TIP_examples

April 10, 2025

0.1 Visualizing EP Patent Filings by CPC Class and Country

This section demonstrates querying and visualizing EP patent filings by applicant country, filtered by CPC classes (e.g., Wind Energy, Geothermal Energy), using PATSTAT data. The output includes choropleth maps showing the ratio of filings across EPC countries.

[4]:
```
import plotly.express as px
from plotly.subplots import make_subplots
from sqlalchemy import func

from epo.tipdata.patstat import PatstatClient
from epo.tipdata.patstat.database.models import (
    TLS201_APPLN, TLS206_PERSON, TLS207_PERS_APPLN, TLS224_APPLN_CPC
)

# ----------------------
# 1) Adjust parameters here
# ----------------------
cpc_filter_1 = "Y02E  10/7%"
cpc_exp_1    = "Wind Energy"
cpc_filter_2 = "Y02E  10/1%"
cpc_exp_2    = "Geothermal Energy"
start_date   = "2015-01-01"


# ----------------------
# 2) DB Connection
# ----------------------
patstat = PatstatClient(env="PROD")
db = patstat.orm()


epc_countries = [
    "AL","AT","BE","BG","CH","CY","CZ","DE","DK","EE","ES","FI","FR",
    "GB","GR","HR","HU","IE","IS","IT","LI","LT","LU","LV","MC","ME",
    "MK","MT","NL","NO","PL","PT","RO","RS","SE","SI","SK","SM","TR"
]
alpha2_to_alpha3 = {
```

```
    "AL": "ALB", "AT": "AUT", "BE": "BEL", "BG": "BGR", "CH": "CHE",
    "CY": "CYP", "CZ": "CZE", "DE": "DEU", "DK": "DNK", "EE": "EST",
    "ES": "ESP", "FI": "FIN", "FR": "FRA", "GB": "GBR", "GR": "GRC",
    "HR": "HRV", "HU": "HUN", "IE": "IRL", "IS": "ISL", "IT": "ITA",
    "LI": "LIE", "LT": "LTU", "LU": "LUX", "LV": "LVA", "MC": "MCO",
    "ME": "MNE", "MK": "MKD", "MT": "MLT", "NL": "NLD", "NO": "NOR",
    "PL": "POL", "PT": "PRT", "RO": "ROU", "RS": "SRB", "SE": "SWE",
    "SI": "SVN", "SK": "SVK", "SM": "SMR", "TR": "TUR"
}

# ----------------------
# 3) Helper functions
# ----------------------
def get_filings_by_country(
    db, patstat, cpc_filter, start_date,
    epc_countries, alpha2_to_alpha3
):
    q = (
        db.query(
            TLS206_PERSON.person_ctry_code.label("country"),
            func.count(TLS201_APPLN.appln_id).label("filings_count")
        )
        .select_from(TLS201_APPLN)
        .join(
            TLS207_PERS_APPLN,
            TLS201_APPLN.appln_id == TLS207_PERS_APPLN.appln_id
        )
        .join(
            TLS206_PERSON,
            TLS207_PERS_APPLN.person_id == TLS206_PERSON.person_id
        )
        .join(
            TLS224_APPLN_CPC,
            TLS224_APPLN_CPC.appln_id == TLS201_APPLN.appln_id
        )
        .filter(
            TLS201_APPLN.appln_auth == "EP",
            TLS201_APPLN.appln_filing_date >= start_date,
            TLS207_PERS_APPLN.applt_seq_nr > 0,
            TLS224_APPLN_CPC.cpc_class_symbol.like(cpc_filter)
        )
        .group_by(TLS206_PERSON.person_ctry_code)
    )
    df = patstat.df(q)
    df = df[df["country"].isin(epc_countries)]
    df["country_alpha3"] = df["country"].map(alpha2_to_alpha3)
    return df.dropna(subset=["country_alpha3"])
```

```python
def normalize_filings(df):
    df["ratio_of_filings"] = df["filings_count"] / df["filings_count"].sum()
    return df

# ----------------------
# 4) Query & Normalize
# ----------------------
df1 = normalize_filings(
    get_filings_by_country(
        db, patstat, cpc_filter_1, start_date,
        epc_countries, alpha2_to_alpha3
    )
)
df2 = normalize_filings(
    get_filings_by_country(
        db, patstat, cpc_filter_2, start_date,
        epc_countries, alpha2_to_alpha3
    )
)

# ----------------------
# 5) Create Maps
# ----------------------
fig1 = px.choropleth(
    df1,
    locations="country_alpha3",
    locationmode="ISO-3",
    color="ratio_of_filings",
    hover_name="country",
    hover_data={
        "filings_count": True,
        "ratio_of_filings": True,
        "country_alpha3": False
    },
    title=f"{cpc_filter_1} - {cpc_exp_1}"
)
fig2 = px.choropleth(
    df2,
    locations="country_alpha3",
    locationmode="ISO-3",
    color="ratio_of_filings",
    hover_name="country",
    hover_data={
        "filings_count": True,
        "ratio_of_filings": True,
        "country_alpha3": False
```

```
    },
    title=f"{cpc_filter_2} - {cpc_exp_2}"
)

# ----------------------
# 6) Combine Subplots
# ----------------------
fig = make_subplots(
    rows=1,
    cols=2,
    subplot_titles=[
        f"{cpc_filter_1} {cpc_exp_1}",
        f"{cpc_filter_2} {cpc_exp_2}"
    ],
    specs=[[{"type": "choropleth"}, {"type": "choropleth"}]]
)
for trace in fig1.data:
    fig.add_trace(trace, 1, 1)
for trace in fig2.data:
    fig.add_trace(trace, 1, 2)

fig.update_layout(
    title={
        "text": (
            f"For the two CPCs, EP patents by applicant's country "
            f"as ratios to total applications in those CPCs "
            f"since {start_date}"
        ),
        "y": 1,
        "x": 0.5,
        "xanchor": "center",
        "yanchor": "top",
        "font": {"size": 16}
    },
    width=1000,
    height=600,
    margin=dict(r=0, t=50, l=0, b=0),
    coloraxis=dict(
        colorscale=[
            [0, "rgba(255, 255, 204, 1)"],
            [0.5, "rgba(255, 165, 0, 1)"],
            [1, "rgba(255, 69, 0, 1)"]
        ],
        colorbar_title="Ratio of Filings"
    )
)
for t in fig.data:
```

```
    t.update(coloraxis="coloraxis")

fig.update_geos(
    scope=None,
    projection_type="mercator",
    center=dict(lat=50, lon=20),
    projection_scale=6,
    showframe=False,
    showcoastlines=True,
    showcountries=True
)
fig.show()
```

0.2 Interactive CPC Treemap and Publication Viewer

This script creates an interactive treemap using CPC filing data. Clicking on any treemap leaf retrieves and displays related publication details dynamically.

[ ]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import ipywidgets as widgets
from IPython.display import display
from sqlalchemy import func
from epo.tipdata.patstat import PatstatClient
from epo.tipdata.patstat.database.models import (
    TLS201_APPLN, TLS211_PAT_PUBLN, TLS224_APPLN_CPC
)

# -------------------------
# 1) Adjust parameters here
# -------------------------
CPC_FILTER = "G06F  16%"
START_DATE = "2020-01-01"
END_DATE   = "2021-01-01"

# -------------------------
# 2) Database Connection
# -------------------------
patstat = PatstatClient(env="PROD")
db = patstat.orm()

def get_cpc_filing_data(
    db, cpc_filter, start_date, end_date
):
    """Counts EP filings per CPC class symbol in a date range."""
    query = (
```

```
        db.query(
            TLS224_APPLN_CPC.cpc_class_symbol.label("cpc.symbol"),
            func.count(TLS201_APPLN.appln_id).label("count")
        )
        .select_from(TLS201_APPLN)
        .join(
            TLS224_APPLN_CPC,
            TLS224_APPLN_CPC.appln_id == TLS201_APPLN.appln_id
        )
        .filter(
            TLS201_APPLN.appln_auth == "EP",
            TLS201_APPLN.appln_filing_date >= start_date,
            TLS201_APPLN.appln_filing_date <= end_date,
            TLS224_APPLN_CPC.cpc_class_symbol.like(cpc_filter)
        )
        .group_by(TLS224_APPLN_CPC.cpc_class_symbol)
    )
    return patstat.df(query)

def get_publications_by_cpc(
    db, cpc_symbol, start_date, end_date
):
    """Returns publication numbers/dates for a given CPC class symbol."""
    query = (
        db.query(
            TLS211_PAT_PUBLN.publn_nr.label("Publication Number"),
            TLS211_PAT_PUBLN.publn_date.label("Publication Date")
        )
        .select_from(TLS211_PAT_PUBLN)
        .join(
            TLS201_APPLN,
            TLS201_APPLN.appln_id == TLS211_PAT_PUBLN.appln_id
        )
        .join(
            TLS224_APPLN_CPC,
            TLS224_APPLN_CPC.appln_id == TLS201_APPLN.appln_id
        )
        .filter(
            TLS201_APPLN.appln_auth == "EP",
            TLS201_APPLN.appln_filing_date >= start_date,
            TLS201_APPLN.appln_filing_date <= end_date,
            TLS224_APPLN_CPC.cpc_class_symbol == cpc_symbol
        )
    )
    return patstat.df(query)

# -------------------------
```

```python
# 3) Retrieve & Display Data
# -------------------------
df = get_cpc_filing_data(
    db, CPC_FILTER, START_DATE, END_DATE
)
print("Sum of counts:", df["count"].sum())

def split_cpc(row):
    """Breaks the CPC symbol into parts for treemap display."""
    parts = row["cpc.symbol"].split("/")
    if len(parts) == 2:
        main, sub = parts[0], parts[1]
        return pd.Series({
            "first": main[0] if len(main) else "",
            "second": (
                main[1:3] if len(main) > 2 else ""
            ),
            "third": (
                main[3] if len(main) > 3 else ""
            ),
            "fourth": (
                main[4:] if len(main) > 4 else ""
            ),
            "fifth": sub
        })
    return pd.Series({
        "first": "", "second": "", "third": "",
        "fourth": "", "fifth": ""
    })

df[["first","second","third","fourth","fifth"]] = df.apply(
    split_cpc, axis=1
)

fig = px.treemap(
    df,
    path=[
        px.Constant("CPC Classification"),
        "first","second","third","fourth","fifth"
    ],
    values="count",
    hover_name="cpc.symbol",
    custom_data=["cpc.symbol"],
    maxdepth=7,
    color_discrete_sequence=px.colors.qualitative.T10,
    width=1000,
    height=500
```

```
)
fig.update_traces(root_color="lightblue")
fig.update_layout(
    margin=dict(t=25, l=25, r=25, b=25)
)

# Make Treemap Interactive
fig_widget = go.FigureWidget(fig)
display(fig_widget)
output_table = widgets.Output()
display(output_table)

def on_treemap_click(trace, points, state):
    if points.point_inds:
        idx = points.point_inds[0]
        clicked_cpc = trace.customdata[idx][0]
        pubs_df = (
            get_publications_by_cpc(
                db, clicked_cpc, START_DATE, END_DATE
            ).drop_duplicates(["Publication Number"])
        )
        output_table.clear_output()
        with output_table:
            if pubs_df.empty:
                print("No publications found for CPC:", clicked_cpc)
            else:
                display(pubs_df)

fig_widget.data[0].on_click(on_treemap_click)
```