In [4]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pylab import mpl
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
x = np.random.random(29)
weights = x / np.sum(x)
print(weights)
```

```
[0.00303938 0.01831665 0.04579018 0.06377905 0.0141596  0.06837427
 0.0086646  0.00918181 0.00780264 0.00353815 0.05479855 0.06563616
 0.05892422 0.02564696 0.01126633 0.04425106 0.07108868 0.05148192
 0.05051865 0.02956066 0.01128385 0.0557344  0.00215386 0.03882185
 0.02242972 0.06614803 0.03620087 0.02380684 0.03760109]
```

In [6]:

```
data = pd.read_excel(r'C:\Users\jzc05\OneDrive\Desktop\test.xlsx', sheet_name = 0, head
er = 0, index_col = 0)
data.head(5)
```

Out[6]:

日期	美国运通	波音	卡特彼勒	雪佛龙	迪士尼	高盛	家得宝	IBM	强生	摩根大通	...	联合技术
2015-01-02	93.02	129.95	91.88	112.58	93.75	194.41	103.43	162.06	104.52	62.49	...	115.04
2015-01-05	90.56	129.05	87.03	108.08	92.38	188.34	101.26	159.51	103.79	60.55	...	113.12
2015-01-06	88.63	127.53	86.47	108.03	91.89	184.53	100.95	156.07	103.28	58.98	...	111.52
2015-01-07	90.30	129.51	87.81	107.94	92.83	187.28	104.41	155.05	105.56	59.07	...	112.73
2015-01-08	91.58	131.80	88.71	110.41	93.79	190.27	106.72	158.42	106.39	60.39	...	114.65

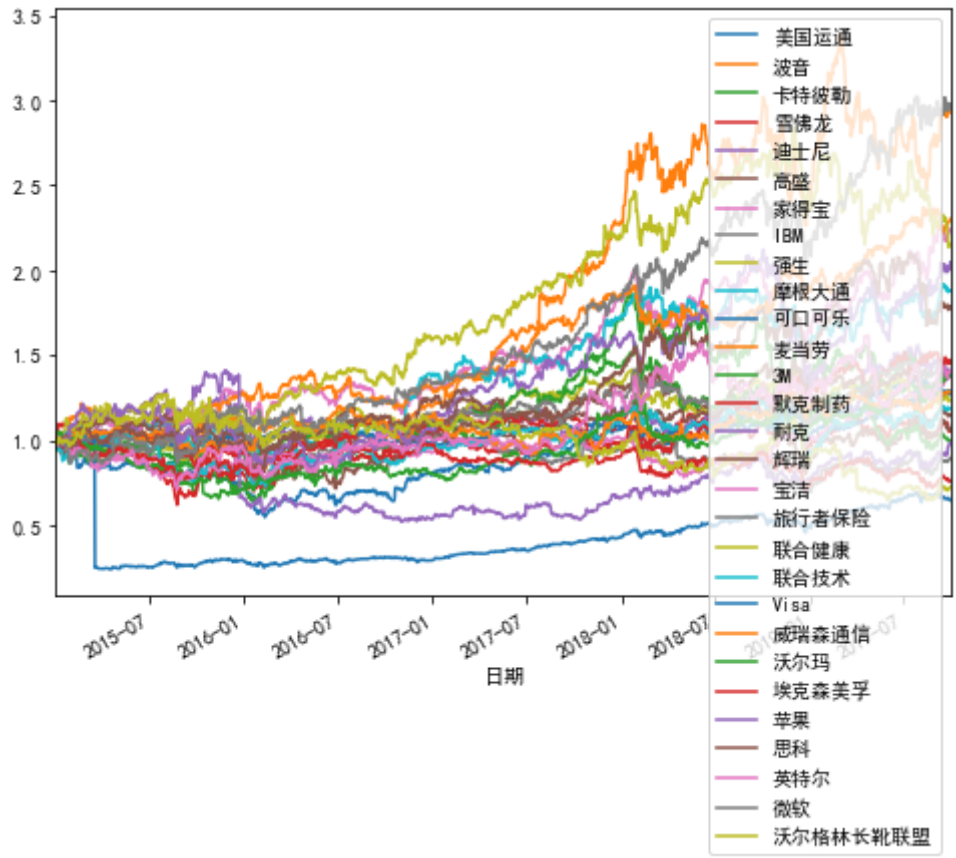
5 rows × 29 columns

In [7]:

```
(data/data.iloc[0]).plot(figsize = (8, 6))
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1eae6533388>
```



In [8]:

```
R = np.log(data / data.shift(1))
R = R.dropna()
R.describe()
```

Out[8]:

	美国运通	波音	卡特彼勒	雪佛龙	迪士尼	高盛	家
count	1193.000000	1193.000000	1193.000000	1193.000000	1193.000000	1193.000000	1193.00
mean	0.000201	0.000900	0.000267	0.000044	0.000276	0.000054	0.00
std	0.013132	0.015854	0.017040	0.014023	0.012477	0.015185	0.01
min	-0.128981	-0.093531	-0.095698	-0.057276	-0.096190	-0.077482	-0.05
25%	-0.005327	-0.007148	-0.008118	-0.006919	-0.005563	-0.007846	-0.00
50%	0.000690	0.001107	0.000347	0.000095	0.000447	0.000462	0.00
75%	0.006940	0.009653	0.009615	0.007267	0.006377	0.008633	0.00
max	0.086440	0.094214	0.075671	0.061446	0.109247	0.091153	0.06

8 rows × 29 columns

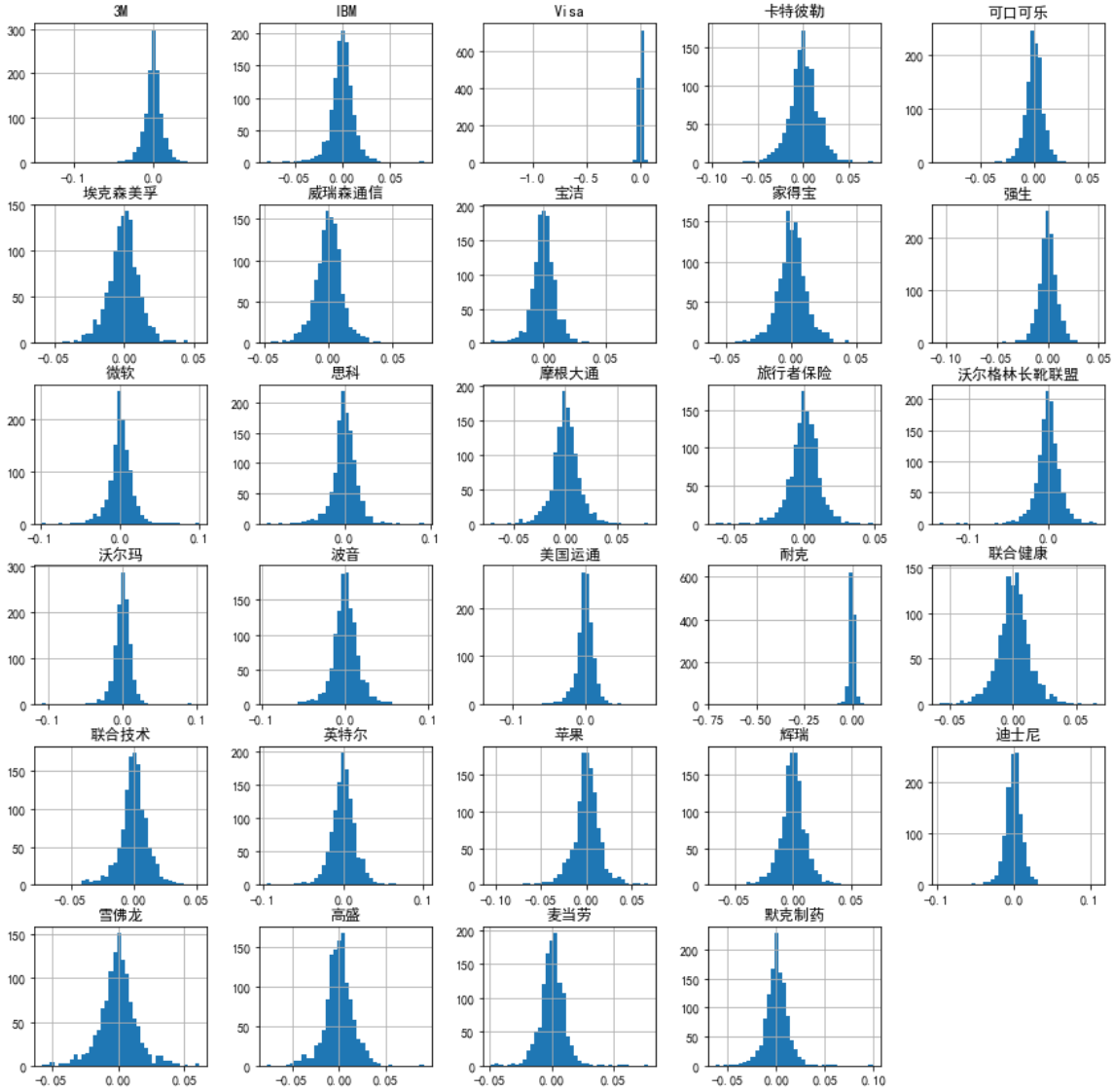
In [13]:

```
R.hist(bins = 40, figsize = (15, 15))
```

Out[13]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001EAEF55290
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF12E5F4
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF1A67BC
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF1C65CC
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF19E774
8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF1A4188
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF1CCC8C
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF1CE79C
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF1CED5C
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF14C378
8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF14ECD0
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF1504DC
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF1526F0
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF155404
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF158D14
8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF15C628
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF15FE38
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF163840
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF166F54
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF16AEE4
8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF16E074
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF171988
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF175194
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF1789A4
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF17C3B4
8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF17FAC8
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF1833D8
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF186CE4
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF18A4F8
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001EAF18E10C
```

```
8>]],
      dtype=object)
```


3M
IBM
Visa
卡特彼勒
可口可乐
埃克森美孚
威瑞森通信
宝洁
家得宝
强生
微软
思科
摩根大通
旅行者保险
沃尔格林长靴联盟
沃尔玛
波音
美国运通
耐克
联合健康
联合技术
英特尔
苹果
辉瑞
迪士尼
雪佛龙
高盛
麦当劳
默克制药

In [15]:

```
R_mean = R.mean() * 252
print(R_mean.head(5))
```

```
美国运通    0.050746
波音      0.226918
卡特彼勒    0.067226
雪佛龙     0.011004
迪士尼     0.069572
dtype: float64
```

In [16]:

```
R_cov = R.cov() * 252
print(R_cov.head(5))
```

```
          美国运通        波音       卡特彼勒        雪佛龙        迪士尼
高盛        家得宝  \
美国运通  0.043460  0.020932  0.025055  0.016009  0.015150  0.028219
0.016999
波音    0.020932  0.063339  0.035432  0.021348  0.018798  0.029371  0.
020636
卡特彼勒  0.025055  0.035432  0.073170  0.031451  0.018738  0.036883
0.022027
雪佛龙   0.016009  0.021348  0.031451  0.049555  0.013550  0.024629
0.015352
迪士尼   0.015150  0.018798  0.018738  0.013550  0.039230  0.020991
0.016054

           IBM        强生       摩根大通  ...       联合技术       Visa
威瑞森通信  \
美国运通  0.018459  0.011594  0.025940  ...  0.018427  0.023355  0.0083
39
波音    0.021530  0.014558  0.026568  ...  0.025566  0.027363  0.01042
6
卡特彼勒  0.024892  0.013439  0.032085  ...  0.027146  0.029344  0.0103
96
雪佛龙   0.018209  0.012837  0.023545  ...  0.016955  0.023194  0.0125
21
迪士尼   0.016027  0.010131  0.019924  ...  0.015688  0.019616  0.0102
92

           沃尔玛       埃克森美孚          苹果          思科        英特尔
微软   沃尔格林长靴联盟
美国运通  0.009955  0.013387  0.018588  0.019495  0.019055  0.021747
0.017749
波音    0.013829  0.019605  0.027119  0.025000  0.026177  0.025572  0.
017815
卡特彼勒  0.012359  0.027573  0.029347  0.029017  0.031589  0.029883
0.019180
雪佛龙   0.009901  0.032530  0.017939  0.021065  0.020770  0.021098
0.015840
迪士尼   0.010661  0.014695  0.018765  0.019939  0.019370  0.018912
0.016088

[5 rows x 29 columns]
```

In [17]:

```
R_vol = R.std() * np.sqrt(252)
print(R_vol.head(5))
```

```
美国运通    0.208471
波音       0.251672
卡特彼勒    0.270499
雪佛龙      0.222611
迪士尼      0.198065
dtype: float64
```

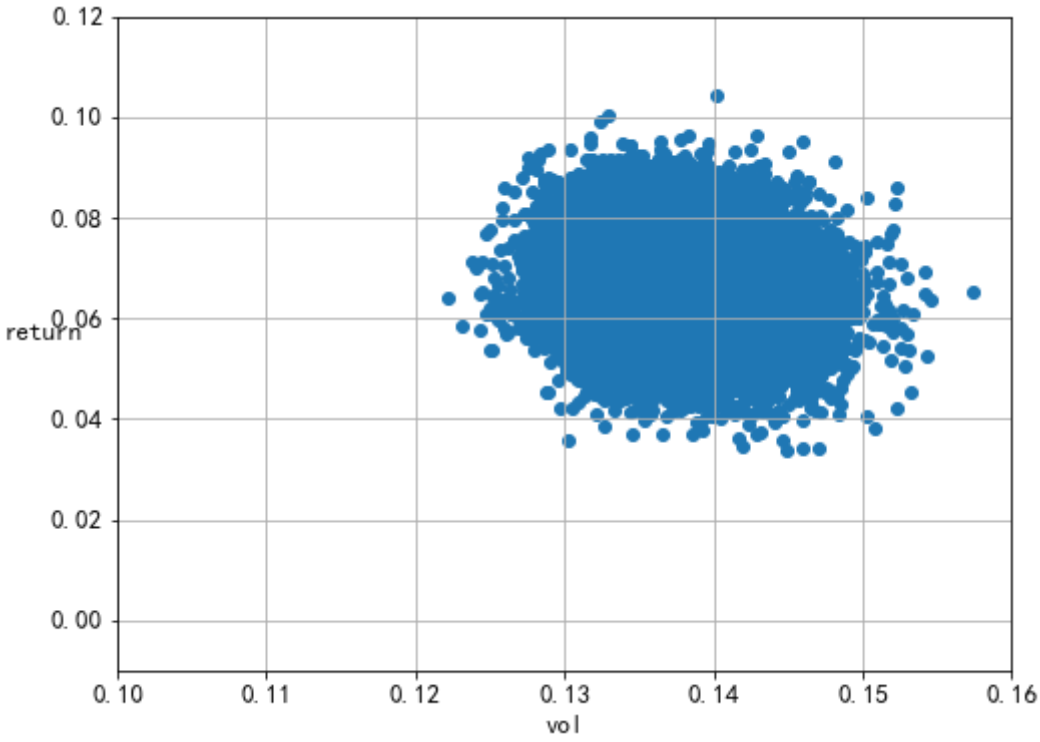
In [19]:

```
R_port = np.sum(weights * R_mean)
vol_port = np.sqrt(np.dot(weights, np.dot(R_cov, weights.T)))
print('expected return of the portfolio is: ', round(R_port, 4))
print('volatility of the portfolio is: ', round(vol_port, 4))
```

```
expected return of the portfolio is:  0.0648
volatility of the portfolio is:  0.1315
```

In [28]:

```
Rp_list = []
Vp_list = []
for i in np.arange(20000):
    x = np.random.random(29)
    weights = x / sum(x)
    Rp_list.append(np.sum(weights * R_mean))
    Vp_list.append(np.sqrt(np.dot(weights, np.dot(R_cov, weights.T))))
plt.figure(figsize = (8, 6))
plt.scatter(Vp_list, Rp_list)
plt.xlabel('vol', fontsize = 13)
plt.ylabel('return', fontsize = 13, rotation = 0)
plt.xticks(fontsize = 13)
plt.yticks(fontsize = 13)
plt.xlim(0.1, 0.16)
plt.ylim(-0.01, 0.12)
plt.grid('True')
plt.show()
```



In [30]:

```
bnds = tuple((0, 1) for x in range(len(R_mean)))
print(bnds)
```

```
((0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1),
(0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1),
(0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1), (0, 1),
(0, 1), (0, 1))
```

In [47]:

```
import scipy.optimize as sco
def f(w):
   # w = np.array(w)
    Rp_opt = np.sum(w * R_mean)
    Vp_opt = np.sqrt(np.dot(w, np.dot(R_cov, w.T)))
    return np.array([Rp_opt, Vp_opt])
def Vmin_f(w):
    return f(w)[1]

# set the return bound is 9%
# two conditions need to be fulfilled: 1. weight sum should be one 2.return is set to b
e 9%
cons = ({'type' : 'eq', 'fun' : lambda x : np.sum(x) - 1}, {'type' : 'eq', 'fun' : lamb
da x : f(x)[0] - 0.09})
bnds = tuple((0, 1) for x in range(len(R_mean)))
result = sco.minimize(Vmin_f, len(R_mean) * [1.0 / len(R_mean), ], method = 'SLSQP', bo
unds = bnds, constraints = cons)
print(np.round(result['x'], 4))
```

```
[0.0373 0.0046 0.     0.     0.056  0.     0.0403 0.     0.0702 0.
 0.2275 0.1811 0.     0.0027 0.     0.049  0.0968 0.0138 0.0525 0.0015
 0.     0.0932 0.0576 0.0004 0.0154 0.     0.     0.     0.     ]
```