_off_result = pd.DataFrame(pac_off_result, columns=columns)
```

In [87]:
```
pac_off_result['team'] = pac_off_result.index
```

In [88]:
```
pac_off_result.reset_index(drop=True, inplace=True)
```

In [89]:
```
pac_off_result.rename(columns={"ppa": "off_ppa"})
```

Out[89]:

	off_ppa	team
0	-0.014443	Arizona
1	0.371231	Arizona State
2	0.320054	California
3	0.055480	Colorado
4	0.271345	Oregon
5	0.406353	Oregon State
6	0.306940	Stanford
7	0.356516	UCLA
8	0.230881	USC
9	0.127414	Utah
10	0.182069	Washington
11	0.132766	Washington State

In [90]:
```
columns = ['ppa']
pac_def_result = pd.DataFrame(pac_def_result, columns=columns)
```

In [91]:
```
pac_def_result['team'] = pac_def_result.index
```

In [92]:
```
pac_def_result.reset_index(drop=True, inplace=True)
```

In [93]:
```
pac_def_result.rename(columns={"ppa": "def_ppa"})
```

Out[93]:

	def_ppa	team
0	0.196498	Arizona
1	0.070503	Arizona State
2	0.234768	California
3	0.182368	Colorado
4	0.153521	Oregon
5	0.122247	Oregon State
6	0.250376	Stanford
7	0.210327	UCLA
8	0.268478	USC
9	0.076489	Utah
10	0.123566	Washington
11	0.209954	Washington State

In [94]:
```
pac_result = pd.merge(pac_off_result, pac_def_result, on='team', how='outer
```

In [95]:
```
pac_result = pac_result.rename(columns={"ppa_x": "off_ppa","ppa_y": "def_pp
```

In [96]:
```
pac_result
```

Out[96]:

	off_ppa	team	def_ppa
0	-0.014443	Arizona	0.196498
1	0.371231	Arizona State	0.070503
2	0.320054	California	0.234768
3	0.055480	Colorado	0.182368
4	0.271345	Oregon	0.153521
5	0.406353	Oregon State	0.122247
6	0.306940	Stanford	0.250376
7	0.356516	UCLA	0.210327
8	0.230881	USC	0.268478
9	0.127414	Utah	0.076489
10	0.182069	Washington	0.123566
11	0.132766	Washington State	0.209954

In [97]:
```
# Need to grab team ids
teams = cfbd.TeamsApi(cfbd.ApiClient(configuration)).get_fbs_teams()
teams_df = pd.DataFrame.from_records([dict(id=t.id, school=t.school) for t
```

In [98]:
```
teams_df.head()
```

Out[98]:

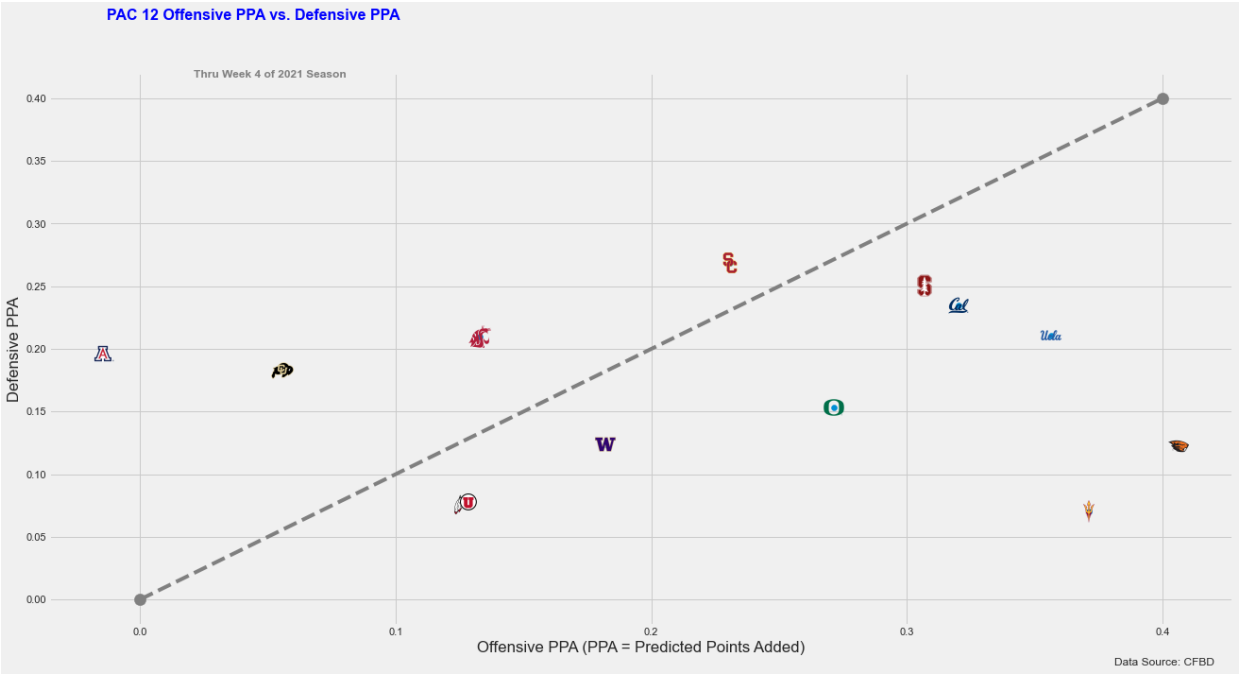
	id	school
0	2005	Air Force
1	2006	Akron
2	333	Alabama
3	2026	Appalachian State
4	12	Arizona

In [99]:
```
pac_result = pac_result.merge(teams_df, left_on='team', right_on='school')[
```

In [101]:

```
plt.style.use('fivethirtyeight')

# Graph sizing
plt.rcParams["figure.figsize"] = [20,10]


# You can download logos from here: https://drive.google.com/drive/folders/
# They come in two sizes. Extract them into the root of your script in a 'l
def getImage(path):
    return OffsetImage(plt.imread("./logos/{0}.png".format(path)))

# Logo file names are <team_id>.png, so we just need ids to map teams to lo
paths = pac_result['team']

# Picking two random stats to plot
x = pac_result['off_ppa']
y = pac_result['def_ppa']

fig, ax = plt.subplots()
ax.scatter(x, y)

# Cycle through each point and add an image annotation
for x0, y0, path in zip(x, y,paths):
    ab = AnnotationBbox(getImage(path), (x0, y0), frameon=False)
    ax.add_artist(ab)


# Define labels and title
# Title
fig.suptitle(
    "PAC 12 Offensive PPA vs. Defensive PPA",
    x = 0.122,
    y = 0.975,
    ha="left",
    fontsize=16,
    color="BLUE",
    weight="bold",
)

# Subtitle
ax.set_title(
    "Thru Week 4 of 2021 Season",
    x = 0.122,
    y = 0.975,
    loc="left",
    ha="left",
    fontsize=12,
    color="GREY",
    weight="bold",
    pad=10
)




#plt.title("Offensive vs Defensive Predicted Points Added")
```

```
plt.xlabel('Offensive PPA (PPA = Predicted Points Added)')
plt.ylabel('Defensive PPA')

# Now add on a line with a fixed slope of 0.03
sl = 1

# A line with a fixed slope can intercept the axis
# anywhere so we're going to have it go through 0,0
x_0 = 0
y_0 = 0

# And we'll have the line stop at x = 5000
x_1 = 0.4
y_1 = sl*(x_1 - x_0) + y_0

# Draw these two points with big triangles to make it clear
# where they lie
ax.scatter([x_0, x_1], [y_0, y_1], s=150, c='grey')

# And now connect them
ax.plot([x_0, x_1], [y_0, y_1], c='grey',linestyle='dashed')

txt="Data Source: CFBD"
plt.figtext(0.9, 0.01, txt, wrap=True, horizontalalignment='center', fontsi

plt.show()
```



Power 5 schools

In [102]:
```
value_list = ["Pac-12","SEC","Big Ten","Big 12","ACC"]
boolean_series = result.offense_conference.isin(value_list)
power_off_result = result[boolean_series]


#power_off_result = result[result['offense_conference'].isin(value_list)
```

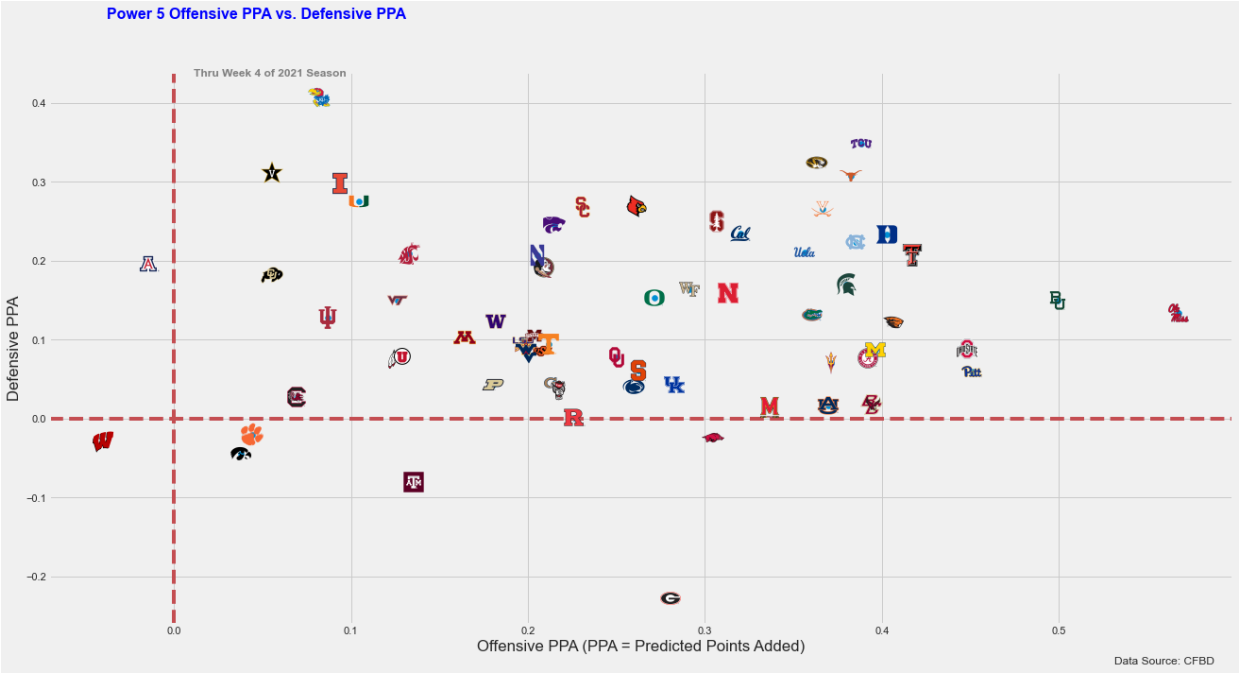
In [103]:
```
value_list = ["Pac-12","SEC","Big Ten","Big 12","ACC"]
boolean_series = result.defense_conference.isin(value_list)
power_def_result = result[boolean_series]
```

In [104]:

```
power_off_result = power_off_result.groupby('offense')['ppa'].mean()
power_def_result = power_def_result.groupby('defense')['ppa'].mean()

columns = ['ppa']
power_off_result = pd.DataFrame(power_off_result, columns=columns)
power_off_result['team'] = power_off_result.index
power_off_result.reset_index(drop=True, inplace=True)
power_off_result.rename(columns={"ppa": "off_ppa"})
power_def_result = pd.DataFrame(power_def_result, columns=columns)
power_def_result['team'] = power_def_result.index
power_def_result.reset_index(drop=True, inplace=True)
power_def_result.rename(columns={"ppa": "def_ppa"})
power_result = pd.merge(power_off_result, power_def_result, on='team', how=
power_result = power_result.rename(columns={"ppa_x": "off_ppa","ppa_y": "de
power_result = power_result.merge(teams_df, left_on='team', right_on='schoo


plt.style.use('fivethirtyeight')

# Graph sizing
plt.rcParams["figure.figsize"] = [20,10]


# You can download logos from here: https://drive.google.com/drive/folders/
# They come in two sizes. Extract them into the root of your script in a 'l
def getImage(path):
    return OffsetImage(plt.imread("./logos/{0}.png".format(path)))

# Logo file names are <team_id>.png, so we just need ids to map teams to lo
paths = power_result['team']

# Picking two random stats to plot
x = power_result['off_ppa']
y = power_result['def_ppa']

fig, ax = plt.subplots()
ax.scatter(x, y)

# Cycle through each point and add an image annotation
for x0, y0, path in zip(x, y,paths):
    ab = AnnotationBbox(getImage(path), (x0, y0), frameon=False)
    ax.add_artist(ab)


# Define labels and title
# Title
fig.suptitle(
    "Power 5 Offensive PPA vs. Defensive PPA",
    x = 0.122,
    y = 0.975,
    ha="left",
    fontsize=16,
    color="BLUE",
    weight="bold",
)
```

```
# Subtitle
ax.set_title(
    "Thru Week 4 of 2021 Season",
    x = 0.122,
    y = 0.975,
    loc="left",
    ha="left",
    fontsize=12,
    color="GREY",
    weight="bold",
    pad=10
)


plt.axhline(y=0, color='r', linestyle='dashed')
plt.axvline(x=0, color='r', linestyle='dashed')
#plt.title("Offensive vs Defensive Predicted Points Added")
plt.xlabel('Offensive PPA (PPA = Predicted Points Added)')
plt.ylabel('Defensive PPA')

txt="Data Source: CFBD"
plt.figtext(0.9, 0.01, txt, wrap=True, horizontalalignment='center', fontsi

# Now add on a line with a fixed slope of 0.03
#sl = 1

# A line with a fixed slope can intercept the axis
# anywhere so we're going to have it go through 0,0
#x_0 = 0
#y_0 = 0

# And we'll have the line stop at x = 5000
#x_1 = 0.4
#y_1 = sl*(x_1 - x_0) + y_0

# Draw these two points with big triangles to make it clear
# where they lie
#ax.scatter([x_0, x_1], [y_0, y_1], s=150, c='grey')

# And now connect them
#ax.plot([x_0, x_1], [y_0, y_1], c='grey',linestyle='dashed')

#plt.show()
```

Out[104]: Text(0.9, 0.01, 'Data Source: CFBD')



Advanced Stats - PAC 12

In [105]:
```
pac_off_result = result[result['offense_conference'] == 'Pac-12']
pac_def_result = result[result['defense_conference'] == 'Pac-12']
```

In [106]:
```
value_list = ["Pac-12","SEC","Big Ten","Big 12","ACC"]
boolean_series = result.offense_conference.isin(value_list)
boolean_series1 = result.defense_conference.isin(value_list)
power_off_result = result[boolean_series]
power_def_result = result[boolean_series1]
```

In [107]:
```
pac_off_result.tail()
```

Out[107]:

	id	drive_id	game_id	drive_number	play_number	offense	offense_
10963	401309862104974502	40130986221	401309862	21	5	Arizona State	
10964	401309862104974503	40130986221	401309862	21	6	Arizona State	
10965	401309862104974504	40130986221	401309862	21	7	Arizona State	
10966	401309862104999901	40130986221	401309862	21	8	Arizona State	
10967	401309862104999902	40130986221	401309862	21	9	Arizona State	

5 rows × 27 columns

In [108]:
```
bad_plays = ["Interception","Pass Interception Return","Interception Return
             "Fumble Return Touchdown"]

def success(play):
    if(play.play_type in bad_plays):
        return False
    if((play.down == 1) & (play.yards_gained >= (0.5 * play.distance))):
        return True
    elif ((play.down == 2)) & (play.yards_gained >= (0.7 * play.distance)):
        return True
    elif ((play.down == 3) & (play.yards_gained >= play.distance)):
        return True
    elif ((play.down == 4) & (play.yards_gained >= play.distance)):
        return True
    else:
        return False

pac_off_result['play_successful'] = pac_off_result.apply(lambda x: success(
pac_off_result['play_explosive'] = pac_off_result.apply(lambda x: x.yards_g
power_off_result['play_successful'] = power_off_result.apply(lambda x: succ
power_off_result['play_explosive'] = power_off_result.apply(lambda x: x.yar

pac_def_result['play_successful'] = pac_def_result.apply(lambda x: success(
pac_def_result['play_explosive'] = pac_def_result.apply(lambda x: x.yards_g
power_def_result['play_successful'] = power_def_result.apply(lambda x: succ
power_def_result['play_explosive'] = power_def_result.apply(lambda x: x.yar
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:18: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:19: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:20: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:21: Sett
ingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:23: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:24: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:25: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:26: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
```

In [109]:

```
pass_types = ["Pass Reception","Pass Incompletion","Passing Touchdown","Int
rush_types = ["Rush","Rushing Touchdown",'Fumble Recovery (Opponent)','Sack

#PAC 12 off
pac_off_result['successful'] = np.where(pac_off_result['play_successful'] =
pac_off_result['explosive'] = np.where(pac_off_result['play_explosive'] ==
pac_off_result['rushing'] = np.where(pac_off_result['play_type'].isin(rush_
pac_off_result['rushing_successful'] = np.where(pac_off_result['play_type']
                                                pac_off_result['play_succes
pac_off_result['rushing_explosive'] = np.where(pac_off_result['play_type'].
                                                pac_off_result['play_explos
pac_off_result['passing'] = np.where(pac_off_result['play_type'].isin(pass_
pac_off_result['passing_successful'] = np.where(pac_off_result['play_type']
                                                pac_off_result['play_succes
pac_off_result['passing_explosive'] = np.where(pac_off_result['play_type'].
                                                pac_off_result['play_explos
#POWER 5 Schools off
power_off_result['successful'] = np.where(power_off_result['play_successful
power_off_result['explosive'] = np.where(power_off_result['play_explosive']
power_off_result['rushing'] = np.where(power_off_result['play_type'].isin(r
power_off_result['rushing_successful'] = np.where(power_off_result['play_ty
                                                power_off_result['play_succ
power_off_result['rushing_explosive'] = np.where(power_off_result['play_typ
                                                power_off_result['play_expl
power_off_result['passing'] = np.where(power_off_result['play_type'].isin(p
power_off_result['passing_successful'] = np.where(power_off_result['play_ty
                                                power_off_result['play_succ
power_off_result['passing_explosive'] = np.where(power_off_result['play_typ
                                                power_off_result['play_expl



#PAC 12 def
pac_def_result['successful'] = np.where(pac_def_result['play_successful'] =
pac_def_result['explosive'] = np.where(pac_def_result['play_explosive'] ==
pac_def_result['rushing'] = np.where(pac_def_result['play_type'].isin(rush_
pac_def_result['rushing_successful'] = np.where(pac_def_result['play_type']
                                                pac_def_result['play_succes
pac_def_result['rushing_explosive'] = np.where(pac_def_result['play_type'].
                                                pac_def_result['play_explos
pac_def_result['passing'] = np.where(pac_def_result['play_type'].isin(pass_
pac_def_result['passing_successful'] = np.where(pac_def_result['play_type']
                                                pac_def_result['play_succes
pac_def_result['passing_explosive'] = np.where(pac_def_result['play_type'].
                                                pac_def_result['play_explos
#POWER 5 Schools off
power_def_result['successful'] = np.where(power_def_result['play_successful
power_def_result['explosive'] = np.where(power_def_result['play_explosive']
power_def_result['rushing'] = np.where(power_def_result['play_type'].isin(r
power_def_result['rushing_successful'] = np.where(power_def_result['play_ty
                                                    power_def_result['play_s
power_def_result['rushing_explosive'] = np.where(power_def_result['play_typ
                                                power_def_result['play_expl
power_def_result['passing'] = np.where(power_def_result['play_type'].isin(p
power_def_result['passing_successful'] = np.where(power_def_result['play_ty
                                                power_def_result['play_succ
power_def_result['passing_explosive'] = np.where(power_def_result['play_typ
```

```
                                                    power_def_result['play_expl
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  """
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
```

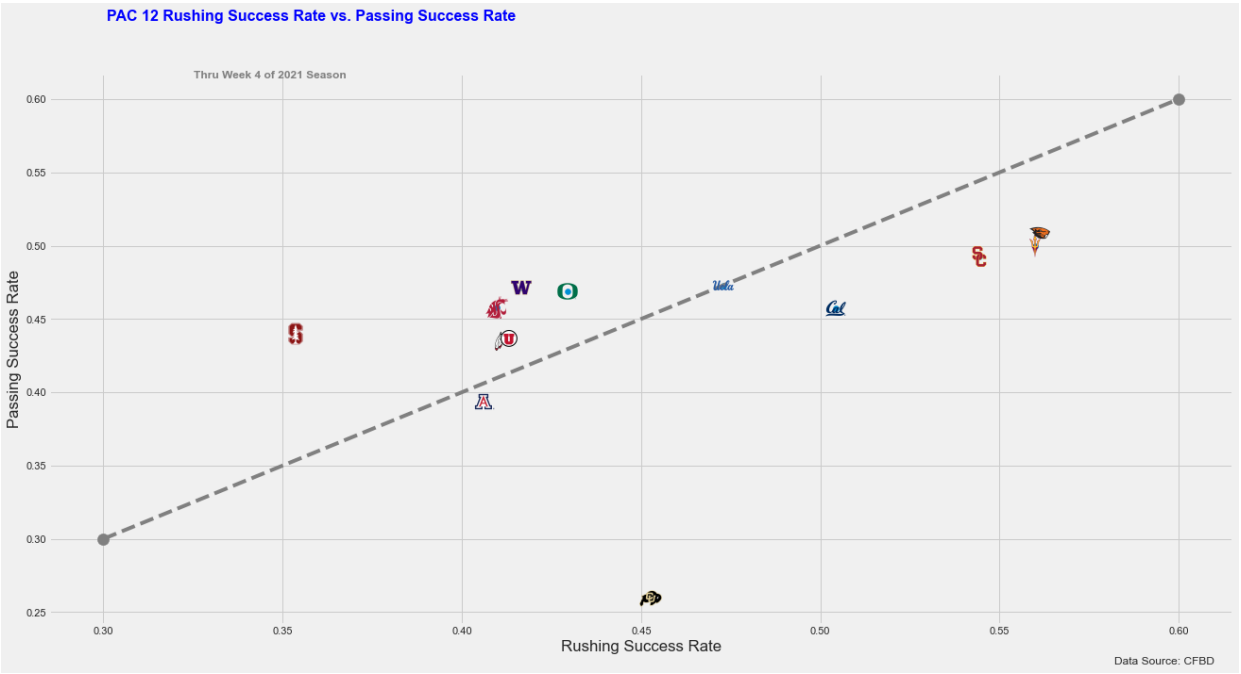
In [110]:

```
#success table
total_plays = pac_off_result.groupby('offense').count()
total_plays = total_plays['game_id']
success = pac_off_result.groupby(['offense'])['successful'].sum().sort_valu
explosive = pac_off_result.groupby(['offense'])['explosive'].sum().sort_val
rushing_plays = pac_off_result.groupby(['offense'])['rushing'].sum().sort_v
rushing_success = pac_off_result.groupby(['offense'])['rushing_successful']
rushing_explosive = pac_off_result.groupby(['offense'])['rushing_explosive'
passing_plays = pac_off_result.groupby(['offense'])['passing'].sum().sort_v
passing_success = pac_off_result.groupby(['offense'])['passing_successful']
passing_explosive = pac_off_result.groupby(['offense'])['passing_explosive'

frames1 = [total_plays,success,explosive,rushing_plays,rushing_success,rush
offensive_ratio = pd.concat(frames1,axis=1,join="outer")
offensive_ratio['team'] = offensive_ratio.index
offensive_ratio.reset_index(drop=True, inplace=True)

offensive_ratio["Success Rate"] = offensive_ratio['successful'] / offensive
offensive_ratio["Explosive Rate"] = offensive_ratio['explosive'] / offensiv
offensive_ratio["Rushing Pct"] = offensive_ratio['rushing'] / offensive_rat
offensive_ratio["Rushing Success Rate"] = offensive_ratio['rushing_successf
offensive_ratio["Rushing Explosive Rate"] = offensive_ratio['rushing_explos
offensive_ratio["Passing Pct"] = offensive_ratio['passing'] / offensive_rat
offensive_ratio["Passing Success Rate"] = offensive_ratio['passing_successf
offensive_ratio["Passing Explosive Rate"] = offensive_ratio['passing_explos

offensive_ratio = offensive_ratio.rename(columns={"game_id":"Total Plays","
#offensive_ratio[['Success Rate']] = pd.Series(["{0:.2f}%".format(val * 100
#offensive_ratio[['Explosive Rate']] = pd.Series(["{0:.2f}%".format(val * 1
#offensive_ratio[['Rushing Pct']] = pd.Series(["{0:.2f}%".format(val * 100)
#offensive_ratio[['Rushing Success Rate']] = pd.Series(["{0:.2f}%".format(v
#offensive_ratio[['Rushing Explosive Rate']] = pd.Series(["{0:.2f}%".format
#offensive_ratio[['Passing Pct']] = pd.Series(["{0:.2f}%".format(val * 100)
#offensive_ratio[['Passing Success Rate']] = pd.Series(["{0:.2f}%".format(v
#offensive_ratio[['Passing Explosive Rate']] = pd.Series(["{0:.2f}%".format
#offensive_ratio.sort_values(by='Success Rate',ascending=False)
#success_ratio[['%']] = pd.Series(["{0:.2f}%".format(val * 100) for val in
```

In [111]:

```
plt.style.use('fivethirtyeight')

# Graph sizing
plt.rcParams["figure.figsize"] = [20,10]


# You can download logos from here: https://drive.google.com/drive/folders/
# They come in two sizes. Extract them into the root of your script in a 'l
def getImage(path):
    return OffsetImage(plt.imread("./logos/{0}.png".format(path)))

# Logo file names are <team_id>.png, so we just need ids to map teams to lo
paths = offensive_ratio['team']

# Picking two random stats to plot
x = offensive_ratio['Rushing Success Rate']
y = offensive_ratio['Passing Success Rate']

fig, ax = plt.subplots()
ax.scatter(x, y)

# Cycle through each point and add an image annotation
for x0, y0, path in zip(x, y,paths):
    ab = AnnotationBbox(getImage(path), (x0, y0), frameon=False)
    ax.add_artist(ab)


# Define labels and title
# Title
fig.suptitle(
    "PAC 12 Rushing Success Rate vs. Passing Success Rate",
    x = 0.122,
    y = 0.975,
    ha="left",
    fontsize=16,
    color="BLUE",
    weight="bold",
)

# Subtitle
ax.set_title(
    "Thru Week 4 of 2021 Season",
    x = 0.122,
    y = 0.975,
    loc="left",
    ha="left",
    fontsize=12,
    color="GREY",
    weight="bold",
    pad=10
)


#plt.axhline(y=0, color='r', linestyle='dashed')
#plt.axvline(x=0, color='r', linestyle='dashed')
#plt.title("Offensive vs Defensive Predicted Points Added")
```

```
plt.xlabel('Rushing Success Rate')
plt.ylabel('Passing Success Rate')

# Now add on a line with a fixed slope of 0.03
sl = 1

# A line with a fixed slope can intercept the axis
# anywhere so we're going to have it go through 0,0
x_0 = 0.3
y_0 = 0.3

# And we'll have the line stop at x = 5000
x_1 = 0.6
y_1 = sl*(x_1 - x_0) + y_0

# Draw these two points with big triangles to make it clear
# where they lie
ax.scatter([x_0, x_1], [y_0, y_1], s=150, c='grey')

# And now connect them
ax.plot([x_0, x_1], [y_0, y_1], c='grey',linestyle='dashed')

txt="Data Source: CFBD"
plt.figtext(0.9, 0.01, txt, wrap=True, horizontalalignment='center', fontsi

plt.show()
```



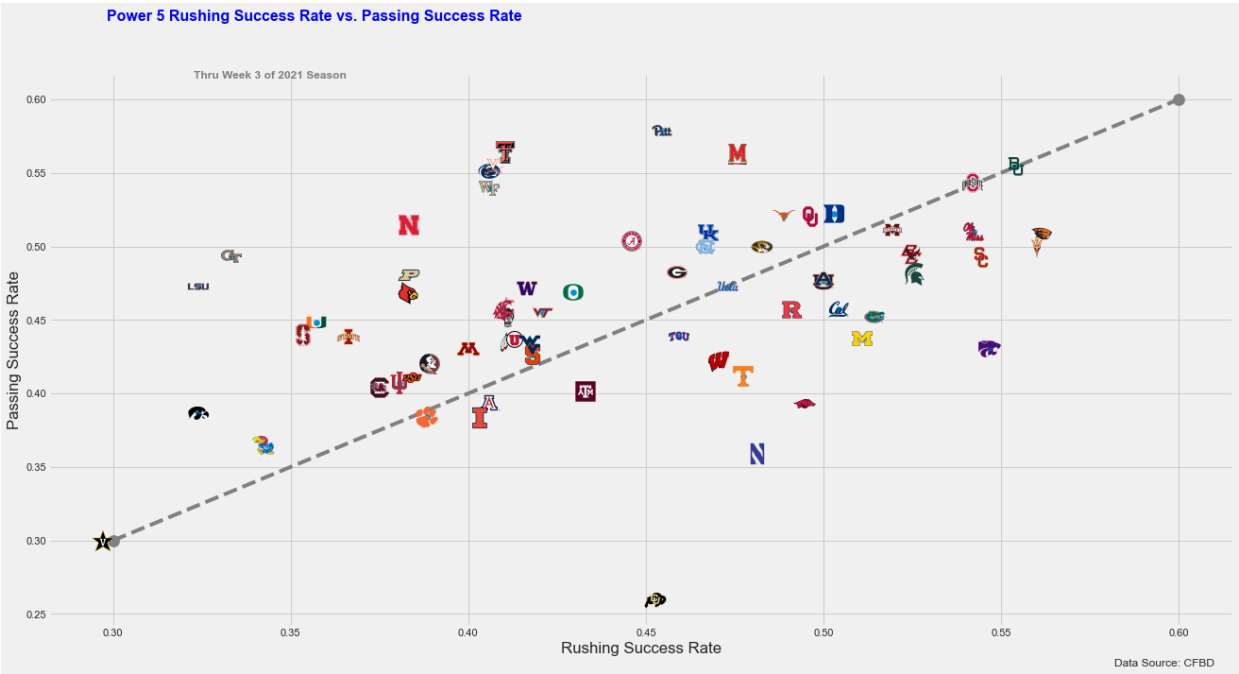
In [112]:

```
#success table
total_plays = power_off_result.groupby('offense').count()
total_plays = total_plays['game_id']
success = power_off_result.groupby(['offense'])['successful'].sum().sort_va
explosive = power_off_result.groupby(['offense'])['explosive'].sum().sort_v
rushing_plays = power_off_result.groupby(['offense'])['rushing'].sum().sort
rushing_success = power_off_result.groupby(['offense'])['rushing_successful
rushing_explosive = power_off_result.groupby(['offense'])['rushing_explosiv
passing_plays = power_off_result.groupby(['offense'])['passing'].sum().sort
passing_success = power_off_result.groupby(['offense'])['passing_successful
passing_explosive = power_off_result.groupby(['offense'])['passing_explosiv

frames1 = [total_plays,success,explosive,rushing_plays,rushing_success,rush
offensive_ratio = pd.concat(frames1,axis=1,join="outer")
offensive_ratio['team'] = offensive_ratio.index
offensive_ratio.reset_index(drop=True, inplace=True)

offensive_ratio["Success Rate"] = offensive_ratio['successful'] / offensive
offensive_ratio["Explosive Rate"] = offensive_ratio['explosive'] / offensiv
offensive_ratio["Rushing Pct"] = offensive_ratio['rushing'] / offensive_rat
offensive_ratio["Rushing Success Rate"] = offensive_ratio['rushing_successf
offensive_ratio["Rushing Explosive Rate"] = offensive_ratio['rushing_explos
offensive_ratio["Passing Pct"] = offensive_ratio['passing'] / offensive_rat
offensive_ratio["Passing Success Rate"] = offensive_ratio['passing_successf
offensive_ratio["Passing Explosive Rate"] = offensive_ratio['passing_explos

offensive_ratio = offensive_ratio.rename(columns={"game_id":"Total Plays","
#offensive_ratio[['Success Rate']] = pd.Series(["{0:.2f}%".format(val * 100
#offensive_ratio[['Explosive Rate']] = pd.Series(["{0:.2f}%".format(val * 1
#offensive_ratio[['Rushing Pct']] = pd.Series(["{0:.2f}%".format(val * 100)
#offensive_ratio[['Rushing Success Rate']] = pd.Series(["{0:.2f}%".format(v
#offensive_ratio[['Rushing Explosive Rate']] = pd.Series(["{0:.2f}%".format
#offensive_ratio[['Passing Pct']] = pd.Series(["{0:.2f}%".format(val * 100)
#offensive_ratio[['Passing Success Rate']] = pd.Series(["{0:.2f}%".format(v
#offensive_ratio[['Passing Explosive Rate']] = pd.Series(["{0:.2f}%".format
#offensive_ratio.sort_values(by='Success Rate',ascending=False)
#success_ratio[['%']] = pd.Series(["{0:.2f}%".format(val * 100) for val in
```

In [113]:

```
plt.style.use('fivethirtyeight')

# Graph sizing
plt.rcParams["figure.figsize"] = [20,10]


# You can download logos from here: https://drive.google.com/drive/folders/
# They come in two sizes. Extract them into the root of your script in a 'l
def getImage(path):
    return OffsetImage(plt.imread("./logos/{0}.png".format(path)))

# Logo file names are <team_id>.png, so we just need ids to map teams to lo
paths = offensive_ratio['team']

# Picking two random stats to plot
x = offensive_ratio['Rushing Success Rate']
y = offensive_ratio['Passing Success Rate']

fig, ax = plt.subplots()
ax.scatter(x, y)

# Cycle through each point and add an image annotation
for x0, y0, path in zip(x, y,paths):
    ab = AnnotationBbox(getImage(path), (x0, y0), frameon=False)
    ax.add_artist(ab)


# Define labels and title
# Title
fig.suptitle(
    "Power 5 Rushing Success Rate vs. Passing Success Rate",
    x = 0.122,
    y = 0.975,
    ha="left",
    fontsize=16,
    color="BLUE",
    weight="bold",
)

# Subtitle
ax.set_title(
    "Thru Week 4 of 2021 Season",
    x = 0.122,
    y = 0.975,
    loc="left",
    ha="left",
    fontsize=12,
    color="GREY",
    weight="bold",
    pad=10
)


#plt.axhline(y=0, color='r', linestyle='dashed')
#plt.axvline(x=0, color='r', linestyle='dashed')
#plt.title("Offensive vs Defensive Predicted Points Added")
```

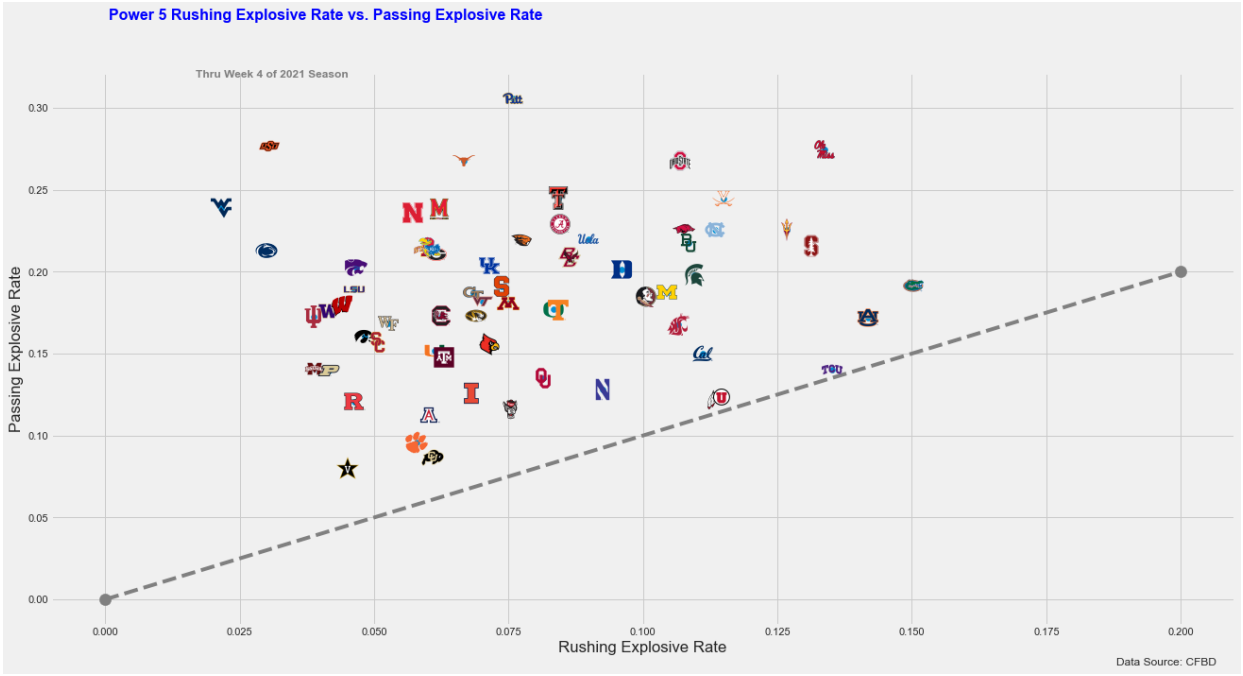
```
plt.xlabel('Rushing Success Rate')
plt.ylabel('Passing Success Rate')

# Now add on a line with a fixed slope of 0.03
sl = 1

# A line with a fixed slope can intercept the axis
# anywhere so we're going to have it go through 0,0
x_0 = 0.3
y_0 = 0.3

# And we'll have the line stop at x = 5000
x_1 = 0.6
y_1 = sl*(x_1 - x_0) + y_0

# Draw these two points with big triangles to make it clear
# where they lie
ax.scatter([x_0, x_1], [y_0, y_1], s=150, c='grey')

# And now connect them
ax.plot([x_0, x_1], [y_0, y_1], c='grey',linestyle='dashed')

txt="Data Source: CFBD"
plt.figtext(0.9, 0.01, txt, wrap=True, horizontalalignment='center', fontsi

plt.show()
```



In [115]:

```
plt.style.use('fivethirtyeight')

# Graph sizing
plt.rcParams["figure.figsize"] = [20,10]


# You can download logos from here: https://drive.google.com/drive/folders/
# They come in two sizes. Extract them into the root of your script in a 'l
def getImage(path):
    return OffsetImage(plt.imread("./logos/{0}.png".format(path)))

# Logo file names are <team_id>.png, so we just need ids to map teams to lo
paths = offensive_ratio['team']

# Picking two random stats to plot
x = offensive_ratio['Rushing Explosive Rate']
y = offensive_ratio['Passing Explosive Rate']

fig, ax = plt.subplots()
ax.scatter(x, y)

# Cycle through each point and add an image annotation
for x0, y0, path in zip(x, y,paths):
    ab = AnnotationBbox(getImage(path), (x0, y0), frameon=False)
    ax.add_artist(ab)


# Define labels and title
# Title
fig.suptitle(
    "Power 5 Rushing Explosive Rate vs. Passing Explosive Rate",
    x = 0.122,
    y = 0.975,
    ha="left",
    fontsize=16,
    color="BLUE",
    weight="bold",
)

# Subtitle
ax.set_title(
    "Thru Week 4 of 2021 Season",
    x = 0.122,
    y = 0.975,
    loc="left",
    ha="left",
    fontsize=12,
    color="GREY",
    weight="bold",
    pad=10
)


#plt.axhline(y=0, color='r', linestyle='dashed')
#plt.axvline(x=0, color='r', linestyle='dashed')
#plt.title("Offensive vs Defensive Predicted Points Added")
```

```
plt.xlabel('Rushing Explosive Rate')
plt.ylabel('Passing Explosive Rate')

# Now add on a line with a fixed slope of 0.03
sl = 1

# A line with a fixed slope can intercept the axis
# anywhere so we're going to have it go through 0,0
x_0 = 0
y_0 = 0

# And we'll have the line stop at x = 5000
x_1 = 0.2
y_1 = sl*(x_1 - x_0) + y_0

# Draw these two points with big triangles to make it clear
# where they lie
ax.scatter([x_0, x_1], [y_0, y_1], s=150, c='grey')

# And now connect them
ax.plot([x_0, x_1], [y_0, y_1], c='grey',linestyle='dashed')

txt="Data Source: CFBD"
plt.figtext(0.9, 0.01, txt, wrap=True, horizontalalignment='center', fontsi

plt.show()
```



In [116]:

```
#success table
total_plays = pac_def_result.groupby('defense').count()
total_plays = total_plays['game_id']
success = pac_def_result.groupby(['defense'])['successful'].sum().sort_valu
explosive = pac_def_result.groupby(['defense'])['explosive'].sum().sort_val
rushing_plays = pac_def_result.groupby(['defense'])['rushing'].sum().sort_v
rushing_success = pac_def_result.groupby(['defense'])['rushing_successful']
rushing_explosive = pac_def_result.groupby(['defense'])['rushing_explosive'
passing_plays = pac_def_result.groupby(['defense'])['passing'].sum().sort_v
passing_success = pac_def_result.groupby(['defense'])['passing_successful']
passing_explosive = pac_def_result.groupby(['defense'])['passing_explosive'

frames2 = [total_plays,success,explosive,rushing_plays,rushing_success,rush
defense_ratio = pd.concat(frames2,axis=1,join="outer")
defense_ratio['team'] = defense_ratio.index
defense_ratio.reset_index(drop=True, inplace=True)

defense_ratio["Success Rate"] = defense_ratio['successful'] / defense_ratio
defense_ratio["Explosive Rate"] = defense_ratio['explosive'] / defense_rati
defense_ratio["Rushing Pct"] = defense_ratio['rushing'] / defense_ratio['ga
defense_ratio["Rushing Success Rate"] = defense_ratio['rushing_successful']
defense_ratio["Rushing Explosive Rate"] = defense_ratio['rushing_explosive'
defense_ratio["Passing Pct"] = defense_ratio['passing'] / defense_ratio['ga
defense_ratio["Passing Success Rate"] = defense_ratio['passing_successful']
defense_ratio["Passing Explosive Rate"] = defense_ratio['passing_explosive'

defense_ratio = defense_ratio.rename(columns={"game_id":"Total Plays","succ
#offensive_ratio[['Success Rate']] = pd.Series(["{0:.2f}%".format(val * 100
#offensive_ratio[['Explosive Rate']] = pd.Series(["{0:.2f}%".format(val * 1
#offensive_ratio[['Rushing Pct']] = pd.Series(["{0:.2f}%".format(val * 100)
#offensive_ratio[['Rushing Success Rate']] = pd.Series(["{0:.2f}%".format(v
#offensive_ratio[['Rushing Explosive Rate']] = pd.Series(["{0:.2f}%".format
#offensive_ratio[['Passing Pct']] = pd.Series(["{0:.2f}%".format(val * 100)
#offensive_ratio[['Passing Success Rate']] = pd.Series(["{0:.2f}%".format(v
#offensive_ratio[['Passing Explosive Rate']] = pd.Series(["{0:.2f}%".format
defense_ratio.sort_values(by='Success Rate',ascending=False)
#success_ratio[['%']] = pd.Series(["{0:.2f}%".format(val * 100) for val in
```

Out[116]:

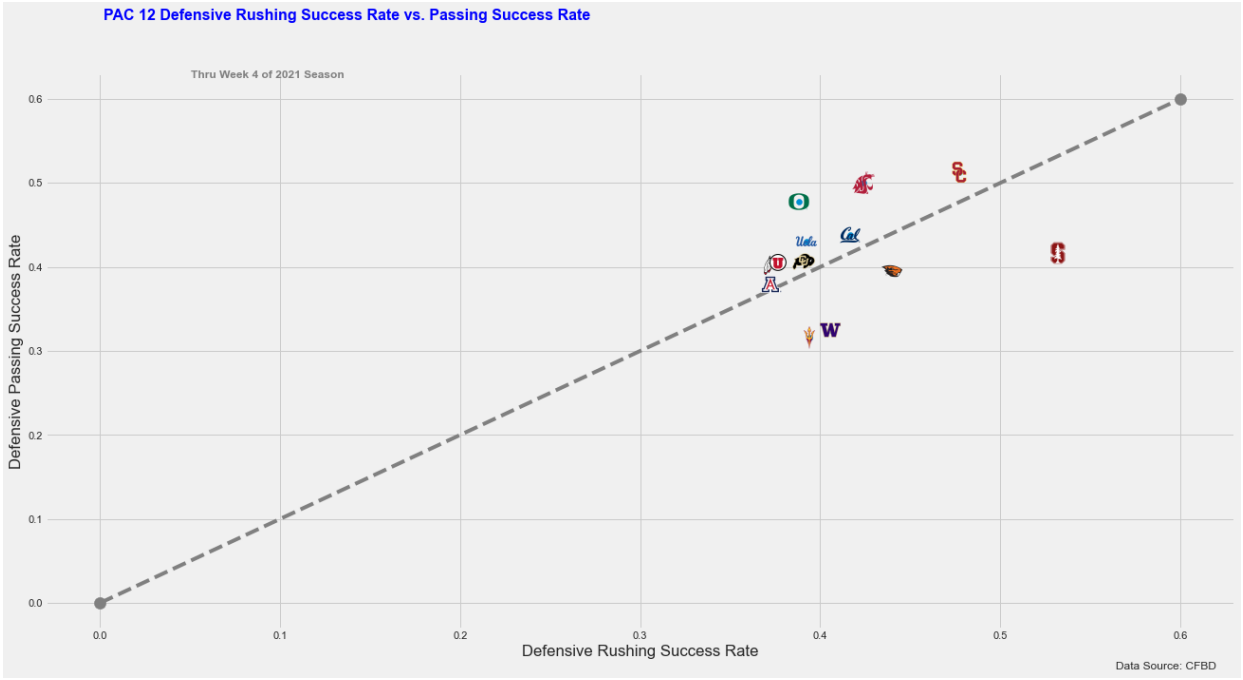
	Total Plays	Successful Play	Explosive Play	Rushing Plays	Successful Rushing Plays	Explosive Rushing Plays	Passing Plays	Successful Passing Plays	Explosive Passing Plays
6	362	160	45	156	83	12	125	52	15
8	250	108	38	109	52	8	76	39	15
11	346	145	40	118	50	5	154	77	19
4	394	156	58	152	59	9	155	74	29
2	372	140	60	120	50	7	153	67	34
7	385	144	58	102	40	7	186	80	31
3	338	124	46	128	50	8	123	50	20

	Total Plays	Successful Play	Explosive Play	Rushing Plays	Successful Rushing Plays	Explosive Rushing Plays	Passing Plays	Successful Passing Plays	Explosive Passing Plays	
1	317	116	45	137	54	10	95	30	11	
5	380	138	46	116	51	4	172	68	27	
0	338	120	48	153	57	14	87	33	12	
9	373	128	39	152	57	10	119	48	11	
10	358	118	41	143	58	9	126	41	18	V

In [117]:

```
plt.style.use('fivethirtyeight')

# Graph sizing
plt.rcParams["figure.figsize"] = [20,10]


# You can download logos from here: https://drive.google.com/drive/folders/
# They come in two sizes. Extract them into the root of your script in a 'l
def getImage(path):
    return OffsetImage(plt.imread("./logos/{0}.png".format(path)))

# Logo file names are <team_id>.png, so we just need ids to map teams to lo
paths = defense_ratio['team']

# Picking two random stats to plot
x = defense_ratio['Rushing Success Rate']
y = defense_ratio['Passing Success Rate']

fig, ax = plt.subplots()
ax.scatter(x, y)

# Cycle through each point and add an image annotation
for x0, y0, path in zip(x, y,paths):
    ab = AnnotationBbox(getImage(path), (x0, y0), frameon=False)
    ax.add_artist(ab)


# Define labels and title
# Title
fig.suptitle(
    "PAC 12 Defensive Rushing Success Rate vs. Passing Success Rate",
    x = 0.122,
    y = 0.975,
    ha="left",
    fontsize=16,
    color="BLUE",
    weight="bold",
)

# Subtitle
ax.set_title(
    "Thru Week 4 of 2021 Season",
    x = 0.122,
    y = 0.975,
    loc="left",
    ha="left",
    fontsize=12,
    color="GREY",
    weight="bold",
    pad=10
)


#plt.axhline(y=0, color='r', linestyle='dashed')
#plt.axvline(x=0, color='r', linestyle='dashed')
#plt.title("Offensive vs Defensive Predicted Points Added")
```

```
plt.xlabel('Defensive Rushing Success Rate')
plt.ylabel('Defensive Passing Success Rate')

# Now add on a line with a fixed slope of 0.03
sl = 1

# A line with a fixed slope can intercept the axis
# anywhere so we're going to have it go through 0,0
x_0 = 0
y_0 = 0

# And we'll have the line stop at x = 5000
x_1 = 0.6
y_1 = sl*(x_1 - x_0) + y_0

# Draw these two points with big triangles to make it clear
# where they lie
ax.scatter([x_0, x_1], [y_0, y_1], s=150, c='grey')

# And now connect them
ax.plot([x_0, x_1], [y_0, y_1], c='grey',linestyle='dashed')

txt="Data Source: CFBD"
plt.figtext(0.9, 0.01, txt, wrap=True, horizontalalignment='center', fontsi

plt.show()
```



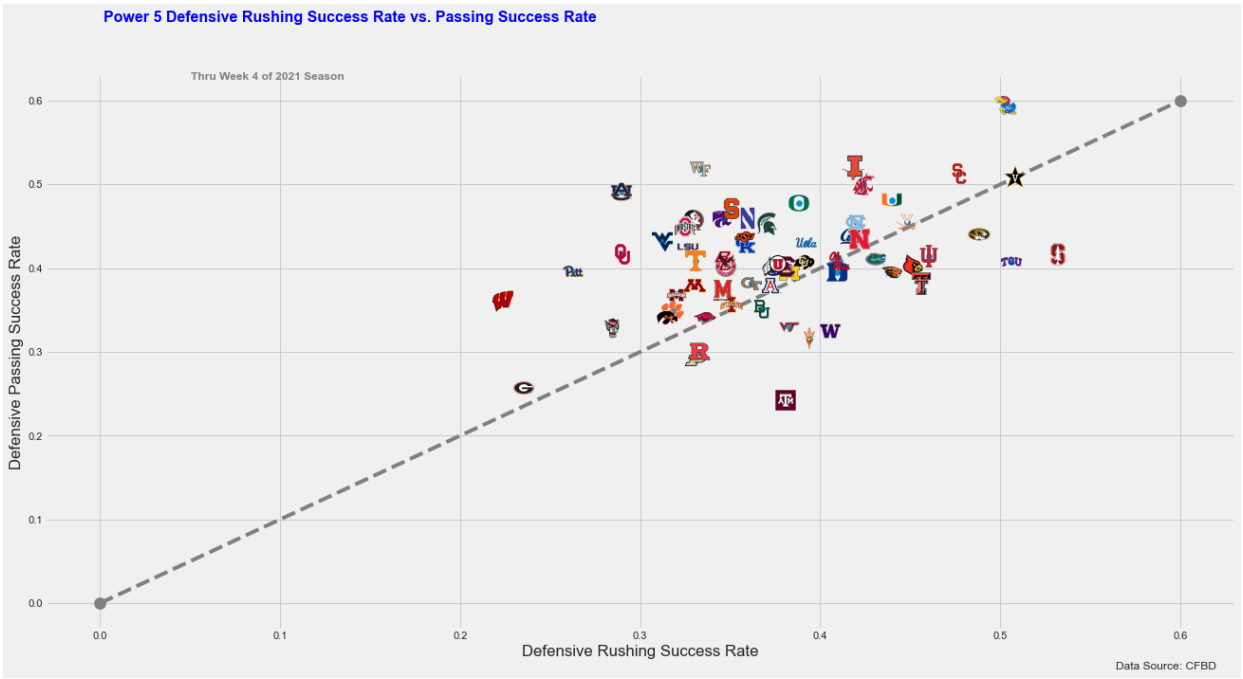
In [118]:

```
#success table
total_plays = power_def_result.groupby('defense').count()
total_plays = total_plays['game_id']
success = power_def_result.groupby(['defense'])['successful'].sum().sort_va
explosive = power_def_result.groupby(['defense'])['explosive'].sum().sort_v
rushing_plays = power_def_result.groupby(['defense'])['rushing'].sum().sort
rushing_success = power_def_result.groupby(['defense'])['rushing_successful
rushing_explosive = power_def_result.groupby(['defense'])['rushing_explosiv
passing_plays = power_def_result.groupby(['defense'])['passing'].sum().sort
passing_success = power_def_result.groupby(['defense'])['passing_successful
passing_explosive = power_def_result.groupby(['defense'])['passing_explosiv

frames3 = [total_plays,success,explosive,rushing_plays,rushing_success,rush
defensive_ratio = pd.concat(frames3,axis=1,join="outer")
defensive_ratio['team'] = defensive_ratio.index
defensive_ratio.reset_index(drop=True, inplace=True)

defensive_ratio["Success Rate"] = defensive_ratio['successful'] / defensive
defensive_ratio["Explosive Rate"] = defensive_ratio['explosive'] / defensiv
defensive_ratio["Rushing Pct"] = defensive_ratio['rushing'] / defensive_rat
defensive_ratio["Rushing Success Rate"] = defensive_ratio['rushing_successf
defensive_ratio["Rushing Explosive Rate"] = defensive_ratio['rushing_explos
defensive_ratio["Passing Pct"] = defensive_ratio['passing'] / defensive_rat
defensive_ratio["Passing Success Rate"] = defensive_ratio['passing_successf
defensive_ratio["Passing Explosive Rate"] = defensive_ratio['passing_explos

defensive_ratio = defensive_ratio.rename(columns={"game_id":"Total Plays","
#offensive_ratio[['Success Rate']] = pd.Series(["{0:.2f}%".format(val * 100
#offensive_ratio[['Explosive Rate']] = pd.Series(["{0:.2f}%".format(val * 1
#offensive_ratio[['Rushing Pct']] = pd.Series(["{0:.2f}%".format(val * 100)
#offensive_ratio[['Rushing Success Rate']] = pd.Series(["{0:.2f}%".format(v
#offensive_ratio[['Rushing Explosive Rate']] = pd.Series(["{0:.2f}%".format
#offensive_ratio[['Passing Pct']] = pd.Series(["{0:.2f}%".format(val * 100)
#offensive_ratio[['Passing Success Rate']] = pd.Series(["{0:.2f}%".format(v
#offensive_ratio[['Passing Explosive Rate']] = pd.Series(["{0:.2f}%".format
#offensive_ratio.sort_values(by='Success Rate',ascending=False)
#success_ratio[['%']] = pd.Series(["{0:.2f}%".format(val * 100) for val in
```

In [119]:

```
plt.style.use('fivethirtyeight')

# Graph sizing
plt.rcParams["figure.figsize"] = [20,10]


# You can download logos from here: https://drive.google.com/drive/folders/
# They come in two sizes. Extract them into the root of your script in a 'l
def getImage(path):
    return OffsetImage(plt.imread("./logos/{0}.png".format(path)))

# Logo file names are <team_id>.png, so we just need ids to map teams to lo
paths = defensive_ratio['team']

# Picking two random stats to plot
x = defensive_ratio['Rushing Success Rate']
y = defensive_ratio['Passing Success Rate']

fig, ax = plt.subplots()
ax.scatter(x, y)

# Cycle through each point and add an image annotation
for x0, y0, path in zip(x, y,paths):
    ab = AnnotationBbox(getImage(path), (x0, y0), frameon=False)
    ax.add_artist(ab)


# Define labels and title
# Title
fig.suptitle(
    "Power 5 Defensive Rushing Success Rate vs. Passing Success Rate",
    x = 0.122,
    y = 0.975,
    ha="left",
    fontsize=16,
    color="BLUE",
    weight="bold",
)

# Subtitle
ax.set_title(
    "Thru Week 4 of 2021 Season",
    x = 0.122,
    y = 0.975,
    loc="left",
    ha="left",
    fontsize=12,
    color="GREY",
    weight="bold",
    pad=10
)


#plt.axhline(y=0, color='r', linestyle='dashed')
#plt.axvline(x=0, color='r', linestyle='dashed')
#plt.title("Offensive vs Defensive Predicted Points Added")
```

```
plt.xlabel('Defensive Rushing Success Rate')
plt.ylabel('Defensive Passing Success Rate')

# Now add on a line with a fixed slope of 0.03
sl = 1

# A line with a fixed slope can intercept the axis
# anywhere so we're going to have it go through 0,0
x_0 = 0
y_0 = 0

# And we'll have the line stop at x = 5000
x_1 = 0.6
y_1 = sl*(x_1 - x_0) + y_0

# Draw these two points with big triangles to make it clear
# where they lie
ax.scatter([x_0, x_1], [y_0, y_1], s=150, c='grey')

# And now connect them
ax.plot([x_0, x_1], [y_0, y_1], c='grey',linestyle='dashed')

txt="Data Source: CFBD"
plt.figtext(0.9, 0.01, txt, wrap=True, horizontalalignment='center', fontsi

plt.show()
```



In [120]:
```
# create an instance of the API class
api_instance = cfbd.PlayersApi(cfbd.ApiClient(configuration))
year = 2021 # int | Year filter (optional)
#team = 'team_example' # str | Team filter (optional)
#conference = 'Pac-12' # str | Conference abbreviation filter (optional)

try:
    # Team returning production metrics
    api_response = api_instance.get_returning_production(year=year)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling PlayersApi->get_returning_production: %s\
```

```
[{'conference': 'Mountain West',
 'passing_usage': 1.0,
 'percent_passing_ppa': 1.0,
 'percent_ppa': 0.742,
 'percent_receiving_ppa': 0.54,
 'percent_rushing_ppa': 0.724,
 'receiving_usage': 0.745,
 'rushing_usage': 0.662,
 'season': 2021,
 'team': 'Air Force',
 'total_passing_ppa': 35.0,
 'total_ppa': 126.8,
 'total_receiving_ppa': 20.8,
 'total_rushing_ppa': 71.0,
 'usage': 0.724},
 {'conference': 'Mid-American',
 'passing_usage': 0.902,
 'percent_passing_ppa': 0.756,
 'percent_ppa': 0.611,
```

In [121]:
```
returning_df = pd.DataFrame.from_records([d.to_dict() for d in api_response
```

In [122]:
```
returning_df
```

Out[122]:

	season	team	conference	total_ppa	total_passing_ppa	total_receiving_ppa	total_rushin
0	2021	Air Force	Mountain West	126.8	35.0	20.8	
1	2021	Akron	Mid-American	27.5	-19.8	23.1	
2	2021	Alabama	SEC	148.6	16.7	90.1	
3	2021	Appalachian State	Sun Belt	279.8	12.1	165.3	
4	2021	Arizona	Pac-12	40.2	-15.2	41.4	
...	...	...	...	...	...	...	...
122	2021	Western Kentucky	Conference USA	63.8	-1.9	59.8	
123	2021	Western Michigan	Mid-American	224.8	101.8	75.7	
124	2021	West Virginia	Big 12	236.3	88.9	114.4	
125	2021	Wisconsin	Big Ten	105.8	20.5	56.8	
126	2021	Wyoming	Mountain West	102.9	8.1	41.7	

127 rows × 15 columns

In [123]:
```
api_instance = cfbd.RecruitingApi(cfbd.ApiClient(configuration))
year = 2021 # int | Recruiting class year (optional)
#team = 'team_example' # str | Team filter (optional)

try:
    # Team recruiting rankings and ratings
    api_response = api_instance.get_recruiting_teams(year=year)
    pprint(api_response)
except ApiException as e:
    print("Exception when calling RecruitingApi->get_recruiting_teams: %s\n
```

```
[{'points': 327.91, 'rank': 1, 'team': 'Alabama', 'year': 2021},
 {'points': 309.49, 'rank': 2, 'team': 'Ohio State', 'year': 2021},
 {'points': 294.55, 'rank': 3, 'team': 'Georgia', 'year': 2021},
 {'points': 291.37, 'rank': 4, 'team': 'LSU', 'year': 2021},
 {'points': 291.2, 'rank': 5, 'team': 'Clemson', 'year': 2021},
 {'points': 287.38, 'rank': 6, 'team': 'Oregon', 'year': 2021},
 {'points': 285.35, 'rank': 7, 'team': 'Texas A&M', 'year': 2021},
 {'points': 280.72, 'rank': 8, 'team': 'USC', 'year': 2021},
 {'points': 269.15, 'rank': 9, 'team': 'Notre Dame', 'year': 2021},
 {'points': 268.77, 'rank': 10, 'team': 'Michigan', 'year': 2021},
 {'points': 267.92, 'rank': 11, 'team': 'Oklahoma', 'year': 2021},
 {'points': 263.12, 'rank': 12, 'team': 'Miami', 'year': 2021},
 {'points': 261.62, 'rank': 13, 'team': 'Florida', 'year': 2021},
 {'points': 253.91, 'rank': 14, 'team': 'North Carolina', 'year': 2021},
 {'points': 240.8, 'rank': 15, 'team': 'Wisconsin', 'year': 2021},
 {'points': 240.4, 'rank': 16, 'team': 'Tennessee', 'year': 2021},
 {'points': 237.09, 'rank': 17, 'team': 'Texas', 'year': 2021},
 {'points': 234.56, 'rank': 18, 'team': 'Ole Miss', 'year': 2021},
 {'points': 230.29, 'rank': 19, 'team': 'Maryland', 'year': 2021},
```

In [124]:
```
recruiting_df = pd.DataFrame.from_records([d.to_dict() for d in api_respons
```

In [125]:
```
recruiting_df
```

Out[125]:

	year	rank	team	points
0	2021	1	Alabama	327.91
1	2021	2	Ohio State	309.49
2	2021	3	Georgia	294.55
3	2021	4	LSU	291.37
4	2021	5	Clemson	291.20
...	...	...	...	...
186	2021	189	Albany	7.31
187	2021	190	Montana	7.30
188	2021	191	Portland State	7.30
189	2021	192	Campbell	6.80
190	2021	193	Wofford	5.80

191 rows × 4 columns

In [126]:
```
production_df = pd.merge(returning_df, recruiting_df, on='team', how='outer
```

In [127]:
```
boolean_series = production_df.conference.isin(value_list)
production_df = production_df[boolean_series]
```

In [128]:
```
recruit_mean = production_df["points"].mean()
production_mean = production_df["percent_ppa"].mean()
```

In [129]:
```
production_mean
```

Out[129]:
```
0.677203125
```

In [130]:
```
production_df[production_df['team'] == 'Notre Dame']
```

Out[130]:

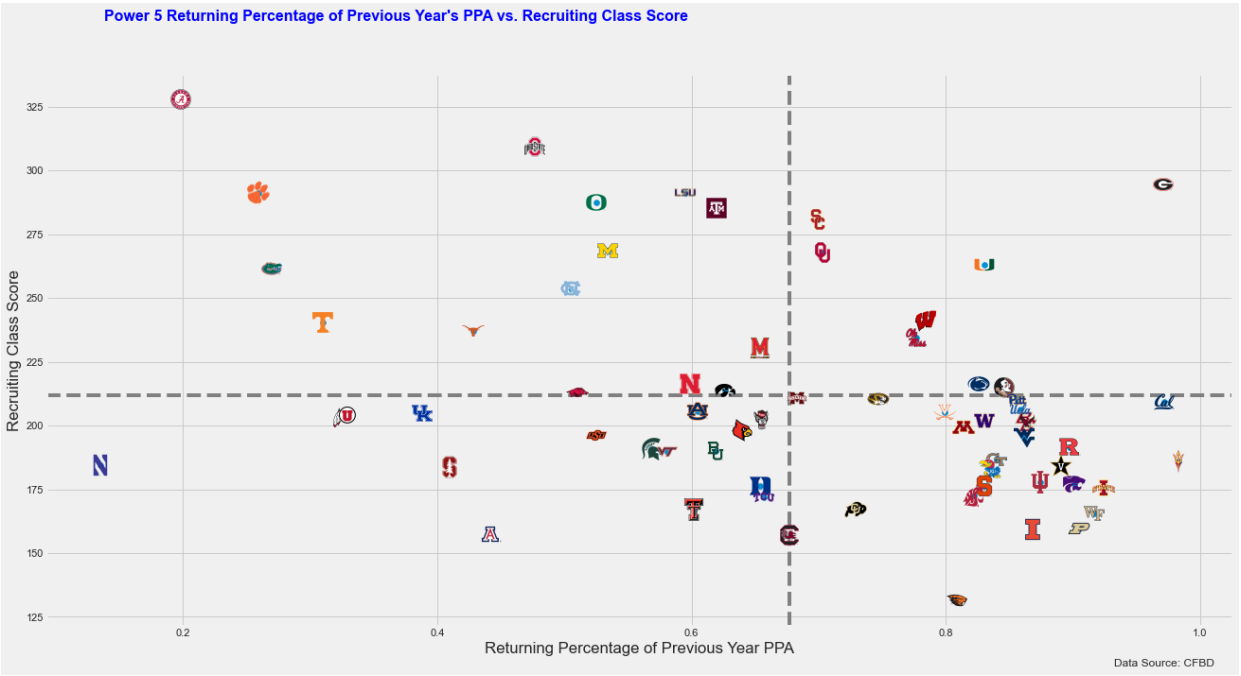
	season	team	conference	total_ppa	total_passing_ppa	total_receiving_ppa	total_rushing_ppa	pe

In [131]:

```
plt.style.use('fivethirtyeight')

# Graph sizing
plt.rcParams["figure.figsize"] = [20,10]


# You can download logos from here: https://drive.google.com/drive/folders/
# They come in two sizes. Extract them into the root of your script in a 'l
def getImage(path):
    return OffsetImage(plt.imread("./logos/{0}.png".format(path)))

# Logo file names are <team_id>.png, so we just need ids to map teams to lo
paths = production_df['team']

# Picking two random stats to plot
x = production_df['percent_ppa']
y = production_df['points']

fig, ax = plt.subplots()
ax.scatter(x, y)

# Cycle through each point and add an image annotation
for x0, y0, path in zip(x, y,paths):
    ab = AnnotationBbox(getImage(path), (x0, y0), frameon=False)
    ax.add_artist(ab)


# Define labels and title
# Title
fig.suptitle(
    "Power 5 Returning Percentage of Previous Year's PPA vs. Recruiting Cla
    x = 0.122,
    y = 0.975,
    ha="left",
    fontsize=16,
    color="BLUE",
    weight="bold",
)

# Subtitle
#ax.set_title(
#    "",
#    x = 0.122,
#    y = 0.975,
#    loc="left",
#    ha="left",
#    fontsize=12,
#    color="GREY",
#    weight="bold",
#    pad=10
#)


plt.axhline(y=recruit_mean, color='grey', linestyle='dashed')
plt.axvline(x=production_mean, color='grey', linestyle='dashed')
#plt.title("Offensive vs Defensive Predicted Points Added")
```

```
plt.xlabel('Returning Percentage of Previous Year PPA')
plt.ylabel('Recruiting Class Score')

# Now add on a line with a fixed slope of 0.03
#sl = 1

# A line with a fixed slope can intercept the axis
# anywhere so we're going to have it go through 0,0
#x_0 = 0
#y_0 = 0

# And we'll have the line stop at x = 5000
#x_1 = 0.6
#y_1 = sl*(x_1 - x_0) + y_0

# Draw these two points with big triangles to make it clear
# where they lie
#ax.scatter([x_0, x_1], [y_0, y_1], s=150, c='grey')

# And now connect them
#ax.plot([x_0, x_1], [y_0, y_1], c='grey',linestyle='dashed')

txt="Data Source: CFBD"
plt.figtext(0.9, 0.01, txt, wrap=True, horizontalalignment='center', fontsi

#plt.show()
```

Out[131]: Text(0.9, 0.01, 'Data Source: CFBD')



In [ ]: