

EPO - Patent Knowledge Forum 2024

04.12.2024

Mario Cañadas Castro

Dt. Patents and Technological Information - OEPM - ES

Email: mario.canadas@oepm.es

Use case at OEPM: Climate Change Mitigation

In this example, a simplified version of a previous study on [climate change mitigation technologies](#) (CCMT) using "PATSTAT online" has been replicated with TIP.

It covers:

- **Patent Trends:** evolution over the past ten years of EP applications.
- **Gender Gap Analysis:** A gender gap study is included as a new aspect, analyzing inventor names with [WIPO gender dictionary](#).
- **Regional Mapping:** A map of Spanish provinces with the highest patent activity is presented, enabled by the TIP's capabilities.

EP patents with origin in Spain about Climate Change Mitigation (CPC Y02)

Collecting EP applications

We define ES origin of an invention (patent applications) if the first applicant has the residence in Spain.

First we import PATSTAT and Pandas libraries

```
from epo.tipdata.patstat import PatstatClient
import pandas as pd
patstat = PatstatClient(env='PROD')
```

In [1]:

In PATSTAT we import the needed Tables

In [2]:

```
from epo.tipdata.patstat.database.models import TLS201_APPLN,  
TLS224_APPLN_CPC, TLS207_PERS_APPLN, TLS206_PERSON
```

Timeframe: published applications between 2014 and 2023 (first publications: "A" kind code)

Two dataframes (Pandas) are obtained as output data:

- **df_epo**: all EP applications
- **df_epo_clim**: EP applications related to climate change mitigation (Y02)

In [3]:

```
# Number of EPO applications from ES origin  
q = patstat.sql_query("""  
    SELECT A.earliest_publn_year AS publ_year, COUNT(DISTINCT A.appln_id)  
AS numb_appl  
    FROM tls201_appln A  
    JOIN tls224_appln_cpc C ON A.appln_id = C.appln_id  
    JOIN tls207_pers_appln Q ON A.appln_id = Q.appln_id  
    JOIN tls206_person P ON Q.person_id = P.person_id  
    WHERE A.earliest_publn_year BETWEEN 2014 AND 2023  
    AND A.appln_auth = 'EP'  
    AND P.person_ctry_code = 'ES'  
    AND A.appln_kind = 'A '  
    AND Q.applt_seq_nr = 1  
    --AND C.cpc_class_symbol LIKE 'Y02%'  
    GROUP BY A.earliest_publn_year  
    ORDER BY A.earliest_publn_year  
  
""", use_legacy_sql = False)
```

In [4]:

```
df_epo = pd.DataFrame(q)  
#df_epo
```

In [5]:

```
# Number of EPO applications from ES origin where CPC = Y02*  
q2 = patstat.sql_query("""  
    SELECT A.earliest_publn_year AS publ_year, COUNT(DISTINCT A.appln_id)  
AS numb_appl  
    FROM tls201_appln A  
    JOIN tls224_appln_cpc C ON A.appln_id = C.appln_id  
    JOIN tls207_pers_appln Q ON A.appln_id = Q.appln_id  
    JOIN tls206_person P ON Q.person_id = P.person_id  
    WHERE A.earliest_publn_year BETWEEN 2014 AND 2023  
    AND A.appln_auth = 'EP'  
    AND P.person_ctry_code = 'ES'  
    AND A.appln_kind = 'A '  
    AND Q.applt_seq_nr = 1
```

```

AND C.cpc_class_symbol LIKE 'Y02%'
GROUP BY A.earliest_publn_year
ORDER BY A.earliest_publn_year

""" , use_legacy_sql = False)

df_epo_clim = pd.DataFrame(q2)
df_epo_clim

```

In [6]:

Out[6]:

	publ_year	numb_appl
0	2014	185
1	2015	199
2	2016	121
3	2017	188
4	2018	155
5	2019	170
6	2020	212
7	2021	257
8	2022	239
9	2023	294

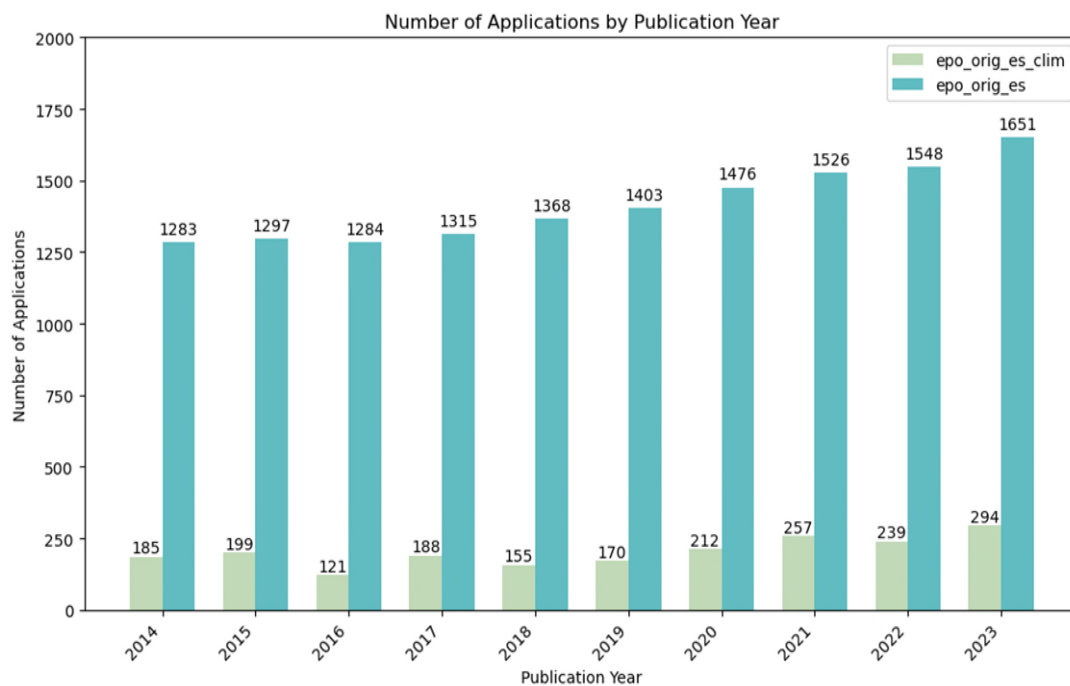
Plotting evolution in the last 10 years

In [7]:

```

import matplotlib.pyplot as plt
...

```



Gender gap study

Adding a gender gap study to the previous case:

Now we want to study the number of women inventors in those patent applications (ES origin and also related to climate change mitigation -CPC Y02-). First we will collect inventor names in PATSTAT, then we will cross that data with an ad-hoc dictionary to assess the gender of each one according to his/her name.

Collecting names of inventors (new PATSTAT query)

Adding inventor names and countries: P.person_etry_code AS V_PAIS, P.person_name AS V_NOMBRE_COMPLETO

In [8]:

```
# Name of inventors of EPO applications from ES origin
qgp = patstat.sql_query("""
SELECT DISTINCT A.earliest_publn_year AS YEAR, P.person_etry_code AS V_PAIS,
P.person_name AS V_NOMBRE_COMPLETO
  FROM tls201_appln A
 JOIN tls207_pers_appln Q ON A.appln_id = Q.appln_id
 JOIN tls206_person P ON Q.person_id = P.person_id
 WHERE Q.invt_seq_nr >= 1
 AND A.appln_id In (   SELECT DISTINCT A.appln_id
                      FROM tls201_appln A
                      JOIN tls224_appln_cpc C ON A.appln_id = C.appln_id
                      JOIN tls207_pers_appln Q ON A.appln_id = Q.appln_id
                      JOIN tls206_person P ON Q.person_id = P.person_id
                      WHERE A.earliest_publn_year BETWEEN 2014 AND 2023
                      AND A.appln_auth = 'EP'
                      AND P.person_etry_code = 'ES'
                      AND A.appln_kind = 'A '
                      AND Q.applt_seq_nr = 1
                      AND C.cpc_class_symbol LIKE 'Y02%')
 ORDER BY  A.earliest_publn_year, P.person_name

""", use_legacy_sql = False)
```

In [9]:

```
df_names_epo_clim = pd.DataFrame()
# Convert to data frame
df_names_epo_clim = pd.DataFrame(qgp)
df_names_epo_clim
```

Out[9]:

	YEAR	V_PAIS	V_NOMBRE_COMPLETO
0	2014	ES	ABANADES GARCÍA, Juan Carlos
1	2014	ES	ABELLÁN SÁEZ, Gonzalo
2	2014	ES	AINZ IBARRONDO, Félix

	YEAR	V_PAIS	V_NOMBRE_COMPLETO
3	2014	ES	ALLONA ALBERICH, Isabel
4	2014	ES	ALONSO BEDATE, Carlos
...
5464	2023	DE	Zeller, Lenz Simon
5465	2023	ES	ÁLVAREZ CARREÑO, Carlos
5466	2023	ES	ÁLVAREZ DE DIEGO, Javier
5467	2023	ES	ÁLVAREZ-HERMS, Jesús
5468	2023	ES	ÁVILA GARCÍA, Pedro

5469 rows × 3 columns

Importing WIPO "Gender name dictionary"

Separated "csv" file obtained from WIPO data, with contributions and refinement by OEPM.

In [10]:

```
# Load the gender dictionary if not already loaded
try:
    df_gender_dic
    print("El Diccionario de género ya estaba cargado.")
except NameError:
    print("Leyendo diccionario...")
    # Load the dictionary with the correct encoding
    df_gender_dic = pd.read_csv("~/generoPAT/DIM_EST_GENERO_202410.csv",
delimeter='}', encoding='latin1')
    print("Diccionario de género cargado.")
Leyendo diccionario...
Diccionario de género cargado.
```

Data wrangling to work with inventor names

In [11]:

```
# Separate V_NOMBRE_COMPLETO in 2 fields by comma
df_names_epo_clim[['V_APELLIDOS', 'V_NOMBRE']] =
df_names_epo_clim['V_NOMBRE_COMPLETO'].str.split(',', expand=True, n=1)
#df_names_epo_clim

# Capitalize V_NOMBRE and remove spaces in front and behind
df_names_epo_clim['V_NOMBRE'] = df_names_epo_clim['V_NOMBRE'].str.upper()
df_names_epo_clim['V_NOMBRE'] = df_names_epo_clim['V_NOMBRE'].str.strip()
df_names_epo_clim
```

Out[11]:

	YEAR	V_PAIS	V_NOMBRE_COMPLETO	V_APELLIDOS	V_NOMBRE
0	2014	ES	ABANADES GARCÍA, Juan Carlos	ABANADES GARCÍA	JUAN CARLOS
1	2014	ES	ABELLÁN SÁEZ, Gonzalo	ABELLÁN SÁEZ	GONZALO
2	2014	ES	AINZ IBARRONDO, Félix	AINZ IBARRONDO	FÉLIX
3	2014	ES	ALLONA ALBERICH, Isabel	ALLONA ALBERICH	ISABEL
4	2014	ES	ALONSO BEDATE, Carlos	ALONSO BEDATE	CARLOS
...
5464	2023	DE	Zeller, Lenz Simon	Zeller	LENZ SIMON
5465	2023	ES	ÁLVAREZ CARREÑO, Carlos	ÁLVAREZ CARREÑO	CARLOS
5466	2023	ES	ÁLVAREZ DE DIEGO, Javier	ÁLVAREZ DE DIEGO	JAVIER
5467	2023	ES	ÁLVAREZ-HERMS, Jesús	ÁLVAREZ-HERMS	JESÚS
5468	2023	ES	ÁVILA GARCÍA, Pedro	ÁVILA GARCÍA	PEDRO

5469 rows \times 5 columns

Determining the inventor's gender

Using "merge" between DataFrames (Pandas library)

In [12]:

```
df_names_epo_clim = pd.merge(df_names_epo_clim, df_gender_dic,
on=["V_NOMBRE", "V_PAIS"], how="left")
#df_names_epo_clim
```

Cleaning data

In [13]:

```
# Replace M by Male, F by Female and NaN by Unknown
# df_names_epo_clim=df
df_names_epo_clim['V_GENERO'] =
df_names_epo_clim['V_GENERO'].fillna('Unknown')
df_names_epo_clim['V_GENERO'] = df_names_epo_clim['V_GENERO'].replace({'F':
'Female', 'M': 'Male'})
df_names_epo_clim
```

Out[13]:

5472 rows \times 6 columns

Second round:

Trying to assess gender for those "Unknown names". Now we look only for the first word in the case of compound names.

In [14]:

```
#New DataFrame to work with:
df_names_epo_clim2 = pd.DataFrame()
df_names_epo_clim2=df_names_epo_clim

# Rename column V_NOMBRE_ORIG with original names (to leave a column
V_NOMBRE for merging with gender dictionary)
df_names_epo_clim2.rename(columns={"V_NOMBRE": "V_NOMBRE_ORIG"},
inplace=True)

# Add the new column `V_NOMBRE` with first word of compound names ONLY if
already "Unkonwn" gender
df_names_epo_clim2['V_NOMBRE'] = df_names_epo_clim2.apply(
    lambda row: row['V_NOMBRE_ORIG'].split()[0] if (
        row['V_GENERO'] == "Unknown" and
        row['V_NOMBRE_ORIG'] and
        isinstance(row['V_NOMBRE_ORIG'], str) and
        len(row['V_NOMBRE_ORIG'].split()) > 1
    ) else "",
    axis=1
)

df_names_epo_clim2
```

Out[14]:

5472 rows × 7 columns

.

Check remaining names (V_NOMBRE) again using "merge" with the Gender dictionary

In [15]:

```
df_names_epo_clim2 = pd.merge(df_names_epo_clim2, df_gender_dic,
on=["V_NOMBRE", "V_PAIS"], how="left")
#df_names_epo_clim2
#NOTE: this merge create two columns: V_GENERO_X and V_GENERO_y, now we
fuse them
```

In [16]:

```
# TO get only one column GENERO:
...
```

Out[16]:

	YEAR	V_PAIS	V_NOMBRE_COMPLETO	V_APELLIDOS	V_NOMBRE_ORIG	V_NOMBRE	V_GENERO
0	2014	ES	ABANADES GARCÍA, Juan Carlos	ABANADES GARCÍA	JUAN CARLOS		Male
1	2014	ES	ABELLÁN SÁEZ, Gonzalo	ABELLÁN SÁEZ	GONZALO		Male
2	2014	ES	AINZ IBARRONDO, Félix	AINZ IBARRONDO	FÉLIX		Male
3	2014	ES	ALLONA ALBERICH, Isabel	ALLONA ALBERICH	ISABEL		Female
4	2014	ES	ALONSO BEDATE, Carlos	ALONSO BEDATE	CARLOS		Male
...

	YEAR	V_PAIS	V_NOMBRE_COMPLETO	V_APELLIDOS	V_NOMBRE_ORIG	V_NOMBRE	V_GENERO
5468	2023	DE	Zeller, Lenz Simon	Zeller	LENZ SIMON	LENZ	Male
5469	2023	ES	ÁLVAREZ CARREÑO, Carlos	ÁLVAREZ CARREÑO	CARLOS		Male
5470	2023	ES	ÁLVAREZ DE DIEGO, Javier	ÁLVAREZ DE DIEGO	JAVIER		Male
5471	2023	ES	ÁLVAREZ-HERMS, Jesús	ÁLVAREZ-HERMS	JESÚS		Male
5472	2023	ES	ÁVILA GARCÍA, Pedro	ÁVILA GARCÍA	PEDRO		Male

5473 rows × 7 columns

Plot gender gap

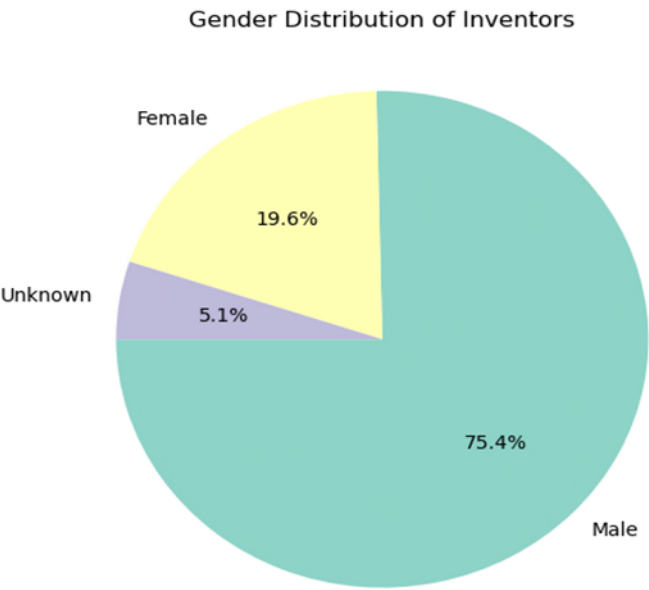
Static graph showing the gender gap in EP inventions with ES origin, over the last 10 years and in the field of CCM.

It counts occurences (in the DataFrame) of each case: Male, Female, Unknown name

In [17]:

```
import matplotlib.pyplot as plt

...
```



Out[17]:

```
V_GENERO
Male      4124
Female    1072
Unknown    277
Name: count, dtype: int64
```

Geographic Distribution of Applicants

Installing "pygwalker" library

In [18]:

```
%pip install pygwalker
import pygwalker as pyg
```

New PATSTAT query including the origin of the applicant: Spanish provinces ([NUTS 3](#))

In [23]:

```
# Number of EPO applications from ES origin
q_ori = patstat.sql_query("""
    SELECT P.nuts AS NUTS, N.nuts_label AS Region, COUNT(DISTINCT A.appln_id) AS
Num_applications
    FROM tls201_appln A
    JOIN tls224_appln_cpc C ON A.appln_id = C.appln_id
    JOIN tls207_pers_appln Q ON A.appln_id = Q.appln_id
    JOIN tls206_person P ON Q.person_id = P.person_id
    JOIN tls904_nuts N ON P.nuts = N.nuts
    WHERE A.earliest_publn_year BETWEEN 2014 AND 2023
    AND A.appln_auth = 'EP'
    AND P.person_ctype_code = 'ES'
    AND A.appln_kind = 'A '
    AND Q.appl_t_seq_nr = 1
    AND C.cpc_class_symbol LIKE 'Y02%'
    AND P.NUTS_LEVEL in (3,4)
    GROUP BY P.nuts, N.nuts_label
    ORDER BY P.nuts

""", use_legacy_sql = False)

#del df_INori
df_INori = pd.DataFrame(q_ori)
df_INori = df_INori.sort_values(by='Num_applications',
ascending=False).reset_index(drop=True)
df_INori.head()
```

Out[23]:

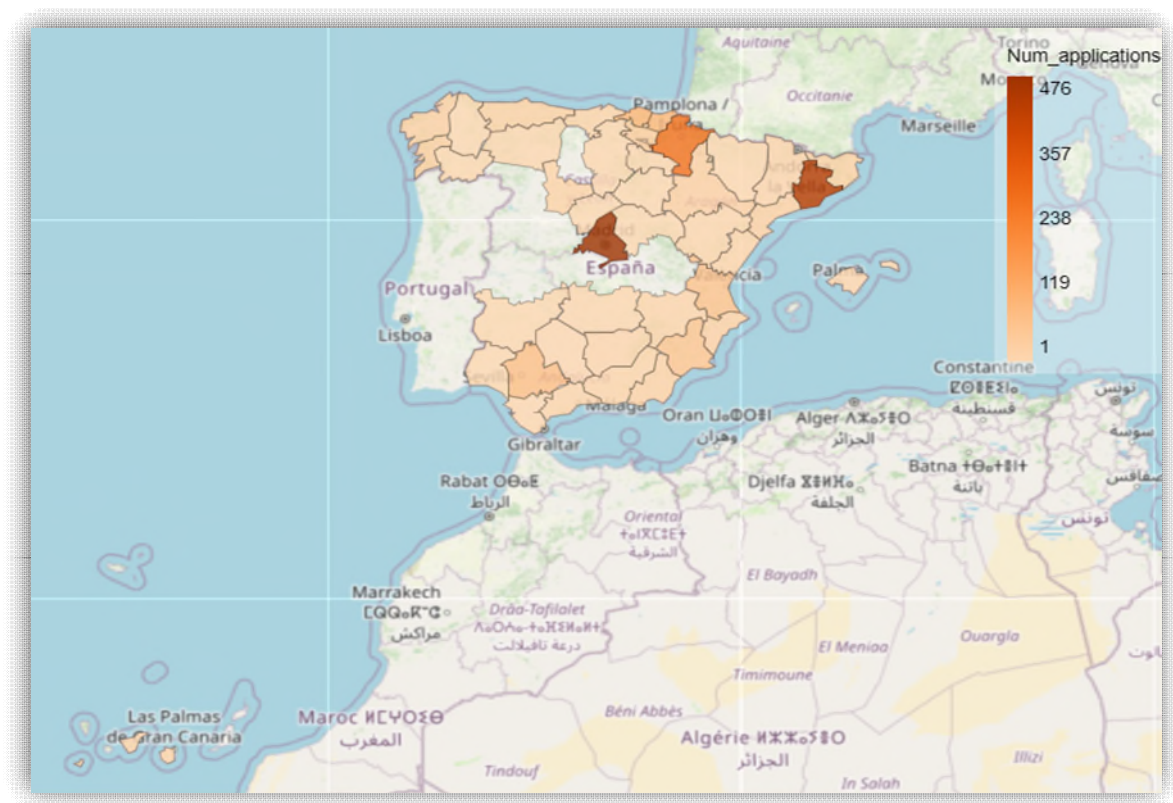
	NUTS	Region	Num_applications
0	ES300	Madrid	476
1	ES511	Barcelona	438
2	ES220	Navarra	272
3	ES213	Bizkaia	100
4	ES212	Gipuzkoa	93

Map of patent applications (CCMT) by Spanish provinces

In [20]:

```
import json
# Step 1: Load the saved JSON configuration
with open('pygwalker_spec_ESPprov.json', 'r') as file: # Replace
'saved_graph.json' with your JSON file path
    saved_spec = json.load(file)

# Step 2: Use the JSON configuration to render the graph
pyg.walk(df_INori, spec=saved_spec)
```



CONCLUSIONS

- **TIP** shows a **very good performance**
- **Python libraries** provide almost unlimited options on output formats, figures, analysis, etc.
- It opens **new possibilities for us** on patent data analysis
- At present: **patent coverage limitations** (looking forward to new versions and improvements)

Thanks!

Special thanks to Carlos Albert and Luis Priegue (OEPM) for the preparation of the code

And to Carlos Aitor Pérez de Unzueta and Borja Rojano (EPO) for all the support with TIP.