In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
```

```
In D:\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_test.mplstyle:
The text.latex.preview rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
In D:\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_test.mplstyle:
The mathtext.fallback_to_cm rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
In D:\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_test.mplstyle: Support for setting the 'mathtext.fallback
_to_cm' rcParam is deprecated since 3.3 and will be removed two minor releases later; use 'mathtext.fallback : 'cm' instead.
In D:\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_test.mplstyle:
The validate_bool_maybe_none function was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
In D:\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_test.mplstyle:
The savefig.jpeg_quality rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
In D:\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_test.mplstyle:
The keymap.all_axes rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
In D:\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_test.mplstyle:
The animation.avconv_path rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
In D:\anaconda3\lib\site-packages\matplotlib\mpl-data\stylelib\_classic_test.mplstyle:
The animation.avconv_args rcparam was deprecated in Matplotlib 3.3 and will be removed two minor releases later.
```

In [2]:
```
data_raw1=pd.read_excel("上海银行间同业拆放利率(SHIBOR)(日).xls",index_col="指标名称").iloc[1:,:]
data_raw1.head()
```

Out[2]:

	SHIBOR:3个月
指标名称	
2013-12-31 00:00:00	5.5565
2014-01-02 00:00:00	5.5657
2014-01-03 00:00:00	5.5661
2014-01-06 00:00:00	5.5732
2014-01-07 00:00:00	5.576

In [3]:
```
data_raw2=pd.read_excel("社会融资规模存量(月).xls",index_col="指标名称").iloc[1:,:]
data_raw2.head()
```

Out[3]:

指标名称	社会融资规模存量:同比
2013-12-31 00:00:00	17.5
2014-12-31 00:00:00	14.3
2015-03-31 00:00:00	13
2015-06-30 00:00:00	11.9
2015-09-30 00:00:00	12.5

In [4]:
```
data_raw=pd.merge(data_raw1,data_raw2,on=["指标名称"]).dropna()
data_raw.head()
```

Out[4]:

指标名称	SHIBOR:3个月	社会融资规模存量:同比
2013-12-31 00:00:00	5.5565	17.5
2014-12-31 00:00:00	5.1351	14.3
2015-03-31 00:00:00	4.8975	13
2015-06-30 00:00:00	3.233	11.9
2015-09-30 00:00:00	3.153	12.5

In [5]:
```
#这里引入一个月未来函数。如果不用把30改为0
import datetime
data_raw.index=data_raw.index+datetime.timedelta(days=0)
data_raw.index=data_raw.index.strftime("%Y-%m-%d %H:%M:%S")
data_raw.head()
```

Out[5]:

	SHIBOR:3个月	社会融资规模存量:同比
2013-12-31 00:00:00	5.5565	17.5
2014-12-31 00:00:00	5.1351	14.3
2015-03-31 00:00:00	4.8975	13
2015-06-30 00:00:00	3.233	11.9
2015-09-30 00:00:00	3.153	12.5

In [6]:
```
#求二次差（加速度）作为好坏的判断标准
data_raw_accelerate=data_raw.diff(1).diff(1).dropna().apply(lambda x:np.where(x>=0,1,0))
#生成四个象限
data_raw_accelerate["state"]=0
for i in range(len(data_raw_accelerate)):
    info=data_raw_accelerate.iloc[i,:]
    if info[0]==1 and info[1]==1:data_raw_accelerate.iloc[i,2]=1
    if info[0]==1 and info[1]==0:data_raw_accelerate.iloc[i,2]=2
    if info[0]==0 and info[1]==1:data_raw_accelerate.iloc[i,2]=3
    if info[0]==0 and info[1]==0:data_raw_accelerate.iloc[i,2]=4
data_raw_accelerate.index.name="日期"
data_raw_accelerate.head()
```

Out[6]:

日期	SHIBOR:3个月	社会融资规模存量:同比	state
2015-03-31 00:00:00	1	1	1
2015-06-30 00:00:00	0	1	3
2015-09-30 00:00:00	1	1	1
2015-12-31 00:00:00	1	0	2
2016-02-29 00:00:00	0	1	3

获取中国四类资产的数据

股票：上证指数，债券：中证全债，商品：南华商品指数，现金：货币基金指数

industry=pd.read_excel("四品种数据.xlsx",index_col="日期") industry.head()

In [7]:
```
#引用申万一级行业指数
industry=pd.read_excel("申万指数.xlsx",index_col="日期")
```

In [8]:
```
#直接merge得到空dataframe，现在先全部转成datatime格式
industry["a"]=industry.index
industry["a"]=industry["a"].apply(pd.to_datetime,format='%Y-%m-%d')
data_raw_accelerate["a"]=data_raw_accelerate.index
data_raw_accelerate["a"]=data_raw_accelerate["a"].apply(pd.to_datetime,format='%Y-%m-%d')
```

In [18]:
```
#合并表格
data=pd.merge(data_raw_accelerate,industry,on=["a"],how='inner').iloc[:-1,2:]
data.index=data["a"]
del data["a"]
data.index.name="日期"
#计算两期间收益率
data.iloc[:,1:]=data.iloc[:,1:]/data.iloc[:,1:].shift(1)-1
data=data.dropna()

#计算各个指标的年化收益率
for i in range(1,len(data)):
    #计算两个周期相差多少天
    day=(data.index[i]-data.index[i-1]).days
    data.iloc[i,1:]=data.iloc[i,1:].apply(lambda x:float(x)/(day/252))
#求年化率增速的话取消下面这行注释
#data.iloc[:,1:]=data.iloc[:,1:].diff(1)
data=data.dropna()
data.head()
```

Out[18]:

日期	state	农林牧渔(申万)801010.SI	采掘(申万)801020.SI	化工(申万)801030.SI	钢铁(申万)801040.SI	有色金属(申万)801050.SI	电子(申万)801080.SI	家用电器(申万)801110.SI	食品饮料(申万)801120.SI	纺织服装(申万)801130.SI	...	建筑装饰(申万)80172(
2015-06-30	3	0.294742	0.230665	0.272302	0.280517	0.108403	0.292620	0.276865	0.256250	0.340913	...	0.108
2015-09-30	1	-0.740251	-1.059129	-0.891747	-1.082408	-1.000148	-0.921325	-1.025533	-0.712004	-0.794018	...	-0.790
2015-12-31	2	0.783720	0.398586	0.958526	0.149762	0.821015	1.123479	0.912438	0.518862	0.968244	...	0.337
2016-02-29	3	-0.959270	-0.618099	-1.197261	-0.965067	-0.960983	-1.281740	-1.031175	-0.879004	-1.170137	...	-1.130
2016-03-31	1	1.183457	0.361372	1.318544	0.567238	1.220931	1.450746	1.020326	1.326727	1.357729	...	1.217

5 rows × 29 columns

In [14]:
```
#统计各区间累计收益
result=data.groupby("state").apply(np.mean).iloc[:,1:].T
print(result.sort_values(by=1,ascending=False).head())
print(result.sort_values(by=2,ascending=False).head())
print(result.sort_values(by=3,ascending=False).head())
print(result.sort_values(by=4,ascending=False).head())
```

```
state                        1         2         3         4
食品饮料(申万)801120.SI   0.170895  0.283117 -0.014160  0.597749
家用电器(申万)801110.SI   0.117799  0.225440 -0.121509  0.501816
电子(申万)801080.SI       0.103140  0.517711 -0.103663  0.244947
医药生物(申万)801150.SI   0.095114  0.335250 -0.209926  0.380913
休闲服务(申万)801210.SI   0.088980  0.388552 -0.239301  0.819947
state                        1         2         3         4
电子(申万)801080.SI       0.103140  0.517711 -0.103663  0.244947
计算机(申万)801750.SI     0.033262  0.501526 -0.192547  0.140548
非银金融(申万)801790.SI   0.019266  0.433100 -0.195352  0.268456
通信(申万)801770.SI       0.032910  0.419963 -0.180740  0.018333
休闲服务(申万)801210.SI   0.088980  0.388552 -0.239301  0.819947
state                        1         2         3         4
有色金属(申万)801050.SI -0.113322  0.286257  0.109306  0.291066
钢铁(申万)801040.SI     -0.059851  0.170135  0.049759 -0.082446
建筑材料(申万)801710.SI   0.053659  0.267081  0.032779  0.331678
食品饮料(申万)801120.SI   0.170895  0.283117 -0.014160  0.597749
国防军工(申万)801740.SI -0.112659  0.265196 -0.054330  0.412312
state                        1         2         3         4
休闲服务(申万)801210.SI   0.088980  0.388552 -0.239301  0.819947
食品饮料(申万)801120.SI   0.170895  0.283117 -0.014160  0.597749
家用电器(申万)801110.SI   0.117799  0.225440 -0.121509  0.501816
国防军工(申万)801740.SI -0.112659  0.265196 -0.054330  0.412312
医药生物(申万)801150.SI   0.095114  0.335250 -0.209926  0.380913
```

In [15]:
```
print(result.T.mean().sort_values(ascending=False).head())
print(result.mean())
```

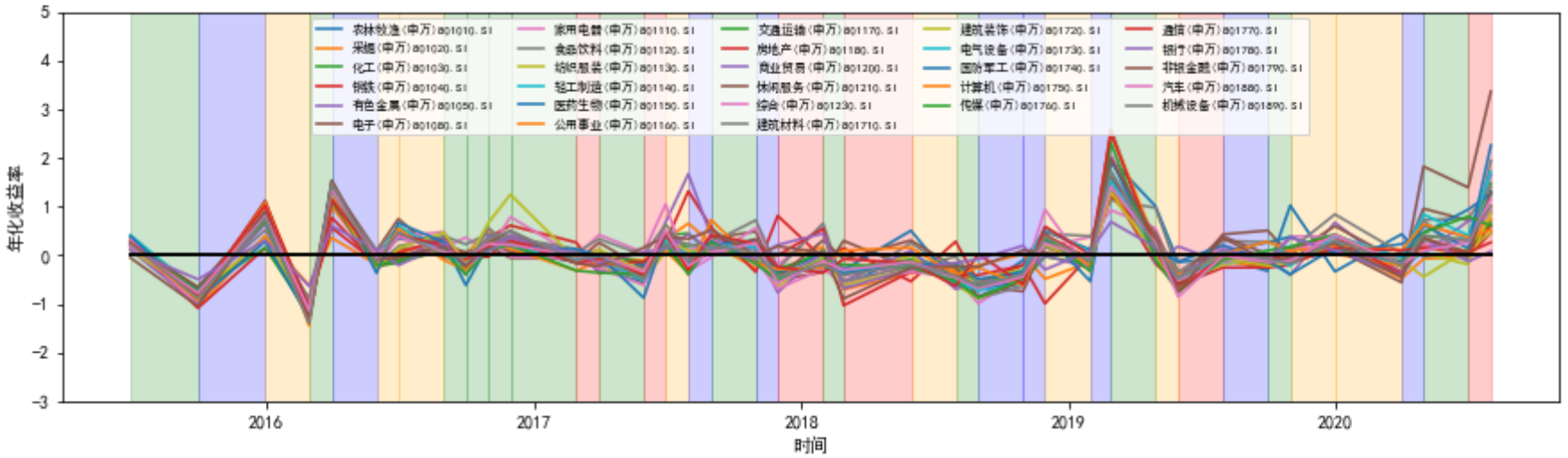
```
休闲服务(申万)801210.SI    0.264545
食品饮料(申万)801120.SI    0.259400
电子(申万)801080.SI       0.190534
家用电器(申万)801110.SI    0.180887
建筑材料(申万)801710.SI    0.171299
dtype: float64
state
1     0.008560
2     0.265156
3    -0.128115
4     0.223144
dtype: float64
```

In [16]:
```
#画状态图
plt.figure(figsize=(15, 4))
x=data.index
for i in range(1,len(data.index)):
    if data.iloc[i,0]==1:
        plt.axvspan(x[i-1], x[i], ymin=1.5, ymax=-1.5, alpha=0.2, color='green')
    if data.iloc[i,0]==2:
        plt.axvspan(x[i-1], x[i], ymin=1.5, ymax=-1.5, alpha=0.2, color='blue')
    if data.iloc[i,0]==3:
        plt.axvspan(x[i-1], x[i], ymin=1.5, ymax=-1.5, alpha=0.2, color='orange')
    if data.iloc[i,0]==4:
        plt.axvspan(x[i-1], x[i], ymin=1.5, ymax=-1.5, alpha=0.2, color='red')
plt.plot(data.iloc[:,1:],)
plt.plot([x[0], x[-1]],[0,0],"black",linewidth=2)
plt.legend(data.columns[1:],loc="upper center",fontsize="x-small",ncol=5)
plt.ylim([-3,5])
plt.xlabel("时间")
plt.ylabel("年化收益率")
print("绿色：货币+信用+，蓝色：货币+信用-，橙色：货币-信用+，红色：货币-信用-")
```

```
D:\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\converter.py:103: FutureWarning: Using an implicitly registered dateti
me converter for a matplotlib plotting method. The converter was registered by pandas on import. Future versions of pandas will r
equire you to explicitly register matplotlib converters.

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```

绿色：货币+信用+，蓝色：货币+信用-，橙色：货币-信用+，红色：货币-信用-



In [19]:
```
#四期rolling mean再画图
data_rolling=data
data_rolling.iloc[:,1:]=data_rolling.iloc[:,1:].rolling(4).mean()
data_rolling=data_rolling.dropna()
plt.figure(figsize=(15, 4))
x=data_rolling.index
for i in range(1,len(data_rolling.index)):
    if data_rolling.iloc[i,0]==1:
        plt.axvspan(x[i-1], x[i], ymin=1.5, ymax=-1.5, alpha=0.2, color='green')
    if data_rolling.iloc[i,0]==2:
        plt.axvspan(x[i-1], x[i], ymin=1.5, ymax=-1.5, alpha=0.2, color='blue')
    if data_rolling.iloc[i,0]==3:
        plt.axvspan(x[i-1], x[i], ymin=1.5, ymax=-1.5, alpha=0.2, color='orange')
    if data_rolling.iloc[i,0]==4:
        plt.axvspan(x[i-1], x[i], ymin=1.5, ymax=-1.5, alpha=0.2, color='red')
plt.plot(data_rolling.iloc[:,1:])
plt.plot([x[0], x[-1]],[0,0],"black",linewidth=2)
plt.ylim([-1,1.5])
plt.legend(data.columns[1:],loc="upper center",fontsize="x-small",ncol=5)
plt.xlabel("时间")
plt.ylabel("年化收益率")
print("绿色：货币+信用+，蓝色：货币+信用-，橙色：货币-信用+，红色：货币-信用-")
```

绿色：货币+信用+，蓝色：货币+信用-，橙色：货币-信用+，红色：货币-信用-