Pandas - pohodlná práca s dátami

Matej Oravec

BAB37ZPR

10. týždeň

Čo sú to dáta

Wikipedia

Dáta sú charakteristiky alebo informácie, zvyčajne číselné, získané *pozorovaním*. Viac technicky povedané ide o súbor hodnôt kvalitatívnych alebo kvantitatívnych premenných, týkajúcich sa jedného alebo viacerých subjektov.

▸ Wikipedia

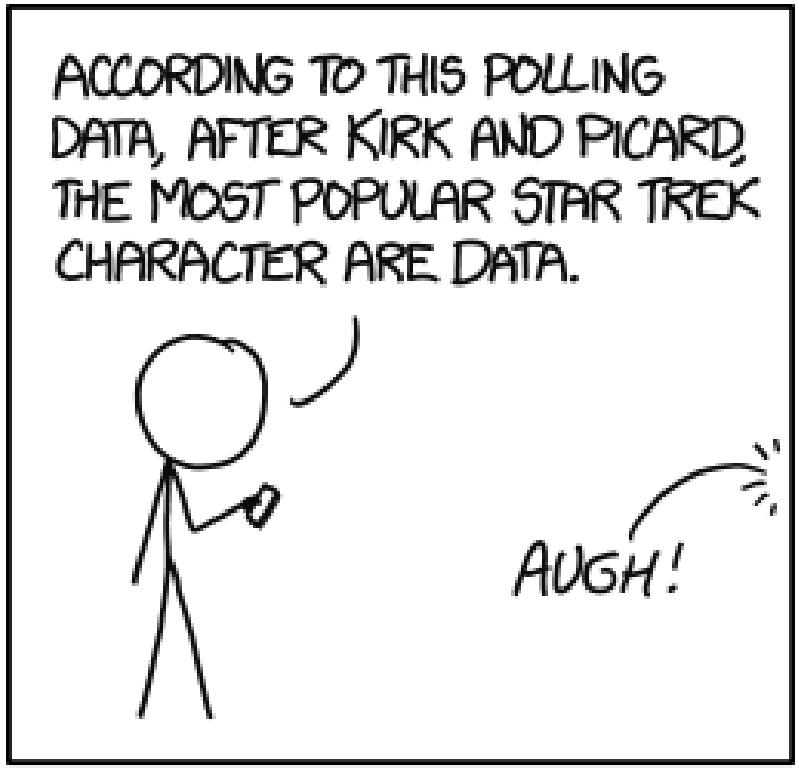
(voľný preklad)

Ako to chápať

Dáta sú výsledky meraní

- Merania nemusia byť presné
- Meradlom môže byť čokoľvek
- Formát výstupu meradla nie je definovaný

Čo sú to dáta



ANNOY GRAMMAR PEDANTS ON ALL SIDES BY MAKING "DATA" SINGULAR *EXCEPT* WHEN REFERRING TO THE ANDROID.

▸ xkcd

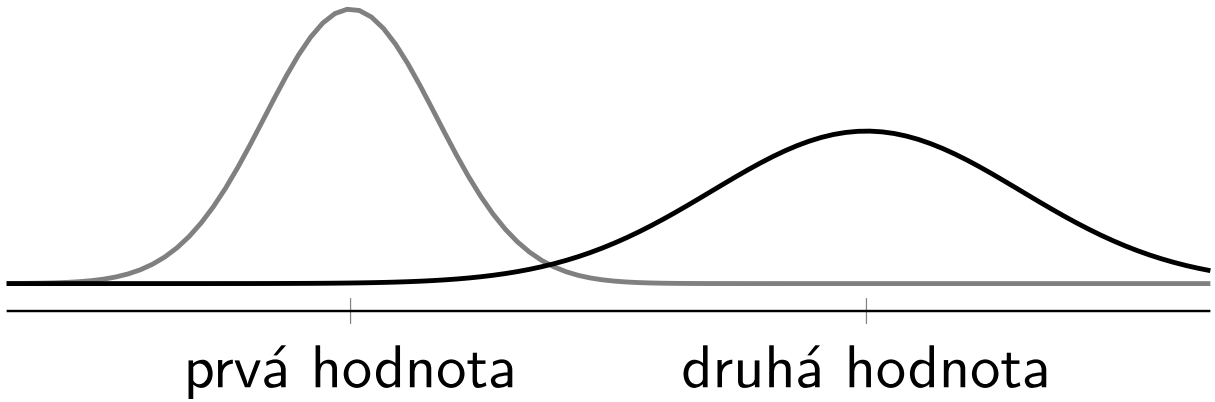
Ako s dátami nakladať

Viac dát – viac muziky

Neexistujú zbytočné dáta

- Musíme vedieť vyhodnocovať ich kvalitu
  - Používame štatistiku

Príklad: Meranie teploty dvoma teplomermi



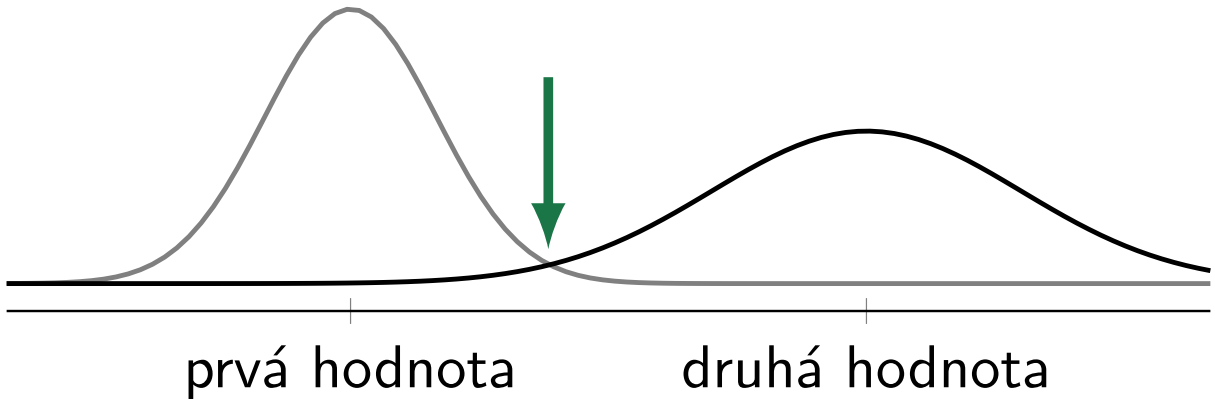
Ako s dátami nakladať

Viac dát – viac muziky

Neexistujú zbytočné dáta

- Musíme vedieť vyhodnocovať ich kvalitu
  - Používame štatistiku

Príklad: Meranie teploty dvoma teplomermi



Dnešná prednáška

Čo sa dnes nenaučíme

- Ako dáta rigorózne analyzovať
  - Predmet *Analýza experimentálních dat*
- Štatistiku

▸ Popis

Čo sa dnes naučíme

- Základy *manipulácie* s dátami
  - Načítať dáta
  - Exportovať dáta
  - Slušne dáta preskupovať
- Analýzovať dáta *pohľadom*
  - Vytvorenie prvej predstavy o dátovom súbore

Pandas

Python data analysis library

Pandas

Python data analysis library

- Balíček na *prácu s dátami*
- Implementácia objektu `DataFrame`
- *Systematizácia* dát
- Nástroje na *okamžitú jednoduchú analýzu*

▸ Pandas ▸ Github

▸ Docs

Pandas

Inštalácia, zahrnutie do programu

Inštalácia

PyPi

- `pip install pandas`

Konvencia pre import

- `import pandas as pd`

pandas.DataFrame

Dátová tabuľka

	1. veličina	2. veličina	3. veličina	$\cdots$	m. veličina
1. realizácia	•	•	•	$\cdots$	•
2. realizácia	•	•	•	$\cdots$	•
3. realizácia	•	•	•	$\cdots$	•
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
n. realizácia	•	•	•	$\cdots$	•

`pandas.DataFrame`

Dátová tabuľka

Stĺpce (`df.columns`)

	1. veličina	2. veličina	3. veličina	⋯	m. veličina
1. realizácia	•	•	•	⋯	•
2. realizácia	•	•	•	⋯	•
3. realizácia	•	•	•	⋯	•
⋮	⋮	⋮	⋮	⋱	⋮
n. realizácia	•	•	•	⋯	•

`pandas.DataFrame`

Dátová tabuľka

	1. veličina	2. veličina	3. veličina	$\cdots$	m. veličina
1. realizácia	•	•	•	$\cdots$	•
2. realizácia	•	•	•	$\cdots$	•
3. realizácia	•	•	•	$\cdots$	•
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
n. realizácia	•	•	•	$\cdots$	•

Indexy (`df.index`)

pandas.DataFrame

Dátová tabuľka

	1. veličina	2. veličina	3. veličina	⋯	m. veličina
1. realizácia	•	•	•	⋯	•
2. realizácia	•	•	•	⋯	•
3. realizácia	•	•	•	⋯	•
⋮	⋮	⋮	⋮	⋱	⋮
n. realizácia	•	•	•	⋯	•

Dáta (`pd.to_numpy(df)`)

pandas.DataFrame

Dátová tabuľka

	1. veličina	2. veličina	3. veličina	⋯	m. veličina
1. realizácia	•	•	•	⋯	•
2. realizácia	•	•	•	⋯	•
3. realizácia	•	•	•	⋯	•
⋮	⋮	⋮	⋮	⋱	⋮
n. realizácia	•	•	•	⋯	•

`pd.DataFrame`

`pandas.DataFrame`

Výhody oproti `numpy.ndarray`

- Prehľadnosť
  - Ku každej z dimenzií je priradený objekt `pandas.Index`
  - Základné štatistiky datasetu dostupné okamžite
    - ★ `DataFrame.describe()`
  - Možnosť časového indexu
    - ★ Veľká výhoda pri práci s časovými postupnosťami
- Integrácia s *matplotlib*
- Rozšírené možnosti indexácie

▸ Docs

pandas.DataFrame

Vytvorenie

Pomocou *listu*

```
1 df = pd.DataFrame([['Adis', 'A', True], ['Beba', 'F', False]],
      columns=['name', 'grade', 'passed'], index=range(2))
```

Pomocou *dictionary*

```
1 df = pd.DataFrame({'name': ['Adis', 'Beba'], 'grade': ['A', 'F'],
        'passed': [True, False]}, index=range(2))
```

Načítanie zo súboru

- Funkcia `pd.read_csv()`

▸ Docs

Objekty typu *index*

Značkovanie riadkov a stĺpcov

`pandas.Index` ▸ Docs

- Základný typ indexu
- Objekt typu `numpy.ndarray`

Špecifikácia indexu

```
1 row_index = pd.Index('a b c d e'.split(' '))
2 column_index = pd.Index(('first', 'second', 'third'))
3 df = pd.DataFrame(np.random.random((5, 3)), index=row_index, columns
      =column_index)
```

Objekty typu *index*

Značkovanie riadkov a stĺpcov

`pandas.Index` ▸ Docs

- Základný typ indexu
- Objekt typu `numpy.ndarray`

Špecifikácia indexu

```
1 df = pd.DataFrame(np.random.random((5, 3)), index='a b c d e'.split(
      ' '), columns=('first', 'second', 'third'))
```

Objekty typu *index*

Značkovanie riadkov a stĺpcov

`pandas.MultiIndex`

Docs

	Zisk			Hodnotenie klientov		
2017	2018	2019	2017	2018	2019
•	•	•	•	•	•
⋮	⋮	⋮	⋮	⋮	⋮
•	•	•	•	•	•

Možnosti vytvorenia

- `pandas.MultiIndex.from_tuples()` Docs
- `pandas.MultiIndex.from_product()` Docs
- `pandas.MultiIndex.from_arrays()` Docs

Indexovanie v pandas.DataFrame

Metódy loc a iloc

Indexovanie ako v numpy

- Možné, ale počíta sa s použitím lokačných metód z pandas

```
1 a = df[3, 10] # Row, column
2 b = df['bank'] # One column. Returns a pd.Series object
3 c = df['parent_col', 'child_col'][3] # Hierarchical indexing
4 d = df[df['client_ranking'] > 5.] # Logical indexing
```

pandas.DataFrame.loc

Docs

- Výber na základe labelu, nie poradia

```
1 a = df.loc[3, 'bank':'client_ranking'] # Row, range of columns
2 b = df.loc[:, 'bank'] # One column. Returns a pd.Series object
3 c = df.loc[3, ('parent_col', 'child_col')] # Hierarchical indexing
4 d = df.loc[df.loc[:, 'client_ranking'] > 5.] # Logical indexing
```

Indexovanie v `pandas.DataFrame`

Metódy `loc` a `iloc`

Indexovanie ako v numpy

- Možné, ale počíta sa s použitím lokačných metód z pandas

```
1 a = df[3, 10] # Row, column
2 b = df['bank'] # One column. Returns a pd.Series object
3 c = df['parent_col', 'child_col'][3] # Hierarchical indexing
4 d = df[df['client_ranking'] > 5.] # Logical indexing
```

`pandas.DataFrame.loc`

Docs

- Pri použití hierarchického indexovania: najvhodnejšie použiť objekt `pandas.IndexSlice`

```
1 idx = pd.IndexSlice
2 a = df.loc[:, idx['client_ranking', '2017':'2018']]
```

Indexovanie v pandas.DataFrame

Metódy loc a iloc

Indexovanie ako v numpy

- Možné, ale počíta sa s použitím lokačných metód z pandas

```
1 a = df[3, 10] # Row, column
2 b = df['bank'] # One column. Returns a pd.Series object
3 c = df['parent_col', 'child_col'][3] # Hierarchical indexing
4 d = df[df['client_ranking'] > 5.] # Logical indexing
```

pandas.DataFrame.iloc

▸ Docs

- Indexovanie poradím
- Pri hierarchickom indexovaní rovnaký postup ako pre pandas.DataFrame.loc

```
1 a = df.iloc[:, :2]
```

Indexovanie v pandas.DataFrame

Metódy loc a iloc

Pozor na číselné indexy

```
>>> df = pd.DataFrame(np.random.random((2, 4)))
>>> print(df.loc[:, :2]) # Stop index INCLUDED
          0         1         2
0  0.799448  0.356222  0.078508
1  0.778320  0.769942  0.317316

>>> print(df.iloc[:, :2]) # Stop index NOT INCLUDED
          0         1
0  0.799448  0.356222
1  0.778320  0.769942
```

Manipulácia so štruktúrou dát

pandas.concat()

df1

```
          a         b
0  0.799448  0.356222
1  0.778320  0.769942
```

df2

```
          c         d
0  0.799448  0.356222
1  0.778320  0.769942
```

pd.concat([df1, df2], axis=1)

Docs

```
          a         b         c         d
0  0.799448  0.356222  0.799448  0.356222
1  0.778320  0.769942  0.778320  0.769942
```

Manipulácia so štruktúrou dát

pandas.pivot()

df1

```
   misc  new    val
0  less    3  small
1  less    4    big
2  more    5  small
3  more    6    big
```

df.pivot(index='misc', columns='val', values='new')

▸ Docs

```
val   big  small
misc
less    4      3
more    6      5
```

Náhľad na dáta

Prvých alebo posledných $n = 5$ riadkov

- `pandas.DataFrame.head()` ▸ Docs
- `pandas.DataFrame.tail()` ▸ Docs

n riadkov s najväčšou alebo najmenšou hodnotou v zadanom stĺpci

- `pandas.DataFrame.nlargest()` ▸ Docs
- `pandas.DataFrame.nsmallest()` ▸ Docs

Základné štatistiky

- `pandas.(DataFrame/Series).value_counts()` ▸ Docs
- `pandas.DataFrame.describe()` ▸ Docs

Náhľad na dáta

Metóda `pandas.DataFrame.describe()`

Aplikácia na dataset *iris*

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Čo s chýbajúcimi dátami

- Dáta môžu chýbať z rôznych dôvodov
  - Absencia merania, meranie mimo konfidenčného intervalu
  - Matematicky nedefinované výrazy (napr. 0/0)
  - Nekompletná pivot tabuľka atď.

Objekty pre prácu s neštandardnými číslami

Numpy docs

- `numpy.Inf` — Pozitívne nekonečno (float)
- `numpy.NINF` — Negatívne nekonečno (float)
- `numpy.NAN` — Nečíslo (float)

Práca s týmito konštantami v Pandas

Pandas

- `pandas.DataFrame.dropna()` — Docs
- `pandas.DataFrame.(isna/notna)()` — Docs (isna)
- `pandas.DataFrame.fillna()` — Docs

Čo s chýbajúcimi dátami

Metóda `pandas.DataFrame.dropna()`

Použitie

```
>>> df
     a  b  c    d  e    f
0  NaN  1  3  2.0  6  3.0
1  9.0  1  1  2.0  5  1.0
2  NaN  8  0  NaN  3  6.0
3  NaN  3  1  7.0  7  8.0
4  7.0  9  0  2.0  5  NaN
```

Čo s chýbajúcimi dátami

Metóda `pandas.DataFrame.dropna()`

Použitie

```
>>> df.dropna()
     a  b  c    d  e    f
1  9.0  1  1  2.0  5  1.0
```

Čo s chýbajúcimi dátami

Metóda `pandas.DataFrame.dropna()`

Použitie

```
>>> df.dropna(axis=1)
   b  c  e
0  1  3  6
1  1  1  5
2  8  0  3
3  3  1  7
4  9  0  5
```

Čo s chýbajúcimi dátami

Metóda pandas.DataFrame.dropna()

Použitie

```
>>> df.dropna(axis=1, thresh=3)
   b  c    d  e    f
0  1  3  2.0  6  3.0
1  1  1  2.0  5  1.0
2  8  0  NaN  3  6.0
3  3  1  7.0  7  8.0
4  9  0  2.0  5  NaN
```

Čo s chýbajúcimi dátami

Metóda `pandas.DataFrame.fillna()`

Použitie

```
>>> df
     a  b  c    d  e    f
0  NaN  1  3  2.0  6  3.0
1  9.0  1  1  2.0  5  1.0
2  NaN  8  0  NaN  3  6.0
3  NaN  3  1  7.0  7  8.0
4  7.0  9  0  2.0  5  NaN
```

Čo s chýbajúcimi dátami

Metóda `pandas.DataFrame.fillna()`

Použitie

```
>>> df.fillna(789)
       a  b  c      d  e      f
0  789.0  1  3    2.0  6    3.0
1    9.0  1  1    2.0  5    1.0
2  789.0  8  0  789.0  3    6.0
3  789.0  3  1    7.0  7    8.0
4    7.0  9  0    2.0  5  789.0
```

Čo s chýbajúcimi dátami

Metóda `pandas.DataFrame.fillna()`

Použitie

```
>>> df.fillna(method='ffill')
     a  b  c    d  e    f
0  NaN  1  3  2.0  6  3.0
1  9.0  1  1  2.0  5  1.0
2  9.0  8  0  2.0  3  6.0
3  9.0  3  1  7.0  7  8.0
4  7.0  9  0  2.0  5  8.0
```

Spájanie dvoch datasetov

- `pandas.concat()` „zliepa“ objekty dokopy ▸ Docs
- Spájanie objektov v zmysle *zlučovania datasetov* → `pandas.DataFrame.merge()` ▸ Docs

Príklad použitia metódy `pandas.DataFrame.merge()`

```
>>> df_bek
                 bekele
weight               56
height              165
nationality    Ethiopia
marathon_time   2:01:41
where            Berlin
year               2019
```

Spájanie dvoch datasetov

- `pandas.concat()` „zliepa“ objekty dokopy ▸ Docs
- Spájanie objektov v zmysle *zlučovania datasetov* → `pandas.DataFrame.merge()` ▸ Docs

Príklad použitia metódy `pandas.DataFrame.merge()`

```
>>> df_kip
                kipchoge
weight                52
height               167
nationality        Kenya
marathon_time   2:01:39
where             Berlin
year                2018
```

Spájanie dvoch datasetov

- `pandas.concat()` „zliepa“ objekty dokopy ▸ Docs
- Spájanie objektov v zmysle *zlučovania datasetov* → `pandas.DataFrame.merge()` ▸ Docs

Príklad použitia metódy `pandas.DataFrame.merge()`

```
>>> df_bek.merge(df_kip, on=df_bek.index).set_index('key_0')
                bekele kipchoge
key_0
weight              56       52
height             165      167
nationality   Ethiopia    Kenya
marathon_time  2:01:41  2:01:39
where           Berlin   Berlin
year              2019     2018
```

Zoskupovanie dát

Metóda `pandas.DataFrame.groupby()`

Príklad: Maratón – najvyšší na jednotlivých miestach

```
>>> df
           weight height     nationality marathon_pb     where  year
bekele         56    165        Ethiopia     2:01:41    Berlin  2019
kipchoge       52    167           Kenya     2:01:39    Berlin  2018
zatopek        72    182  Czechoslovakia     2:23:04  Helsinki  1952
pavlista       62    179  Czech Republic     2:16:30    Prague  2019
kosgei        NaN    NaN           Kenya     2:14:04   Chicago  2019
radcliffe      54    173   Great Britain     2:15:25    London  2003
```

Zoskupovanie dát

Metóda `pandas.DataFrame.groupby()`

Príklad: Maratón – najvyšší na jednotlivých miestach

```
>>> df.loc[:, 'height'].groupby(df.loc[:, 'where']).max()
where
Berlin      167.0
Chicago       NaN
Helsinki    182.0
London      173.0
Prague      179.0
Name: height, dtype: float64
```

Aplikácia funkcií na dáta

pandas.(Series/DataFrame).apply()

Docs

```
>>> df
          A         B
0  0.897537  0.278127
1  0.081324  0.845928
2  0.848448  0.609961
3  0.741872  0.735974
4  0.768731  0.647151
5  0.094650  0.905735
```

Aplikácia funkcií na dáta

pandas.(Series/DataFrame).apply()

Docs

```
>>> df['C'] = df.loc[:, 'B'].apply(lambda x: int(100 * x))
>>> df
          A         B   C
0  0.897537  0.278127  27
1  0.081324  0.845928  84
2  0.848448  0.609961  60
3  0.741872  0.735974  73
4  0.768731  0.647151  64
5  0.094650  0.905735  90
```

Agregácia funkcií

pandas.DataFrame.agg()

Docs

```
>>> df2
          A         B
0  0.897537  0.278127
1  0.081324  0.845928
2  0.848448  0.609961
3  0.741872  0.735974
4  0.768731  0.647151
5  0.094650  0.905735
6  0.499432  0.319021
7  0.346645  0.952854
8  0.953975  0.874165
9  0.690989  0.807034
```

Agregácia funkcií

pandas.DataFrame.agg()

Docs

```
>>> df2.agg(['max', 'std', 'mean'])
              A         B
max    0.953975  0.952854
std    0.321550  0.236785
mean   0.592360  0.697595
```

Ďakujem za pozornosť.