Stock Prices II: Distributions of Returns

In [1]:

```
import pandas as pd

from tiingo import TiingoClient
tiingo = TiingoClient({'api_key':'XXX'})

import quandl
quandl.ApiConfig.api_key = 'YYY'

import matplotlib.pyplot as plt                          # Basic plot library.
plt.style.use('ggplot')                                  # Make plots look nice.
```

Get data for S&P 500 ETF:

In [2]:

```
data = tiingo.get_dataframe('SPY', '1900-1-1')
data
```

Out[2]:

date	close	high	low	open	volume	adjClose	adjHigh
1993-01-29 00:00:00+00:00	43.9375	43.9687	43.7500	43.9687	1003200	25.961948	25.980383
1993-02-01 00:00:00+00:00	44.2500	44.2500	43.9687	43.9687	480500	26.146599	26.146599
1993-02-02 00:00:00+00:00	44.3437	44.3750	44.1250	44.2187	201300	26.201965	26.220459
1993-02-03 00:00:00+00:00	44.8125	44.8437	44.3750	44.4062	529400	26.478971	26.497407
1993-02-04 00:00:00+00:00	45.0000	45.0937	44.4687	44.9687	531500	26.589762	26.645128
...	...	...	...	...	...	...	...
2021-02-25 00:00:00+00:00	382.3300	391.8800	380.7789	390.4100	144712701	382.330000	391.880000
2021-02-26 00:00:00+00:00	380.3600	385.5800	378.2300	384.3500	149530614	380.360000	385.580000
2021-03-01 00:00:00+00:00	389.5800	390.9200	380.5720	385.5900	105348798	389.580000	390.920000
2021-03-02 00:00:00+00:00	386.5400	390.0700	386.0000	389.8200	79595332	386.540000	390.070000
2021-03-03 00:00:00+00:00	381.4200	386.8300	381.3100	385.7900	119940211	381.420000	386.830000

7074 rows × 12 columns

Get rid of time zone:

In [3]:

```
data.index = data.index.tz_convert(None)
data[-3:]
```

Out[3]:

date	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVo
2021-03-01	389.58	390.92	380.572	385.59	105348798	389.58	390.92	380.572	385.59	10534
2021-03-02	386.54	390.07	386.000	389.82	79595332	386.54	390.07	386.000	389.82	7959
2021-03-03	381.42	386.83	381.310	385.79	119940211	381.42	386.83	381.310	385.79	11994

Calculate daily returns:

In [4]:

```
r_daily = data[['adjClose']].pct_change()
r_daily
```

Out[4]:

date	adjClose
1993-01-29	NaN
1993-02-01	0.007112
1993-02-02	0.002118
1993-02-03	0.010572
1993-02-04	0.004184
...	...
2021-02-25	-0.024096
2021-02-26	-0.005153
2021-03-01	0.024240
2021-03-02	-0.007803
2021-03-03	-0.013246

7074 rows × 1 columns

What are the annual returns?

In [5]:

```
# With groupby:
data[['adjClose']].groupby(data.index.year).last().pct_change()
```

Out[5]:

date	adjClose
1993	NaN

	adjClose
date	
1994	0.004019
1995	0.380251
1996	0.225543
1997	0.334780
1998	0.286873
1999	0.203875
2000	-0.097293
2001	-0.117525
2002	-0.215877
2003	0.281765
2004	0.107028
2005	0.048258
2006	0.158482
2007	0.051356
2008	-0.368069
2009	0.263661
2010	0.150577
2011	0.018879
2012	0.159917
2013	0.323067
2014	0.134621
2015	0.012523
2016	0.120013
2017	0.217003
2018	-0.045571
2019	0.312217
2020	0.183732
2021	0.020167

In [6]:

```
# With resample:
data[['adjClose']].resample('A').last().pct_change()[-5:]
```

Out[6]:

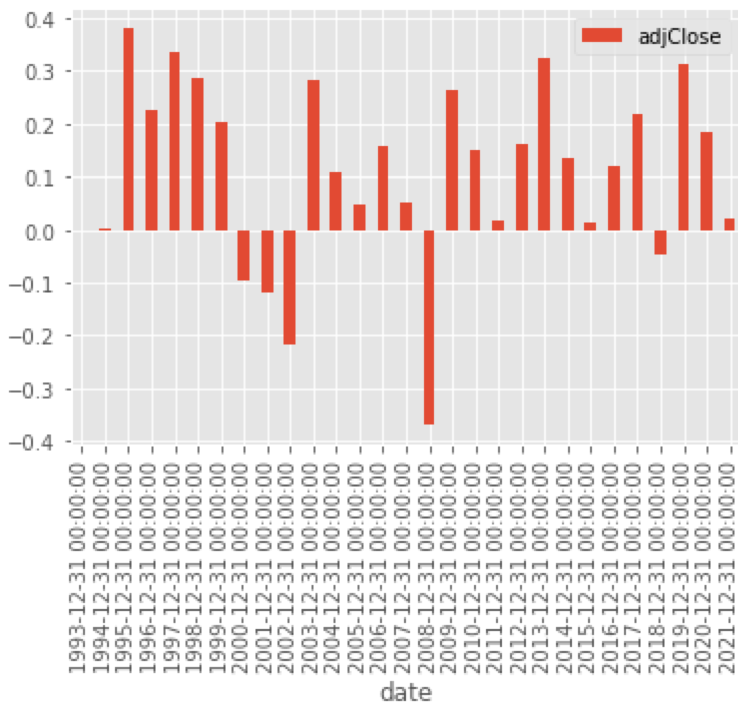
	adjClose
date	
2017-12-31	0.217003

	adjClose
date	
2018-12-31	-0.045571
2019-12-31	0.312217
2020-12-31	0.183732
2021-12-31	0.020167

Bar-plot this:

In [7]:
```
data[['adjClose']].resample('A').last().pct_change().plot.bar()
```

Out[7]: <AxesSubplot:xlabel='date'>



Monthly returns:

In [8]:
```
# With groupby
data[['adjClose']].groupby([data.index.year,data.index.month]).last().pct_change
```

Out[8]:

		adjClose
date	**date**	
1993	**1**	NaN
	2	0.010667
	3	0.022412
	4	-0.025589
	5	0.026970
...	**...**	...

		adjClose
date	**date**	
2020	**11**	0.108777
	12	0.037066
2021	**1**	-0.010190
	2	0.027806
	3	0.002787

339 rows × 1 columns

In [9]:
```
# With resample:
data[['adjClose']].resample('M').last().pct_change()[-5:]
```

Out[9]:

	adjClose
date	
2020-11-30	0.108777
2020-12-31	0.037066
2021-01-31	-0.010190
2021-02-28	0.027806
2021-03-31	0.002787

Or calculate monthly returns directly from daily returns:

In [10]:
```
# Compound daily returns:
r_daily.add(1).resample('M').prod().sub(1)     [-5:]
```

Out[10]:

	adjClose
date	
2020-11-30	0.108777
2020-12-31	0.037066
2021-01-31	-0.010190
2021-02-28	0.027806
2021-03-31	0.002787

All frequencies:

In [11]:
```
r_daily   = data['adjClose'].pct_change()                  # Daily returns as a
r_monthly = r_daily.add(1).resample('M').prod().sub(1)
r_annual  = r_daily.add(1).resample('A').prod().sub(1)
```

Do returns predict returns?

In [12]:
```
t = pd.DataFrame(index=r_daily.index)
t['today']    = r_daily
t['tomorrow'] = r_daily.shift(-1)  # Shift column r_daily 1 row up.
t
```

Out[12]:

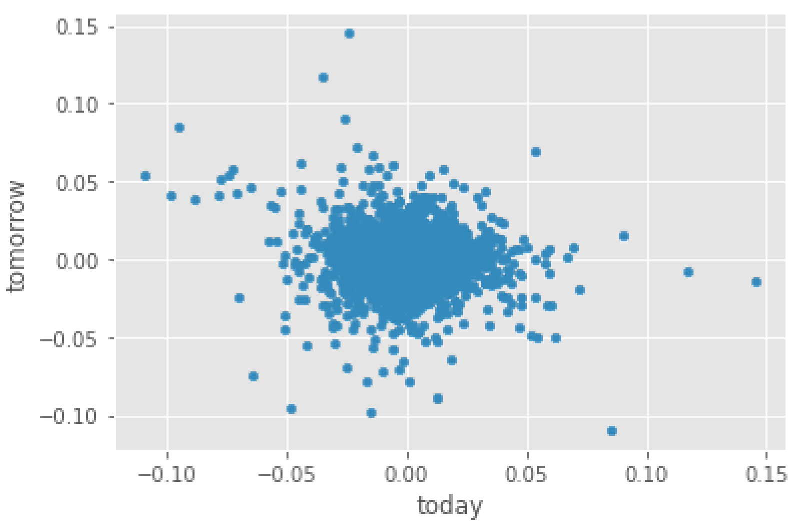
	today	tomorrow
date		
1993-01-29	NaN	0.007112
1993-02-01	0.007112	0.002118
1993-02-02	0.002118	0.010572
1993-02-03	0.010572	0.004184
1993-02-04	0.004184	-0.000696
...	...	...
2021-02-25	-0.024096	-0.005153
2021-02-26	-0.005153	0.024240
2021-03-01	0.024240	-0.007803
2021-03-02	-0.007803	-0.013246
2021-03-03	-0.013246	NaN

7074 rows × 2 columns

Scatter-plot this:

In [13]:
```
t.plot.scatter('today','tomorrow')
```

Out[13]: <AxesSubplot:xlabel='today', ylabel='tomorrow'>



Autocorrelation:

In [14]:
```
r_daily.autocorr(1)
```

Out[14]: -0.09045401581967706

In [15]:
```
r_monthly.autocorr(1)
```

Out[15]: 0.03712685214044506

In [16]:
```
r_annual.autocorr(1)
```

Out[16]: 0.06361313843053643

Volatility of returns:

In [17]:
```
# Annual data
r_annual.std()
```

Out[17]: 0.17310321288226904

In [18]:
```
# Monthly data
r_monthly.std() * 12**0.5
```

Out[18]: 0.14595985869704467

In [19]:
```
# Daily data:
r_daily.std() * 252**0.5
```

Out[19]: 0.18877850118346676

Get U.S. treasury rates from Quandl:

https://www.quandl.com/data/FRED/DGS10-10-Year-Treasury-Constant-Maturity-Rate

In [20]:
```
quandl.get(['FRED/DGS10'])    # 10-year treasury
```

Out[20]:

	FRED/DGS10 - Value
Date	
1962-01-02	4.06
1962-01-03	4.03
1962-01-04	3.99
1962-01-05	4.02
1962-01-08	4.03
...	...
2021-02-24	1.38
2021-02-25	1.54
2021-02-26	1.44
2021-03-01	1.45

	FRED/DGS10 - Value
Date	
2021-03-02	1.42

14775 rows × 1 columns

Get multiple rates:

In [21]:
```
rates = quandl.get(['FRED/FEDFUNDS','FRED/DGS1','FRED/DGS5','FRED/DGS10','FRED/D
rates
```

Out[21]:

	FRED/FEDFUNDS - Value	FRED/DGS1 - Value	FRED/DGS5 - Value	FRED/DGS10 - Value	FRED/DGS30 - Value
Date					
1954-07-01	0.0080	NaN	NaN	NaN	NaN
1954-08-01	0.0122	NaN	NaN	NaN	NaN
1954-09-01	0.0107	NaN	NaN	NaN	NaN
1954-10-01	0.0085	NaN	NaN	NaN	NaN
1954-11-01	0.0083	NaN	NaN	NaN	NaN
...	...	...	...	...	...
2021-02-24	NaN	0.0008	0.0062	0.0138	0.0224
2021-02-25	NaN	0.0009	0.0081	0.0154	0.0233
2021-02-26	NaN	0.0008	0.0075	0.0144	0.0217
2021-03-01	NaN	0.0008	0.0071	0.0145	0.0223
2021-03-02	NaN	0.0008	0.0067	0.0142	0.0221

15123 rows × 5 columns

Forward-fill missing values:

In [22]:
```
rates = rates.ffill()
rates[-5:]
```

Out[22]:

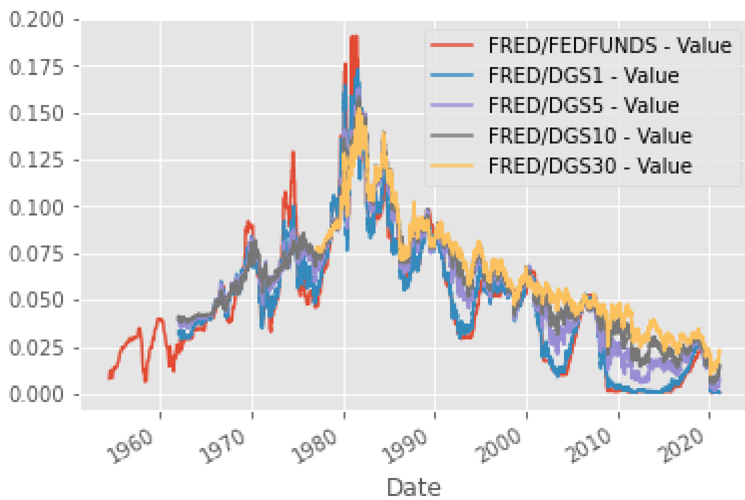
	FRED/FEDFUNDS - Value	FRED/DGS1 - Value	FRED/DGS5 - Value	FRED/DGS10 - Value	FRED/DGS30 - Value
Date					

Date	FRED/FEDFUNDS - Value	FRED/DGS1 - Value	FRED/DGS5 - Value	FRED/DGS10 - Value	FRED/DGS30 - Value
2021-02-24	0.0008	0.0008	0.0062	0.0138	0.0224
2021-02-25	0.0008	0.0009	0.0081	0.0154	0.0233
2021-02-26	0.0008	0.0008	0.0075	0.0144	0.0217
2021-03-01	0.0008	0.0008	0.0071	0.0145	0.0223
2021-03-02	0.0008	0.0008	0.0067	0.0142	0.0221

Plot this table:

In [23]:
```
rates.plot()
```

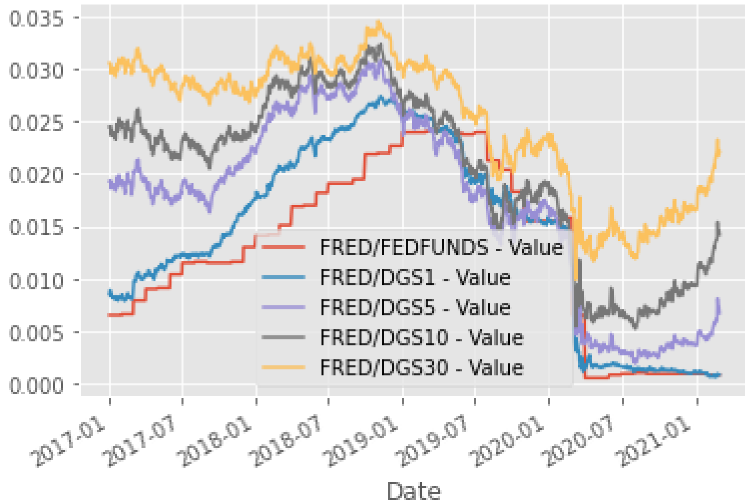
Out[23]: <AxesSubplot:xlabel='Date'>



Plot recent rates:

In [24]:
```
rates['2017':].plot()
```

Out[24]: <AxesSubplot:xlabel='Date'>



Annual 1-year rate::

In [25]:

```
rates['FRED/DGS1 - Value'].resample('A').first()  # First: we buy at start of ye
```

Out[25]:

```
Date
1954-12-31        NaN
1955-12-31        NaN
1956-12-31        NaN
1957-12-31        NaN
1958-12-31        NaN
                ...
2017-12-31     0.0085
2018-12-31     0.0176
2019-12-31     0.0263
2020-12-31     0.0159
2021-12-31     0.0010
Freq: A-DEC, Name: FRED/DGS1 - Value, Length: 68, dtype: float64
```

Annual market excess returns:

In [26]:

```
rx_annual = r_annual - rates['FRED/DGS1 - Value'].resample('A').first()
rx_annual
```

Out[26]:

```
1954-12-31          NaN
1955-12-31          NaN
1956-12-31          NaN
1957-12-31          NaN
1958-12-31          NaN
                  ...
2017-12-31     0.208503
2018-12-31    -0.063171
2019-12-31     0.285917
2020-12-31     0.167832
2021-12-31     0.019167
Freq: A-DEC, Length: 68, dtype: float64
```

Historical risk premium, 1993-2020:

In [27]:

```
rx_annual.mean()
```

Out[27]:

```
0.08552159372632122
```