

Interaktivní mapa

Návrh řešení:

jsem se rozhodl vytvořit interaktivní mapu ČR. Vybral jsem si mně blízké téma, prodej osobních automobilů za rok 2021.

Pro vytvoření mapy s pomocí Pythonu potřebujeme některou ze specializovaných knihoven. Já jsem si vybral Folium. Dále potřebujeme nějakou knihovnu, která obstará zpracování dat z tabulky. V mém tomto případě jsem použil knihovnu Pandas. Další důležitou součástí práce bylo obstarání vhodných dat. Na to skvěle posloužil státní portál Registr vozidel. A poslední nepostradatelnou komponentou jsou data hranic územních celků. Já použil okresy a kraje z Geoportálu ČÚZK.

Celou práci na projektu jsem zahájil asi dvoudenním průzkumem návodů na internetu. Je jich mnoho a většina vede k cíli. Ale mnohdy velmi odlišnými cestami.

Během výzkumu jsem narazil na pěkné [návod ke knihovně Folium](#). Začal jsem tedy tam, a nakonec to byla velmi dobrá volba, jak se později ukázalo. Mnoho jiných návodů se na tuto stránku odkazuje.

Pro mou vizi je asi nejvhodnější funkce Choropleth map. Jedná se o barevně rozčleněné oblasti mapy podle načtených dat.

Nicméně pro tvorbu takovéto mapy nám bohužel většinou nestačí data v takovém formátu, v jakém se dají najít. Proto jsem musel geografická data konvertovat na formát JSON a excelovou tabulku doplnit o ID kódy krajů a okresů, které v ní standardně (data z registru vozidel) nejsou.

Pro získání dat o přihlášených vozech za rok 2021 jsme využili volného přístupu k informacím na stránce ministerstva dopravy [Registr vozidel](#).

Na postup získání Geo dat a jejich konverzi jsem objevil [velmi pěkný návod](#). Na jeho základě jsem se dostal na již zmíněnou stránku [Geoportál ČÚZK](#) a na stránku určenou ke konverzi a zjednodušení geografických souborů [Mapshaper.org](#). Po zjednodušení až na 1% a tím zmenšení velikosti souboru na rozumnou velikost jsem ručně upravil Excelovou tabulku. Asi by se to dalo zařídit pomocí programování, nebo maker, ale to bylo v tu chvíli nad moje síly (a nervy).

Teprve poté došlo na samotné programování dle výše uvedeného [návodu ke knihovně Folium](#).

Z tohoto návodu jsem využil tento kód:

```
import pandas as pd

url = (
    "https://raw.githubusercontent.com/python-visualization/folium/master/examples/data"
)
state_geo = f"{url}/us-states.json"
state_unemployment = f"{url}/US_Unemployment_Oct2012.csv"
state_data = pd.read_csv(state_unemployment)

m = folium.Map(location=[48, -102], zoom_start=3)

folium.Choropleth(
    geo_data=state_geo,
    name="choropleth",
    data=state_data,
    columns=["State", "Unemployment"],
    key_on="feature.id",
    fill_color="YlGn",
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name="Unemployment Rate (%)",
).add_to(m)

folium.LayerControl().add_to(m)

m
```

Jeho modifikací na moje podmínky vzniklo toto:

```
import pandas as pd
import folium
```

```
state_geo1 = (r"SPH_KRAJ1.json")
state_cars1 = (r"data.xlsx")
state_data1 = pd.read_excel(state_cars1)
state_data.head()
```

```
m = folium.Map(location=[49.8171606, 15.4766247], zoom_start=8)
```

```
folium.Choropleth(
    geo_data=state_geo1,
    name="Kraje",
    data=state_data1,
    columns=["ID", "Auto"],
    key_on="properties.ID",
    fill_color="Paired",
    fill_opacity=0.7,
    line_opacity=0.3,
    legend_name="Nově přihlášená osobní vozidla podle krajů v roce 2021",
    bins=bins,
    reset=True,
).add_to(m)
```

```
m.save("CZ_cars_2021.html")
```

Dalším postupem docházelo k ladění a vylepšování.

Doplnil jsem tedy upravenou legendu s pomocí tohoto kódu:

```
bins = list(state_data["Unemployment"].quantile([0, 0.25, 0.5, 0.75, 1]))

m = folium.Map(location=[48, -102], zoom_start=3)

folium.Choropleth(
    geo_data=state_geo,
    data=state_data,
    columns=["State", "Unemployment"],
    key_on="feature.id",
    fill_color="BuPu",
    fill_opacity=0.7,
    line_opacity=0.5,
    legend_name="Unemployment Rate (%)",
    bins=bins,
    reset=True,
).add_to(m)

m
```

Moje verze:

```
bins = list(state_data1["Auta"].quantile([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 1]))
```

Na [této stránce](#) jsem objevil krásný a podrobný popis jak do mapy přidat překryvnou vrstvu s popisky odpovídajícími jednotlivým oblastem a k nim patřícím datům.

```
style_function = lambda x: {'fillColor': '#ffffff',
                             'color': '#000000',
                             'fillOpacity': 0.1,
                             'weight': 0.1}
highlight_function = lambda x: {'fillColor': '#000000',
                                 'color': '#000000',
                                 'fillOpacity': 0.50,
                                 'weight': 0.1}

NIL = folium.features.GeoJson(
    nilpop,
    style_function=style_function,
    control=False,
    highlight_function=highlight_function,
    tooltip=folium.features.GeoJsonTooltip(
        fields=['NAME_NIL', 'PER_FOREIGN'],
        aliases=['Neighborhood: ', 'Resident foreign population in %: '],
        style=("background-color: white; color: #333333; font-family: arial; font-size: 12px; padding: 10px;")
    )
)
mymap.add_child(NIL)
mymap.keep_in_front(NIL)
folium.LayerControl().add_to(mymap)
mymap
```

Bohužel jsem narazil na několik problému a nepodařilo se mi tuto funkcionalitu využít tak, jak jsem zamýšlel. Použil jsem ji jen v základní formě.

```

style_function = lambda x: {'fillColor': '#ffffff',
                             'color': '#000000',
                             'fillOpacity': 0.1,
                             'weight': 0.1}
highlight_function = lambda x: {'fillColor': '#000000',
                                 'color': '#000000',
                                 'fillOpacity': 0.50,
                                 'weight': 0.1}

NIL = folium.features.GeoJson(
    state_geo1,
    style_function=style_function,
    control=False,
    highlight_function=highlight_function,
    tooltip=folium.features.GeoJsonTooltip(
        fields=['NAZEV_NUTS'],
        aliases=['Kraj: '],
        style=("background-color: white; color: #333333; font-family: arial; font-size: 12px; padding: 10px;")
    )
)
m.add_child(NIL)
m.keep_in_front(NIL)
folium.LayerControl().add_to(m)
m

```

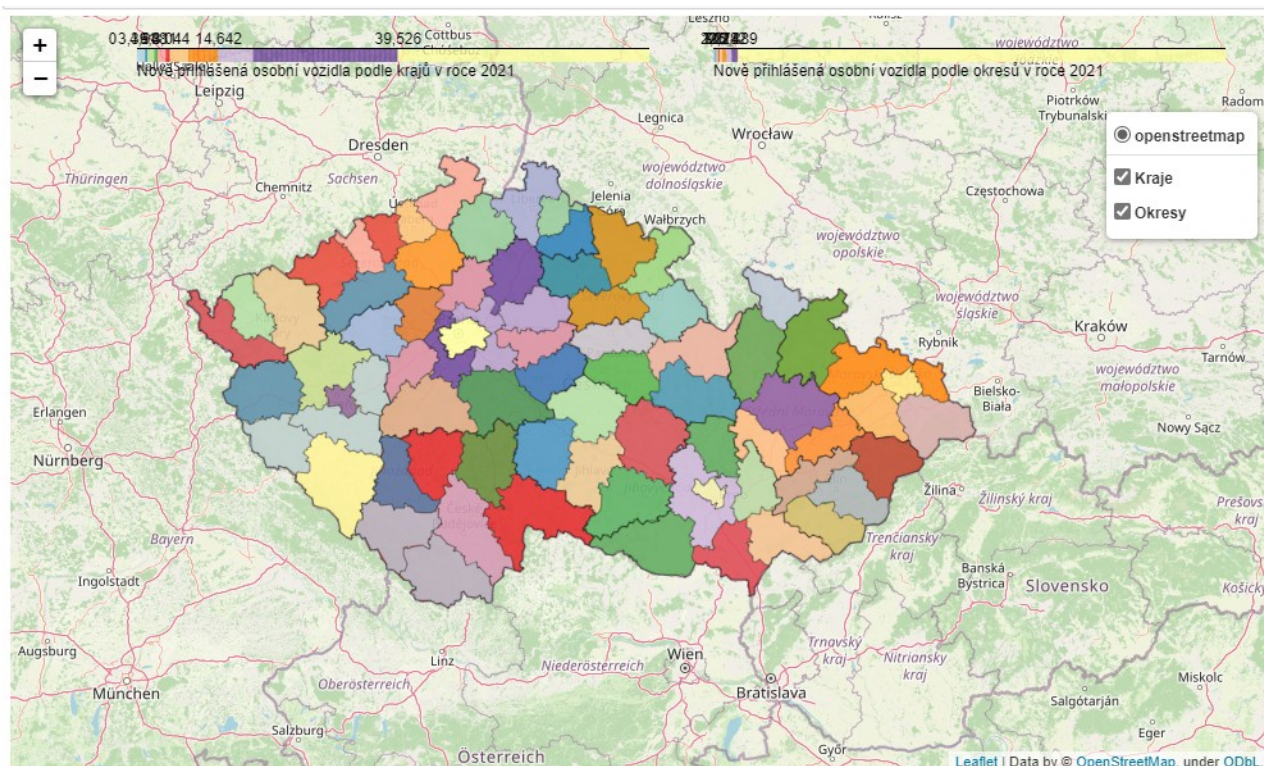
Nepodařilo se totiž nainstalovat knihovnu Geopandas, na které je závislé spojení Geo souboru s tabulkou do jednoho datového prvku. Folium totiž umí načíst jako zdroj dat pouze Geo data. Ne třeba excel nebo csv. Tím pádem mám k dispozici pouze názvy okresů a krajů, ale už ne počty prodaných aut.

Dalším nevyřešeným problémem je, že se mi nepodařilo přidat dvě překryvné vrstvy tak, aby byly závislé na přepínači vrstev. Tedy aby se při zobrazení okresů ukazovaly popisky pro okresy a u krajů jen pro kraje. Vždy byla aktivní jedna pro obě verze a druhá byla potlačena.

Také jsem narazil na problematiku velmi odlišných počtů prodaných vozů vzhledem k zobrazení v mapě. Např. Praha má prodeje kolem 70 000ks aut ročně. Druhý Středočeský kraj asi 20 000ks a poslední Karlovarský kraj pouze 3000ks. To jsou pro zobrazení v Choroplethu strašně rozdílná čísla. Ve standardním nastavení ukáže mapa cokoliv od cca 10 000ks nahoru jako jednu barvu. Tím poněkud ztrácíme vypovídající hodnotu mapy. Dá se použít např. zobrazení logaritmu hodnot, čímž se rozsah zmenší, ale zase v mapě nejsou reálná čísla. Já jsem tedy raději použil úpravu nastavení legendy. K ideálu bohužel chybí výše zmiňovaná plně funkční překryvná vrstva.

Výsledky a fakta:

Výsledkem je myslím celkem povedená mapka ČR s barevně vyznačenými počty prodaných aut v ČR. Níže je uveden snímek, funkční mapa bude součástí příloh.



Pro tvorbu byly využity knihovny:

- Pandas
- Folium

Při tvorbě jsem využil následující zdroje:

- [Geoportál ČÚZK](#)
- [Registr vozidel](#)
- [Návod na knihovnu Folium](#)
- [Návod na konverzi Geo dat](#)
- [Konvertor Geo dat - Mapshaper](#)
- [Návod - Interactive choropleth with Python and Folium](#)

Závěr:

Přípravou této práce jsem strávil několik dní, ale musím přiznat, že jsem se naučil hodně o programování v Pythonu a o zdrojích informací pro tuto práci. Většina mnou použitých informací se nachází na serverech Githubu.

Začínal jsem programovat v Pycharm, ale postupně jsem se přesunul do Jupyter Notebooku, který umí napsaný kód i vizualizovat. Což bylo zrovna u tvorby mapy výrazné usnadnění práce.