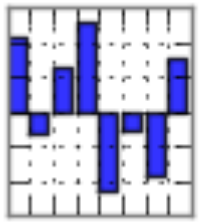
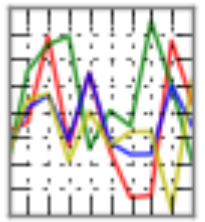
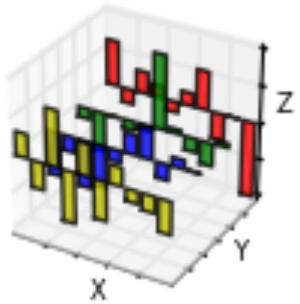
pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$







Pandas(1) - Series, DataFrame

이승준 fb.com/plusjune

Pandas: 파이썬에서 가장 강력한 데이터 분석 도구

http://pandas.pydata.org

- “panel data analysis” 다차원 구조화된 데이터 (계량경제학)
- 오픈소스 데이터 분석 라이브러리 (데이터 분석, 클리닝, 모델링 등)
- 웨스 맥키니(Wes Mckinney), 2009년 금융 데이터 분석을 위해 설계
- 시계열 등 다양한 금융 데이터를 처리 기능을 제공
- R 혹은 Matlab 사용자들이 접근하기 용이 (DataFrame과 R의 data.frame 유사)

pandas 자료구조

- Series(1차원), DataFrame(2차원), Panel(3차원)
- 가장 많이 사용하는 구조는 DataFrame
- TimeSeries는 인덱스에 datetimes를 가지고 있는 Series
- Panel은 3D 라벨 배열 (여러 DataFrame을 포함)

시리즈 (Series)

- 1차원 배열 모습의 자료구조이며, 각 요소는 NumPy의 데이터 타입
- numpy.ndarray 의 서브클래스
- 값과 값에 대한 인덱스(index)로 구성
- Series 생성은 리스트, 딕셔너리, 시리즈로 부터 생성

- **import** pandas **as** pd
- **from** pandas **import** Series, DataFrame

리스트에서 생성

```
s = Series([7, 0, -3, 8, 1])
s.values  # array([ 7,  0, -3,  8,  1])
s.index   # Int64Index([0, 1, 2, 3, 4], dtype='int64')
```

시리즈를 생성하면서 인덱스를 지정

```
s = Series([7, 0, -3, 8, 1], index=['A', 'B', 'C', 'D', 'E'])
```

값에 접근할 때는 라벨을 인덱스로 이용

```
s['D']          # 8
s[['B', 'D']]   # B 0, D 8
s[0:3]          # s[0:3]에서는 끝점을 포함하지 않지만
s['A':'C']      # 라벨 이름은 끝점을 포함
```

필터링, 값의 비교를 통해 추출

```
s[s > 0]

s10 = s * 10   # 스칼라 곱
```

산술 연산

```
a = Series([2, 3, 6, -4], index=['A', 'B', 'C', 'D'])
b = Series([10, 2, 3, 8], index=['B', 'C', 'D', 'E'])


c = a + b
c
```

```
A    NaN
B     13
C      8
D     -1
E    NaN
```

누락 데이터(missing data)

```
# 누락 데이터(NaN)를 0으로 채우기
fill_c = c.fillna(0)

# NA 요소 삭제, dropna()
drop_c = c.dropna()
```

DataFrame

- 다양한 데이터를 포함할 수 있는 2차원 자료구조
- 간단한게 시리즈(Series) 객체에 대한 딕셔너리 구조
- 스프레드시트와 유사

DataFrame 생성

```
month = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
exp_kt = [7039177, 6960795, 6438333, 5024973, 7327214, 5891620]
exp_skt = [5277359, 9523039, 8420870, 8114839, 10809910, 8650904]

data = {'month': month, 'kt': exp_kt, 'skt': exp_skt}
df = DataFrame(data, columns=['month', 'kt', 'skt'])
```

	month	kt	skt
0	Jan	7039177	5277359
1	Feb	6960795	9523039
2	Mar	6438333	8420870
3	Apr	5024973	8114839
4	May	7327214	10809910
5	Jun	5891620	8650904

DataFrame 구성

- df.index
- df.values
- df.columns

컬럼

```
df = DataFrame(data, columns=['month', 'kt', 'skt'])


# 컬럼 접근

df.month

df.kt # df['kt']
```

컬럼 연산

```
# 컬럼 추가
df['etc'] = 100

# 컬럼간 연산
df['etc'] = df['etc'] + 10

#  100000로 나누어 천원→억원으로 단위 변경
df['kt'] =  df['kt'] / 100000.

# 컬럼삭제
del df['etc']
```

Row, Column 삭제

```
# 로우 삭제는 drop을 사용

r_df = df.drop([4,5])


# 컬럼 삭제는 drop 사용, 축 지정

r_df = df.drop(['sum'], axis=1)
```

ix

- data.ix[rows] *rows* 부분의 로우 선택
- data.ix[:, cols] *cols* 부분의 컬럼 선택
- data.ix[rows, cols] 대응하는 *rows*와 *cols*에 선택

ix 선택

```
# 컬럼 'skt'~'sum' 선택

df.ix[:, 'skt':'sum']
```

```
# 로우 처음부터~2까지, 컬럼 'kt','sum' 선택

df.ix[:2, ['kt','sum']]
```

인덱스와 **ix**

- df[3:5] *# row 3~4*

	month	kt	skt	sum
3	Apr	50.24973	81.14839	131.4
4	May	73.27214	108.09910	181.4

- df.ix[3:5] *# row 3~5*

	month	kt	skt	sum
3	Apr	50.24973	81.14839	131.4
4	May	73.27214	108.09910	181.4
5	Jun	58.91620	86.50904	145.4

불린 인덱스

- df[df.skt > 90]
- df[(df.skt > 50) & (df.kt > 70)]
- df[(df.skt > 90) | (df.kt > 70)]

인덱스

- 인덱스는 로우, 컬럼의 이름과 정보를 저장하는 객체
- 색인은 변경할 수 없다 (공유) immutable
- 종류: Index, Int64Index, MultiIndex (다중), DatetimeIndex (날짜시간), PeriodIndex (기간)

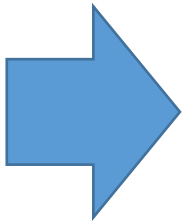
소트

```
# 인덱스로 소트 (내림차순)
df.sort_index(ascending=False)

# 컬럼으로 소트 (오름차순)
df.sort_index(axis=1, ascending=True)

# skt 컬럼으로 (내림차순), skt광고비 지출은 5월, 2월, 6월 순
df.sort_values(by='skt', ascending=False)
```

축 변환 (Pivot)

- df.T

	month	kt	skt	sum
0	Jan	70.39177	52.77359	123.2
1	Feb	69.60795	95.23039	164.8
2	Mar	64.38333	84.20870	148.6
3	Apr	50.24973	81.14839	131.4
4	May	73.27214	108.09910	181.4
5	Jun	58.91620	86.50904	145.4



	0	1	2	3	4	5
month	Jan	Feb	Mar	Apr	May	Jun
kt	70.3918	69.608	64.3833	50.2497	73.2721	58.9162
skt	52.7736	95.2304	84.2087	81.1484	108.099	86.509
sum	123.2	164.8	148.6	131.4	181.4	145.4

리뷰

- Series, 필터링
- DataFrame (pandas에서 가장 자주 사용되는 구조)
- Ix를 사용하는 인덱싱
- 소트
- 축 변환