Economic Health Analysis

Mark Styx | IT542 | Final Project | 3/18/2020

Code: https://github.com/meow1928/economic_health/blob/master/analysis.py (https://github.com/meow1928/economic_health/blob/master/analysis.py)

Collect the headers and summary data output from HDFS/Spark

In [3]:
```
import os
os.chdir('S:\Anchor\economic_health')
```

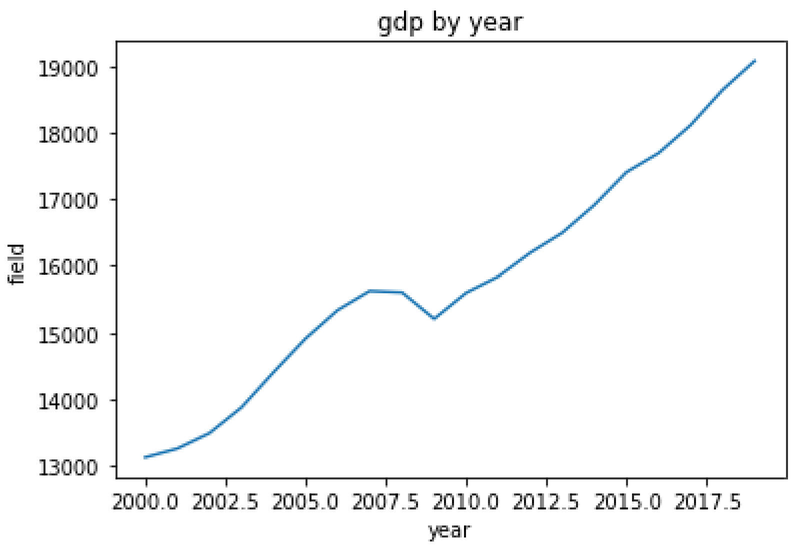
In [4]:
```
import pandas as pd

#get headers
with open('headers.txt','r') as f:
    headers = f.read()
headers = headers.split(',')
headers = [x for x in headers if x != '']

#load data
df = pd.read_csv('summary.csv',names=headers)

#normalize data types
for field in df.columns:
    df[field] = pd.to_numeric(df[field],errors='coerce')
```

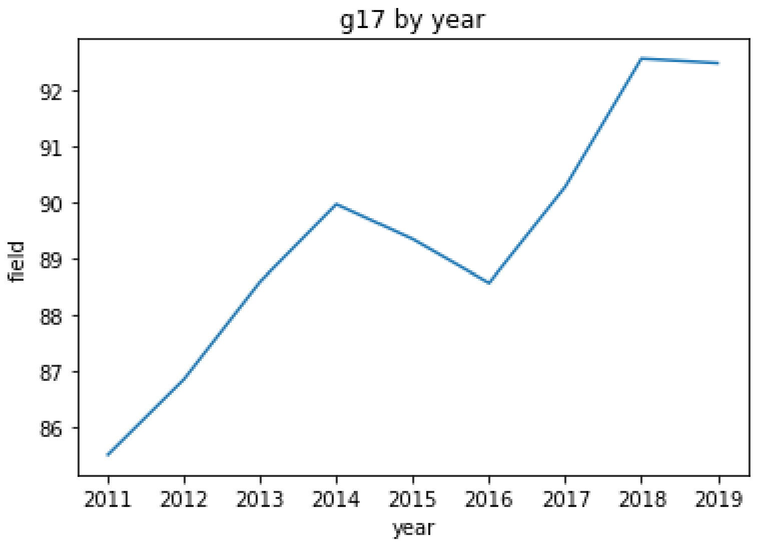
Create a function to generate graphs:

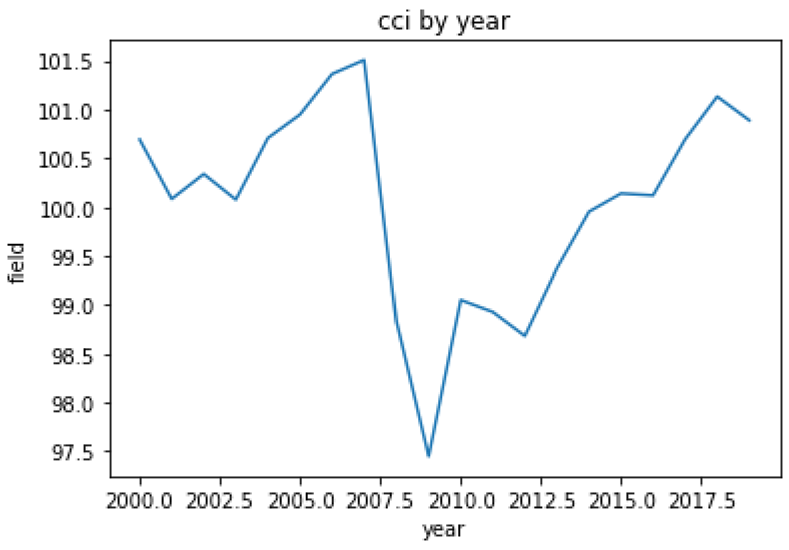
In [6]:
```
import matplotlib.pyplot as plt
import seaborn as sns

def plotvsyear(field):
    if field == 'year':
        return
    plt.plot(df['year'],df[field])
    plt.xlabel('year')
    plt.ylabel('field')
    plt.title('{0} by year'.format(field))
    plt.show()
    return
```

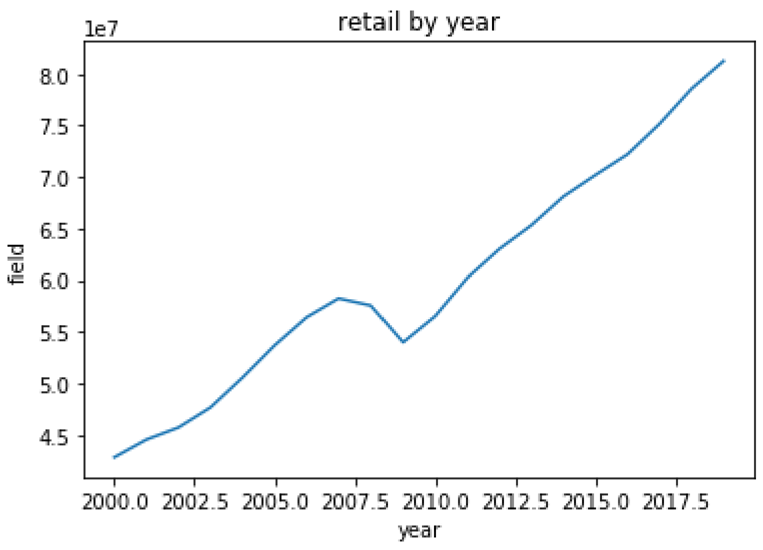
Create graphs (measure v year):

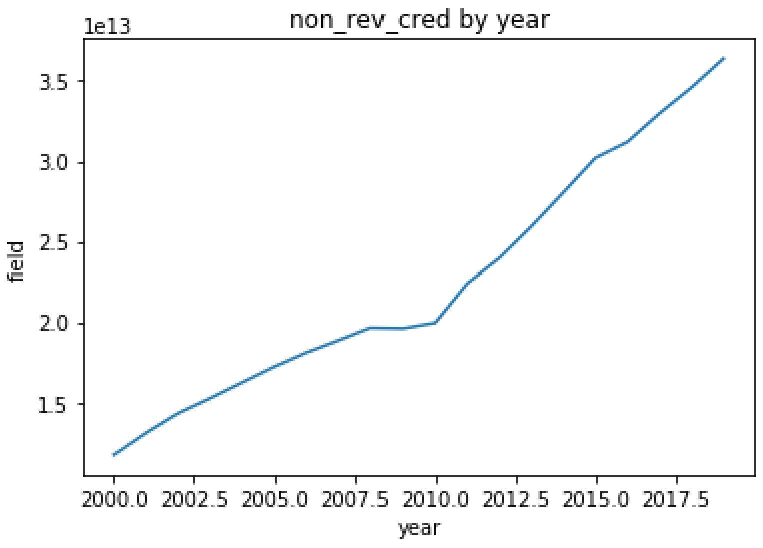
In [7]:
```
#plot by year
for field in df.columns:
    plotvsyear(field)
```

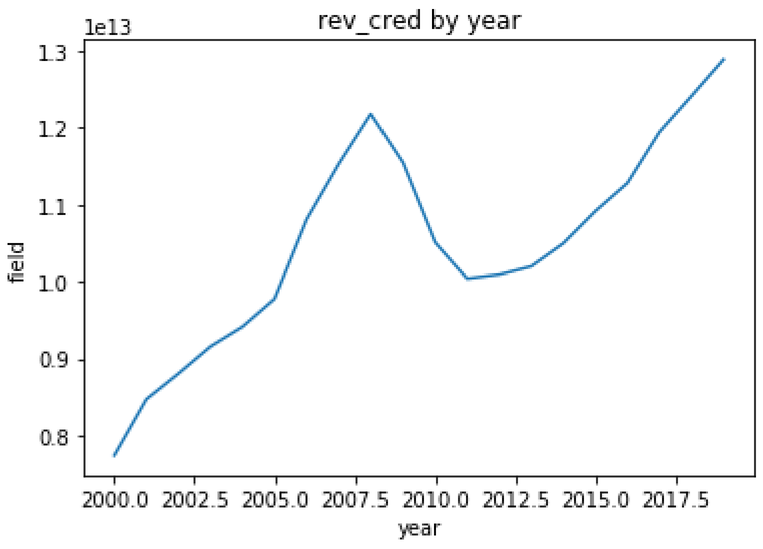

gdp by year
field
year
19000
18000
17000
16000
15000
14000
13000
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5

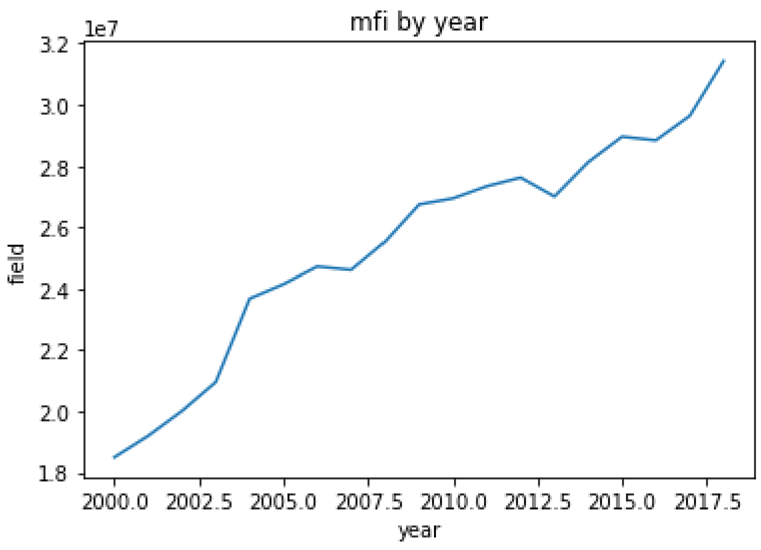

g17 by year
field
year
92
91
90
89
88
87
86
2011
2012
2013
2014
2015
2016
2017
2018
2019


cci by year
field
year
101.5
101.0
100.5
100.0
99.5
99.0
98.5
98.0
97.5
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5


retail by year
1e7
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
field
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
year


non_rev_cred by year
1e13
3.5
3.0
2.5
2.0
1.5
field
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
year


rev_cred by year
1e13
1.3
1.2
1.1
1.0
0.9
0.8
field
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
year



Since the scaling is off, it will make it hard to visualize the differences between measures so I'll take the z-score to normalize.

Create function to generate z-score:

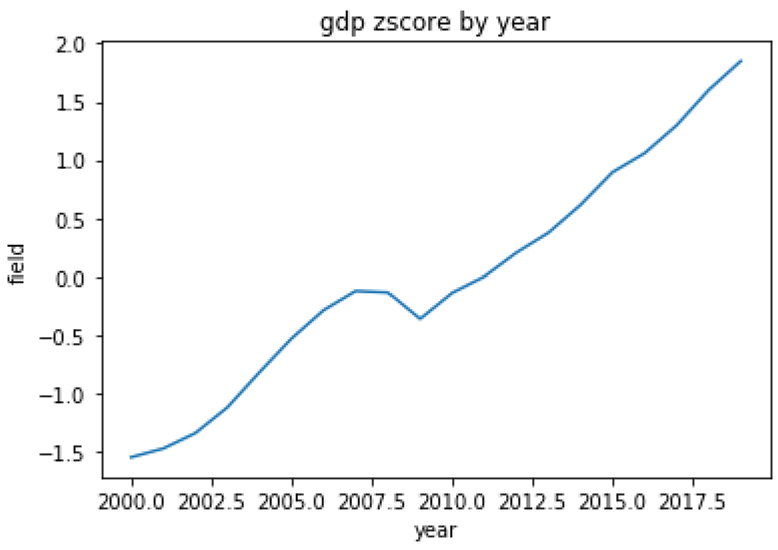
In [8]:
```
def zscores(field):
    if field == 'year':
        return
    mean = df[field].mean()
    std = df[field].std()
    xvalues = df[field].tolist()
    zscore = []
    for x in xvalues:
        zscore.append((x-mean)/std)
    new_field = str(field) + ' zscore'
    df[new_field] = zscore
    return
```

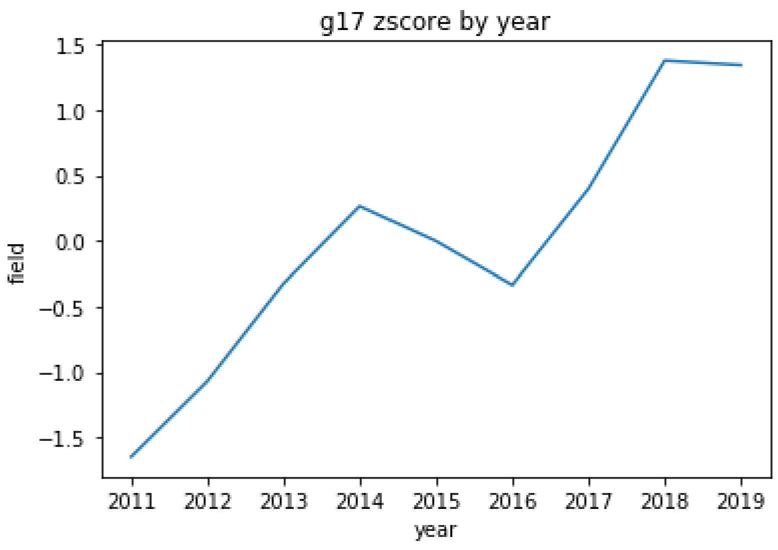
Generate z-scores:

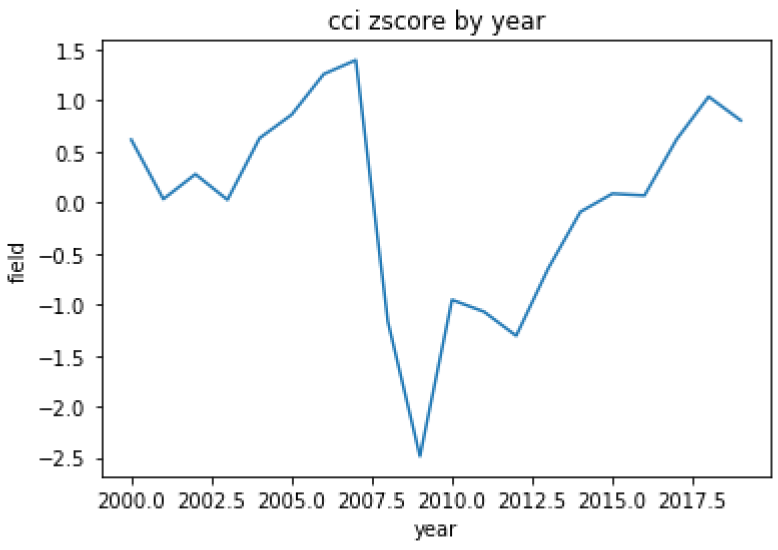
In [9]:
```
#generate zscores
for field in df.columns:
    zscores(field)
```

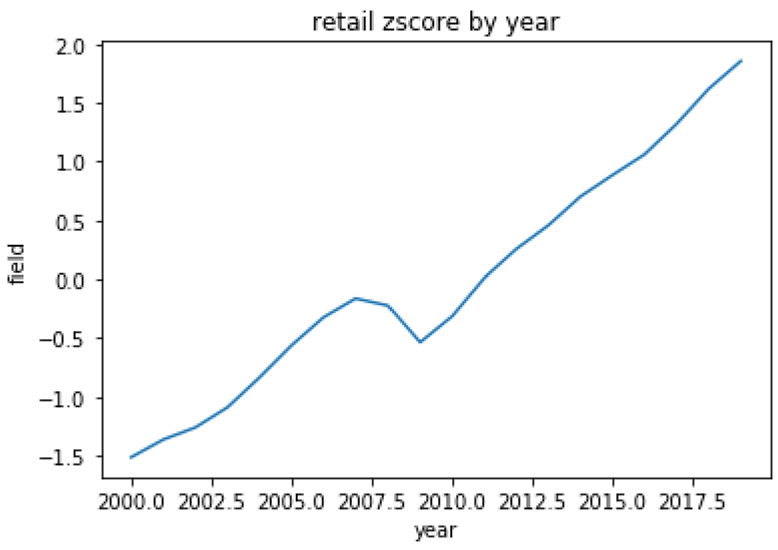
Plot z-scores by year:

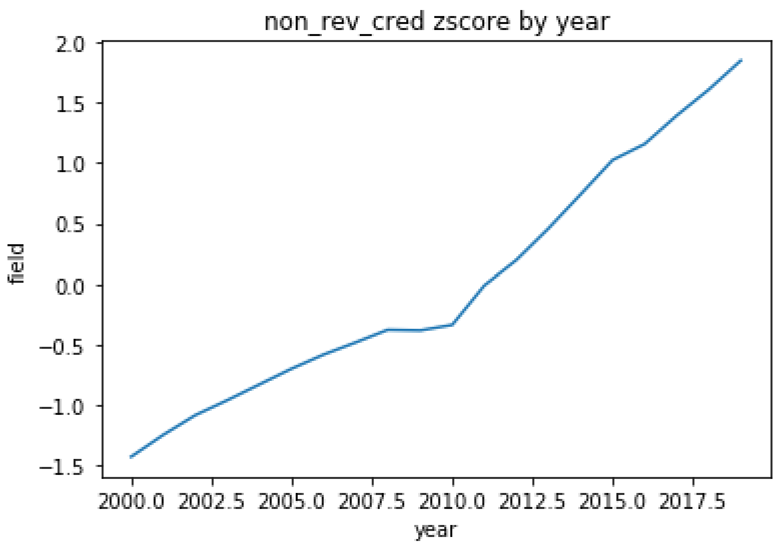
In [10]:
```
#plot zscores by year
for field in df.columns:
    if field.find('zscore') != -1:
        plotvsyear(field)
```

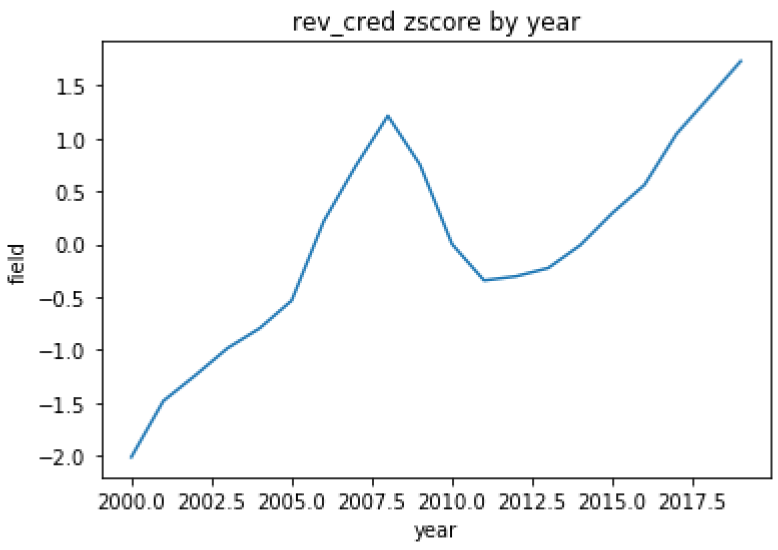

gdp zscore by year
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
field
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
year

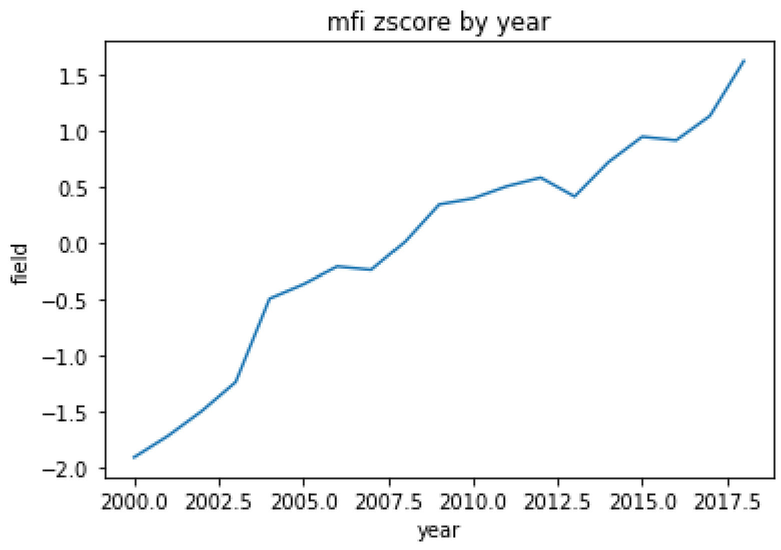

g17 zscore by year
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
field
2011
2012
2013
2014
2015
2016
2017
2018
2019
year


cci zscore by year
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
field
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
year


retail zscore by year
field
year


non_rev_cred zscore by year
field
year


rev_cred zscore by year
field
year



Check the correlations between measures:

In [11]:
```
zscores = [x for x in df.columns if x.find('zscore') != -1]

#correlations
corr = df[zscores].corr(method='pearson')
corr
```

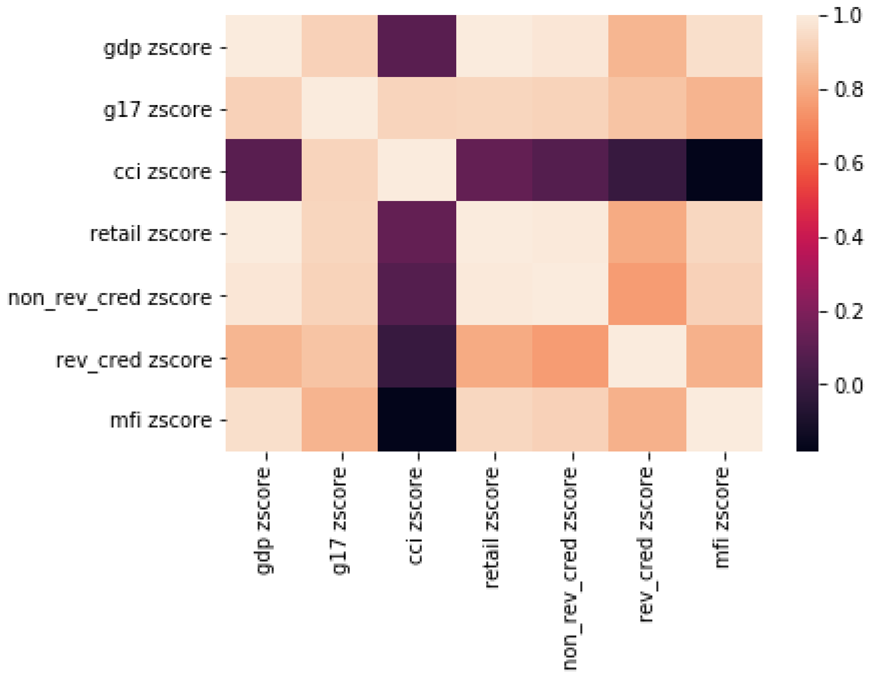
Out[11]:

	gdp zscore	g17 zscore	cci zscore	retail zscore	non_rev_cred zscore	rev_cred zscore	mfi zscore
gdp zscore	1.000000	0.918829	0.095377	0.997046	0.985192	0.837321	0.958801
g17 zscore	0.918829	1.000000	0.927307	0.933867	0.922410	0.877054	0.832536
cci zscore	0.095377	0.927307	1.000000	0.122321	0.079532	-0.007990	-0.179348
retail zscore	0.997046	0.933867	0.122321	1.000000	0.990911	0.806202	0.940031
non_rev_cred zscore	0.985192	0.922410	0.079532	0.990911	1.000000	0.763318	0.921576
rev_cred zscore	0.837321	0.877054	-0.007990	0.806202	0.763318	1.000000	0.820334
mfi zscore	0.958801	0.832536	-0.179348	0.940031	0.921576	0.820334	1.000000

Generate heatmap of correlation:

```
In [12]: sns.heatmap(corr)
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x24ff69a1978>
```



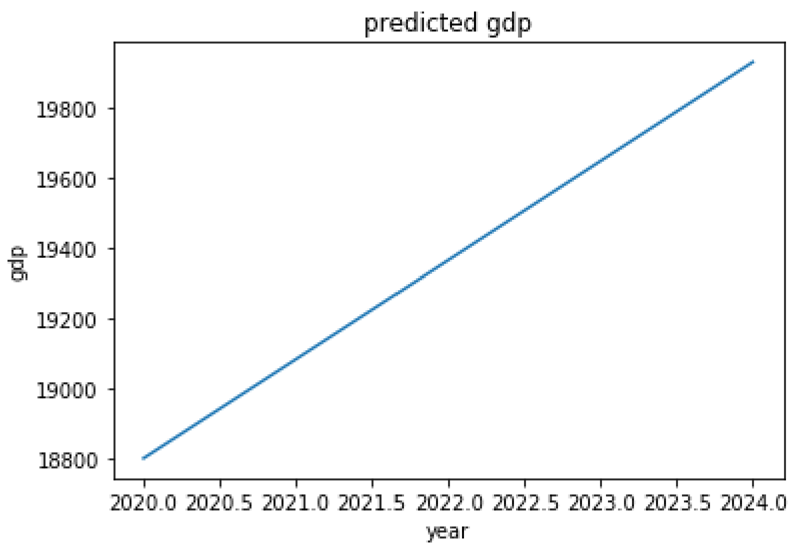
Create simple linear regression to predict gdp:

```
In [13]: #linear regression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

y = df['gdp'].values.reshape(-1,1)
X = df['year'].values.reshape(-1,1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, rando
m_state=0)
reg = LinearRegression()
gdp_pred = reg.fit(X_train,y_train)
results = gdp_pred.predict(X_test)
print('''
actual: {0}
predict: {1}'''.format(y_test,results))
```

```
actual: [[18638.2]
 [13262.1]]
predict: [[18234.73941723]
 [13422.73836055]]
```

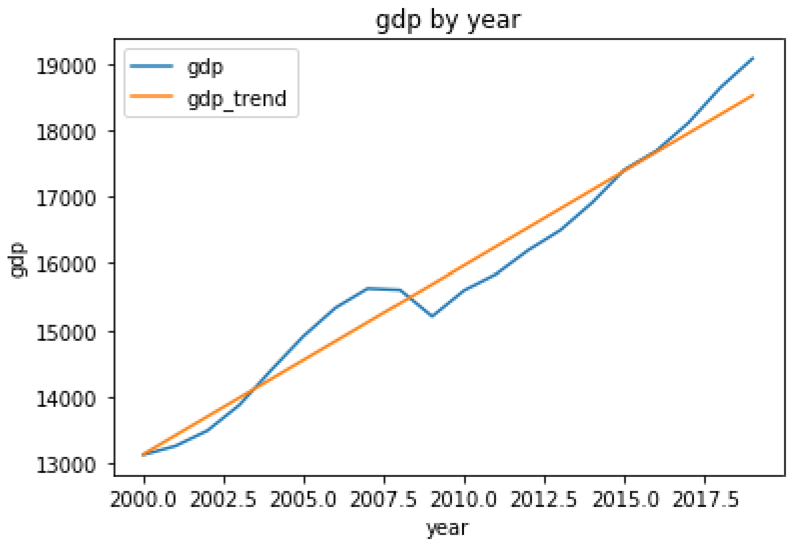
Predict gdp for the next five years:

In [14]:
```
#next five years gdp
next_five_years = pd.Series([2020,2021,2022,2023,2024]).values.reshape(-1,1)
n5 = gdp_pred.predict(next_five_years)
plt.plot(next_five_years,n5)
plt.xlabel('year')
plt.ylabel('gdp')
plt.title('predicted gdp')
plt.show()
```



Check trend vs actual:

In [15]:
```
#trend vs actual
trend = gdp_pred.predict(X)
df['gdp_trend'] = trend
plt.plot(df['year'],df[['gdp','gdp_trend']])
plt.xlabel('year')
plt.ylabel('gdp')
plt.title('gdp by year')
plt.legend(['gdp','gdp_trend'])
plt.show()
```



In [ ]: