# Druhá část projektu - základní analýza dat

Tato část projektu navazuje na první část. Cílem je provést základní analýzu dat (Statistika nehodovosti Policie ČR), která máme stažena a předzpracována. Řešení se skládá ze 4 úkolů, přičemž každý úkol bude jedna samostatná funkce implementovaná v jazyce Python.

## Vstupní data

Aby byly stejné podmínky pro všechny a nemuseli jste si opravovat první část projektu, budeme **pracovat se souborem accidents.pkl.gz**, který stáhnete z adresy: http://ehw.fit.vutbr.cz/izv/accidents.pkl.gz (stahování souboru není součástí tohoto úkolu, není třeba proto implementovat automaticky).

Obsah souboru byl získán následovně:

```
import pandas as pd
dw = DataDownloader()
df = pd.DataFrame(dw.get_dict())
df.to_pickle("accidents.pkl.gz")
```

Serializovaný DataFrame obsahuje data organizovaná ve sloupích, které jsou pojmenované tak, jak jsou uvedeny v popisu datového souboru (*p1*, *p13a*, …). Data jsou uložena jako `float`, `int` nebo `str` podle sloupce. Neznámé hodnoty jsou u typů float reprezentované jako `np.nan`, u int hodnotou `-1` a v případě řetězců jako prázdný řetězec.

## Požadovaný výstup

Cílem je vytvořit kód, který vizualizuje tři různé závislosti v datech. Veškeré kódy budou součástí jednoho souboru `analysis.py`, jehož **kostru naleznete v souborovém skladu** ve WIS. Předpokládá se, že budete **primárně pracovat s knihovnami Pandas a Seaborn** + je dovolené využít všechny knihovny zmiňované během přednášek. Matplotlib používejte pouze pro doladění vizuální podoby.

## Odevzdávání a hodnocení

Soubor `analysis.py` odevzdejte do 10. 12. 2021. Hodnotit se bude zejména:

- správnost výsledků
- vizuální zpracování grafů
- kvalita kódu
    - efektivita implementace (nebude hodnocena rychlost, ale bude kontrolováno, zda nějakým způsobem řádově nezvyšujete složitost)
    - korektní práce s Pandas a Seaborn
    - přehlednost kódu
    - dodržení standardů a zvyklostí pro práci s jazykem Python (PEP8)
    - dokumentace kódu

Celkem lze získat až 20 bodů, přičemž k zápočtu je nutné získat z této části minimálně 2 body.

## Úkol 1: Načtení dat (až 5 bodů)

**Signatura funkce**, která bude odpovídat tomuto úkolu:

```
def get_dataframe(filename : str = "accidents.pkl.gz",
                  verbose : bool = False
                  ) -> pd.DataFrame:
```

**Funkcionalita**

- Funkce načte lokálně uložený soubor se statistikou nehod (odpovídající http://ehw.fit.vutbr.cz/izv/accidents.pkl.gz), jehož umístění je specifikováno argumentem filename.
- Funkce vytvoří v DataFrame sloupec *date*, který bude ve formátu pro reprezentaci data (berte v potaz pouze datum, t.j. sloupec *p2a*)
- S výjimkou sloupce region reprezentujte vhodné sloupce pomocí kategorického datového typu. Měli byste se dostat pod velikost v paměti 0.5 GB. Sloupec *region* je vhodné ponechat v původní podobě pro lepší práci s figure-level funkcemi Seaborn.
- Při povoleném výpisu ( `verbose == True` ) spočítejte kompletní (hlubokou) velikost všech sloupců v datovém rámci před a po vaší úpravě a vypište na standardní výstup pomocí funkce print následující 2 řádky
  `orig_size=X MB`
  `new_size=X MB`
  Čísla vypisujte na 1 desetinné místo a počítejte, že 1 MB = $10^6$ B.

**Upozornění:** Další skripty budou využívat vaši implementaci parsování datového rámce. Bez této implementace není možné hodnotit další úkoly**.**

## Úkol 2: Počty nehod podle druhů silnic (až 5 bodů)

**Signatura funkce**, která bude odpovídat tomuto úkolu:

```
def plot_roadtype(df: pd.DataFrame, fig_location: str = None,
                  show_figure: bool = False):
```

**Funkcionalita**

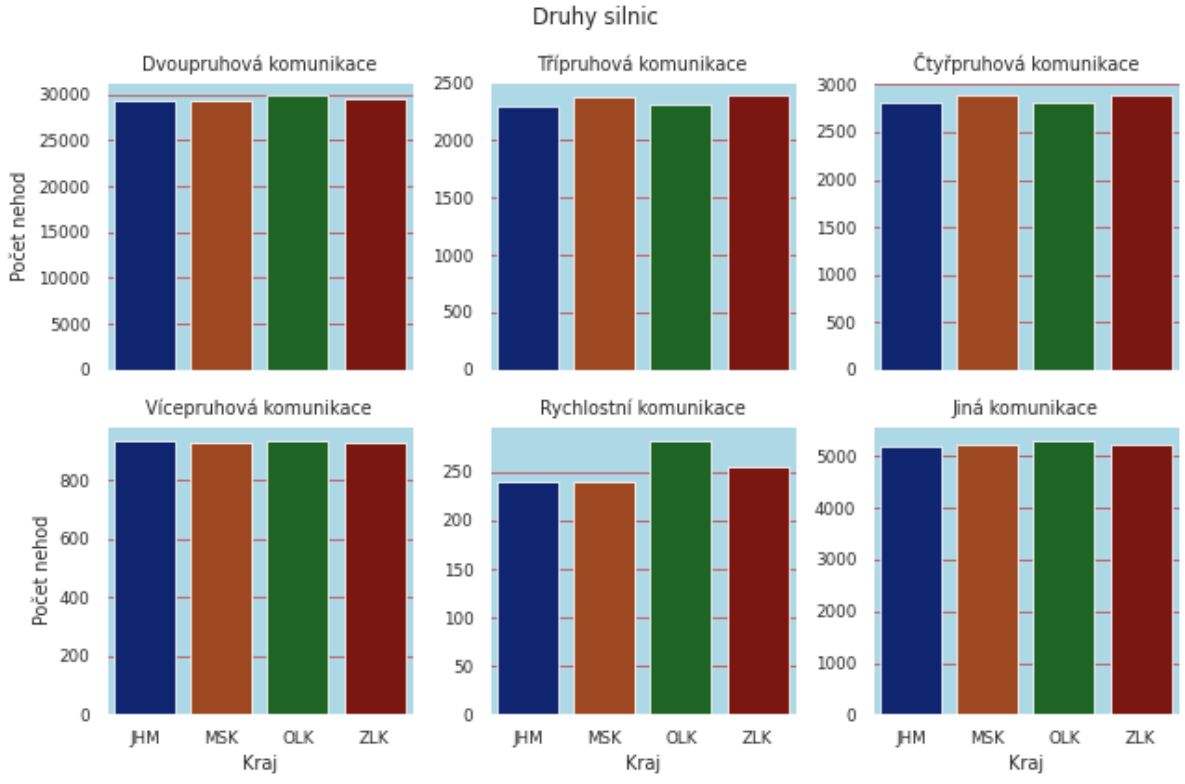
Vytvořte graf počtu nehod v jednotlivých regionech, který uložte do souboru specifikovaného argumentem `fig_location` a případně zobrazte pokud `show_figure je True`. Argument `df` odpovídá DataFrame jenž je výstupem funkce `get_dataframe`. Graf se bude skládat z 6 podgrafů uspořádaných do mřížky o dvou řádcích a třech sloupcích.

Požadavky:

1) Vyberte si čtyři různé kraje.
2) Pracujte se sloupcem *p21*, nerozlišuje mezi různými druhy čtyřpruhové komunikace (hodnoty 3 a 4).
3) Nastavte správně titulky jednotlivých podgrafů.
4) Graf upravte tak, aby popisky na osách, titulky atd. dávaly smysl. Dle zásad dobré vizualizace (viz přednáška 4) zvolte vhodnou barvu a vhodný styl. U podgrafu nastavte vlastní pozadí.

**Tip:** nejdříve je vhodné si nahradit celočíselné hodnoty ve sloupci *p21* vhodnými řetězci a vytvořit nějaký *pomocný* sloupec, který potom budete sčítat při agregaci `groupby`. Vhodným figure-level grafem pak můžete tato data vizualizovat.

**Příklad výstupu:** (použita náhodná data a záměrně zcela nevhodná grafická forma)



## Úkol 3: Nehody zaviněné zvěří (až 5 bodů)

**Signatura funkce**, která bude odpovídat tomuto úkolu:

```
def plot_animals(df: pd.DataFrame, fig_location: str = None,
                 show_figure: bool = False):
```

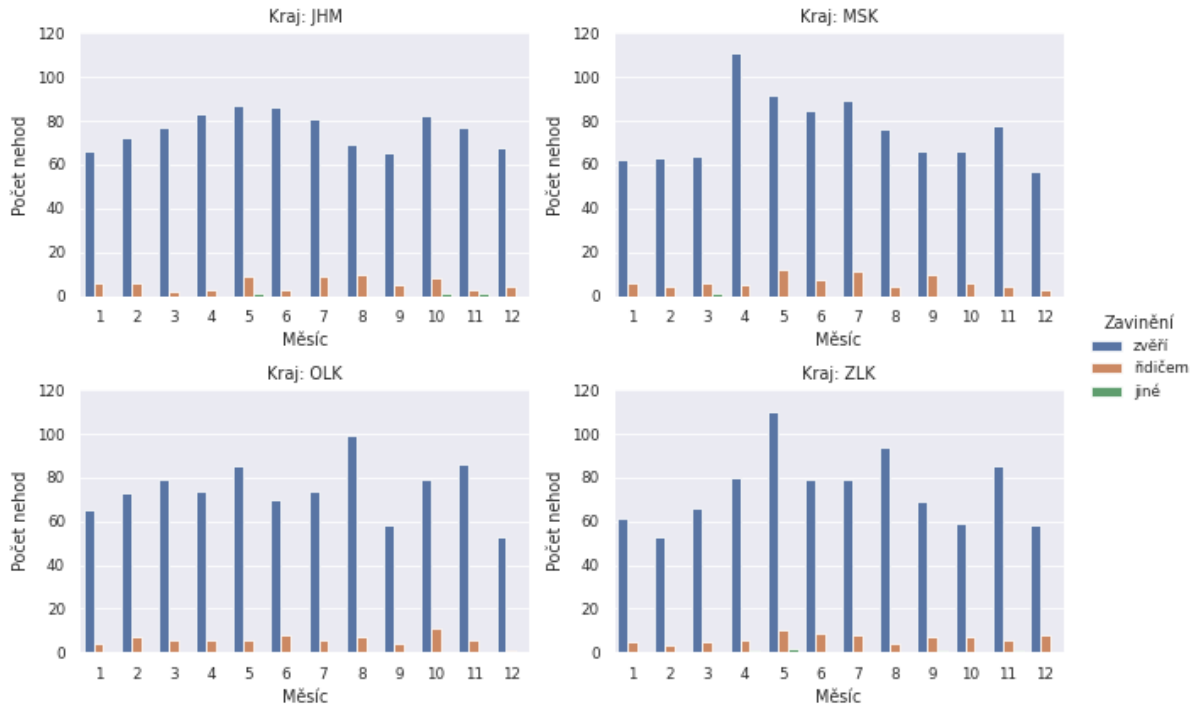
**Funkcionalita**

Vytvořte graf počtu nehod v jednotlivých regionech, který uložte do souboru specifikovaného argumentem `fig_location` a případně zobrazte pokud `show_figure je True`. Argument `df` odpovídá DataFrame jenž je výstupem funkce `get_dataframe`. Pro čtyři vámi vybrané kraje znázorněte počet nehod, které se staly v důsledku vyhýbání lesní zvěři, domácímu zvířectvu atp. *(p58 == 5)*. Určite počty nehod, které zavinil řidič, zvěř či někdo jiný pro jednotlivé měsíce v roce (celkový počet přes všechny roky 2016 - 2020 (jelikož 2021 není kompletní).

**Požadavky a doporučený postup:**

1) Zavinění (p10) konvertujte jako 1 a 2 => “řidičem”, 4 => “zvěří”, zbytek jako “jiné”.
2) Přes `dt` accessor zjistěte měsíc, kdy se stala nehoda a vyfiltrujte pryč nehody, které se staly v roce 2021
3) Data správně agregujte pomocí `groupby` a vykreslete vhodným figure-level grafem.
4) Upravte zobrazení tak, aby se grafy a popisky nepřekrývaly.
5) Graf upravte tak, aby popisky na osách, titulky atd. dával smysl.

**Příklad výstupu:** (použita náhodná data)



## Úkol 4: Povětrnostní podmínky v jednotlivých měsících (až 5 bodů)

**Signatura funkce**, která bude odpovídat tomuto úkolu:

```
def plot_conditions(df: pd.DataFrame, fig_location: str = None,
                    show_figure: bool = False):
```

**Funkcionalita**

Vytvořte graf počtu nehod v jednotlivých regionech pro různé povětrnostní podmínky, který uložte do souboru specifikovaného argumentem `fig_location` a případně zobrazte pokud `show_figure je True`. Argument `df` odpovídá DataFrame jenž je výstupem funkce `get_dataframe`. Pro čtyři vámi vybrané kraje pro různé měsíce vykreslete čárový graf, který bude zobrazovat pro jednotlivé měsíce (osa X- sloupec *date*) počet nehod při různých povětrnostních podmínkách.

**Požadavky a doporučený postup:**

1. Vyberte čtyři kraje a vyfiltrujte všechny nehody, kde byly povětrnostní podmínky *jiné* (p18 == 0)
2. Přepište celočíselné hodnoty podmínek *p18* na textovou reprezentaci
3. Transformujte tabulku tak, aby pro každý den a region byl v každém sloupci odpovídající podmínkám počet nehod (`pivot_table`)
4. Pro každý kraj proveďte podvzorkování na úroveň měsíců a převeďte správně do *stacked formátu*.
5. Vykreslete čárový graf a omezte osu X od 1. 1. 2016 do 1. 1. 2020 (jelikož 2021 není kompletní)
6. Vykreslete patřičné grafy upravené tak, aby popisky na osách, titulky atd. dávaly smysl.

**Příklad výstupu:** (použita náhodná data)

## Poznámky k implementaci

Soubor, který vytvoříte, bude při hodnocení importovaný a budou volány jednotlivé funkce. Mimo tyto funkce, část importů a dokumentační řetězce nepište žádný funkční kód. Blok na konci souboru ohraničený podmínkou

```
if __name__ == "__main__":
    pass
```

naopak můžete upravit libovolně pro testovací účely. Dále můžete přidat další funkce (pokud budete potřebovat), pro názvy těchto funkcí použijte prefix "_".

Stručnou dokumentaci všech částí (souboru a funkcí) uveďte přímo v odevzdaných souborech. Respektuje konvenci pro formátování kódu PEP 257 [PEP 257 -- Docstring Conventions] a PEP 8 [PEP 8 -- Style Guide for Python Code].

Grafy by měly splňovat všechny náležitosti, které u grafu očekáváme, měl by být přehledný a jeho velikost by měla být taková, aby se dal čitelně použít v šířce A4 (t.j. cca 18 cm). Toto omezení není úplně striktní, ale negenerujte grafy, které by byly přes celý monitor.

Grafy v zadání jsou pouze ukázkové. Data byla randomizována a vaše výsledky budou vypadat jinak. Není nutné ani chtěné, aby grafy vizuálně vypadaly stejně - vlastní invence směrem k větší přehlednosti a hezčímu vzhledu se cení! Co není přímo specifikováno v zadání můžete vyřešit podle svého uvážení.

Doporučený postup není nutné dodržet. Důležité je však to, aby výsledky odpovídaly zadání (včetně podvýběru dat a podobně). U argumentů funkcí `fig_location` můžete počítat s tím, že adresář, kam se mají data ukládat, již existuje.

## Dotazy a připomínky

Dotazy a připomínky směřujte na fórum ve WIS případně na mail mrazek@fit.vutbr.cz.