Visualising STIP Compass data in Python

Daniela Valenzuela and Rei Tomoe 2025-08

This data story provides an overview of how to visualise STIP Compass data using Python. It offers an accessible guide on how to access the datasets available via the STIP Data Lab, including their structure, the important fields, and the types of charts and visualisations that can be developed.

A similar Data Story: "Working with STIP Compass data in R: An Introduction" focused on step-by-step guide to accessing STIP using R. That complementary resource provides a guide to accessing the data. In this Data Story however, we aim to build on this and go one step further — focusing on how Python can be used to generate visualisations that help uncover relationships, detect patterns, and track trends.

Accessing the dataset

The following scripts are available to get from this GitHub Link.

To begin our analysis, we first need to load the STIP Compass data into a Python environment. This section provides a minimal set of code to get you started quickly. We will use the pandas library, an essential tool for data manipulation in Python.

For a more detailed explanation of the data model and variables, please refer to the original R data story.

First, let's load the data directly from the STIP Compass website and perform some basic cleaning.

```ptyhon
import pandas as pd
import numpy as np
url = "https://stip.oecd.org/assets/downloads/STIP_Survey.csv"
stip_survey = pd.read_csv(url,sep="|")


# Remove the description row to keep only observational data
stip_survey = stip_survey.iloc[1:].reset_index(drop=True)

# Convert theme and target group columns to a numeric format
th_tg_cols = [col for col in stip_survey.columns if col.startswith('TH') or
col.startswith('TG')]
stip_survey[th_tg_cols] = stip_survey[th_tg_cols].apply(pd.to_numeric,
errors='coerce').fillna(0)
```

```
# The dataset has one row per policy instrument. To analyze policy
initiatives,
# we create a separate DataFrame with unique initiatives only.
stip_survey_unique =
stip_survey.drop_duplicates(subset=['InitiativeID']).copy()

print("Data loaded and prepared.")
print(f"Total policy instruments (rows): {len(stip_survey)}")
print(f"Total unique policy initiatives: {len(stip_survey_unique)}")
stip_survey_unique.head()
```

```

# Understanding the dataset

The 'live dataset' comprises more than 10,000 rows and more than 800 columns. The columns fall into four groups; further context can be found in the publicly available survey data model. For each survey wave, the data model is slightly updated.

- **Policy initatives**: The leftmost columns provide information on policy initiatives, the STIP Survey's main unit of reporting, and correspond to the 'policy initiative fiche' described in the data model. They contain qualitative as well as quantitative information.
- **Themes**: A set of columns with names starting with "TH". These are dummy columns indicating membership of policy initiatives with different themes. The current STIP Survey has 58 themes, which correspond to the different questions addressed to survey respondents. Policy initiatives are linked to one or more themes.
- **Target groups**: A set of columns with names starting with "TG". These are dummy columns indicating the direct beneficiaries (or 'Target Groups') of policy initiatives. The current STIP Survey specifies 33 types of direct beneficiaries. Policy initiatives have one or more direct beneficiaries.
- **Policy instruments**: The remaining columns describe the policy instruments associated with policy initiatives. The data model comprises 28 instrument types, denoted in the 'InstrumentTypeLabel' column. All columns starting with letter F followed by a number describe instrument facets as specified in the data model. There are more than 600 columns of this type, and analysts mainly interested in policy initiatives might disregard their content.

- **Coding schemes**: The data model specifies that policy themes, direct beneficiaries and policy instruments are organised using two-level coding schemes, with smaller
```

sets of broad categories at the first level, and more specific instances under these categories at the second level. The dataset only provides information on the second, more granular level. The first row of the dataset below the header contains additional detail on the content of columns, rather than observational data.

```python
#To facilitate working with the dataset, we generate a separate 'Codebook' dataframe listing the column names and the detail given in the first row, for variables on themes and direct beneficiaries

# 1. Get column names (Code) and the first row (Meaning) from the DataFrame
columns = stip_survey.columns
meanings = stip_survey.iloc[0].values

# 2. Create a DataFrame pairing each column name with its description
codebook = pd.DataFrame({
    "Code": columns,
    "Meaning": meanings
})

# 3. Filter only columns whose names start with "TH" or "TG" (policy themes and direct beneficiaries)
codebook =
codebook[codebook["Code"].str.match(r"^TH|^TG")].reset_index(drop=True)

# 4. Display the first 10 rows of the codebook
codebook.head(10)
```

Code	Meaning
TH101	Net zero transitions policy debates
TH102	Government capabilities for net zero transitions
TH103	Net zero transitions in transport and mobility
TH104	Net zero transitions in food and agriculture
TH105	Cross-sectoral policies for net zero
TH106	Digital transformation of research-performing ...
TH107	Open and enhanced access to publications
TH108	Open and enhanced access to research data
TH109	Research security
TH13	STI plan or strategy

Visualisations: Example Analysis — Comparing Policy Initiatives Across Countries

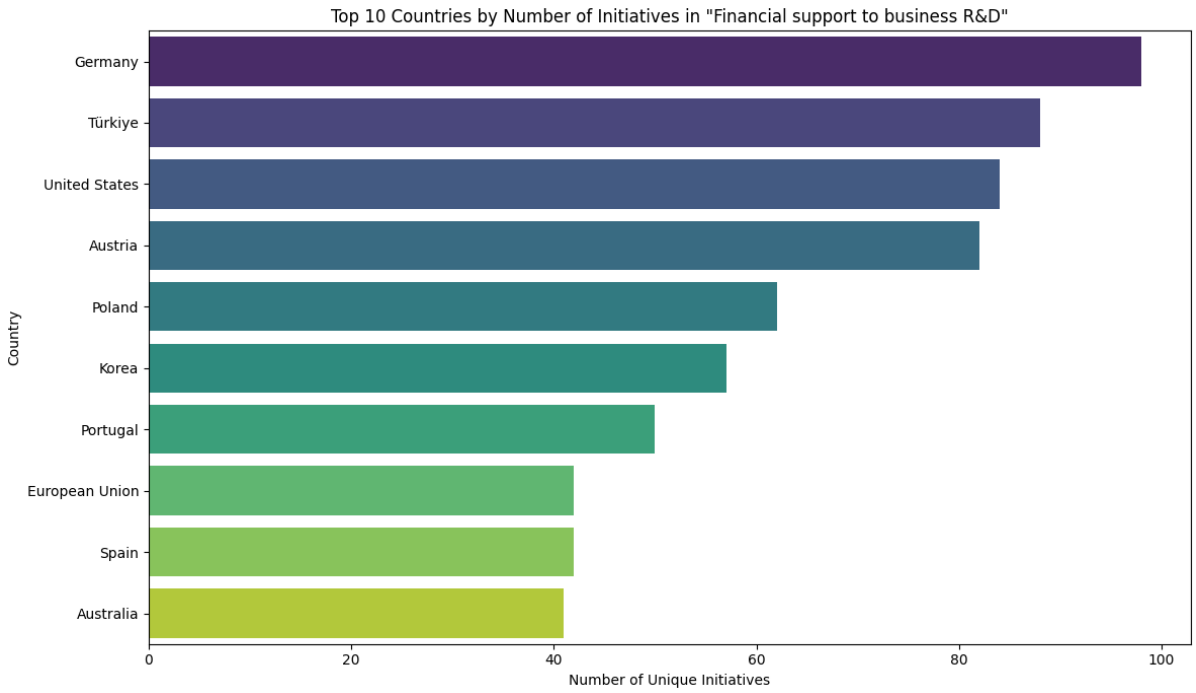
One of the powerful applications of the STIP Compass dataset is conducting comparative analysis to understand how policy priorities differ across countries. The structured data on themes and target groups makes it straightforward to investigate national policy landscapes.

As an example, let's identify which countries have reported the highest number of policy initiatives related to the theme "Financial support to business R&D and innovation" (Theme ID: TH31). This type of analysis can reveal national priorities and highlight countries with a strong focus on funding private sector innovation.

To do this, we will use the stip_survey_unique DataFrame to ensure we are counting each policy initiative only once. We will filter for the relevant theme, group the data by country, and then visualize the results for the top 10 countries.

```python
Ensure theme columns are numeric before filtering

th_cols = [col for col in financing_innovation.columns if col.startswith('TH')]
financing_innovation[th_cols] = financing_innovation[th_cols].apply(pd.to_numeric)

# 1. Filter for unique initiatives related to the theme 'Financial support to business R&D and innovation' (TH31).

th31_initiatives = financing_innovation[financing_innovation['TH31'] == 1]

# 2. Count the number of initiatives per country and get the top 10.

top_countries_th31 = th31_initiatives['CountryLabel'].value_counts().head(10)

# 3. Print the resulting counts.

print("Top 10 countries by number of initiatives for 'Financial support to business R&D':")
print(top_countries_th31)

# 4. Visualize the results using a bar chart for better comparison.
```

```
plt.figure(figsize=(12, 7))

sns.barplot(x=top_countries_th31.values, y=top_countries_th31.index, palette='viridis')

plt.title('Top 10 Countries by Number of Initiatives in "Financial support to business R&D"')
plt.xlabel('Number of Unique Initiatives')

 plt.ylabel('Country')

plt.tight_layout()

# Adjust plot to ensure everything fits without overlapping

plt.show()
```



This code snippet first filters our unique initiatives dataset for the specific theme TH31(Financial support to business R&D and innovation. It then counts the number of initiatives for each country and selects the top 10. The resulting bar chart provides an immediate, clear comparison of how different countries prioritize direct financial support for business R&D within their policy frameworks, offering a valuable insight for policy analysts and researchers.

Visualisations: Example Analysis Mapping Policy Priorities with a Heatmap

A heatmap is an excellent tool for visualizing the relationships between different policy themes. It can reveal which themes are often addressed together in a single policy initiative, suggesting common policy "packages" or priorities.

Here, we will create a hypothetical analysis for the new 2025 theme, **"Strategic autonomy and promotion of critical technologies."** We will explore how this theme might co-occur with other governance-related themes once the data is available.

```python
import matplotlib.pyplot as plt
import seaborn as sns

# For demonstration, simulate columns if not present
if 'TH18' not in stip_survey_unique.columns:
    stip_survey_unique['TH18'] = np.random.randint(0, 2,
size=len(stip_survey_unique))
if 'TH16' not in stip_survey_unique.columns:
    stip_survey_unique['TH16'] = np.random.randint(0, 2,
size=len(stip_survey_unique))

# 1. Define the governance themes to analyze (use correct codes)
governance_themes = [
    'TH18',  # Strategic autonomy (New)
    'TH16',  # Dynamic capabilities (New)
    'TH13',  # STI plan or strategy
    'TH15'   # Evaluation and impact assessment
]

# 2. English labels for the heatmap
label_mapping = {
    'TH18': 'Strategic Autonomy',
    'TH16': 'Dynamic Capabilities',
    'TH13': 'STI Plan/Strategy',
    'TH15': 'Evaluation & Impact Assessment'
}

# 3. Create a DataFrame containing only these theme columns and set English
labels
```

```
governance_df = stip_survey_unique[governance_themes].copy()
governance_df.columns = [label_mapping[col] for col in governance_df.columns]

# 4. Calculate the co-occurrence matrix
co_occurrence_matrix = governance_df.T.dot(governance_df)

# 5. Draw the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(co_occurrence_matrix, annot=True, fmt='d', cmap='viridis')
plt.title('Co-occurrence Heatmap of Governance Policy Themes')
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```

```

# Visualisations: Example Analysis — Network analysis co-ocurrence of policy instruments

Network analysis is a powerful technique for visualizing complex relationships between different entities. In the context of the STIP Compass data, we can use it to map which countries are using which policy instruments for a specific theme. This approach transforms a flat table of data into an intuitive graph, revealing national policy toolkits and international patterns at a glance.

For our example, we will create a network graph to explore the new 2025 theme, **"Dynamic skills and capabilities for policymaking."** This visualization will show us which policy instruments are being used by different countries to enhance their policymaking processes.

```Python

```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
```

```
# --- HYPOTHETICAL ANALYSIS FOR 2025 DATA ---

# We will use the full `stip_survey` DataFrame because we are interested in
the
# relationship between each initiative and its specific instruments.

# Let's assume the new theme "Dynamic skills and capabilities for
policymaking" has the code 'TH16'.
# To make this example runnable, we'll first simulate this column if it
doesn't exist.
if 'TH16' not in stip_survey.columns:
    # Create a small, random sample of initiatives for this theme to make the
graph readable
    np.random.seed(42) # for reproducibility
    stip_survey['TH16'] = 0
    sample_indices = np.random.choice(stip_survey.index, 50, replace=False)
    stip_survey.loc[sample_indices, 'TH16'] = 1

# 1. Filter for instruments belonging to initiatives with the target theme.
dynamic_skills_instruments = stip_survey[stip_survey['TH16'] == 1]

# 2. Create a DataFrame of the relationships (edges) between countries and
instruments.
#    We drop any rows with missing values in these key columns.
edges = dynamic_skills_instruments[['CountryLabel',
'InstrumentTypeLabel']].dropna().reset_index(drop=True)

# 3. Create a graph object from this list of edges using the networkx library.
G = nx.from_pandas_edgelist(edges, source='CountryLabel',
target='InstrumentTypeLabel')

# 4. Prepare for plotting by defining node properties.
plt.figure(figsize=(16, 16)) # Use a large figure size for clarity
pos = nx.spring_layout(G, k=0.4, iterations=50) # Position nodes using a
force-directed layout

# Differentiate nodes by type (country vs. instrument) for better visual
interpretation.
node_colors = []
node_sizes = []
country_nodes = edges['CountryLabel'].unique()

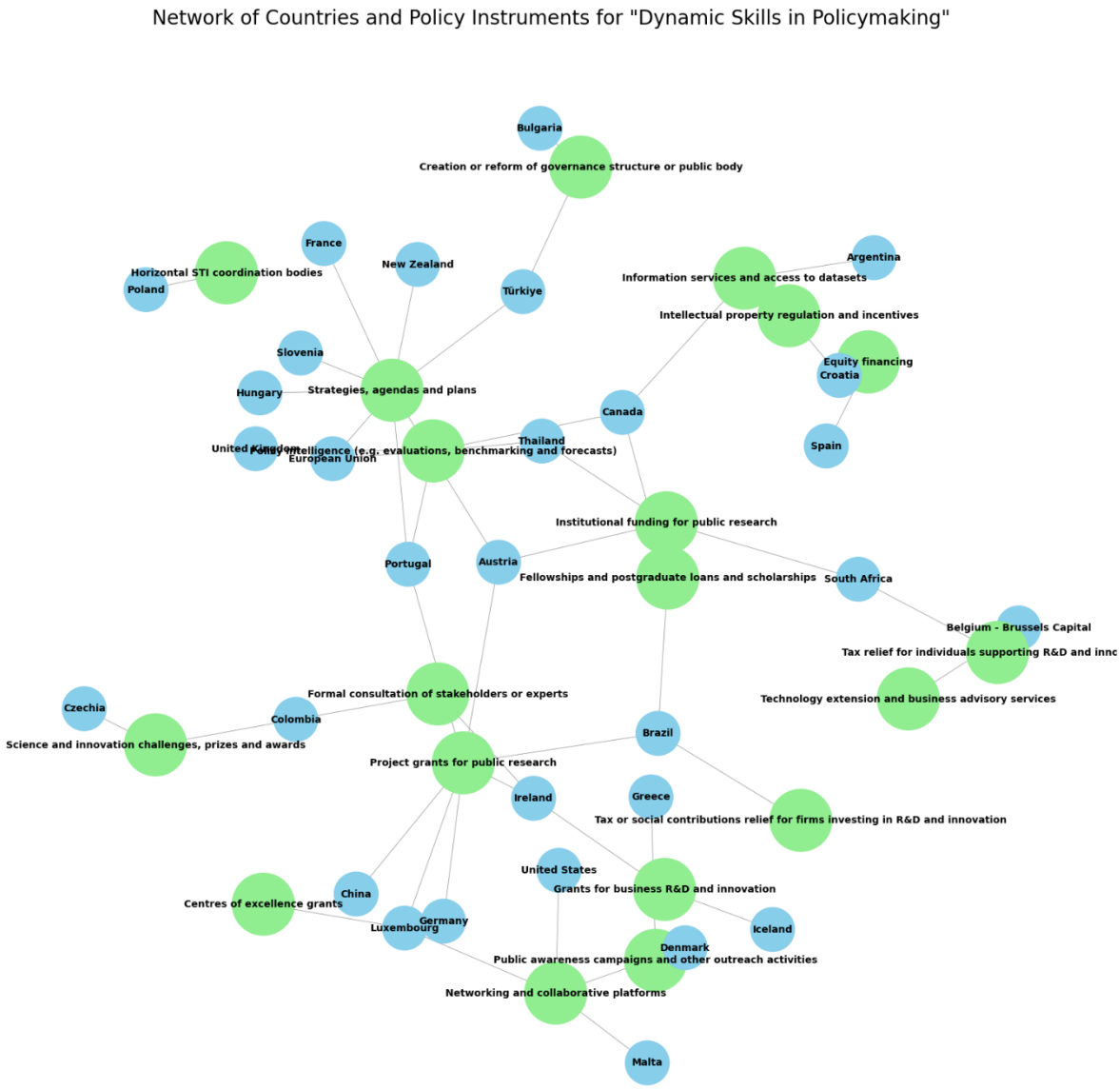
for node in G.nodes():
    if node in country_nodes:
        node_colors.append('skyblue') # Color for countries
        node_sizes.append(2000)
    else:
```

```
        node_colors.append('lightgreen') # Color for instruments
        node_sizes.append(4000) # Larger nodes for instrument labels

# 5. Draw the network graph.
nx.draw(G, pos,
        with_labels=True,
        node_color=node_colors,
        node_size=node_sizes,
        font_size=10,
        font_weight='bold',
        edge_color='gray',
        width=0.5)

plt.title('Network of Countries and Policy Instruments for "Dynamic Skills in Policymaking"', size=20)
plt.show()
```

```
```

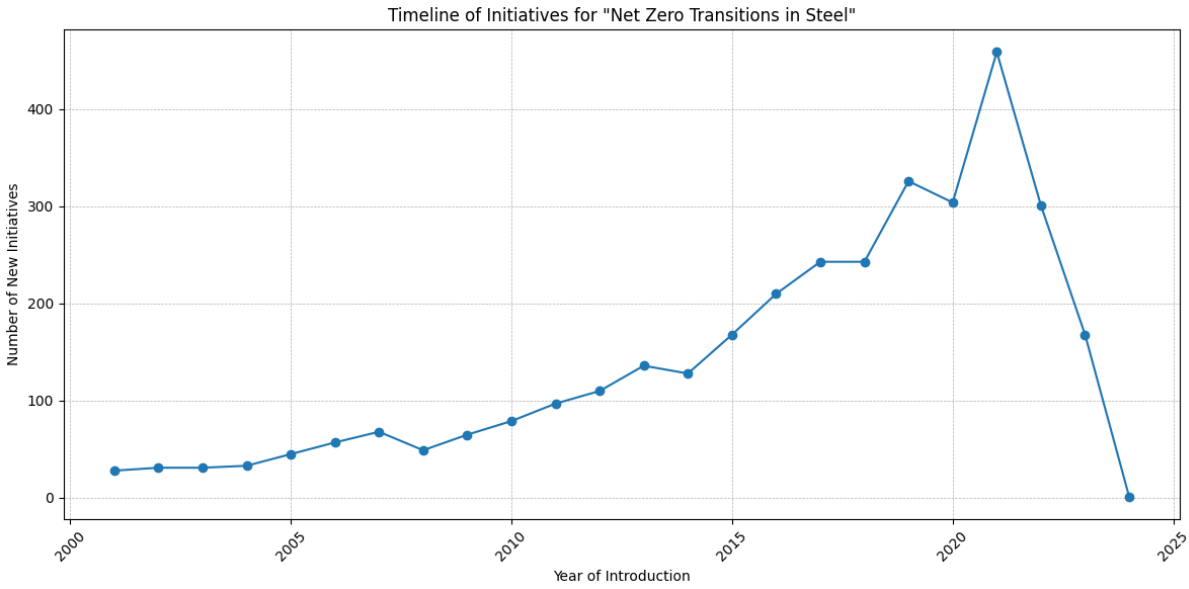


Visualisations: Example Analysis — Timeline visualisation

Timelines are powerful for understanding how policy priorities evolve. By plotting the number of new initiatives over time for a specific theme, we can identify trends, such as increased focus after a major event or technological breakthrough.

Let's create a timeline for the new theme **"Net zero transitions in steel."** This analysis could reveal if efforts to decarbonize the steel industry have intensified in recent years, especially after international climate agreements.

```python
steel_theme_code = 'TH76'

# Simulate the new theme column in our unique initiatives DataFrame
if steel_theme_code not in stip_survey_unique.columns:
    stip_survey_unique[steel_theme_code] = np.random.randint(0, 2,
size=len(stip_survey_unique))

# Filter for initiatives related to the steel theme
steel_initiatives = stip_survey_unique[stip_survey_unique[steel_theme_code] ==
1].copy()

# Convert 'StartDateYear' to a numeric type, coercing errors to NaN
steel_initiatives['StartDateYear'] =
pd.to_numeric(steel_initiatives['StartDateYear'], errors='coerce')

# Drop rows where the start year is missing and filter for a reasonable time
frame
steel_initiatives.dropna(subset=['StartDateYear'], inplace=True)
steel_initiatives = steel_initiatives[steel_initiatives['StartDateYear'] >
2000]

# Count the number of new initiatives per year
timeline_data = steel_initiatives['StartDateYear'].value_counts().sort_index()

# Create the timeline plot
plt.figure(figsize=(12, 6))
plt.plot(timeline_data.index, timeline_data.values, marker='o', linestyle='-')
plt.title('Timeline of Initiatives for "Net Zero Transitions in Steel"')
plt.xlabel('Year of Introduction')
plt.ylabel('Number of New Initiatives')
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Conclusion

This data story has provided an introduction to using STIP Survey data with Python, from downloading the dataset to explaining its structure and contents and to presenting basic steps for its analysis. These should serve as initial steps into analysing the wealth of data available in STIP Compass. Working directly with the dataset can be highly beneficial, especially for users who prefer more control, a custom approach, or need reproducible results. As highlighted, data users should be cautious with regard to the structure of the dataset and be mindful of the differences between policy initiatives and policy instruments when analysing the dataset.

This story is linked to another data story introducing an analysis of text data describing policy initiatives from the STIP Survey dataset: Working with STIP Compass data in R: An Introduction. The STIP Compass team strives to support users of the survey data sets and will develop more useful guidance materials for data users and analysts in the future.