Momentum

- 어떤 주식을 살 것인가?
  - 주가가 오른 주식을 산다
  - 주가가 내린 주식을 산다
- 주식에 대하여
  - 쌀 때 사서 비쌀 때 팔아라
  - 무릎에 사서 어깨에 팔아라
  - 달리는 말에 올라 타라
- Short-term reversal (1개월 이내)
- Mid-term momentum (3개월 – 1년)
- Long-term reversal (3년 – 5년)
- Jegadeesh and Titman (1993)
  - Journal of Finance
  - “Returns to buying winners and selling losers: Implications for stock market efficiency”

여러 종류의 모멘텀 전략

- J (formation period)
  - J = 3, 6, 9, 12 months
- K (holding period)
  - K = 3, 6, 9, 12 months
- short-term reversal 감안 여부
  - 1 week, 1 month
- Portfolio 구성 방법
  - Equal weight
  - Value weight
- 모멘텀은 왜 생기는가?
  - 정보의 전달 속도
  - Underreact to information
  - Momentum crash

Missing data 처리 방법

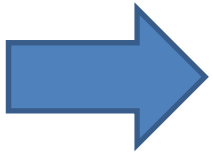
- Fillna(0)
- Formation period에서 제외
- Holding portfolio에서 제외

	No time lag	1-month lag
Formation: fillna(0) **Holding**: fillna(0)	수업시간	Homework 1
Formation: fillna(0) **Holding**: 해당월의 holding portfolio에서 제외	Homework 2	

Input and output

	permno	date	ret
0	10006	1963-01-31	0.047002
1	10006	1963-02-28	0.038700
2	10006	1963-03-29	-0.009009
3	10006	1963-04-30	0.084848
4	10006	1963-05-31	0.091620
...	...	...	...
730674	93201	1989-11-30	0.209677
730675	93201	1989-12-29	0.040000
730676	93172	1986-07-31	NaN
730677	93172	1986-08-29	-0.772727
730678	93172	1986-09-30	-0.400000

730679 rows × 3 columns



J = 6, K = 6 months
1965.1월 – 1989.12월(25년)

momr	count	mean	std
1	300.0	0.008253	0.085689
2	300.0	0.011243	0.068947
3	300.0	0.012731	0.062932
4	300.0	0.012297	0.058580
5	300.0	0.012507	0.056925
6	300.0	0.013490	0.055938
7	300.0	0.013600	0.055506
8	300.0	0.014442	0.057620
9	300.0	0.015186	0.060354
10	300.0	0.016745	0.068974

	momr	mean	t-stat	p-value
0	winners	0.016745	4.204959	0.000035
1	losers	0.008253	1.668114	0.096340
2	long_short	0.008492	2.796524	0.005500

(m1) lambda, (m2) qcut

(m1) lambda function

lambda function

lambda function with dataframe

lambda function with series

df.apply()

df.transform()

(m2) qcut function

np.random.seed(0)

np.random.randn(10)

pd.qcut()

df.transform(lambda x: pd.qcut(x, 10, labels=False))

(m3) cumulative return

$$+2\%, +3\%, -3\%, -2\%$$

$$100(1.02)(1.03)(0.97)(0.98) = 99.87$$

$$(1.02)(1.03)(0.97)(0.98) = 0.9987$$

$$\log(1.02) + \log(1.03) + \log(0.97) + \log(0.98) = \log(0.9987)$$

$$\exp(\log(1.02) + \log(1.03) + \log(0.97) + \log(0.98)) = 0.9987$$

$$\text{cumret} = \frac{99.87}{100} - 1 = -0.001299 = -0.1299\%$$

(m3) cumulative return

cumret = np.exp(np.log(1+x).sum()) – 1 = -0.001299

(m4) rolling

```
df.rolling(2).sum()
df['logret'] = np.log(1+df['ret’])
df['cumret'] = np.exp(df['logret'].rolling(3).sum()) - 1
df.dropna(subset=['cumret'], inplace=True)
```

(m5) MonthEnd(), MonthBegin()

pd.to_datetime(string)

pd.to_datetime(list of strings)

timestamp datatype

datetime64[ns] datatype

date.dt.year

date.dt.month

date.dt.day

from pandas.tseries.offsets import *

date + MonthEnd(0)

date + MonthBegin(1)

date + MonthEnd(6)

(m6) merge(), (m7) concat()

(m6) merge()

df1.merge(df2, on='name', how='inner')

df1.merge(df2, on='name', how='outer')

df1.merge(df2, on='name', how='left')

df1.merge(df2, on='name', how='right')

df1.merge(df2, how='cross')

(m7) concat()

pd.concat([df1, df2])

pd.concat([df1, df2], ignore_index=True)

(m8) t-test(), (m9) pivot()

(m8) t-test()

```
np.random.seed(0)
data = 5 + 10 * np.random.randn(50)
stats.ttest_1samp(data, 5.0)
stats.ttest_1samp(data, 0.0)
```

(m9) pivot()

```
df.pivot(index='foo', columns='bar', values='baz’)
df.pivot(index='foo', columns='bar')
```

(m10) set_index(), reset_index()

```
df.set_index('month’)
df.reset_index()
df.reset_index(drop=True)
```

(pandas)

info()
to_datetime()
group_by()
count()
plot()
sort_values()
set_index()
reset_index()
mean()
dt.year
dt.year.min()
describe()[[‘count’, ‘mean’, ‘std’]]

(1) input data 점검

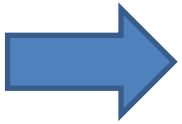
- Data type 적정성 검토

  crsp['date'] = pd.to_datetime(crsp['date'])

- Missing data 처리 방법

  crsp['ret'] = crsp['ret'].fillna(0)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730679 entries, 0 to 730678
Data columns (total 3 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   permno  730679 non-null  int64
 1   date    730679 non-null  object
 2   ret     707466 non-null  float64
dtypes: float64(1), int64(1), object(1)
memory usage: 16.7+ MB
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 730679 entries, 0 to 730678
Data columns (total 3 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   permno  730679 non-null  int64
 1   date    730679 non-null  datetime64[ns]
 2   ret     730679 non-null  float64
dtypes: datetime64[ns](1), float64(1), int64(1)
memory usage: 16.7 MB
```

(2) cumulative return

	permno	date	logret	cumret
0	10006	1963-01-31	NaN	NaN
1	10006	1963-02-28	NaN	NaN
2	10006	1963-03-29	NaN	NaN
3	10006	1963-04-30	NaN	NaN
4	10006	1963-05-31	NaN	NaN
...	...	...	...	...
730674	93201	1989-08-31	0.305382	0.357143
730675	93201	1989-09-29	0.270875	0.311111
730676	93201	1989-10-31	-0.031749	-0.031250
730677	93201	1989-11-30	0.190354	0.209677
730678	93201	1989-12-29	0.367725	0.444444

730679 rows × 4 columns

```
crsp.sort_values(['permno', 'date'])
crsp['logret'] = np.log(1 + crsp['ret'])
crsp = crsp.set_index('date')
umd = crsp.groupby('permno')['logret'].rolling(J).sum()
umd = umd.reset_index()
umd['cumret'] = np.exp(umd['logret'])-1
umd = umd[['permno', 'date', 'cumret']]
```

(3) momentum

```
umd.dropna(subset=['cumret'], inplace=True)
umd['momr’] = umd.groupby('date')['cumret'].transform(lambda x: pd.qcut(x, 10, labels=False))
umd['momr'] = umd['momr’] + 1
```

	permno	date	logret	cumret	momr
5	10006	1963-06-28	0.242664	0.274640	9
6	10006	1963-07-31	0.150521	0.162439	9
7	10006	1963-08-30	0.226845	0.254635	9
8	10006	1963-09-30	0.155042	0.167707	8
9	10006	1963-10-31	0.129883	0.138695	8
...	...	...	...	...	...
730674	93201	1989-08-31	0.305382	0.357143	9
730675	93201	1989-09-29	0.270875	0.311111	9
730676	93201	1989-10-31	-0.031749	-0.031250	4
730677	93201	1989-11-30	0.190354	0.209677	9
730678	93201	1989-12-29	0.367725	0.444444	10

704292 rows × 5 columns

(4) merge

	permno	form_date	momr	hdate1	hdate2
5	10006	1964-06-30	8	1964-07-01	1964-12-31
6	10006	1964-07-31	8	1964-08-01	1965-01-31
7	10006	1964-08-31	8	1964-09-01	1965-02-28
8	10006	1964-09-30	9	1964-10-01	1965-03-31
9	10006	1964-10-30	9	1964-11-01	1965-04-30

	permno	date	ret
0	10006	1964-01-31	0.068548
1	10006	1964-02-28	0.043774
2	10006	1964-03-31	0.000000
3	10006	1964-04-30	0.003636
4	10006	1964-05-28	-0.010870

	permno	form_date	momr	hdate1	hdate2	date	ret
0	10006	1964-06-30	8	1964-07-01	1964-12-31	1964-01-31	0.068548
1	10006	1964-06-30	8	1964-07-01	1964-12-31	1964-02-28	0.043774
2	10006	1964-06-30	8	1964-07-01	1964-12-31	1964-03-31	0.000000
3	10006	1964-06-30	8	1964-07-01	1964-12-31	1964-04-30	0.003636
4	10006	1964-06-30	8	1964-07-01	1964-12-31	1964-05-28	-0.010870
...	...	...	...	...	...	...	...
148507349	93201	1989-12-29	10	1990-01-01	1990-06-30	1989-08-31	0.187500
148507350	93201	1989-12-29	10	1990-01-01	1990-06-30	1989-09-29	0.035088
148507351	93201	1989-12-29	10	1990-01-01	1990-06-30	1989-10-31	0.050847
148507352	93201	1989-12-29	10	1990-01-01	1990-06-30	1989-11-30	0.209677
148507353	93201	1989-12-29	10	1990-01-01	1990-06-30	1989-12-29	0.040000

148507354 rows × 7 columns

(5) delete

```
port = port[(port['hdate1’] <= port['date']) & (port['date’] <= port['hdate2’])]
port = port.sort_values(by=['date','momr','form_date','permno’])
```

	permno	form_date	momr	hdate1	hdate2	date	ret
680268	10217	1963-06-28	1	1963-07-01	1963-12-31	1963-07-31	0.000000
790260	10233	1963-06-28	1	1963-07-01	1963-12-31	1963-07-31	-0.033613
2569312	10874	1963-06-28	1	1963-07-01	1963-12-31	1963-07-31	-0.163934
3069570	11084	1963-06-28	1	1963-07-01	1963-12-31	1963-07-31	-0.017143
3269314	11113	1963-06-28	1	1963-07-01	1963-12-31	1963-07-31	0.000000
...	...	...	...	...	...	...	...
1536962	79250	1989-11-30	10	1989-12-01	1990-05-31	1989-12-29	-0.060086
1630657	85869	1989-11-30	10	1989-12-01	1990-05-31	1989-12-29	-0.022059
1648663	86765	1989-11-30	10	1989-12-01	1990-05-31	1989-12-29	0.076923
1650055	86976	1989-11-30	10	1989-12-01	1990-05-31	1989-12-29	0.001860
1678398	91380	1989-11-30	10	1989-12-01	1990-05-31	1989-12-29	0.097842

4117538 rows × 7 columns

(6) Portfolio return

```
umd2 = port.groupby(['date','momr','form_date'])['ret'].mean().reset_index()
ewret = umd2.groupby(['date','momr'])['ret'].mean().reset_index()
ewret.rename(columns={'ret':'ewret'}, inplace=True)
```

	date	momr	ewret
0	1965-01-29	1	0.125148
1	1965-01-29	2	0.087203
2	1965-01-29	3	0.072248
3	1965-01-29	4	0.064964
4	1965-01-29	5	0.063503
...	...	...	...
2995	1989-12-29	6	0.007974
2996	1989-12-29	7	0.009907
2997	1989-12-29	8	0.010142
2998	1989-12-29	9	0.005789
2999	1989-12-29	10	-0.007858

3000 rows × 3 columns

(7) Pivot table

```
ewret2 = ewret.pivot(index='date', columns='momr', values='ewret')
```

momr	1	2	3	4	5	6	7	8	9	10
date										
1965-01-29	0.125148	0.087203	0.072248	0.064964	0.063503	0.059188	0.057857	0.057466	0.060715	0.075762
1965-02-26	0.023498	0.036074	0.039101	0.029394	0.031543	0.029517	0.031333	0.035150	0.043514	0.032679
1965-03-31	0.029623	0.014346	0.011204	0.015541	0.011194	0.004043	0.007294	0.011785	0.018297	0.023156
1965-04-30	0.030717	0.043300	0.040916	0.033639	0.037757	0.041739	0.040175	0.044794	0.052073	0.053390
1965-05-28	-0.018215	-0.012755	-0.008787	-0.008654	-0.007943	-0.006797	-0.004161	-0.001007	0.005152	0.004448
...	...	...	...	...	...	...	...	...	...	...
1989-08-31	-0.011274	0.012553	0.015148	0.013318	0.024014	0.021361	0.019848	0.019984	0.034438	0.036031
1989-09-29	-0.037375	-0.014215	-0.005144	-0.009116	-0.006785	-0.000103	-0.005796	0.000427	0.006646	0.009885
1989-10-31	-0.106492	-0.075305	-0.046067	-0.048517	-0.053980	-0.046414	-0.043922	-0.050186	-0.057677	-0.075002
1989-11-30	-0.050089	-0.019166	-0.007705	-0.005809	0.005670	0.011542	0.011329	0.015344	0.013121	0.011550
1989-12-29	-0.063588	-0.018569	-0.012114	-0.009317	0.001935	0.007974	0.009907	0.010142	0.005789	-0.007858

300 rows × 10 columns

(8) t-test

```
mom_mean = ewret2[['winners', 'losers', 'long_short']].mean().to_frame()
mom_mean = mom_mean.rename(columns={0:'mean'}).reset_index()
t_losers = pd.Series(stats.ttest_1samp(ewret2['losers'], 0.0)).to_frame().T
t_winners = pd.Series(stats.ttest_1samp(ewret2['winners'], 0.0)).to_frame().T
t_long_short = pd.Series(stats.ttest_1samp(ewret2['long_short'],0.0)).to_frame().T
t_output['momr'] = ['winners', 'losers', 'long_short']
pd.merge(mom_mean, t_output, on=['momr'], how='inner')
```

	momr	mean	t-stat	p-value
0	winners	0.016745	4.204959	0.000035
1	losers	0.008253	1.668114	0.096340
2	long_short	0.008492	2.796524	0.005500

Homework

J = 6, K = 6 months
1965.1월 – 1989.12월(25년)

수업시간

momr	count	mean	std
1	300.0	0.008253	0.085689
2	300.0	0.011243	0.068947
3	300.0	0.012731	0.062932
4	300.0	0.012297	0.058580
5	300.0	0.012507	0.056925
6	300.0	0.013490	0.055938
7	300.0	0.013600	0.055506
8	300.0	0.014442	0.057620
9	300.0	0.015186	0.060354
10	300.0	0.016745	0.068974

	momr	mean	t-stat	p-value
0	winners	0.016745	4.204959	0.000035
1	losers	0.008253	1.668114	0.096340
2	long_short	0.008492	2.796524	0.005500

Homework 1

momr	count	mean	std
1	300.0	0.006403	0.082809
2	300.0	0.010447	0.067502
3	300.0	0.012508	0.062177
4	300.0	0.012683	0.058208
5	300.0	0.012622	0.057204
6	300.0	0.013490	0.055878
7	300.0	0.014071	0.055988
8	300.0	0.015057	0.058301
9	300.0	0.015826	0.061337
10	300.0	0.017565	0.070062

	momr	mean	t-stat	p-value
0	winners	0.017565	4.342423	0.000019
1	losers	0.006403	1.339180	0.181529
2	long_short	0.011163	4.026172	0.000072

Homework 2

momr	count	mean	std
1	300.0	0.008516	0.087463
2	300.0	0.011573	0.070373
3	300.0	0.012949	0.064714
4	300.0	0.012884	0.060796
5	300.0	0.013387	0.058817
6	300.0	0.013912	0.057256
7	300.0	0.013997	0.056963
8	300.0	0.014599	0.058297
9	300.0	0.015529	0.061508
10	300.0	0.016914	0.069698

	momr	mean	t-stat	p-value
0	winners	0.016914	4.203222	0.000035
1	losers	0.008516	1.686480	0.092746
2	long_short	0.008398	2.760694	0.006124