

파이썬 라이브러리를 활용한 데이터 분석

14장 데이터 분석 예제

2020.07.09목 2h

14장 데이터 분석 예제

URL 축약 서비스 정보 분석

2h

- 파이참에서 확인 가능

```
mple1.py × bksample2.py × example.txt ×
{
  "a": "Mozilla\\5.0 (Windows NT 6.1; WOW64) AppleWebKit\\535.11 (KHTML, like Gecko) Chrome\\17.0.963.78 Safari\\535.11", "c": "US", "nk": 1, "tz": "America\\New_York",
  "a": "GoogleMaps\\RochesterNY", "c": "US", "nk": 0, "tz": "America\\Denver", "gr": "UT", "g": "mwszkS", "h": "mwszkS", "l": "bitly", "hh": "j.mp", "r": "http://www",
  "a": "Mozilla\\4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident\\4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)", "c": "US", "nk": 0, "tz": "America\\New_York",
  "a": "Mozilla\\5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit\\534.52.7 (KHTML, like Gecko) Version\\5.1.2 Safari\\534.52.7", "c": "BR", "nk": 0, "tz": "America\\New_York",
  "a": "Mozilla\\5.0 (Windows NT 6.1; WOW64) AppleWebKit\\535.11 (KHTML, like Gecko) Chrome\\17.0.963.79 Safari\\535.11", "c": "US", "nk": 0, "tz": "America\\New_York",
  "a": "Mozilla\\5.0 (Windows NT 6.1; WOW64) AppleWebKit\\535.11 (KHTML, like Gecko) Chrome\\17.0.963.79 Safari\\535.11", "c": "US", "nk": 0, "tz": "America\\New_York",
  "a": "Mozilla\\5.0 (Windows NT 5.1) AppleWebKit\\535.11 (KHTML, like Gecko) Chrome\\17.0.963.79 Safari\\535.11", "c": "PL", "nk": 0, "tz": "Europe\\Warsaw", "gr": "UT",
  "a": "Mozilla\\5.0 (Windows NT 6.1; rv:2.0.1) Gecko\\20100101 Firefox\\4.0.1", "c": null, "nk": 0, "tz": "", "g": "wcnDER", "h": "zpkJBR", "l": "bnjacobs", "al": "en-US",
  "a": "Opera\\9.80 (X11; Linux zboy; U; en) Presto\\2.10.254 Version\\12.00", "c": null, "nk": 0, "tz": "", "g": "wcnDER", "h": "zpkJBR", "l": "bnjacobs", "al": "en-US",
  "a": "Mozilla\\5.0 (Windows NT 6.1; WOW64) AppleWebKit\\535.11 (KHTML, like Gecko) Chrome\\17.0.963.79 Safari\\535.11", "c": null, "nk": 0, "tz": "", "g": "zCaLwp", "h": "zCaLwp", "l": "bnjacobs", "al": "en-US",
  "a": "Mozilla\\5.0 (Windows NT 6.1; WOW64; rv:10.0.2) Gecko\\20100101 Firefox\\10.0.2", "c": "US", "nk": 1, "tz": "America\\Los_Angeles", "gr": "WA", "g": "vNJS4H", "h": "vNJS4H", "l": "bnjacobs", "al": "en-US",
  "a": "Mozilla\\5.0 (Macintosh; U; Intel Mac OS X 10.4; en-US; rv:1.9.2.27) Gecko\\20120216 Firefox\\3.6.27", "c": "US", "nk": 0, "tz": "America\\New_York", "gr": "D", "g": "vNJS4H", "h": "vNJS4H", "l": "bnjacobs", "al": "en-US",
  "a": "Mozilla\\5.0 (Windows NT 6.1; WOW64; rv:10.0.2) Gecko\\20100101 Firefox\\10.0.2", "c": "US", "nk": 1, "tz": "America\\New_York", "gr": "VA", "g": "vNJS4H", "h": "vNJS4H", "l": "bnjacobs", "al": "en-US",
  "heartbeat": "1331923261"
}
```

JSON(JavaScript Object Notation) 형식

• 특징

- JSON은 경량(Lightweight)의 DATA-교환 형식
- Javascript에서 객체를 만들 때 사용하는 표현식을 의미
- 사람과 기계 모두 이해하기 쉬우며 용량이 작음
 - **최근에는 JSON이 XML을 대체해서 데이터 전송 등에 많이 사용**
- 특정 언어에 종속되지 않으며, 대부분의 프로그래밍 언어에서 JSON 포맷의 데이터를 핸들링 할 수 있는 라이브러리를 제공

• 파이썬의 사전 형식

```
{  
    "firstName": "Kwon",  
    "lastName": "YoungJae",  
    "email": "kyoje11@gmail.com",  
    "hobby": ["puzzles", "swimming"]  
}
```

파일 읽기

• 스냅 샷 파일의 행

- JSON
- 한 줄 검사

• records 객체

- 파이썬 사전 리스트
- 총 3560개

```
In [5]: import json
path = 'datasets/bitly_usagov/example.txt'
open(path).readline()
```

```
Out[5]: '{ "a": "Mozilla###/5.0 (Windows NT 6.1; WOW64) AppleWebKit###/535.11 (KHTML, like Ge
cko) Chrome###/17.0.963.78 Safari###/535.11", "c": "US", "nk": 1, "tz": "America###/Ne
w_York", "gr": "MA", "g": "A6q0VH", "h": "wFLQtf", "l": "orofrog", "al": "en-US,en;
q=0.8", "hh": "1.usa.gov", "r": "http:###/###/www.facebook.com###/l###/7AQEFzjSi###/1.us
a.gov###/wFLQtf", "u": "http:###/###/www.ncbi.nlm.nih.gov###/pubmed###/22415991", "t": 1
331923247, "hc": 1331822918, "cy": "Danvers", "ll": [ 42.576698, -70.954903 ] }#n'
```

```
In [10]: records = [json.loads(line) for line in open(path, encoding="utf-8")]
records[0]
```

```
Out[10]: {'a': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.11 (KHTML, like Gecko) C
hrome/17.0.963.78 Safari/535.11',
'c': 'US',
'nk': 1,
'tz': 'America/New_York',
'gr': 'MA',
'g': 'A6q0VH',
'h': 'wFLQtf',
'l': 'orofrog',
'al': 'en-US,en;q=0.8',
'hh': '1.usa.gov',
'r': 'http://www.facebook.com/l/7AQEFzjSi/1.usa.gov/wFLQtf',
'u': 'http://www.ncbi.nlm.nih.gov/pubmed/22415991',
't': 1331923247,
'hc': 1331822918,
'cy': 'Danvers',
'll': [42.576698, -70.954903]}
```

표준 시간대 파악

표준 시간대

- tz 필드
- 모든 행이 키 'tz'가 있는 건 아님
 - 오류 발생

```
In [15]: records[0]['tz']
```

```
Out[15]: 'America/New_York'
```

```
In [13]: time_zones = [rec['tz'] for rec in records]
```

```
-----
KeyError                                Traceback (most recent call last)
<ipython-input-13-f3fbbc37f129> in <module>
----> 1 time_zones = [rec['tz'] for rec in records]

<ipython-input-13-f3fbbc37f129> in <listcomp>(.0)
----> 1 time_zones = [rec['tz'] for rec in records]

KeyError: 'tz'
```

```
In [25]: records[:10]
```

```
{
  'hh': '1.usa.gov',
  'r': 'http://www.facebook.com/l.php?u=http%3A%2F%2F1.usa.gov%2FzpkJBR&h=fAQG5ntSGAQHqKPIWzuJKUA9LYeckHZCUxvjQipJDd7Rmmw',
  'u': 'http://www.nasa.gov/mission_pages/nustar/main/index.html',
  't': 1331923254,
  'hc': 1331922854},
{
  'a': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.79 Safari/535.11',
  'c': None,
  'nk': 0,
  'tz': '',
  'g': 'zCaLwp',
  'h': 'zUtuOu',
  'l': 'alex88',
  'al': 'pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4',
  'hh': '1.usa.gov',
  'r': 'http://t.co/o1Pd0WeV',
  'u': 'http://apod.nasa.gov/apod/ap120312.html',
  't': 1331923255,
  'hc': 1331923068}]
```

파이썬으로 표준시간대 세기(1)

- 키 tz 없는 행 처리
 - 키가 있는 것만
 - time_zones에 추가
 - 키 'tz'의 값으로
 - "도 많음
 - 총 3560개
 - 키 'tz'가 있는 것은
 - 3440 개

```
In [29]: time_zones = [rec['tz'] for rec in records if 'tz' in rec]
         time_zones[:10]
```

```
Out[29]: ['America/New_York',
          'America/Denver',
          'America/New_York',
          'America/Sao_Paulo',
          'America/New_York',
          'America/New_York',
          'America/New_York',
          'Europe/Warsaw',
          ' ',
          ' ']
```

```
In [30]: len(time_zones)
```

```
Out[30]: 3440
```

타임존이
있는 갯수

```
In [31]: time_zones2 = [rec.get('tz', None) for rec in records]
         len(time_zones2)
```

```
Out[31]: 3560
```

타임존이 없는
None도 추가

defaultdict와 counter

표준 패키지 collections

- defaultdict
 - 딕셔너리(dictionary)와 거의 비슷하지만 키 값이 없을 경우 미리 지정해 놓은 초기(default) 값을 지정하는 dictionary
- Counter
 - 컨테이너에 동일한 값의 자료가 몇개인지 파악하는데 사용하는 객체
 - A Counter is a dict subclass for counting hashable objects.
 - collections.Counter()의 결과 값(return)은 딕셔너리 형태
 - 메소드 most_common(n)
 - 빈도 수 내림차순으로

```
In [9]: from collections import defaultdict
s = ['a', 'b', 'c', 'b', 'a', 'b', 'c']
d = defaultdict(int)

for k in s:
    d[k] += 1
d
```

```
Out[9]: defaultdict(int, {'a': 2, 'b': 3, 'c': 2})
```

```
In [20]: c = Counter(s)
c
```

```
Out[20]: Counter({'a': 2, 'b': 3, 'c': 2})
```

```
In [13]: from collections import Counter
lst = ['aa', 'cc', 'dd', 'aa', 'bb', 'ee']
print(Counter(lst))
```

```
Counter({'aa': 2, 'cc': 1, 'dd': 1, 'bb': 1, 'ee': 1})
```

```
In [17]: Counter({'가': 3, '나': 2, '다': 4})
```

```
Out[17]: Counter({'가': 3, '나': 2, '다': 4})
```


파이썬으로 표준시간대 세기(2)

• 사전 counts에는 표준 시간대 수가 저장

```
In [32]: def get_counts(sequence):
        counts = {}
        for x in sequence:
            if x in counts:
                counts[x] += 1
            else:
                counts[x] = 1
        return counts
```

```
In [34]: counts['America/New_York']
```

```
Out [34]: 1251
```

```
In [35]: len(time_zones)
```

```
Out [35]: 3440
```

```
In [33]: from collections import defaultdict

        def get_counts2(sequence):
            counts = defaultdict(int) # values will initialize to 0
            for x in sequence:
                counts[x] += 1
            return counts
```

딕셔너리(dictionary)와 거의 비슷하지만 키 값이 없을 경우 미리 지정해 놓은 초기(default) 값을 지정하는 dictionary

```
In [22]: counts = get_counts(time_zones)
        counts
```

```
Out [22]: {'America/New_York': 1251,
            'America/Denver': 191,
            'America/Sao_Paulo': 33,
            'Europe/Warsaw': 16,
            '': 521,
            'America/Los_Angeles': 382,
            'Asia/Hong_Kong': 10,
            'Europe/Rome': 27,
```

파이썬으로 표준시간대 세기(3)

- 상위 10 개의 표준시간대

```
In [38]: def top_counts(count_dict, n=10):
          value_key_pairs = [(count, tz) for tz, count in count_dict.items()]
          value_key_pairs.sort()
          return value_key_pairs[-n:]
```

```
In [39]: top_counts(counts)
```

```
Out [39]: [(33, 'America/Sao_Paulo'),
           (35, 'Europe/Madrid'),
           (36, 'Pacific/Honolulu'),
           (37, 'Asia/Tokyo'),
           (74, 'Europe/London'),
           (191, 'America/Denver'),
           (382, 'America/Los_Angeles'),
           (400, 'America/Chicago'),
           (521, ''),
           (1251, 'America/New_York')]
```

- 표준 라이브러리
collections.Counter
사용

```
In [40]: from collections import Counter
          counts = Counter(time_zones)
          counts.most_common(10)
```

```
Out [40]: [('America/New_York', 1251),
           ('', 521),
           ('America/Chicago', 400),
           ('America/Los_Angeles', 382),
           ('America/Denver', 191),
           ('Europe/London', 74),
           ('Asia/Tokyo', 37),
           ('Pacific/Honolulu', 36),
           ('Europe/Madrid', 35),
           ('America/Sao_Paulo', 33)]
```

데이터프레임으로 보기

• 메소드 info()

– 데이터프레임의 요약 정보

• 행 3560, 열 18 개

In [48]: frame

Out [48]:

| | | a | c | nk | | tz | gr | g | h | | l | al | hh | |
|------|--|-----|-----|---------------------|-----|--------|--------|--------------|----------------|-----------|-----|-----|-----|-----|
| 0 | Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit... | US | 1.0 | America/New_York | MA | A6qOVH | wfLQtF | orofrog | en-US,en;q=0.8 | 1.usa.gov | htt | | | |
| 1 | GoogleMaps/RochesterNY | US | 0.0 | America/Denver | UT | mwszkS | mwszkS | bitly | NaN | j.mp | | | | |
| 2 | Mozilla/4.0 (compatible; MSIE 8.0; Windows NT ... | US | 1.0 | America/New_York | DC | xxr3Qb | xxr3Qb | bitly | en-US | 1.usa.gov | | | | |
| 3 | Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8)... | BR | 0.0 | America/Sao_Paulo | 27 | zCaLwp | zUtuOu | alelex88 | pt-br | 1.usa.gov | | | | |
| 4 | Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit... | US | 0.0 | America/New_York | MA | 9b6kNI | 9b6kNI | bitly | en-US,en;q=0.8 | bit.ly | | | | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 3555 | Mozilla/4.0 (compatible; MSIE 9.0; Windows NT ... | US | 1.0 | America/New_York | NJ | e5SvKE | fQPSr9 | tweetdeckapi | en | 1.usa.gov | | | | |
| 3556 | Mozilla/5.0 (Windows NT 5.1) AppleWebKit/535.1... | US | 0.0 | America/Chicago | OK | jQLtP4 | jQLtP4 | bitly | en-US,en;q=0.8 | 1.usa.gov | | | | |
| 3557 | GoogleMaps/RochesterNY | US | 0.0 | America/Denver | UT | mwszkS | mwszkS | bitly | NaN | j.mp | | | | |
| 3558 | GoogleProducer | US | 0.0 | America/Los_Angeles | CA | zjtI4X | zjtI4X | bitly | NaN | 1.usa.gov | | | | |
| 3559 | Mozilla/4.0 (compatible; MSIE 8.0; Windows NT ... | US | 0.0 | America/New_York | VA | qxKrTK | qxKrTK | bitly | en-US | 1.usa.gov | | | | |

3560 rows × 18 columns

In [45]:

```
import pandas as pd
frame = pd.DataFrame(records)
frame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3560 entries, 0 to 3559
Data columns (total 18 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   a               3440 non-null   object
1   c               2919 non-null   object
2   nk              3440 non-null   float64
3   tz              3440 non-null   object
4   gr              2919 non-null   object
5   g               3440 non-null   object
6   h               3440 non-null   object
7   l               3440 non-null   object
8   al              3094 non-null   object
9   hh              3440 non-null   object
10  r               3440 non-null   object
11  u               3440 non-null   object
12  t               3440 non-null   float64
13  hc              3440 non-null   float64
14  cy              2919 non-null   object
15  ll              2919 non-null   object
16  _heartbeat_     120 non-null    float64
17  kw              93 non-null     object
dtypes: float64(4), object(14)
memory usage: 500.8+ KB
```

In [46]: frame['tz'][:10]

Out [46]:

| | |
|---|-------------------|
| 0 | America/New_York |
| 1 | America/Denver |
| 2 | America/New_York |
| 3 | America/Sao_Paulo |
| 4 | America/New_York |
| 5 | America/New_York |
| 6 | Europe/Warsaw |
| 7 | |
| 8 | |
| 9 | |

Name: tz, dtype: object

판다스로 표준시간대 세기

• 간단히 처리

- 필드 tz 아예 빠진 것은
 - **Missing**으로 넣고
- 필드 tz가 ""인 것은
 - **시간대 이름을 unknown**으로

```
In [49]: tz_counts = frame['tz'].value_counts()
         tz_counts[:10]
```

```
Out [49]: America/New_York      1251
          America/Chicago      400
          America/Los_Angeles   382
          America/Denver        191
          Europe/London         74
          Asia/Tokyo            37
          Pacific/Honolulu      36
          Europe/Madrid         35
          America/Sao_Paulo     33
          Name: tz, dtype: int64
```

```
In [54]: clean_tz = frame['tz'].fillna('Missing')
         clean_tz[clean_tz == ''] = 'Unknown'
         tz_counts = clean_tz.value_counts()
         tz_counts[:10]
```

```
Out [54]: America/New_York      1251
          Unknown              521
          America/Chicago      400
          America/Los_Angeles   382
          America/Denver        191
          Missing              120
          Europe/London         74
          Asia/Tokyo            37
          Pacific/Honolulu      36
          Europe/Madrid         35
          Name: tz, dtype: int64
```

수평 막대 그리기

가장 많이 나타난 시간대 10개

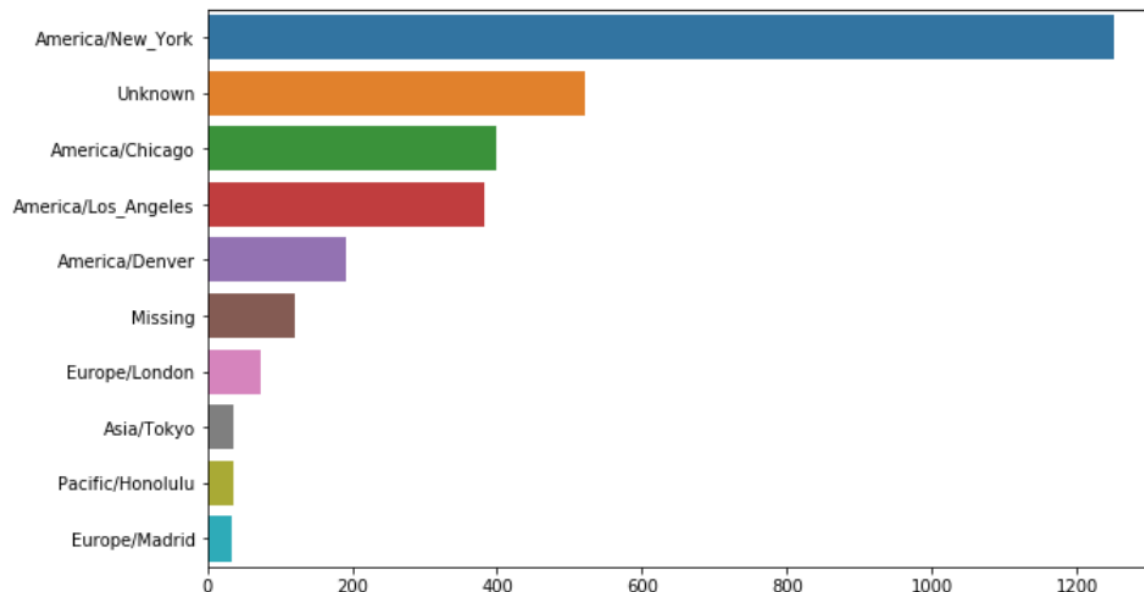
```
In [55]: plt.figure(figsize=(10, 4))
```

```
Out[55]: <Figure size 720x288 with 0 Axes>
```

```
<Figure size 720x288 with 0 Axes>
```

```
In [56]: import seaborn as sns
subset = tz_counts[:10]
sns.barplot(y=subset.index, x=subset.values)
```

```
Out[56]: <matplotlib.axes._subplots.AxesSubplot at 0x2515b4c3f88>
```



필드 a 분석

- URL 단축을 실행하는 정보
 - 브라우저
 - 단말기
 - 애플리케이션
- 브라우저의 종류와 수 알기
 - 첫 토큰(문자열)

열 'a' 자료를 분리하여
첫 번째 내용(브라우저
종류)만을 저장한 시리
즈 생성

```
In [57]: frame['a'][0:2]
Out[57]: 0    Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit
          1    GoogleMaps/RochesterNY
          Name: a, dtype: object

In [58]: frame['a'][1]
Out[58]: 'GoogleMaps/RochesterNY'

In [59]: frame['a'][50]
Out[59]: 'Mozilla/5.0 (Windows NT 5.1; rv:10.0.2) Gecko/20100101 Firefox/10.0.2'

In [60]: frame['a'][51][:50] # long line
Out[60]: 'Mozilla/5.0 (Linux; U; Android 2.2.2; en-us; LG-P9'
```

In [61]: results = pd.Series([x.split()[0] for x in frame.a.dropna()])
Out[61]: 0 Mozilla/5.0
 1 GoogleMaps/RochesterNY
 2 Mozilla/4.0
 3 Mozilla/5.0
 4 Mozilla/5.0
 dtype: object

```
In [63]: results.value_counts()[:8]
Out[63]: Mozilla/5.0          2594
          Mozilla/4.0          601
          GoogleMaps/RochesterNY  121
          Opera/9.80           34
          TEST_INTERNET_AGENT   24
          GoogleProducer        21
          Mozilla/6.0           5
          BlackBerry8520/5.0.0.681 4
          dtype: int64
```

시간대와 원도 사용자

표준 시간대를

– 원도/비원도 사용자로 비교

- 필드 a(agent 문자열)에 Windows 포함 여부에 따라

```
In [80]: cframe
```

```
Out [80]:
```

| | a | c | nk |
|------|--|-----|-----|
| 0 | Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit... | US | 1.0 |
| 1 | GoogleMaps/RochesterNY | US | 0.0 |
| 2 | Mozilla/4.0 (compatible; MSIE 8.0; Windows NT ... | US | 1.0 |
| 3 | Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8)... | BR | 0.0 |
| 4 | Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit... | US | 0.0 |
| ... | ... | ... | ... |
| 3555 | Mozilla/4.0 (compatible; MSIE 9.0; Windows NT ... | US | 1.0 |
| 3556 | Mozilla/5.0 (Windows NT 5.1) AppleWebKit/535.1... | US | 0.0 |
| 3557 | GoogleMaps/RochesterNY | US | 0.0 |
| 3558 | GoogleProducer | US | 0.0 |
| 3559 | Mozilla/4.0 (compatible; MSIE 8.0; Windows NT ... | US | 0.0 |

3440 rows × 4 columns

```
In [85]: by_tz_os = cframe.groupby(['tz', 'os'])
by_tz_os.size()
```

```
Out [85]: tz          os
          Africa/Cairo  Windows    3
          Africa/Casablanca  Windows    1
          Africa/Ceuta    Windows    2
          Europe/Warsaw    Windows   15
          Europe/Zurich    Not Windows    4
          Pacific/Auckland  Not Windows    3
          Pacific/Honolulu  Windows    8
          Length: 149, dtype: int64
```

“으로 지정된
시간대의 수

```
In [65]: cframe = frame[frame.a.notnull()]
```

```
In [66]: cframe = cframe.copy()
```

```
In [67]: cframe['os'] = np.where(cframe['a'].str.contains('Windows'),
                                'Windows', 'Not Windows')
cframe['os'][:5]
```

```
Out [67]: 0      Windows
          1    Not Windows
          2      Windows
          3    Not Windows
          4      Windows
          Name: os, dtype: object
```

```
8]: by_tz_os = cframe.groupby(['tz', 'os'])
```

```
9]: agg_counts = by_tz_os.size().unstack().fillna(0)
agg_counts[:10]
```

| | os | Not Windows | Windows |
|--------------------------------|----|-------------|---------|
| tz | | | |
| | | 245.0 | 276.0 |
| Africa/Cairo | | 0.0 | 3.0 |
| Africa/Casablanca | | 0.0 | 1.0 |
| Africa/Ceuta | | 0.0 | 2.0 |
| Africa/Johannesburg | | 0.0 | 1.0 |
| Africa/Lusaka | | 0.0 | 1.0 |
| America/Anchorage | | 4.0 | 1.0 |
| America/Argentina/Buenos_Aires | | 1.0 | 0.0 |
| America/Argentina/Cordoba | | 0.0 | 1.0 |
| America/Argentina/Mendoza | | 0.0 | 1.0 |

데이터프레임 정렬

• 메소드

**sort_values(by='열명',
ascending=False)**

```
In [152]: data = {
            "도시": ["서울", "서울", "서울", "부산", "부산", "부산", "인천", "인천"],
            "연도": ["2015", "2010", "2005", "2015", "2010", "2005", "2015", "2010"],
            "인구": [9904312, 9631482, 9762546, 3448737, 3393191, 3512547, 2890451, 263203],
            "지역": ["수도권", "수도권", "수도권", "경상권", "경상권", "경상권", "수도권", "수도권"],
            }
            columns = ["도시", "연도", "인구", "지역"]
            df1 = pd.DataFrame(data, columns=columns)
            df1
```

Out [152]:

```
In [161]: df1.sort_values(by=['연도', '인구'], ascending=False)
```

Out [161]:

| | 도시 | 연도 | 인구 | 지역 |
|---|----|------|---------|-----|
| 0 | 서울 | 2015 | 9904312 | 수도권 |
| 3 | 부산 | 2015 | 3448737 | 경상권 |
| 6 | 인천 | 2015 | 2890451 | 수도권 |
| 1 | 서울 | 2010 | 9631482 | 수도권 |
| 4 | 부산 | 2010 | 3393191 | 경상권 |
| 7 | 인천 | 2010 | 263203 | 수도권 |
| 2 | 서울 | 2005 | 9762546 | 수도권 |
| 5 | 부산 | 2005 | 3512547 | 경상권 |

```
In [159]: df1.인구.sort_values()
```

Out [159]:

```
7      263203
6      2890451
4      3393191
3      3448737
5      3512547
1      9631482
2      9762546
0      9904312
Name: 인구, dtype: int64
```

| | 도시 | 연도 | 인구 | 지역 |
|---|----|------|---------|-----|
| 0 | 서울 | 2015 | 9904312 | 수도권 |
| 1 | 서울 | 2010 | 9631482 | 수도권 |
| 2 | 서울 | 2005 | 9762546 | 수도권 |
| 3 | 부산 | 2015 | 3448737 | 경상권 |
| 4 | 부산 | 2010 | 3393191 | 경상권 |
| 5 | 부산 | 2005 | 3512547 | 경상권 |
| 6 | 인천 | 2015 | 2890451 | 수도권 |
| 7 | 인천 | 2010 | 263203 | 수도권 |

```
Out [159]: df1.sort_values(by='인구', ascending=False)
```

Out [159]:

| | 도시 | 연도 | 인구 | 지역 |
|---|----|------|---------|-----|
| 0 | 서울 | 2015 | 9904312 | 수도권 |
| 2 | 서울 | 2005 | 9762546 | 수도권 |
| 1 | 서울 | 2010 | 9631482 | 수도권 |
| 5 | 부산 | 2005 | 3512547 | 경상권 |
| 3 | 부산 | 2015 | 3448737 | 경상권 |
| 4 | 부산 | 2010 | 3393191 | 경상권 |
| 6 | 인천 | 2015 | 2890451 | 수도권 |
| 7 | 인천 | 2010 | 263203 | 수도권 |

```
In [159]: df1.인구.sort_values()
```

```
Out [159]: 7      263203
6      2890451
4      3393191
3      3448737
5      3512547
1      9631482
2      9762546
0      9904312
Name: 인구, dtype: int64
```


데이터프레임 정렬 요약



[Python pandas DataFrame]

Sort by value → Select Top N per Group

| index | group | value |
|-------|-------|-------|
| 0 | A | 1 |
| 1 | A | 3 |
| 2 | A | 4 |
| 3 | A | 5 |
| 4 | A | 2 |
| 5 | B | 7 |
| 6 | B | 10 |
| 7 | B | 8 |
| 8 | B | 9 |
| 9 | B | 6 |

```
df2 = df.sort_values(by="val",
                    ascending=False).W
        groupby("grp").head(3)
```



```
df2.sort_values(by=["grp", "val"],
               ascending=[True, False])
```

<http://rfriend.tistory.com>

| index | group | value |
|-------|-------|-------|
| 3 | A | 5 |
| 2 | A | 4 |
| 1 | A | 3 |
| 6 | B | 10 |
| 8 | B | 9 |
| 7 | B | 8 |

메소드 argsort()

- 값을 (기본) 올림차순으로 정렬한 인덱스 시리즈를 반환
 - 첫 번째 행은 순위 7

| | 도시 | 연도 | 인구 | 지역 |
|---|----|------|---------|-----|
| 0 | 서울 | 2015 | 9904312 | 수도권 |
| 1 | 서울 | 2010 | 9631482 | 수도권 |
| 2 | 서울 | 2005 | 9762546 | 수도권 |
| 3 | 부산 | 2015 | 3448737 | 경상권 |
| 4 | 부산 | 2010 | 3393191 | 경상권 |
| 5 | 부산 | 2005 | 3512547 | 경상권 |
| 6 | 인천 | 2015 | 2890451 | 수도권 |
| 7 | 인천 | 2010 | 263203 | 수도권 |

```
In [157]: indexer = df1.인구.argsort()
          indexer
```

```
Out[157]: 0    7
          1    6
          2    4
          3    3
          4    5
          5    1
          6    2
          7    0
          Name: 인구, dtype: int64
```

```
In [158]: df1.take(indexer)
```

```
Out[158]:
```

| | 도시 | 연도 | 인구 | 지역 |
|---|----|------|---------|-----|
| 7 | 인천 | 2010 | 263203 | 수도권 |
| 6 | 인천 | 2015 | 2890451 | 수도권 |
| 4 | 부산 | 2010 | 3393191 | 경상권 |
| 3 | 부산 | 2015 | 3448737 | 경상권 |
| 5 | 부산 | 2005 | 3512547 | 경상권 |
| 1 | 서울 | 2010 | 9631482 | 수도권 |
| 2 | 서울 | 2005 | 9762546 | 수도권 |
| 0 | 서울 | 2015 | 9904312 | 수도권 |

전체 표준시간대 순위

- 먼저 표준시간대 합 구하고
 - 순위의 arg 구하기
 - 순위 첨자인 indexer로 자료를 take
 - 마지막 10개 가장 큰 값

```
In [129]: count_subset = agg_counts.take(indexer[-10:])
count_subset
```

Out[129]:

| | os | Not Windows | Windows |
|---------------------|----|-------------|---------|
| tz | | | |
| America/Sao_Paulo | | 13.0 | 20.0 |
| Europe/Madrid | | 16.0 | 19.0 |
| Pacific/Honolulu | | 0.0 | 36.0 |
| Asia/Tokyo | | 2.0 | 35.0 |
| Europe/London | | 43.0 | 31.0 |
| America/Denver | | 132.0 | 59.0 |
| America/Los_Angeles | | 130.0 | 252.0 |
| America/Chicago | | 115.0 | 285.0 |
| | | 245.0 | 276.0 |
| America/New_York | | 339.0 | 912.0 |

```
In [91]: agg_counts
```

Out[91]:

| | os | Not Windows | Windows |
|---------------------|----|-------------|---------|
| tz | | | |
| | | 245.0 | 276.0 |
| Africa/Cairo | | 0.0 | 3.0 |
| Africa/Casablanca | | 0.0 | 1.0 |
| Africa/Ceuta | | 0.0 | 2.0 |
| Africa/Johannesburg | | 0.0 | 1.0 |
| ... | | ... | ... |
| Europe/Volgograd | | 0.0 | 1.0 |
| Europe/Warsaw | | 1.0 | 15.0 |
| Europe/Zurich | | 4.0 | 0.0 |
| Pacific/Auckland | | 3.0 | 8.0 |
| Pacific/Honolulu | | 0.0 | 36.0 |

97 rows × 2 columns

```
In [72]: # Use to sort in ascending order
indexer = agg_counts.sum(1).argsort()
indexer[:10]
```

Out[72]: tz

```
Africa/Cairo
Africa/Casablanca
Africa/Ceuta
Africa/Johannesburg
Africa/Lusaka
America/Anchorage
America/Argentina/Buenos_Aires
America/Argentina/Cordoba
America/Argentina/Mendoza
dtype: int64
```

시간대의 수
로 정렬한 첨
자를 준비,
가장 작은 10
개 출력

전체 표준시간대 순위 간단히

• 판다스의 `nlargest()`

- 다음으로 간단히 처리

```
In [130]: agg_counts.sum(1).nlargest(10)
```

```
Out[130]: tz
America/New_York    1251.0
                  521.0
America/Chicago     400.0
America/Los_Angeles 382.0
America/Denver      191.0
Europe/London        74.0
Asia/Tokyo           37.0
Pacific/Honolulu     36.0
Europe/Madrid        35.0
America/Sao_Paulo    33.0
dtype: float64
```

```
In [101]: agg_counts.take(agg_counts.sum(1).argsort()[::-11:-1])
```

```
Out[101]:
```

| os | Not Windows | Windows |
|---------------------|-------------|---------|
| tz | | |
| America/New_York | 339.0 | 912.0 |
| | 245.0 | 276.0 |
| America/Chicago | 115.0 | 285.0 |
| America/Los_Angeles | 130.0 | 252.0 |
| America/Denver | 132.0 | 59.0 |
| Europe/London | 43.0 | 31.0 |
| Asia/Tokyo | 2.0 | 35.0 |
| Pacific/Honolulu | 0.0 | 36.0 |
| Europe/Madrid | 16.0 | 19.0 |
| America/Sao_Paulo | 13.0 | 20.0 |

중첩 막대 그래프

In [103]: count_subset

Out[103]:

| | os | Not Windows | Windows |
|---------------------|----|-------------|---------|
| tz | | | |
| America/Sao_Paulo | | 13.0 | 20.0 |
| Europe/Madrid | | 16.0 | 19.0 |
| Pacific/Honolulu | | 0.0 | 36.0 |
| Asia/Tokyo | | 2.0 | 35.0 |
| Europe/London | | 43.0 | 31.0 |
| America/Denver | | 132.0 | 59.0 |
| America/Los_Angeles | | 130.0 | 252.0 |
| America/Chicago | | 115.0 | 285.0 |
| | | 245.0 | 276.0 |
| America/New_York | | 339.0 | 912.0 |

In [165]:

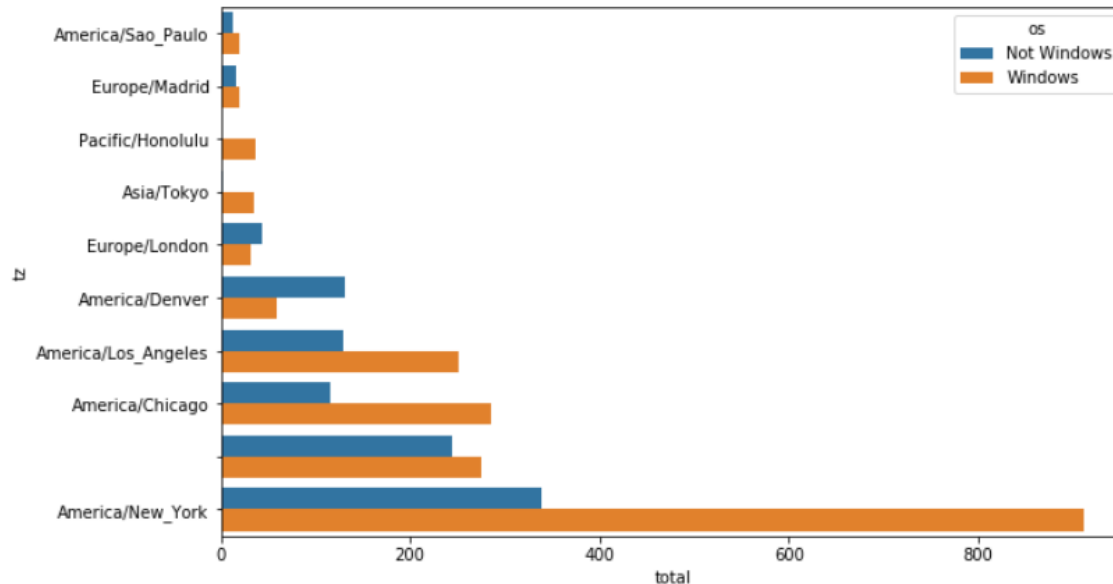
```
# Rearrange the data for plotting
count_subset = count_subset.stack()
count_subset.name = 'total'
count_subset = count_subset.reset_index()
count_subset[:10]
```

Out[165]:

| | tz | os | total |
|---|-------------------|-------------|-------|
| 0 | America/Sao_Paulo | Not Windows | 13.0 |
| 1 | America/Sao_Paulo | Windows | 20.0 |
| 2 | Europe/Madrid | Not Windows | 16.0 |
| 3 | Europe/Madrid | Windows | 19.0 |
| 4 | Pacific/Honolulu | Not Windows | 0.0 |
| 5 | Pacific/Honolulu | Windows | 36.0 |
| 6 | Asia/Tokyo | Not Windows | 2.0 |
| 7 | Asia/Tokyo | Windows | 35.0 |
| 8 | Europe/London | Not Windows | 43.0 |
| 9 | Europe/London | Windows | 31.0 |

In [166]: sns.barplot(x='total', y='tz', hue='os', data=count_subset)

Out[166]: <matplotlib.axes._subplots.AxesSubplot at 0x25157a3bf08>



시간대를 모두 정규화시킨 그래프

• 시간대 사용자 총합을 1로 한 정규화된 그래프

```
In [77]: def norm_total(group):
        group['normed_total'] = group.total / group.total.sum()
        return group

        results = count_subset.groupby('tz').apply(norm_total)
```

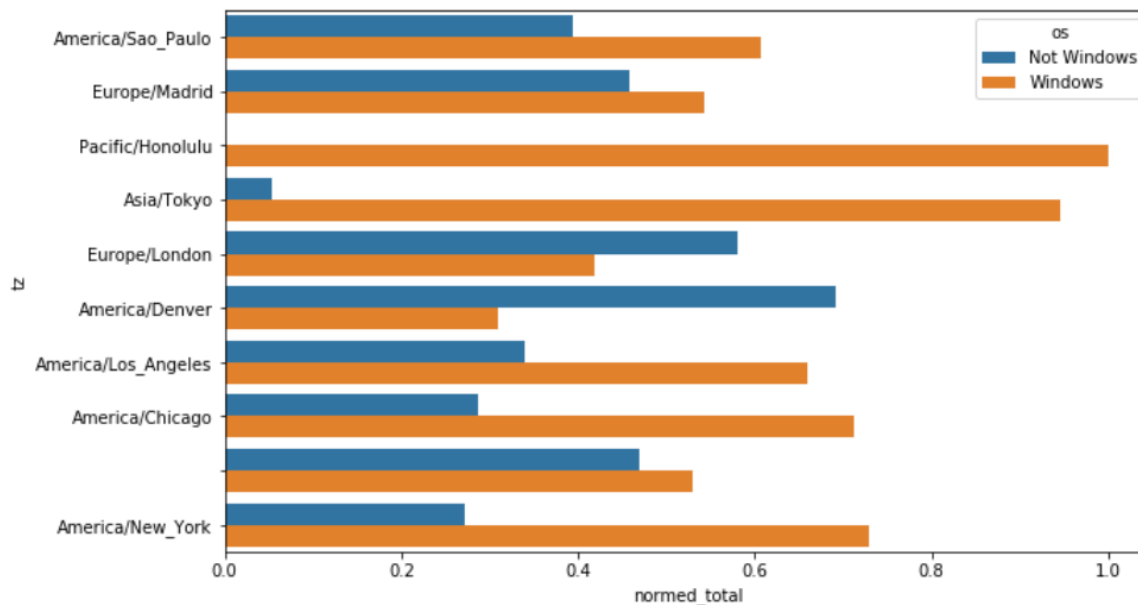
```
In [78]: plt.figure()
```

```
Out[78]: <Figure size 720x432 with 0 Axes>
```

```
<Figure size 720x432 with 0 Axes>
```

```
In [79]: sns.barplot(x='normed_total', y='tz', hue='os', data=results)
```

```
Out[79]: <matplotlib.axes._subplots.AxesSubplot at 0x2515b9366c8>
```



정규화 계산 효율화

• 메소드 groupby와 transform 사용

```
In [89]: g = count_subset.groupby('tz')
results2 = count_subset.total / g.total.transform('sum')
results2
```

```
Out[89]: 0    0.393939
1    0.606061
2    0.457143
3    0.542857
4    0.000000
5    1.000000
6    0.054054
7    0.945946
8    0.581081
9    0.418919
10   0.691099
11   0.308901
12   0.340314
13   0.659686
14   0.287500
15   0.712500
16   0.470250
17   0.529750
18   0.270983
19   0.729017
Name: total, dtype: float64
```

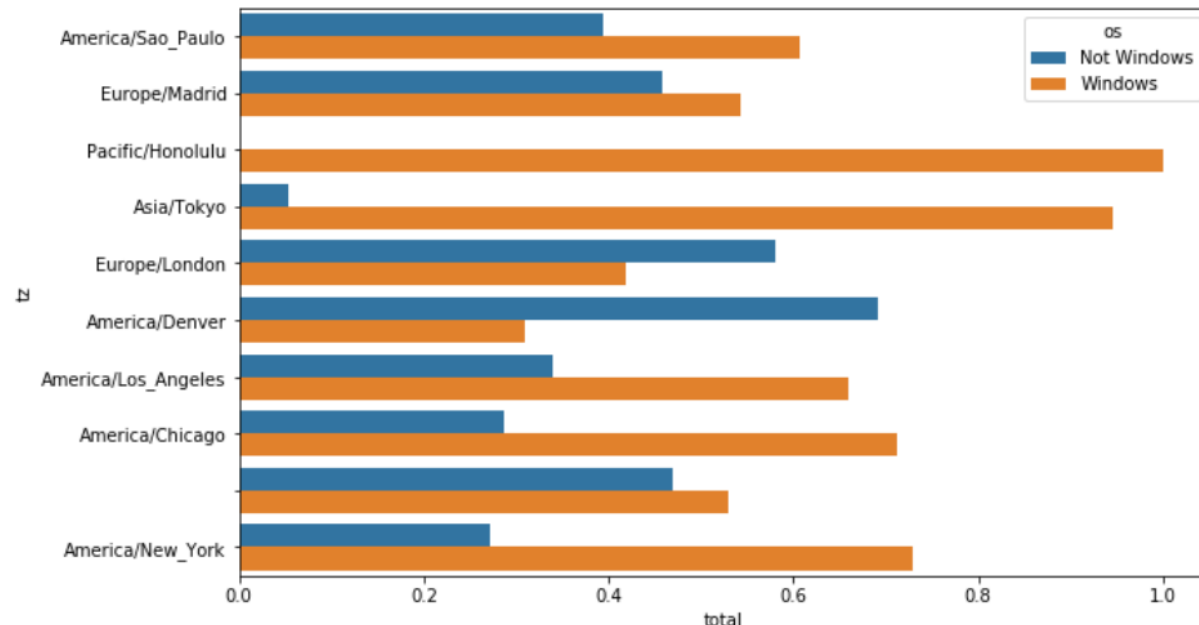
```
In [124]: count_subset
```

```
Out[124]:
```

| | | tz | os | total |
|---|-------------------|---------|---------|-------|
| 0 | America/Sao_Paulo | Not | Windows | 13.0 |
| 1 | America/Sao_Paulo | Windows | | 20.0 |
| 2 | Europe/Madrid | Not | Windows | 16.0 |

```
In [90]: sns.barplot(x=results2, y='tz', hue='os', data=results)
```

```
Out[90]: <matplotlib.axes._subplots.AxesSubplot at 0x2515bbb1308>
```



메소드 transform()

- 그룹별 대표 값을 만드는 것이 아니라 그룹별 계산을 통해 데이터프레임 자체를 변화
- 만들어진 데이터프레임의 크기는 원래 데이터프레임과 같음

```
In [132]: df = pd.DataFrame({'Year': [1997, 1997, 1997, 1998, 1999],
                             'Japan': [100, 100, 300, 200, 100],
                             'USA': [200, 100, 300, 400, 500],
                             'Canada': [400, 300, 200, 100, 400]},
                             index=['1', '2', '3', '4', '5'])
```

Out[132]:

| | Year | Japan | USA | Canada |
|---|------|-------|-----|--------|
| 1 | 1997 | 100 | 200 | 400 |
| 2 | 1997 | 100 | 100 | 300 |
| 3 | 1997 | 300 | 300 | 200 |
| 4 | 1998 | 200 | 400 | 100 |
| 5 | 1999 | 100 | 500 | 400 |

```
In [131]: df.groupby('Year').transform(np.sum)
```

Out[131]:

| | Japan | USA | Canada |
|---|-------|-----|--------|
| 1 | 500 | 600 | 900 |
| 2 | 500 | 600 | 900 |
| 3 | 500 | 600 | 900 |
| 4 | 200 | 400 | 100 |
| 5 | 100 | 500 | 400 |

```
In [133]: df.groupby('Year').sum()
```

Out[133]:

| | Japan | USA | Canada |
|------|-------|-----|--------|
| Year | | | |
| 1997 | 500 | 600 | 900 |
| 1998 | 200 | 400 | 100 |
| 1999 | 100 | 500 | 400 |