In [ ]:
```python
import pandas as pd
import numpy as np

from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics import tsaplots

from pmdarima.arima import auto_arima
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab

# mape
from sklearn.metrics import mean_absolute_percentage_error

# dickey fuller
from statsmodels.tsa.stattools import adfuller
```

In [ ]:
```python
# # change font properties for plots
# font = {'size'   : 12}
# plt.rc('font', **font)
params = {'legend.fontsize': 'small',
          'figure.figsize': (15, 5),
         'axes.labelsize': 'medium',
         'axes.titlesize':'x-large',
         'xtick.labelsize':'medium',
         'ytick.labelsize':'medium'}

pylab.rcParams.update(params)
```

In [ ]:
```python
df = pd.read_csv('seasonSummaries.csv')
```

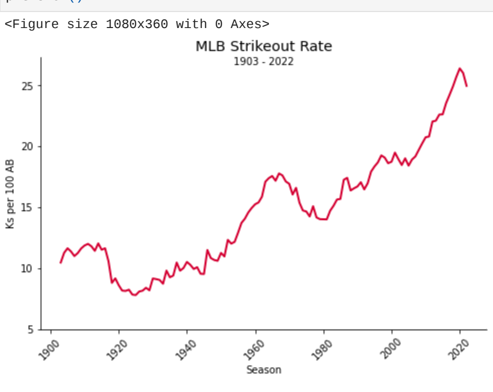
In [ ]:
```python
df.index = pd.period_range('1903','2022',freq='Y')
df = df[['Rate']]
```

In [ ]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
PeriodIndex: 120 entries, 1903 to 2022
Freq: A-DEC
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Rate    120 non-null    float64
dtypes: float64(1)
memory usage: 1.9 KB
```

In [ ]:
```python
# plot strikeout rate
plt.clf()
fig = plt.figure(figsize=(8,5))
ax = fig.add_subplot(111)
plt.plot(np.arange(1903,2023,1),df['Rate'],c='#D50032',linewidth=2)
plt.xlabel('Season')
plt.ylabel('Ks per 100 AB')
plt.title('MLB Strikeout Rate')
ax.text(y=.97,x=.5,s='1903 - 2022',transform=ax.transAxes, ha="center")
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.tick_params(bottom=True, top=False, left=True, right=False)
ax.tick_params(labelbottom=True, labeltop=False, labelleft=True, labelright=False)
plt.yticks(ticks=np.arange(5,26,5))
plt.xticks(np.arange(1900,2021,20))
plt.xticks(rotation=45)
plt.show()
```

```
<Figure size 1080x360 with 0 Axes>
```




In [ ]:
```python
# check if process is stationary
dickey = adfuller(df['Rate'])
print(f'Test Statistic: {dickey[0]}\nP-Value: {dickey[1]}')
```

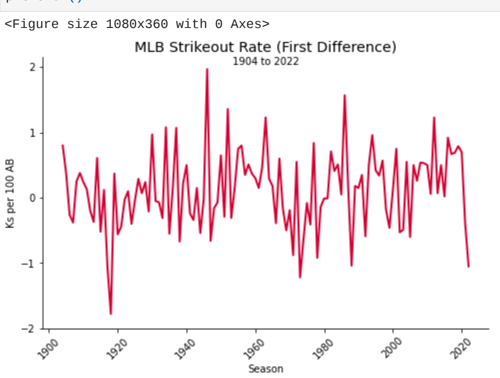
```
Test Statistic: 0.803940105582916
P-Value: 0.9916995032284984
```

not stationary

In [ ]:
```python
firstDifference = pd.DataFrame({'Rate':df['Rate'].diff()})
firstDifference = firstDifference.iloc[1:,]
```

In [ ]:
```python
# plot strikeout rate
plt.clf()
fig = plt.figure(figsize=(8,5))
ax = fig.add_subplot(111)
plt.plot(np.arange(1904,2023,1),firstDifference['Rate'],c='#D50032',linewidth=2)
plt.xlabel('Season')
plt.ylabel('Ks per 100 AB')
plt.title('MLB Strikeout Rate (First Difference)')
ax.text(y=.97,x=.5,s='1904 to 2022',transform=ax.transAxes, ha="center")
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.tick_params(bottom=True, top=False, left=True, right=False)
ax.tick_params(labelbottom=True, labeltop=False, labelleft=True, labelright=False)
plt.yticks(ticks=np.arange(-2,2.5,1))
plt.xticks(np.arange(1900,2021,20))
plt.xticks(rotation=45)
plt.show()
```

```
<Figure size 1080x360 with 0 Axes>
```

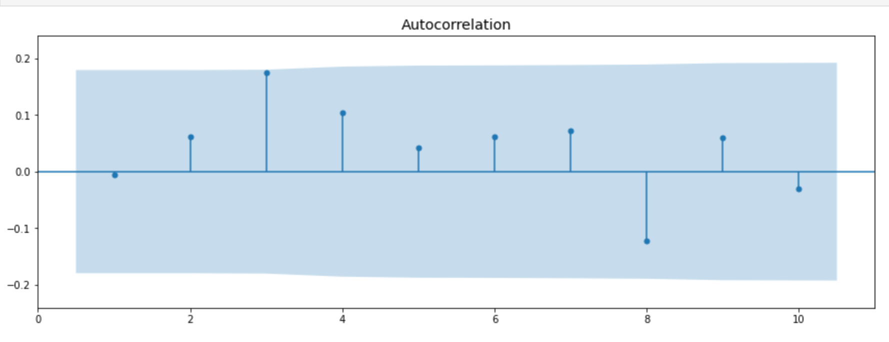



In [ ]:
```python
# check if process is stationary
dickey = adfuller(firstDifference['Rate'])
print(f'Test Statistic: {dickey[0]}\nP-Value: {dickey[1]}')
```

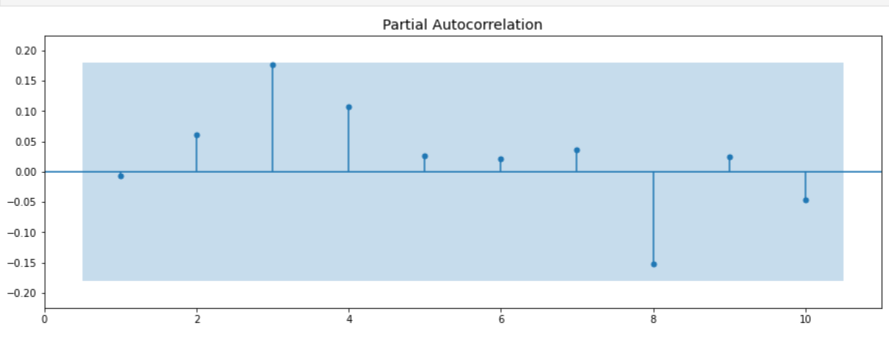
```
Test Statistic: -4.727596686209665
P-Value: 7.466119310757497e-05
```

Stationary

In [ ]:
```python
# plot acf function
tsaplots.plot_acf(firstDifference['Rate'],lags=10, auto_ylims=True, zero=False);
```




In [ ]:
```python
# plot pacf function
tsaplots.plot_pacf(firstDifference['Rate'], method='ywm',lags=10, auto_ylims=True, zero=False);
```




In [ ]:
```python
# split into a test and train dataset
train = df.iloc[:int(len(df)*.7),]
test = df.iloc[int(len(df)*.7)-1:,]
```

In [ ]:
```python
auto = auto_arima(train,
                  max_p=5,
                  max_q=5,
                  max_d=2,
                  max_order=14,
                  information_criterion='aicc',
                  seasonal=False,
                  supress_warings=True,
                  approximation=False,
                  error_action='warn',
                  stepwise=False,
                  trace=True)
```

```
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AICC=158.515, Time=0.07 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AICC=160.406, Time=0.04 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AICC=161.036, Time=0.03 sec
 ARIMA(0,1,3)(0,0,0)[0] intercept   : AICC=160.659, Time=0.05 sec
 ARIMA(0,1,4)(0,0,0)[0] intercept   : AICC=160.611, Time=0.06 sec
 ARIMA(0,1,5)(0,0,0)[0] intercept   : AICC=160.346, Time=0.07 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AICC=160.338, Time=0.02 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AICC=162.397, Time=0.04 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AICC=157.279, Time=0.06 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AICC=157.394, Time=0.08 sec
 ARIMA(1,1,4)(0,0,0)[0] intercept   : AICC=159.624, Time=0.12 sec
 ARIMA(1,1,5)(0,0,0)[0] intercept   : AICC=160.998, Time=0.14 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AICC=161.517, Time=0.03 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AICC=159.864, Time=0.07 sec
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AICC=156.710, Time=0.11 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AICC=155.630, Time=0.27 sec
 ARIMA(2,1,4)(0,0,0)[0] intercept   : AICC=162.027, Time=0.16 sec
 ARIMA(2,1,5)(0,0,0)[0] intercept   : AICC=163.510, Time=0.28 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AICC=159.428, Time=0.04 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AICC=159.917, Time=0.07 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AICC=inf, Time=0.22 sec
 ARIMA(3,1,3)(0,0,0)[0] intercept   : AICC=inf, Time=0.28 sec
 ARIMA(3,1,4)(0,0,0)[0] intercept   : AICC=inf, Time=0.33 sec
 ARIMA(3,1,5)(0,0,0)[0] intercept   : AICC=inf, Time=0.43 sec
 ARIMA(4,1,0)(0,0,0)[0] intercept   : AICC=160.443, Time=0.05 sec
 ARIMA(4,1,1)(0,0,0)[0] intercept   : AICC=162.171, Time=0.10 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AICC=157.812, Time=0.28 sec
 ARIMA(4,1,3)(0,0,0)[0] intercept   : AICC=inf, Time=0.32 sec
 ARIMA(4,1,4)(0,0,0)[0] intercept   : AICC=inf, Time=0.36 sec
 ARIMA(4,1,5)(0,0,0)[0] intercept   : AICC=inf, Time=0.42 sec
 ARIMA(5,1,0)(0,0,0)[0] intercept   : AICC=162.218, Time=0.07 sec
 ARIMA(5,1,1)(0,0,0)[0] intercept   : AICC=164.383, Time=0.11 sec
 ARIMA(5,1,2)(0,0,0)[0] intercept   : AICC=166.622, Time=0.26 sec
 ARIMA(5,1,3)(0,0,0)[0] intercept   : AICC=inf, Time=0.35 sec
 ARIMA(5,1,4)(0,0,0)[0] intercept   : AICC=inf, Time=0.39 sec
 ARIMA(5,1,5)(0,0,0)[0] intercept   : AICC=inf, Time=0.53 sec

Best model:  ARIMA(2,1,3)(0,0,0)[0] intercept
Total fit time: 6.363 seconds
```

In [ ]:
```python
model = ARIMA(train,order=auto.get_params()['order'])
results = model.fit(method_kwargs={"warn_convergence": False})
```

In [ ]:
```python
results.summary()
```

Out[ ]:

SARIMAX Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | Rate | **No. Observations:** | 84 |
| **Model:** | ARIMA(2, 1, 3) | **Log Likelihood** | -70.196 |
| **Date:** | Thu, 20 Oct 2022 | **AIC** | 152.391 |
| **Time:** | 13:59:10 | **BIC** | 166.904 |
| **Sample:** | 12-31-1903 | **HQIC** | 158.222 |
| | - 12-31-1986 | | |
| **Covariance Type:** | opg | | |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **ar.L1** | 1.8384 | 0.093 | 19.839 | 0.000 | 1.657 | 2.020 |
| **ar.L2** | -0.9202 | 0.093 | -9.908 | 0.000 | -1.102 | -0.738 |
| **ma.L1** | -2.0862 | 0.489 | -4.267 | 0.000 | -3.045 | -1.128 |
| **ma.L2** | 1.4404 | 0.641 | 2.246 | 0.025 | 0.184 | 2.697 |
| **ma.L3** | -0.2406 | 0.183 | -1.313 | 0.189 | -0.600 | 0.118 |
| **sigma2** | 0.2999 | 0.172 | 1.748 | 0.080 | -0.036 | 0.636 |

| | | | |
|---|---|---|---|
| **Ljung-Box (L1) (Q):** | 0.01 | **Jarque-Bera (JB):** | 7.68 |
| **Prob(Q):** | 0.94 | **Prob(JB):** | 0.02 |
| **Heteroskedasticity (H):** | 0.93 | **Skew:** | 0.18 |
| **Prob(H) (two-sided):** | 0.84 | **Kurtosis:** | 4.45 |

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [ ]:
```python
nAhead = len(test)
forecast = results.get_forecast(steps=nAhead)
alphaVal = .1
forecastFrame = forecast.summary_frame(alpha=alphaVal)
forecastFrame.columns = ['mean','se','lower','upper']
forecastFrame.head()
```

Out[ ]:

| | mean | se | lower | upper |
|---|---|---|---|---|
| **1987** | 17.370278 | 0.552176 | 16.462030 | 18.278527 |
| **1988** | 17.755143 | 0.695907 | 16.610477 | 18.899808 |
| **1989** | 18.101269 | 0.836009 | 16.726157 | 19.476381 |
| **1990** | 18.383414 | 0.986717 | 16.760409 | 20.006419 |
| **1991** | 18.583587 | 1.154534 | 16.684548 | 20.482625 |

In [ ]:
```python
forecastFrame.tail()
```

Out[ ]:

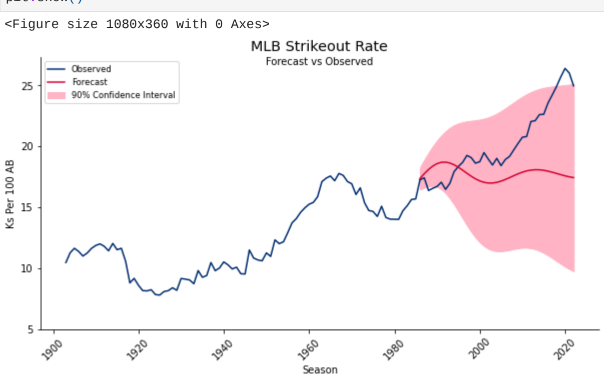
| | mean | se | lower | upper |
|---|---|---|---|---|
| **2019** | 17.780534 | 4.364382 | 10.601765 | 24.959304 |
| **2020** | 17.678707 | 4.446265 | 10.365252 | 24.992162 |
| **2021** | 17.583216 | 4.525319 | 10.139728 | 25.026704 |
| **2022** | 17.501373 | 4.600584 | 9.934085 | 25.068661 |
| **2023** | 17.438789 | 4.671483 | 9.754883 | 25.122696 |

In [ ]:
```python
trainYears = np.arange(1903,1987,1)
testYears = np.arange(1986,2023,1)
totalYears = np.arange(1903,2023,1)
```

In [ ]:
```python
# plot forecast over the test data set with all other seasons plotted too
plt.clf()
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(111)
plt.plot(trainYears,train['Rate'],label='Observed', color='#002D72')
plt.plot(testYears,test['Rate'][0:nAhead], color='#002D72')
plt.plot(testYears,forecastFrame['mean'],color='#D50032',label='Forecast')
plt.fill_between(testYears,forecastFrame['lower'],forecastFrame['upper'],color='#ffb3c4',label=f'{(1.0-alphaVal)*100:.0f}% Confidence Interval')
plt.xlabel('Season')
plt.ylabel('Ks Per 100 AB')

plt.yticks(ticks=np.arange(5,26,5))
plt.xticks(rotation = 45)
plt.legend(loc='upper left')
plt.title('MLB Strikeout Rate',loc='center')
#ax.text(y=.875,s='Predicted vs Observed', transform=ax.transAxes, ha="center")
ax.text(y=.97,x=.5,s='Forecast vs Observed',transform=ax.transAxes, ha="center")
#plt.suptitle('Predicted vs Observed', y=.875, fontsize=16)
#plt.suptitle('1987 to 2022', y=.875, fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.show()
```

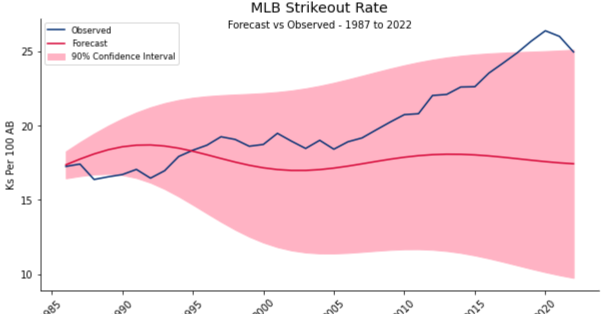
```
<Figure size 1080x360 with 0 Axes>
```




In [ ]:
```python
# plot forecast over the test data set with all other seasons plotted too
plt.clf()
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(111)
plt.plot(testYears,test['Rate'][0:nAhead], color='#002D72',label='Observed')
#plt.scatter(testYears,test['Rate'][0:nAhead], color='#002D72',label='Observed')
plt.plot(testYears,forecastFrame['mean'],color='#D50032',label='Forecast')
plt.fill_between(testYears,forecastFrame['lower'],forecastFrame['upper'],color='#ffb3c4',label=f'{(1.0-alphaVal)*100:.0f}% Confidence Interval')
plt.xlabel('Season')
plt.ylabel('Ks Per 100 AB')

plt.xticks(rotation = 45)
plt.legend(loc='upper left')
plt.title('MLB Strikeout Rate')
ax.text(y=.965,x=.5,s='Forecast vs Observed - 1987 to 2022',transform=ax.transAxes, ha="center")
plt.yticks(ticks=np.arange(10,26,5))
#plt.suptitle('1987 to 2022', y=.875, fontsize=16)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.show()
```

```
<Figure size 1080x360 with 0 Axes>
```




In [ ]:
```python
actual = test['Rate']
APE = []
for i in range(len(forecastFrame['mean'])):
    if(actual[i] != 0): #otherwise we get an infinite MAPE
        error = abs(actual[i] - forecastFrame['mean'][i])/(actual[i])
        #print(error)
        APE.append(error)
MAPE = np.mean(APE)
print(f'Mean Absolute Percentage Error = {round(MAPE*100,2)}%')
```

```
Mean Absolute Percentage Error = 13.65%
```

In [ ]:
```python
MAPE = mean_absolute_percentage_error(
            y_true = actual,
            y_pred=forecastFrame['mean'])
print(f'Mean Absolute Percentage Error = \
{round(MAPE*100,2)}%')
```

```
Mean Absolute Percentage Error = 13.65%
```

In [ ]:
```python
forecastLength = 10
forecastModel = ARIMA(df,order=auto.get_params()['order'])
forecastResults = forecastModel.fit()
futureForecast = forecastResults.get_forecast(steps=forecastLength)
futureForecastFrame = futureForecast.summary_frame(alpha=alphaVal)
futureForecastFrame.columns = ['mean','se','lower','upper']
futureForecastFrame.head()
```

Out[ ]:

| | mean | se | lower | upper |
|---|---|---|---|---|
| **2023** | 25.292433 | 0.585612 | 24.329187 | 26.255679 |
| **2024** | 25.277519 | 0.812124 | 23.941694 | 26.613343 |
| **2025** | 25.048211 | 1.009815 | 23.387212 | 26.709209 |
| **2026** | 24.899258 | 1.240154 | 22.859387 | 26.939129 |
| **2027** | 24.792224 | 1.475296 | 22.365579 | 27.218870 |

In [ ]:
```python
futureForecastFrame.loc[0] = [24.967 ,0,24.967,24.967 ]
```

In [ ]:
```python
futureForecastFrame.index = (list(np.arange(1,forecastLength+1)) + [0])
futureForecastFrame.sort_index(inplace=True)
```

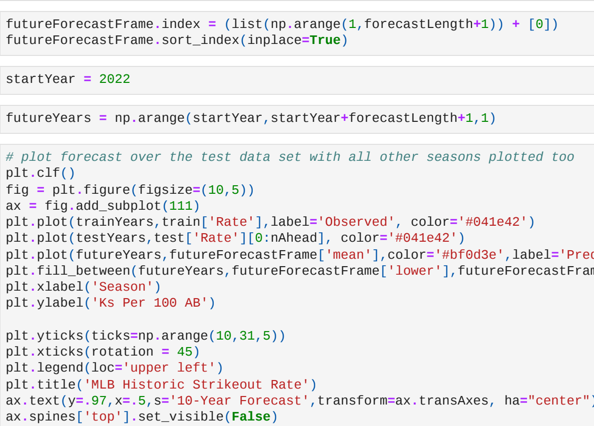
In [ ]:
```python
startYear = 2022
```

In [ ]:
```python
futureYears = np.arange(startYear,startYear+forecastLength+1,1)
```

In [ ]:
```python
# plot forecast over the test data set with all other seasons plotted too
plt.clf()
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(111)
plt.plot(trainYears,train['Rate'],label='Observed', color='#041e42')
plt.plot(testYears,test['Rate'][0:nAhead], color='#041e42')
plt.plot(futureYears,futureForecastFrame['mean'],color='#bf0d3e',label='Prediction')
plt.fill_between(futureYears,futureForecastFrame['lower'],futureForecastFrame['upper'],color='#ffb3c4',label=f'{(1.0-alphaVal)*100:.0f}% Confidence Interval')
plt.xlabel('Season')
plt.ylabel('Ks Per 100 AB')

plt.yticks(ticks=np.arange(10,31,5))
plt.xticks(rotation = 45)
plt.legend(loc='upper left')
plt.title('MLB Historic Strikeout Rate')
ax.text(y=.97,x=.5,s='10-Year Forecast',transform=ax.transAxes, ha="center")
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.show()
```

```
<Figure size 1080x360 with 0 Axes>
```




In [ ]:
```python
# plot forecast over the test data set with all other seasons plotted too
plt.clf()
fig = plt.figure(figsize=(10,5))
ax = fig.add_subplot(111)
#plt.plot(trainYears,train['Rate'],label='Observed', color='#041e42')
#plt.plot(testYears,test['Rate'][0:nAhead],label='Observed', color='#041e42')
plt.plot(futureYears,futureForecastFrame['mean'],color='#bf0d3e',label='Forecast')
plt.fill_between(futureYears,futureForecastFrame['lower'],futureForecastFrame['upper'],color='#ffb3c4',label=f'{(1.0-alphaVal)*100:.0f}% Confidence Interval')
plt.xlabel('Season')
plt.ylabel('Ks Per 100 AB')

plt.yticks(ticks=np.arange(20,31,2))
plt.xticks(ticks=np.arange(2022,2033,2))
plt.xticks(rotation = 45)
plt.legend(loc='upper left')
plt.title('MLB Strikeout Rate')
ax.text(y=.965,x=.5,s='10-Year Forecast',transform=ax.transAxes, ha="center")
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.show()
```

```
<Figure size 1080x360 with 0 Axes>
```