

# Analiza danych z wodowskazów Kraków-Bielany i Czernichów-Prom w latach hydrologicznych 2016-2018

Filip Hajdyla

May 3, 2022

## 1 Zaimportowane moduły i pliki

```
[1]: import time
import zipfile
from glob import glob
import pandas as pd
import matplotlib.pyplot as plt
```

```
[2]: data_dir = 'dane.imgw.pl/data/dane_pomiarowo_obserwacyjne/dane_hydrologiczne/
      ↪dobowe'
zip_files = glob(f'{data_dir}/**/*.zip') # list of .zip files
zip_files[:5] # first five elements
```

```
[2]: ['dane.imgw.pl/data/dane_pomiarowo_obserwacyjne/dane_hydrologiczne/dobowe\\2016\\
      \codz_2016_01.zip',
      'dane.imgw.pl/data/dane_pomiarowo_obserwacyjne/dane_hydrologiczne/dobowe\\2016\\
      \codz_2016_02.zip',
      'dane.imgw.pl/data/dane_pomiarowo_obserwacyjne/dane_hydrologiczne/dobowe\\2016\\
      \codz_2016_03.zip',
      'dane.imgw.pl/data/dane_pomiarowo_obserwacyjne/dane_hydrologiczne/dobowe\\2016\\
      \codz_2016_04.zip',
      'dane.imgw.pl/data/dane_pomiarowo_obserwacyjne/dane_hydrologiczne/dobowe\\2016\\
      \codz_2016_05.zip']
```

## 2 Ekstrakcja z archiwów .zip

Definicja funkcji rozpakowującej zawartości pliku .zip (plik .csv) do tej samej ścieżki i tworzącej listę rozpakowanych plików .csv.

```
[3]: def unpack(files: list) -> list:
      """Unpacks given file list in the same parent dir.
      Returns list of unpacked files."""

      for file in files:
          with zipfile.ZipFile(file) as zip_file:
```

```

zip_file.extractall(file[:77])

csv_files = glob(f'{data_dir}/**/*.csv') # list of .csv files

return sorted(csv_files)

```

### 3 Analiza danych

Poniższy skrypt rozpakowuje pliki .zip w pętli, tworzy z nich obiekty DataFrame (tabele), łączy je ze sobą w jedną wielką tabelę, a następnie przyłącza filtrowane po nazwach stacji fragmenty tabeli do słownika wraz z przypisanym im kluczem — nazwą stacji. W zależności od hardware'u i ilości danych do analizy może to zająć trochę czasu, ale za to daje nam później możliwość pracowania z danymi z dowolnej stacji. Dla danych z lat 2016-2018 powinno to zająć około minuty.

```

[4]: start_time = time.time()
codz_names = (
    'Kod stacji',
    'Nazwa stacji',
    'Nazwa rzeki/jeziora',
    'Rok hydrologiczny',
    'Wskaźnik miesiąca w roku hydrologicznym',
    'Dzień',
    'Stan wody [cm]',
    'Przepływ [m^3/s]',
    'Temperatura wody [st. C]',
    'Miesiąc kalendarzowy'
)

DF = pd.DataFrame()
dates = []
stations = []
for file in unpack(zip_files):
    if 'codz_' in file:
        df = pd.read_csv(
            file,
            encoding='windows-1250',
            names=codz_names
        )

        DF = pd.concat([DF, df])

    for station in df['Nazwa stacji']:
        if station not in stations:
            stations.append(station)

DF = {station: (DF[DF['Nazwa stacji'] == station]) for station in stations}

```

```
print(f'execution time: {time.time() - start_time} s')
```

execution time: 52.51819729804993 s

Definicja funkcji konwertującej 3 kolumny zawierające osobne dane o roku, miesiącu i dniu wykonania pomiaru w jedną kolumnę “Data hydrologiczna”. Funkcja zwraca tabelę z przyłączoną kolumną “Data” oraz usuniętymi kolumnami “Rok hydrologiczny”, “Wskaźnik miesiąca w roku hydrologicznym”, “Dzień” oraz “Miesiąc kalendarzowy”.

```
[5]: def date_creator(df: pd.DataFrame) -> pd.DataFrame:
    """Takes year, month and day cols and combines them into date col_
    ↪ (YYYY-MM-dd format).
    Inserts date col into df and drops year, month and day cols."""

    dates = []
    df = df.set_index(i for i in range(len(df['Kod stacji'])))
    for idx, row in enumerate(df['Rok hydrologiczny']):
        date = f"{row}.{df['Wskaźnik miesiąca w roku hydrologicznym'][idx]:02d}.
    ↪ {df['Dzień'][idx]:02d}"
        dates.append(date)

    df.insert(value=dates, column='Data hydrologiczna', loc=3)

    df = df.drop(
        axis=1,
        labels=[
            'Rok hydrologiczny',
            'Wskaźnik miesiąca w roku hydrologicznym',
            'Dzień',
            'Miesiąc kalendarzowy'
        ]
    )
    return df
```

Definicja funkcji rysującej sformatowany wykres poziomu wody w czasie hydrologicznym na podstawie zadanej tabeli (df).

```
[6]: def plot_wlevel(station: str):
    """Draws formatted plot of water level in time."""

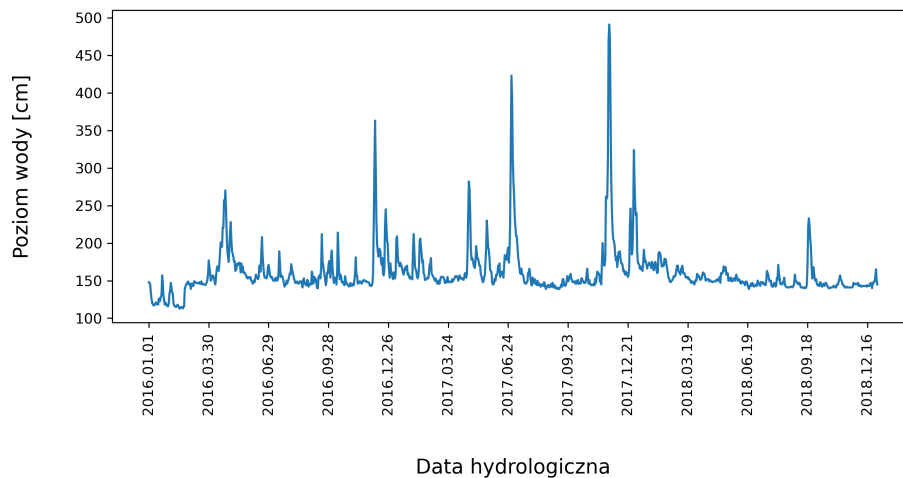
    df = date_creator(Df[station]) # Tabel with data from KRAKÓW-BIELANY station

    fig = plt.figure(figsize=(10, 4), dpi=300, facecolor='#ffffff')
    plt.plot(
        df['Data hydrologiczna'],
        df['Stan wody [cm]'],
    )
```

```
plt.title(f'Poziom wody w Wiśle w latach hydrologicznych 2016-2018 (stacja_
↳{df["Nazwa stacji"][0]}'),
        fontdict={'fontsize': 18}, pad=30)
plt.ylabel('Poziom wody [cm]', fontdict={'fontsize': 14}, labelpad=30)
plt.xlabel('Data hydrologiczna', fontdict={'fontsize': 14}, labelpad=30)
plt.xticks(df['Data hydrologiczna'][:,90], rotation=90)
plt.savefig(f'output/fig-{df["Nazwa stacji"][0]}.png', bbox_inches='tight')
plt.show()
plt.close(fig)
```

```
plot_wlevel('KRAKÓW-BIELANY')
```

Poziom wody w Wiśle w latach hydrologicznych 2016-2018 (stacja KRAKÓW-BIELANY)



Ile wody przepłynęło przez Wisłę w 2017 roku?

```
[7]: czernichow_prom = DF['CZERNICHÓW-PROM']
czernichow_prom2017 = czernichow_prom[czernichow_prom['Rok hydrologiczny'] ==
↳2017]
czernichow_prom2017.set_index(idx for idx, _ in
↳enumerate(czernichow_prom2017['Kod stacji']))
mean_flow = czernichow_prom2017['Przepływ [m^3/s]'].mean()
f'W roku 2017 przez Kraków (stacja CZERNICHÓW-PROM) przepłynęło {mean_flow * 60_
↳* 60 * 24 * len(czernichow_prom2017["Kod stacji"]):.2f} m^3 wody.'
```

```
[7]: 'W roku 2017 przez Kraków (stacja CZERNICHÓW-PROM) przepłynęło 2283965856.00 m^3
wody.'
```