

CENDOJ

CENTRO DE
DOCUMENTACIÓN JUDICIAL

Juzgado de Violencia
de Género
Sala de Espera



Legal Data Analysis Presentation
Juzgado de Violencia Sobre la Mujer
December 11th 2023
Anna Cantos Prats & Lisa Milani

Plan:

- I. Context of « Violencia de Género » in Spain
- II. Data Analysis of Judgments of the « *Juzgado de Violencia sobre la Mujer* »
 - i. Efficiency analysis*
 - ii. Judgement analysis*

I. Context of « Violencia de Género » in Spain

1. Number of deaths by “Violencia de Género”

Code :

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Set the data
data = {
    "Year": [2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019],
    "Deaths": [71, 72, 57, 69, 71, 76, 57, 73, 62, 51, 54, 55, 60, 49, 50, 53, 55]
}
df = pd.DataFrame(data)

# Use a seaborn color palette with red tones
colors = sns.color_palette("Reds_r", n_colors=len(df["Year"]))

# Set a dark theme with a white grid
sns.set(style="darkgrid")

# Create a line plot
plt.figure(figsize=(14, 8))

# Use relplot for faceted line plots
g = sns.relplot(x="Year", y="Deaths", kind="line", data=df, height=6, aspect=2, marker="o", errorbar=None, palette=colors)

# Add regression line with transparency
sns.replot(x="Year", y="Deaths", data=df, scatter=False, ax=g.ax, color="darkred", line_kws={"alpha": 0.7})

# Add mean line
mean_line = df["Deaths"].mean()
g.ax.axhline(mean_line, color="navy", linestyle="--", label="Mean", linewidth=2)

# Set titles and labels
plt.title("Deaths of women due to gender-based violence (2003-2019)", fontsize=20, fontweight="bold", color="darkred")
plt.xlabel("Year", fontsize=14, fontweight="bold", color="darkslategray")
plt.ylabel("Number of deaths", fontsize=14, fontweight="bold", color="darkslategray")

# Add a legend
g.ax.legend(frameon=False, loc="upper left", fontsize=12)

# Annotate the plot with
g.ax.annotate("Reg Line: Linear Regression", xy=(2008, 75), xytext=(2008, 78),
              arrowprops=dict(facecolor='black', shrink=0.05),
              fontsize=12, color='black')

g.ax.annotate("Mean Line: Mean of Deaths", xy=(2015, mean_line), xytext=(2015, mean_line + 2),
              arrowprops=dict(facecolor='black', shrink=0.05),
              fontsize=12, color='black')

# Customize grid
plt.grid(True, linestyle="--", alpha=0.5)

# Customize ticks
plt.xticks(fontsize=12, color="darkslategray")
plt.yticks(fontsize=12, color="darkslategray")

# Remove spines
sns.despine()

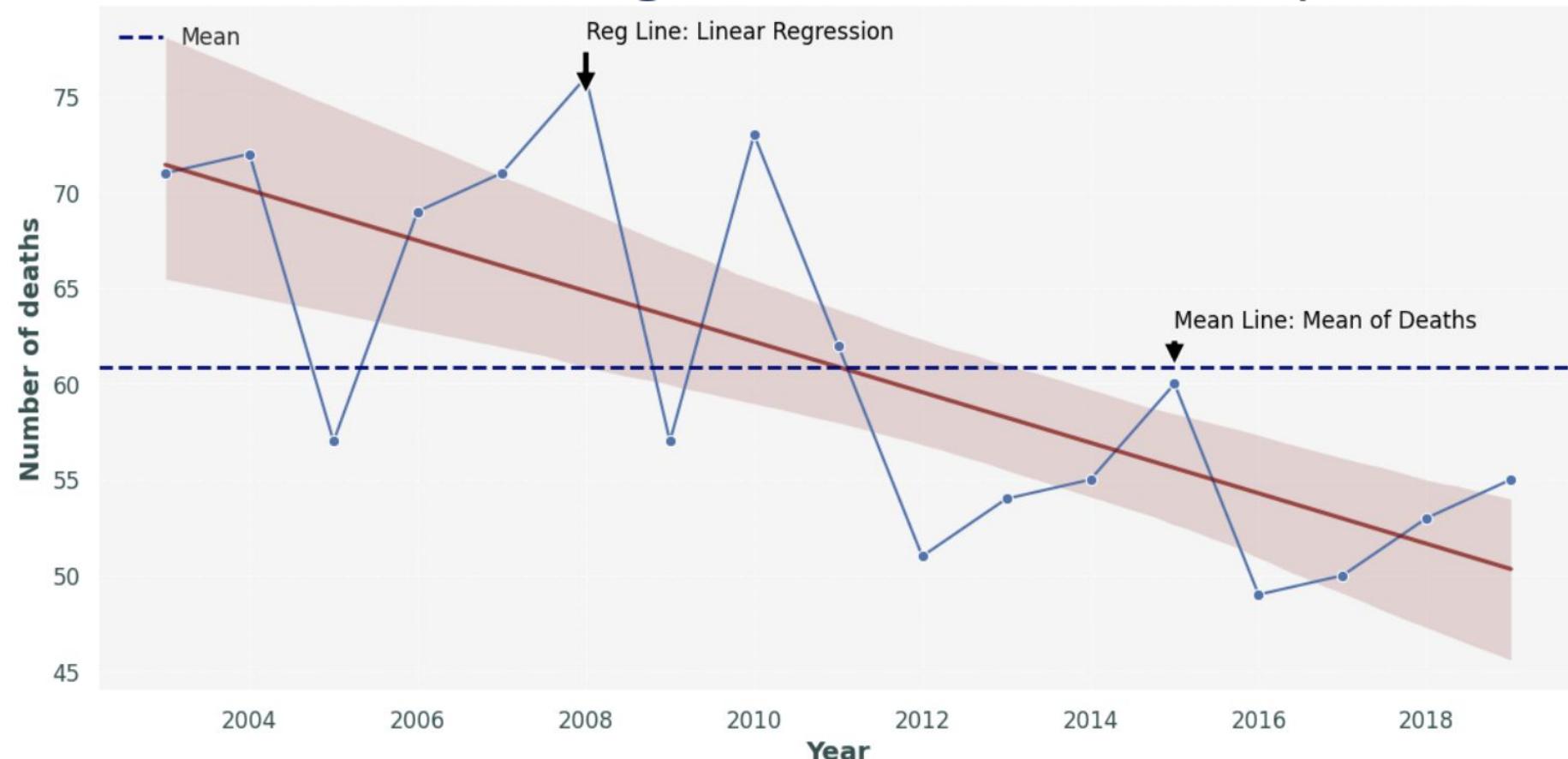
# Set a background color for the plot
plt.gca().set_facecolor('whitesmoke')

# Display the plot with a shadow
plt.show()
```

I. Context of « Violencia de Género » in Spain

1. Number of deaths by “Violencia de Género”

Deaths of women due to gender-based violence (2003-2019)



I. Context of « Violencia de Género » in Spain

2. Victims and Accused

Code :

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

# Data
data = {
    'Year': [2019, 2018, 2017, 2016, 2015, 2014, 2013, 2012, 2011],
    'Victims (Women)': [31911, 31286, 29008, 28281, 27624, 27087, 27122, 29146, 32242],
    'Accused (Men)': [31805, 31250, 28987, 28201, 27562, 26987, 27017, 29048, 32142]
}

df = pd.DataFrame(data)

# Set the figure size
plt.figure(figsize=(12, 8))

# Plot Victims and Accused
plt.plot(df['Year'], df['Victims (Women)'], marker='o', color="#e74c3c", label='Victims (Women)', linewidth=2)
plt.plot(df['Year'], df['Accused (Men)'], marker='s', color="#3498db", label='Accused (Men)', linewidth=2)

# Add titles and labels
plt.title('Victims and Accused of Gender-Based Violence with Protection Orders (2011-2019)', fontsize=18, fontweight='bold', color="#34495e")
plt.xlabel('Year', fontsize=16, fontweight='bold', color="#34495e")
plt.ylabel('Count', fontsize=16, fontweight='bold')

# Add legend
plt.legend(fontsize=12)

# Customize the appearance
plt.grid(True, linestyle='--', alpha=0.5, color="#95a5a6")
plt.xticks(df['Year'], fontsize=12, color="#34495e")
plt.yticks(fontsize=12, color="#34495e")

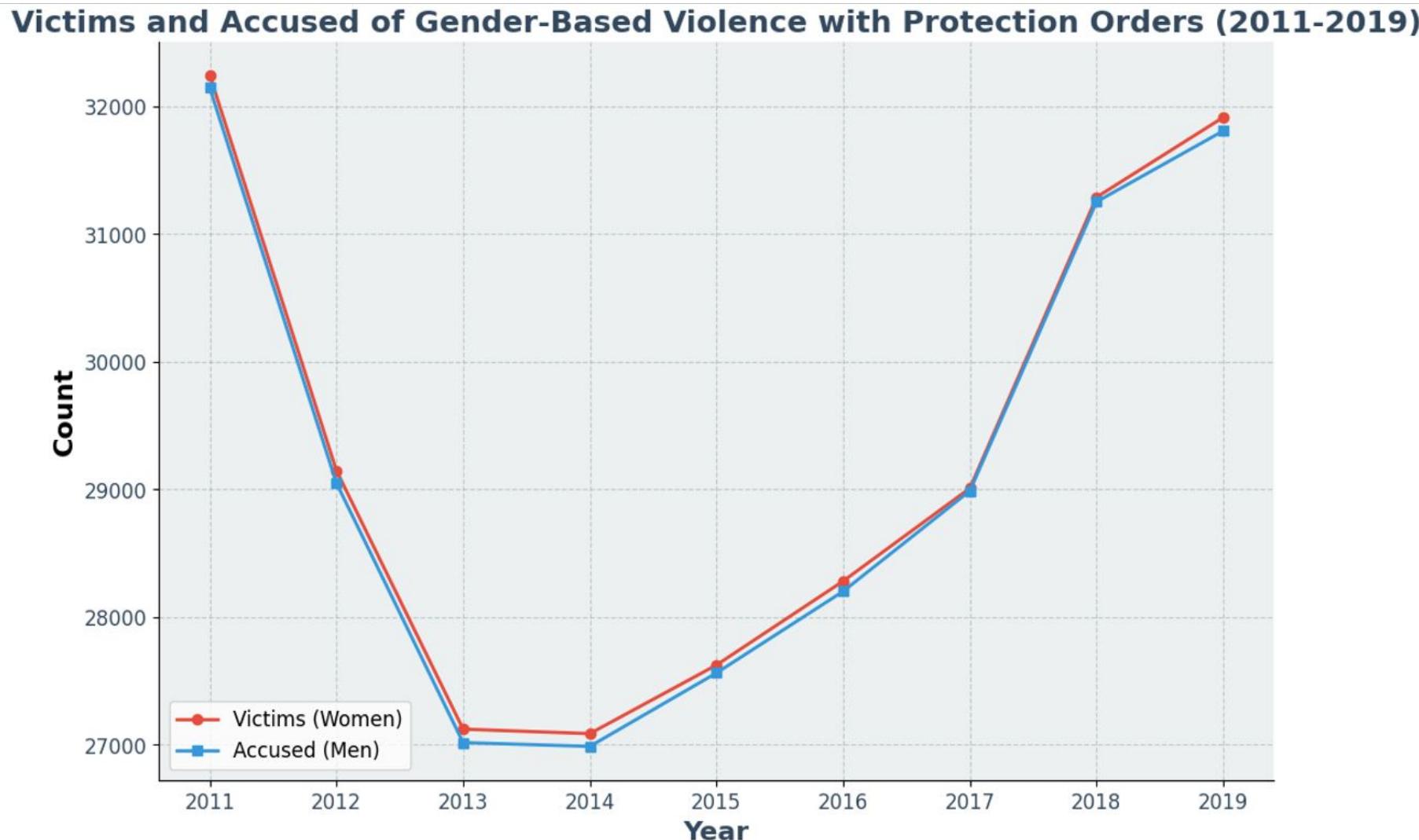
# Add a background color
plt.gca().set_facecolor('#ecf0f1')

# Remove spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Show the plot
plt.show()
```

I. Context of « Violencia de Género » in Spain

2. Victims and Accused



I. Context of « Violencia de Género » in Spain

3. Total Complaints and Daily Average

Code :

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

# Data
data = {
    "Year": [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019],
    "Total Complaints": [126293, 142125, 135540, 134105, 134002, 128543, 124894, 126742, 129193, 142893, 166260, 166961, 168057],
    "Daily Average": [346, 388, 371, 367, 367, 351, 342, 347, 354, 390, 456, 457, 460]
}

df = pd.DataFrame(data)

# Set the figure size
plt.figure(figsize=(12, 8))

# Plot Total Complaints with blue bars
plt.bar(df["Year"], df["Total Complaints"], color="#3498db", label='Total Complaints')

# Create a secondary y-axis for Daily Average line plot
ax2 = plt.gca().twinx()
ax2.plot(df["Year"], df["Daily Average"], marker='o', color="#e74c3c", label='Daily Average', linewidth=2)

# Add titles and labels
plt.title('Total Complaints and Daily Average Over the Years', fontsize=18, fontweight='bold', color="#34495e")
plt.xlabel('Year', fontsize=16, fontweight='bold', color="#34495e")
plt.ylabel('Total Complaints', fontsize=16, fontweight='bold', color="#3498db")
ax2.set_ylabel('Daily Average', fontsize=16, fontweight='bold', color="#e74c3c")

# Add legend
lines, labels = plt.gca().get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
plt.legend(labels + labels2, loc='upper left', fontsize=12)

# Customize the appearance
plt.grid(True, linestyle='--', alpha=0.5, color="#95a5a6")
plt.xticks(df["Year"], fontsize=12, color="#34495e")
plt.yticks(fontsize=12, color="#34495e")
ax2.set_yticks(ax2.get_yticks())
ax2.set_yticklabels([int(label) for label in ax2.get_yticks()], fontsize=12, color="#e74c3c")

# Add a background color
plt.gca().set_facecolor("#ecf0f1")

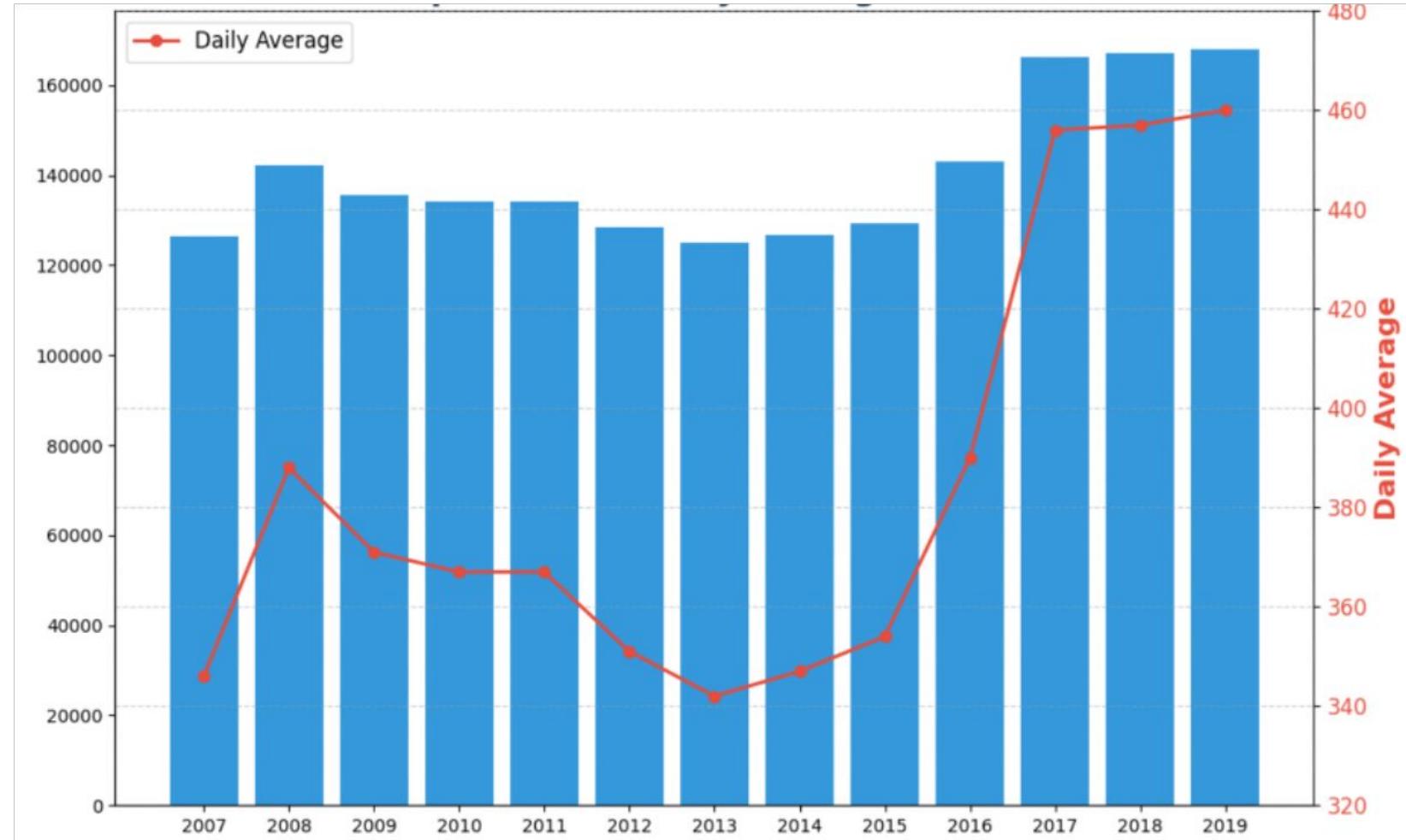
# Remove spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Show the plot
plt.show()
```

I. Context of « Violencia de Género » in Spain

3. Total Complaints and Daily Average

Total Complaints and Daily Average Over the Years (2007-2019)



I. Context of « Violencia de Género » in Spain

4. File of complaints – Percentage distribution

Code :

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

# Data
data = {
    'Year': ['2015', '2016', '2017', '2018', '2019', '2006–2019'],
    'Complaint filed': [21.70, 32.70, 24.00, 28.30, 20.00, 26.00],
    'No Complaint filed': [78.30, 67.30, 76.00, 71.70, 80.00, 74.00]
}

df = pd.DataFrame(data)

# Set the figure size and style
plt.figure(figsize=(10, 6))
plt.style.use('seaborn-whitegrid')

# Create a dark background
plt.gca().set_facecolor('lightgray')

# Create an empty plot
bars = plt.bar(df['Year'], df['Complaint filed'], color="#1f78b4", label='Complaint filed')
bars2 = plt.bar(df['Year'], df['No Complaint filed'], color="#e31a1c", label='No Complaint filed', bottom=df['Complaint filed'])

# Add labels and title
plt.xlabel('year', fontsize=14, fontweight='bold', color='black')
plt.ylabel('Percentage', fontsize=14, fontweight='bold', color='black')
plt.title("Percentage distribution of female victims of gender-based violence who died according to whether or not a complaint was lodged against the perpetrator (2003–2021)", fontsize=16, fontweight='bold', color='black')

# Add data labels
for bar in bars + bars2:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, yval + 1, f'{round(yval, 2)}%', ha='center', va='bottom', fontsize=12, color='black')

# Add legend
plt.legend(title="Legend", bbox_to_anchor=(0.5, -0.15), loc="upper center", ncol=2, fontsize=12)

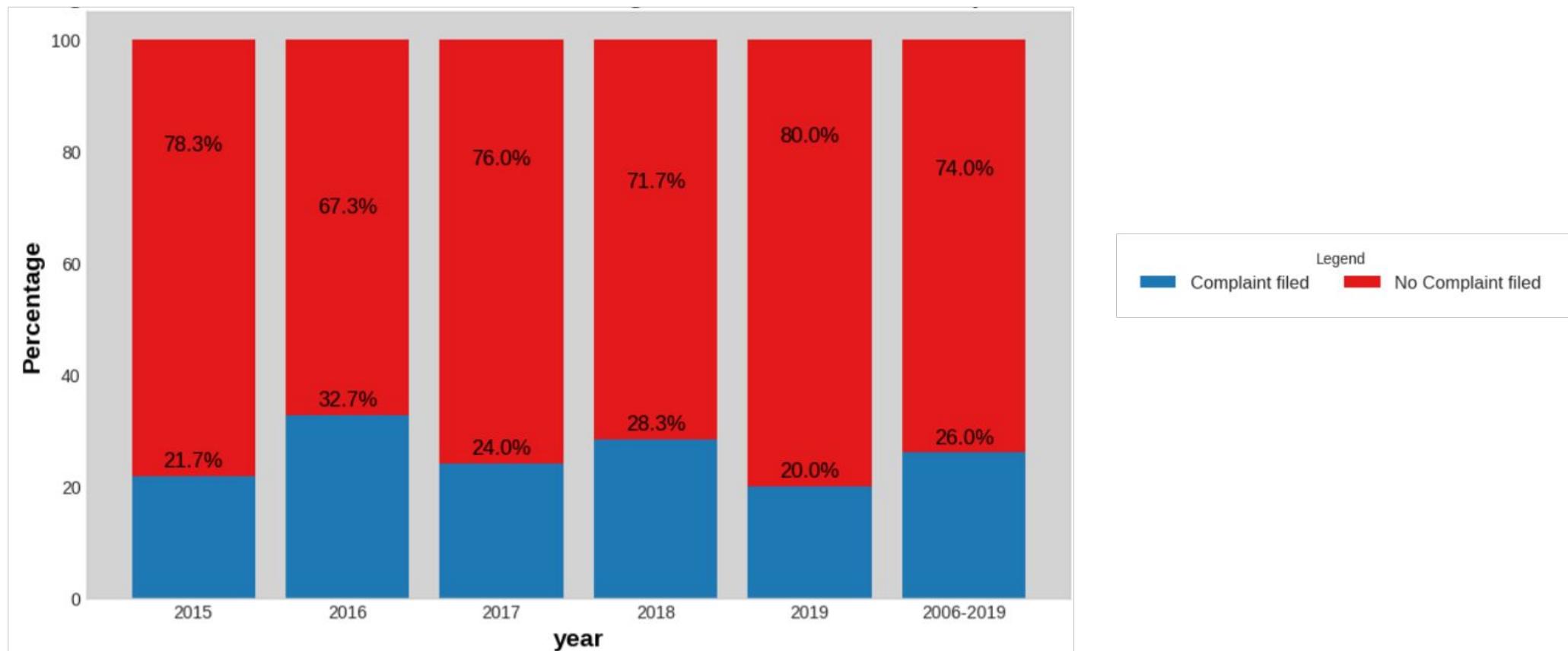
# Remove grid lines
plt.grid(False)

# Show the plot
plt.show()
```

I. Context of « Violencia de Género » in Spain

4. File of complaints – Percentage distribution

Percentage distribution of female victims of gender-based violence who died according to whether or not a complaint was lodged against the perpetrator (2006-2019)



I. Context of « Violencia de Género » in Spain

5. Complaints by origin

Code :

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# Data
data = {
    'Origin': ['Police report of the complaint lodged by the victim', 'Direct intervention police report',
               'Injury certificate', 'Victims in court', 'Support services and third parties in general',
               'Police report containing a complaint about a relative', 'Relatives of victims on trial'],
    'Total 2007-2021 period': [64.90, 14.10, 10.80, 6.50, 2.10, 1.20, 0.40],
    '2021': [70.40, 13.10, 9.60, 2.20, 2.80, 1.70, 0.20]
}

df = pd.DataFrame(data)

# Set the figure size and style
plt.figure(figsize=(12, 8))
plt.style.use('seaborn-whitegrid')

# Create horizontal bar chart for the total period
bars = plt.barh(df['Origin'], df['Total 2007-2021 period'], color='skyblue', label='Total 2007-2021 period', edgecolor='black', linewidth=1)

# Add data labels with percentage formatting
for bar in bars:
    plt.text(bar.get_width(), bar.get_y() + bar.get_height() / 2, f'{bar.get_width():.2f}%', ha='left', va='center', fontsize=10, color='black')

# Customize the plot
plt.xlabel('Percentage (%)', fontsize=12)
plt.title('Complaints of gender-based violence by origin (2007-2021)', fontsize=16, fontweight='bold')
plt.legend()
plt.tight_layout()

# Remove unnecessary spines
plt.gca().set_frame_on(False)

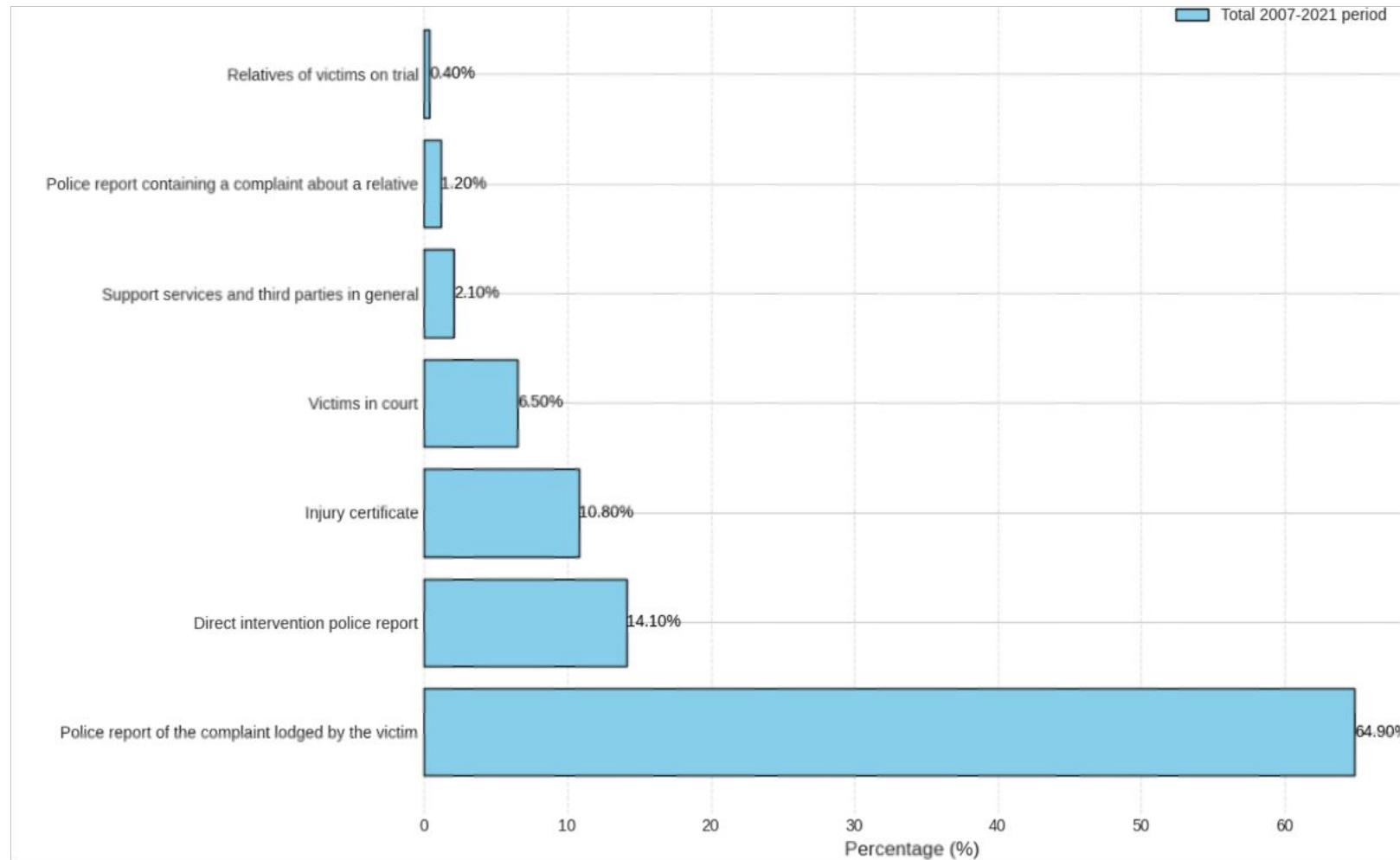
# Add a grid for better readability
plt.grid(axis='x', linestyle='--', alpha=0.6)

# Show the plot
plt.show()
```

I. Context of « Violencia de Género » in Spain

5. Complaints by origin

Complaints of gender-based violence by origin (2007-2021)



I. Context of « Violencia de Género » in Spain

6. RAI (Renta Activa de Inserción)

Code :

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# Your data
data = {
    "Year": [2019, 2018, 2017, 2016, 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 2006],
    "Number of beneficiaries": [29396, 29802, 31398, 33565, 34695, 34550, 32596, 30065, 29065, 25512, 22010, 16883, 13291, 10924],
    "Average monthly number of beneficiaries": [18334, 17815, 19039, 20309, 21763, 21785, 20630, 18710, 17819, 15839, 13461, 9444, 7601, 5673]
}

df = pd.DataFrame(data)

# Create a figure and axis
fig, ax = plt.subplots(figsize=(10, 6))

# Scatterplot for the number of beneficiaries with different sizes and shadow background
sns.scatterplot(x="Year", y="Number of beneficiaries", size="Number of beneficiaries", data=df, ax=ax, label="Number of beneficiaries", legend=False, alpha=0.7)

# Lineplot for the moyenne
sns.lineplot(x="Year", y="Average monthly number of beneficiaries", data=df, ax=ax, color='red', marker='o', label="Monthly average", alpha=0.8)

# Add mean line
mean_line = df["Number of beneficiaries"].mean()
ax.axhline(y=mean_line, color='blue', linestyle='--', label='Monthly mean of beneficiaries', alpha=0.7)

# Add regression line
sns.regplot(x="Year", y="Number of beneficiaries", data=df, ax=ax, scatter=False, color='green', label='Linear regression', line_kws={'alpha':0.7})

# Set labels and title
ax.set_xlabel("Year")
ax.set_ylabel("Number of beneficiaries")
ax.set_title("Female victims of violence receiving RAI")

# Add legend
ax.legend(loc="upper left")

# Add shadow background
ax.set_facecolor('#f5f5f5')
fig.set_facecolor('#f5f5f5')

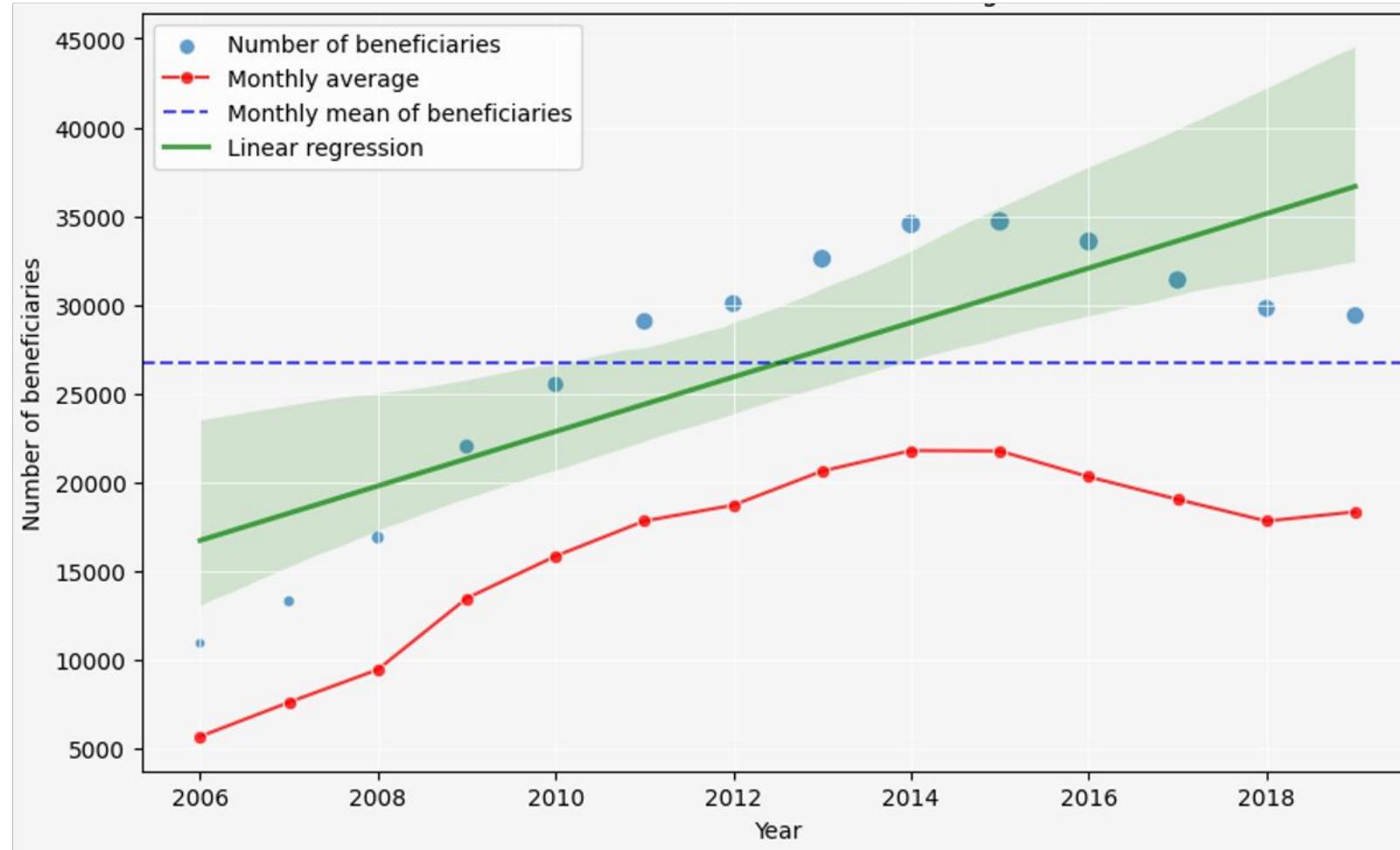
# Set grid lines
ax.grid(color='white', linestyle='-', linewidth=0.5)

# Show the plot
plt.show()
```

I. Context of « Violencia de Género » in Spain

6. RAI (Renta Activa de Inserción)

Female Victims of violence receiving RAI (2006-2018)



I. Context of « Violencia de Género » in Spain

7. ATENPRO (Servicio telefónico de atención y protección)

Code :

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.stats import linregress

# Data
data = {
    'Year': [2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019],
    'Active Users': [2374, 3287, 8787, 12274, 13696, 8830, 9939, 9405, 10426, 10502, 10887, 11491, 12477, 13376, 14472],
    'New Registrations': [2658, 1917, 8412, 7332, 7624, 5430, 5891, 5512, 8003, 7151, 7421, 8007, 8667, 9156, 9822],
    'Cancelled Registrations': [284, 1004, 2912, 3845, 6202, 10296, 4782, 6046, 6982, 7075, 7036, 7403, 7681, 8257, 8726]
}
df = pd.DataFrame(data)

# Set the figure size
plt.figure(figsize=(12, 8))

# Define bar width
bar_width = 0.25

# Plot Active Users
plt.bar(df['Year'] - bar_width, df['Active Users'], width=bar_width, color="#3498db", label='Active Users')

# Plot New Registrations
plt.bar(df['Year'], df['New Registrations'], width=bar_width, color="#2ecc71", label='New Registrations', alpha=0.7)

# Plot Cancelled Registrations
plt.bar(df['Year'] + bar_width, df['Cancelled Registrations'], width=bar_width, color="#e74c3c", label='Cancelled Registrations', alpha=0.7)

# Add data labels
for i, value in enumerate(df['Active Users']):
    plt.text(df['Year'][i] - bar_width, value + 200, str(value), ha='center', va='bottom', fontsize=10)
for i, value in enumerate(df['New Registrations']):
    plt.text(df['Year'][i], value + 200, str(value), ha='center', va='bottom', fontsize=10)
for i, value in enumerate(df['Cancelled Registrations']):
    plt.text(df['Year'][i] + bar_width, value + 200, str(value), ha='center', va='bottom', fontsize=10)
```

```
# Add regression curves
slope_active, intercept_active, _, _ = linregress(df['Year'], df['Active Users'])
reg_active = np.poly1d([slope_active, intercept_active])
plt.plot(df['Year'], reg_active(df['Year']), linestyle='--', color="#3498db", label='Active Users Regression')

slope_new, intercept_new, _, _ = linregress(df['Year'], df['New Registrations'])
reg_new = np.poly1d([slope_new, intercept_new])
plt.plot(df['Year'], reg_new(df['Year']), linestyle='--', color="#2ecc71", label='New Registrations Regression')

slope_cancelled, intercept_cancelled, _, _ = linregress(df['Year'], df['Cancelled Registrations'])
reg_cancelled = np.poly1d([slope_cancelled, intercept_cancelled])
plt.plot(df['Year'], reg_cancelled(df['Year']), linestyle='--', color="#e74c3c", label='Cancelled Registrations Regression')

# Add titles and labels
plt.title('Active Users and Registrations in ATENPRO (2005-2019)', fontsize=18, fontweight='bold', color="#34495e")
plt.xlabel('Year', fontsize=16, fontweight='bold', color="#34495e")
plt.ylabel('Counts', fontsize=16, fontweight='bold')

# Add legend
plt.legend(title="Legend", bbox_to_anchor=(0.5, -0.15), loc="upper center", ncol=2, fontsize=12)

# Customize the appearance
plt.grid(True, linestyle='--', alpha=0.5, color="#95a5a6")
plt.xticks(df['Year'], fontsize=12, color="#34495e")

# Add a background color
plt.gca().set_facecolor('#ecf0f1')

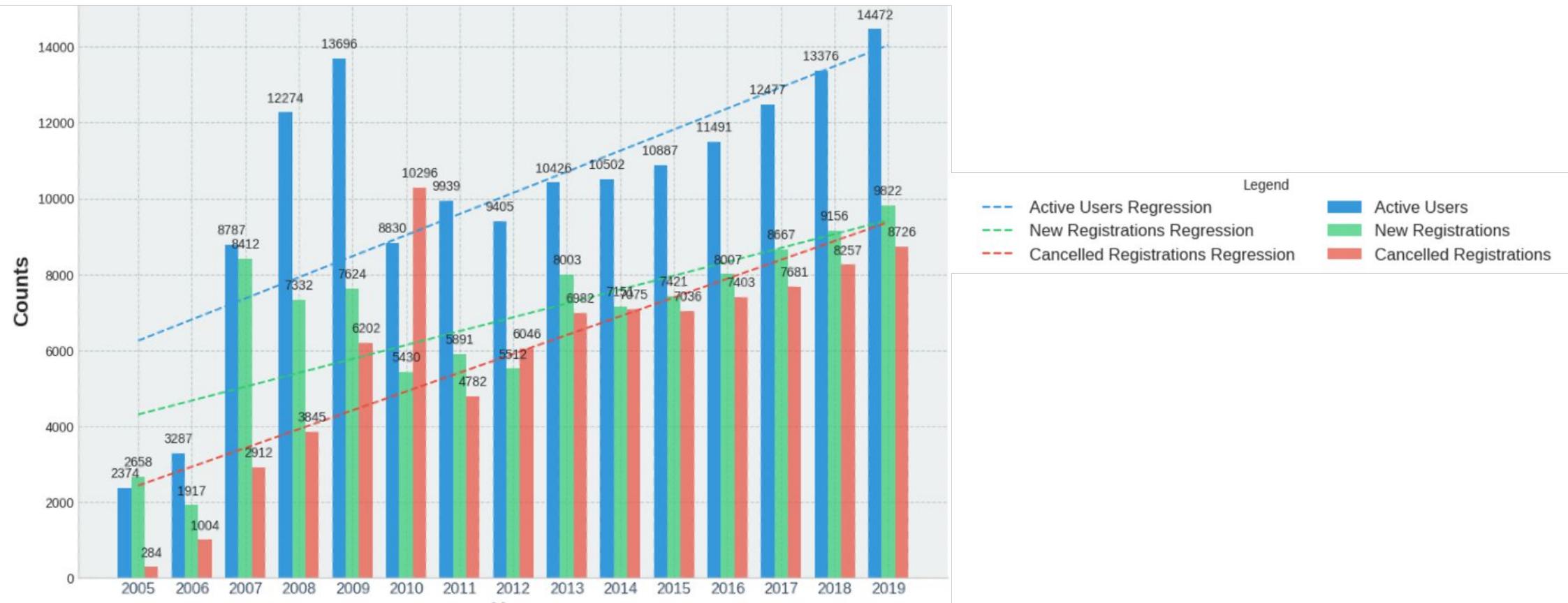
# Remove spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Show the plot
plt.show()
```

I. Context of « Violencia de Género » in Spain

7. ATENPRO (Servicio telefónico de atención y protección)

Active Users and Registrations in ATENPRO (2005-2019)



I. Context of « Violencia de Género » in Spain

8. Calls for Violencia de Généro

Code :

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Data
data = {
    'Day': ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'],
    '2015': [14000, 12900, 12500, 12100, 11200, 9800, 9800],
    '2016': [14900, 13000, 12700, 12200, 12200, 10200, 10200],
    '2017': [13500, 12000, 11500, 10900, 10900, 9100, 9800],
    '2018': [12700, 11100, 10900, 10700, 10100, 9100, 9000],
    '2019': [11900, 10700, 10100, 9900, 9100, 8000, 8900]
}

df = pd.DataFrame(data)

# Melt the DataFrame for Seaborn's violin plot
melted_df = pd.melt(df, id_vars=['Day'], var_name='Year', value_name='Number of Calls')

# Set the figure size and style
plt.figure(figsize=(12, 8))
plt.style.use('seaborn-whitegrid')

# Create a violin plot for each day with enhanced aesthetics
sns.violinplot(x='Day', y='Number of Calls', data=melted_df, palette='muted', inner='quartile', saturation=0.7)

# Customize the plot
plt.title('Violin Plot of Calls for Violence Based on Gender (2015-2019)', fontsize=18, fontweight='bold', color='navy')
plt.xlabel('Day of the Week', fontsize=16, fontweight='bold')
plt.ylabel('Number of Calls', fontsize=16, fontweight='bold')
plt.xticks(rotation=45, ha='right', fontsize=14) # Rotate x-axis labels for better readability
plt.yticks(fontsize=14) # Adjust y-axis tick font size

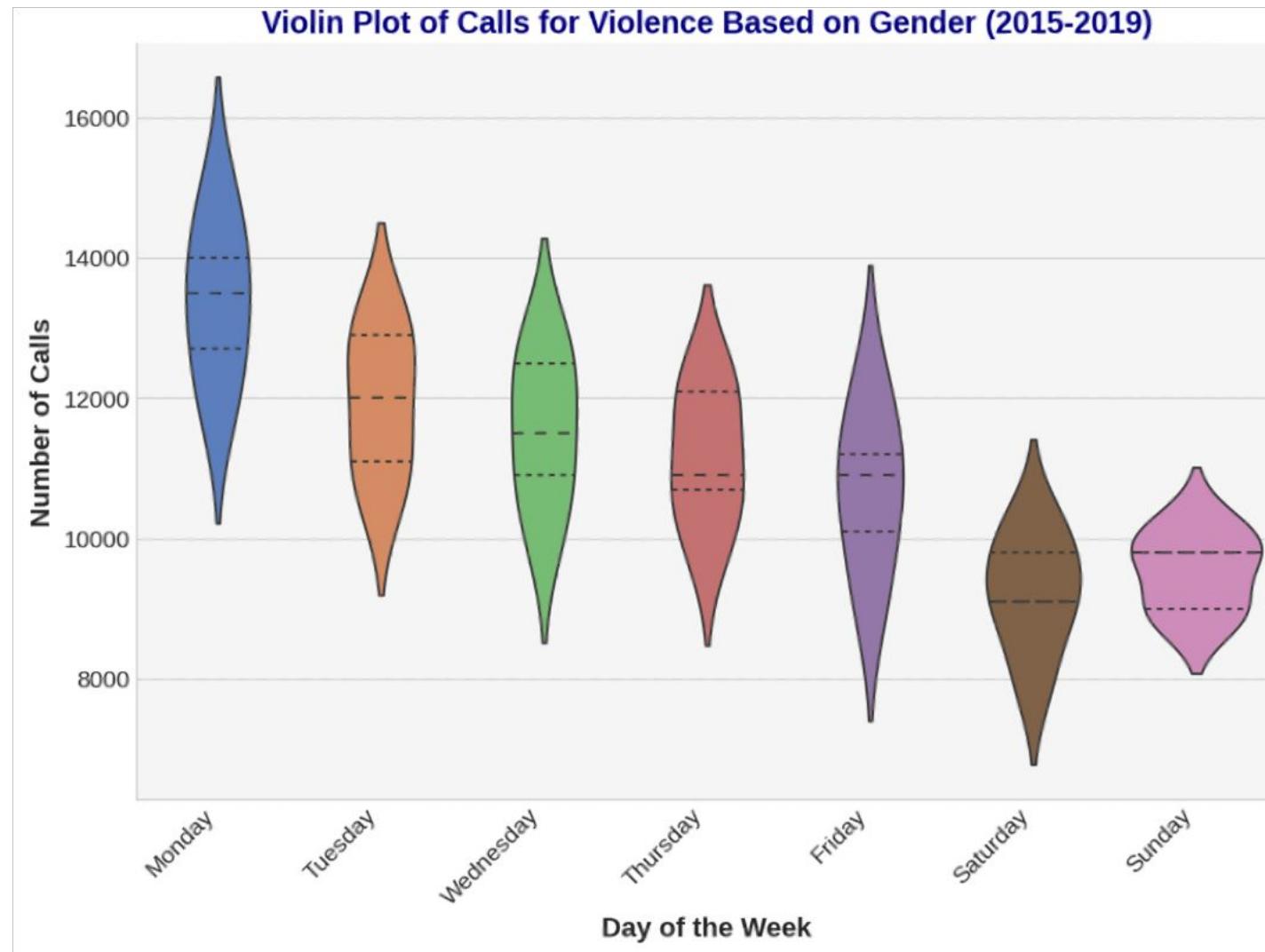
# Add a background color for a professional look
plt.gca().set_facecolor('whitesmoke')

# Remove top and right spines for cleanliness
sns.despine()

# Show the plot
plt.show()
```

I. Context of « Violencia de Género » in Spain

8. Calls for Violencia de Género



I. Context of « Violencia de Género » in Spain

9. Financial Aid (Article 27 of the « Ley integral »)

Code :

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np

# Data
data = {
    'Year': [2019, 2018, 2017, 2016, 2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008, 2007, 2006],
    'Beneficiaries': [1156, 826, 782, 730, 685, 606, 483, 463, 430, 350, 360, 284, 199, 61]
}

df = pd.DataFrame(data)

# Set the figure size
plt.figure(figsize=(12, 8))

# Plot the regression curve with filled area for confidence interval
sns.regplot(x='Year', y='Beneficiaries', data=df, color='#3498db', ci=95, scatter_kws={'s': 100},
             line_kws={'color': '#e74c3c', 'linewidth': 2, 'label': 'Regression Curve'})
plt.fill_between(df['Year'], df['Beneficiaries'] - 50, df['Beneficiaries'] + 50, color='#3498db', alpha=0.2)

# Annotate key points
for i, row in df.iterrows():
    plt.text(row['Year'] + 0.1, row['Beneficiaries'], f"{row['Year']}: {row['Beneficiaries']}", fontsize=10, color="#2c3e50")

# Add titles and labels
plt.title("Women Receiving Financial Aid under Article 27 of the Ley Integral (2006-2019)", fontsize=18, fontweight='bold', color="#34495e")
plt.xlabel('Year', fontsize=16, fontweight='bold', color="#34495e")
plt.ylabel('Number of Beneficiaries', fontsize=16, fontweight='bold')

# Customize the appearance
plt.grid(True, linestyle='--', alpha=0.5, color="#95a5a6")
plt.xticks(df['Year'], fontsize=12, color="#34495e")
plt.yticks(fontsize=12, color="#34495e")
plt.legend(fontsize=12, loc='upper left')

# Add a background color for a professional look
plt.gca().set_facecolor('#ecf0f1')

# Remove spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Fine-tune layout
plt.tight_layout()

# Show the plot
plt.show()
```

I. Context of « Violencia de Género » in Spain

9. Financial Aid (Article 27 of the « Ley integral »)

Women Receiving Financial Aid under Article 27 of the « Ley Integral »



I. Context of « Violencia de Género » in Spain

10. GeoJSON visualization

```
# Load Google Drive to have access to the GE0dataframe
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import geopandas as gpd

# Set the filepath and load in a shapefile
spain_data = '/content/drive/MyDrive/Colab Notebooks/georef-spain-municipio-millesime.geojson'
VDG_data = '/content/drive/MyDrive/Colab Notebooks/map-4.geojson'

# Read shapefiles
spain = gpd.read_file(spain_data)
VDG = gpd.read_file(VDG_data)

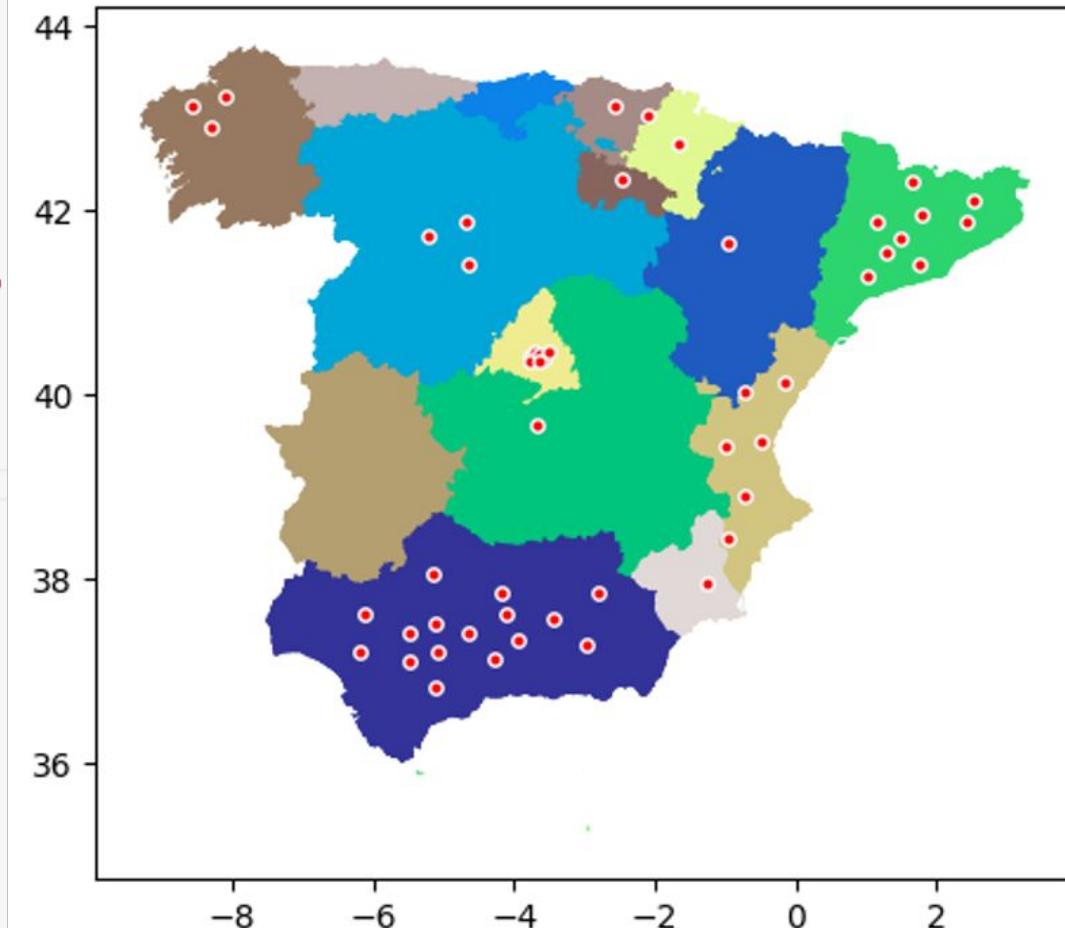
# Exclude the Canary Islands from the Spain GeoDataFrame
spain = spain[spain['acom_name'] != 'Canarias']

# Exclude the Baleares Islands from the Spain GeoDataFrame
spain = spain[spain['acom_name'] != 'Illes Balears']

# Plot Spain GeoDataFrame with different colors for each 'acom_name'
spain.plot(column='acom_name', cmap='terrain')

# Plot VDG GeoDataFrame with white dots and black border
VDG.plot(ax=plt.gca(), marker='o', color='red', edgecolor='white', markersize=15)

# Show the plot
plt.show()
```



II. Data Analysis of Judgments of the « Juzgado de Violencia sobre la Mujer »

i. Efficiency analysis

Code :

```
import pandas as pd
import matplotlib.pyplot as plt
from google.colab import auth

# Autenticar en Google Colab
auth.authenticate_user()

# Enlace a tu hoja de cálculo de Google Sheets (asegúrate de que sea compartida públicamente)
gsheet_url = "https://docs.google.com/spreadsheets/d/1a7-hqPUsXDWpTSfEshh4ti2yZWXbGZxC/edit?usp=sharing&ouid=110850547870905770175&rtpof=true&sd=true"

# Cargar la hoja de cálculo en un DataFrame de pandas
datos = pd.read_csv(f"{gsheet_url.replace('/edit?usp=sharing', '/export?format=csv')}")

# Limpiar valores no finitos en la columna "Anos"
datos['Anos'] = pd.to_numeric(datos['Anos'], errors='coerce')

# Redondear y convertir la columna "Anos" a números enteros
datos['Anos'] = datos['Anos'].round().astype('Int64') # Usamos 'Int64' para permitir NaN

# Filtrar las filas de "Resueltos" y "Pendientes al finalizar"
resueltos = datos.loc[datos['Anos'].notnull(), ['Anos', 'Resueltos']]
pendientes = datos.loc[datos['Anos'].notnull(), ['Anos', 'Pendientes al finalizar']]

# Agrupar por año y sumar los casos
total_casos = pd.merge(resueltos, pendientes, on='Anos', how='outer').fillna(0)
total_casos['Total'] = total_casos['Resueltos'] + total_casos['Pendientes al finalizar']

# Graficar
fig, ax = plt.subplots(figsize=(12, 6))

# Gráfico de barras apiladas con años en el eje x
ax.bar(total_casos['Anos'], total_casos['Resueltos'], color='pink', label='Resueltos')
ax.bar(total_casos['Anos'], total_casos['Pendientes al finalizar'], bottom=total_casos['Resueltos'], color='purple', label='Pendientes')

# Añadir líneas horizontales cada 50 unidades en el eje y
for y_value in range(0, int(total_casos['Total'].max()) + 1, 50):
    ax.axhline(y=y_value, color='gray', linestyle='--', linewidth=0.8)

# Añadir etiquetas y leyenda
ax.set_xlabel('Año')
ax.set_ylabel('Número de casos')
ax.set_title('Número de casos resueltos y pendientes por año')
ax.legend(title='Estado', loc='upper right', bbox_to_anchor=(1.2, 1))

# Ajustar el diseño para evitar solapamiento
plt.tight_layout()

# Convertir la columna 'Anos' a una lista de enteros para establecer las etiquetas del eje x manualmente
years = total_casos['Anos'].dropna().astype(int).tolist()

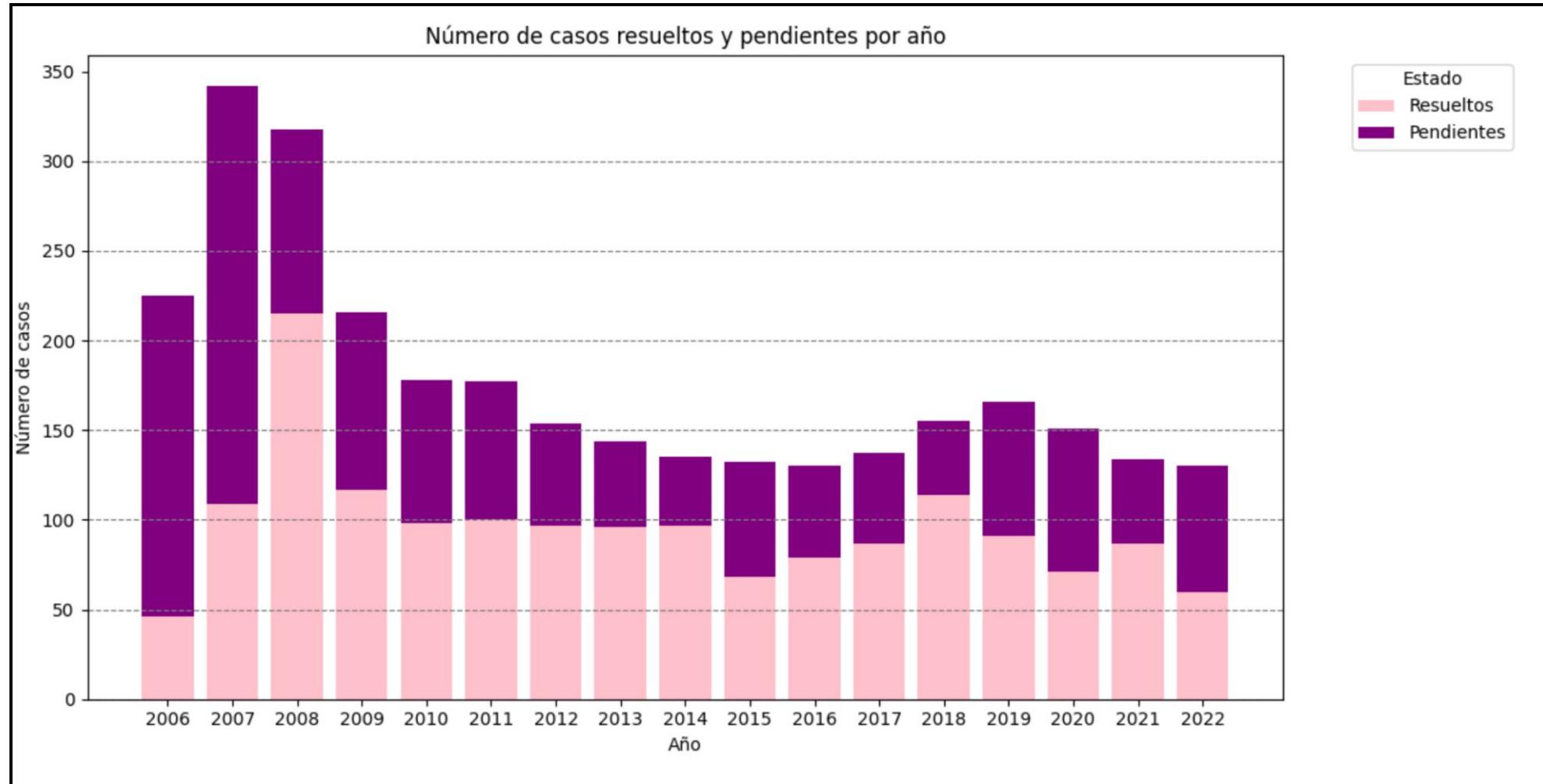
# Cambiar las etiquetas del eje x manualmente
ax.set_xticks(years)
ax.set_xticklabels(['2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022'])

# Mostrar la gráfica
plt.show()
```

i. Efficiency analysis

1. Number of cases solved vs still pending

Output:



i. Efficiency analysis

2. Percentage of cases solved vs pending

Code:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from google.colab import auth

# Autenticar en Google Colab
auth.authenticate_user()

# Enlace a tu hoja de cálculo de Google Sheets (asegúrate de que sea compartida públicamente)
gsheet_url = "https://docs.google.com/spreadsheets/d/1a7-hqPUsXDWPtSfEshh4ti2yZXbGZxC/edit?usp=sharing&ouid=110850547870905770175&rtpof=true&sd=true"

# Cargar la hoja de cálculo en un DataFrame de pandas
datos = pd.read_csv(f"{gsheet_url.replace('/edit?usp=sharing', '/export?format=csv')}")

# Filtrar las filas de "Resueltos" y "Pendientes al finalizar"
resueltos = datos['Resueltos'].sum()
pendientes = datos['Pendientes al finalizar'].sum()

# Calcular el porcentaje
porcentaje_resueltos = (resueltos / (resueltos + pendientes)) * 100
porcentaje_pendientes = 100 - porcentaje_resueltos

# Configuración de colores más contrastantes
colors = ['#FF6B81', '#6A89CC'] # Rosa oscuro y azul pálido
explode = (0.1, 0) # Separación de la porción "Resueltos" para resaltar

# Configuración del gráfico de pastel
fig, ax = plt.subplots(figsize=(10, 6))
wedges, texts, autotexts = ax.pie([porcentaje_resueltos, porcentaje_pendientes],
                                    labels=['Resueltos', 'Pendientes'],
                                    autopct=lambda p: '{:.1f}%'.format(p) if p > 0 else '',
                                    colors=colors, startangle=90, explode=explode, wedgeprops=dict(width=0.3, edgecolor='w'))

# Personalizar el texto en cada porción
for text, autotext in zip(texts, autotexts):
    text.set_size(14)
    text.set_weight('bold')
    autotext.set_size(12)

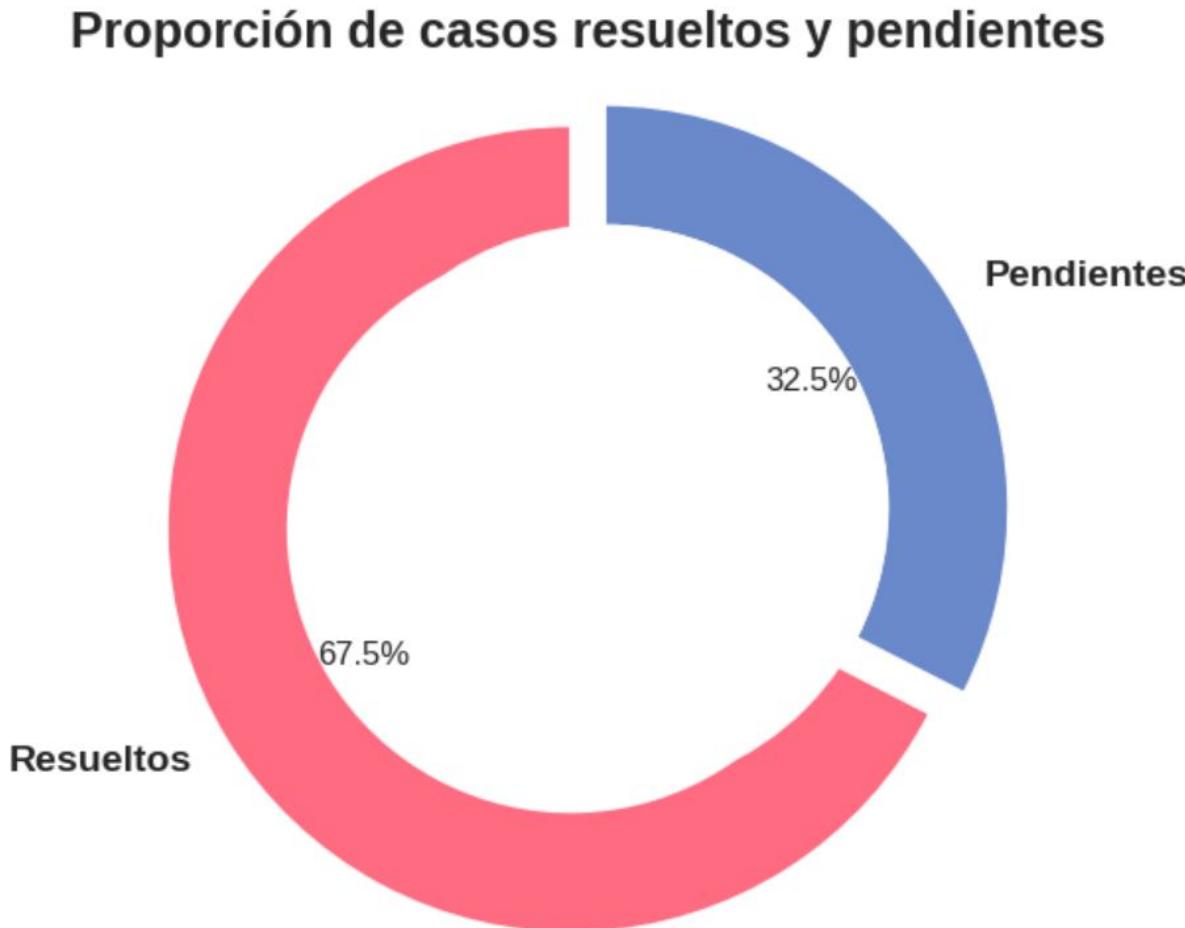
# Añadir un círculo blanco en el centro para hacer un donut chart
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# Configuración adicional
plt.title('Proporción de casos resueltos y pendientes', fontsize=18, fontweight='bold')
plt.axis('equal') # Aspecto circular
plt.show()
```

i. Efficiency analysis

2. Percentage of cases solved vs pending

Output:



i. Efficiency analysis

3. Boxplot matrix showing the efficiency of each procedure

Code :

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import auth

# Autenticar en Google Colab
auth.authenticate_user()

# Enlace a tu hoja de cálculo de Google Sheets (asegúrate de que sea compartida públicamente)
gsheet_url = "https://docs.google.com/spreadsheets/d/1a7-hqPUsXDWPfEShh4ti2yZWxbGzxC/edit?usp=sharing&ouid=110850547870905770175&rtpof=true&sd=true"

# Cargar la hoja de cálculo en un DataFrame de pandas
datos = pd.read_csv(f'{gsheet_url.replace("/edit?usp=sharing", "/export?format=csv")}')

# Limpiar valores no finitos en la columna "Años"
datos['Años'] = pd.to_numeric(datos['Años'], errors='coerce')

# Redondear y convertir la columna "Años" a números enteros
datos['Años'] = datos['Años'].round().astype('Int64') # Usamos 'Int64' para permitir NaN

# Crear una nueva columna 'Grupo' para marcar los bloques de años
datos['Grupo'] = datos['Años'].ffill()

# Obtener la lista única de años
años = datos['Años'].unique()

# Configurar subgráficos
num_anios = len(años)
num_cols = 3 # Puedes ajustar el número de columnas según tus preferencias
num_filas = (num_anios // num_cols) + (num_anios % num_cols)

# Ajustar el tamaño de los subgráficos y las etiquetas
fig, axes = plt.subplots(num_filas, num_cols, figsize=(20, 8 * num_filas))
fig.suptitle('Distribución de casos resueltos por procedimiento y año', fontsize=24, fontweight='bold')

# Iterar sobre cada año y crear un boxplot en el subgráfico correspondiente
for i, anio in enumerate(años):
    fil = i // num_cols
    col = i % num_cols

    datos_anio = datos[datos['Grupo'] == anio]

    # Verificar si hay datos para el año específico
    if not datos_anio.empty:
        # Mostrar solo las iniciales en el eje x y rotarlas verticalmente
        datos_anio['Procedimientos'] = datos_anio['Procedimientos'].str.split().str[0]

        sns.boxplot(data=datos_anio, x='Procedimientos', y='Resueltos', ax=axes[fil, col], palette="Set3")

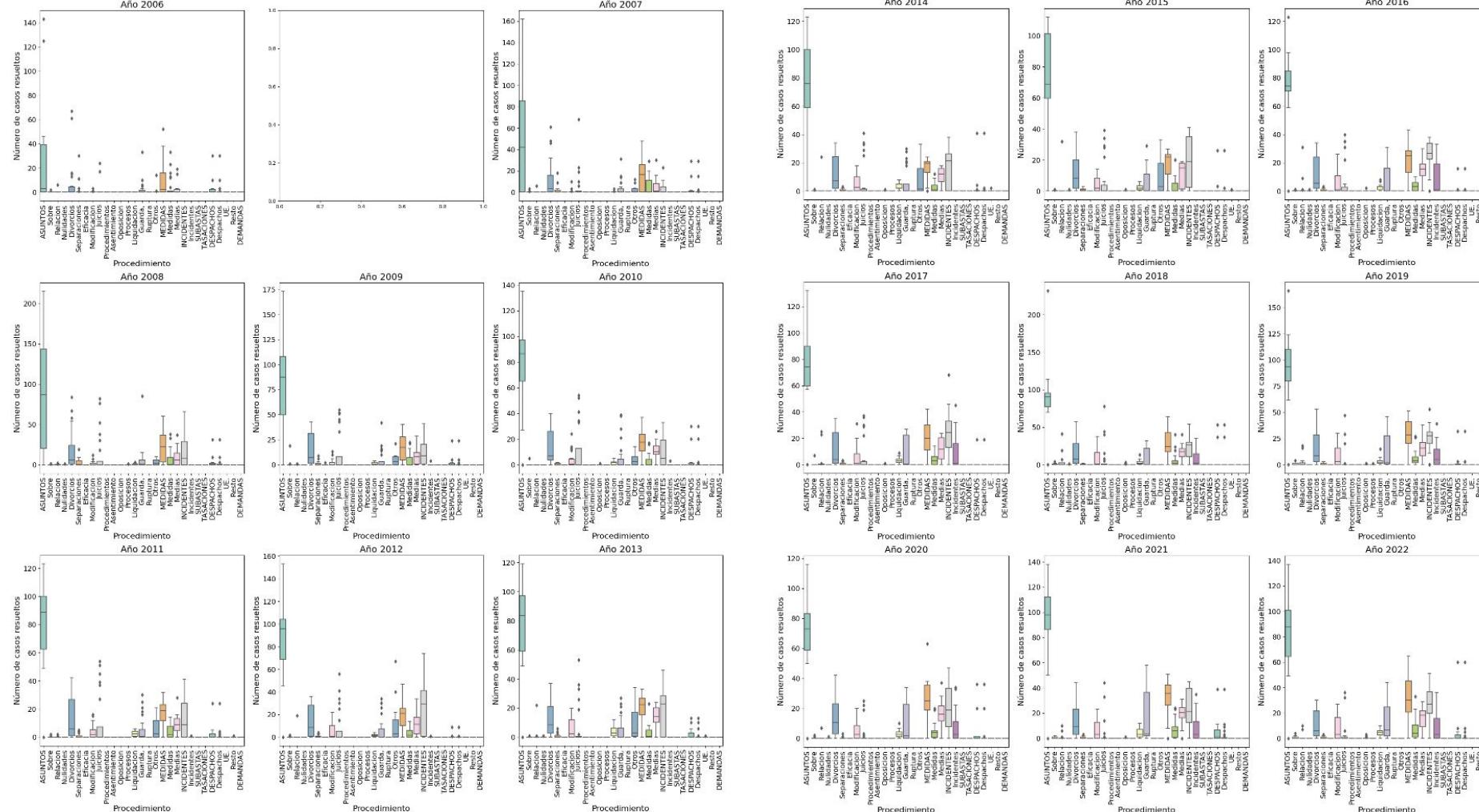
        axes[fil, col].set_title(f'Año {anio}', fontsize=18)
        axes[fil, col].set_xlabel('Procedimiento', fontsize=16)
        axes[fil, col].set_ylabel('Número de casos resueltos', fontsize=16)
        axes[fil, col].tick_params(axis='x', rotation=90, labelsize=14) # Etiquetas completamente verticales
        axes[fil, col].tick_params(axis='y', labelsize=14) # Ajustar tamaño de las etiquetas en el eje y

# Ajustar el diseño para evitar solapamiento
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

i. Efficiency analysis

3. Boxplot matrix showing the efficiency of each procedure

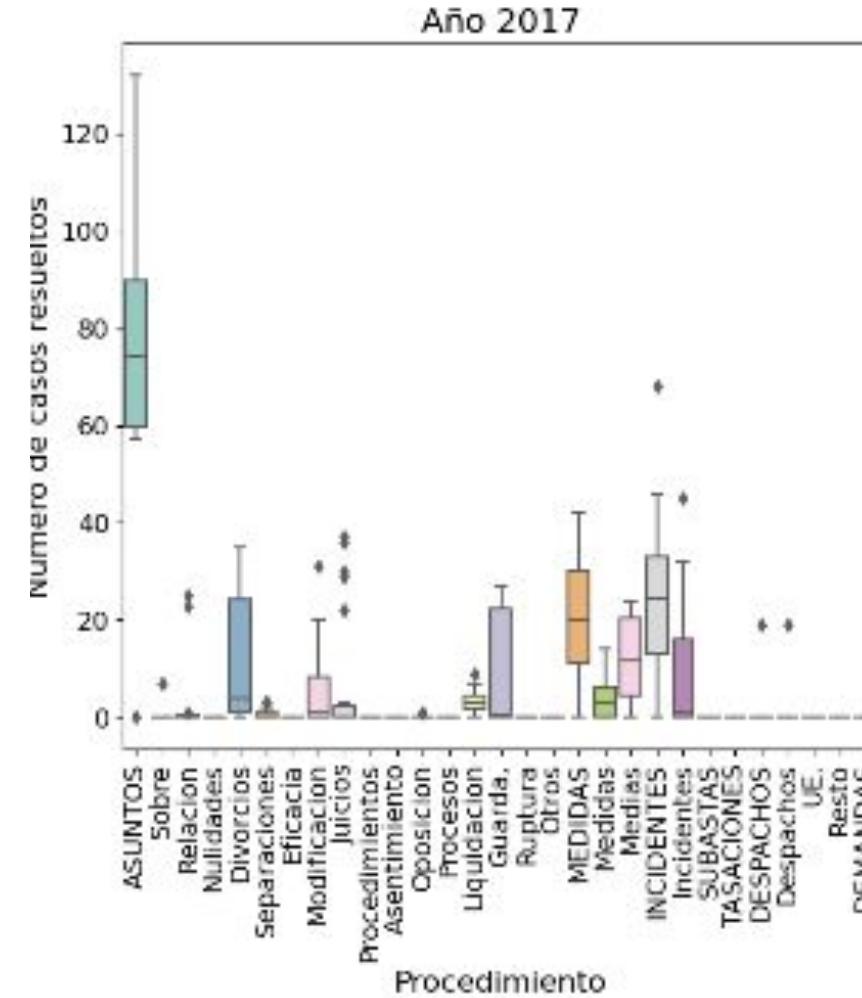
Output:



i. Efficiency analysis

3. Boxplot matrix showing the efficiency of each procedure

Closeup:



i. Efficiency analysis

4. Chronological evolution of solved cases

Code :

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import auth

# Autenticar en Google Colab
auth.authenticate_user()

# Enlace a tu hoja de cálculo de Google Sheets (asegúrate de que sea compartida públicamente)
gsheet_url = "https://docs.google.com/spreadsheets/d/1a7-hqPUsXDWpTSfEshh4ti2yZWXbGZxC/edit?usp=sharing&ouid=110850547870905770175&rtpof=true&sd=true"

# Cargar la hoja de cálculo en un DataFrame de pandas
datos = pd.read_csv(f"{gsheet_url.replace('/edit?usp=sharing', '/export?format=csv')}")

# Filtrar las filas de "Resueltos"
resueltos = datos.loc[datos['Anos'].notnull(), ['Anos', 'Resueltos']]

# Configurar el estilo del gráfico
sns.set(style="whitegrid")

# Configurar el gráfico de línea con marcadores
plt.figure(figsize=(14, 8))
sns.lineplot(x='Anos', y='Resueltos', data=resueltos, marker='o', color="#1f78b4", label='Casos Resueltos')

# Agregar líneas de media y desviación estándar
mean_line = resueltos['Resueltos'].mean()
std_dev_line = resueltos['Resueltos'].std()

plt.axhline(mean_line, color='red', linestyle='--', label='Media')
plt.axhline(mean_line + std_dev_line, color='orange', linestyle='--', label='Media + 1 Desviación Estándar')
plt.axhline(mean_line - std_dev_line, color='orange', linestyle='--', label='Media - 1 Desviación Estándar')

# Personalizar el gráfico
plt.title('Evolución del número de casos resueltos a lo largo de los años', fontsize=18, fontweight='bold')
plt.xlabel('Año', fontsize=14)
plt.ylabel('Número de casos resueltos', fontsize=14)
plt.xticks(rotation=45, ha='right') # Rotar etiquetas del eje x para mayor legibilidad
plt.legend(fontsize=12)

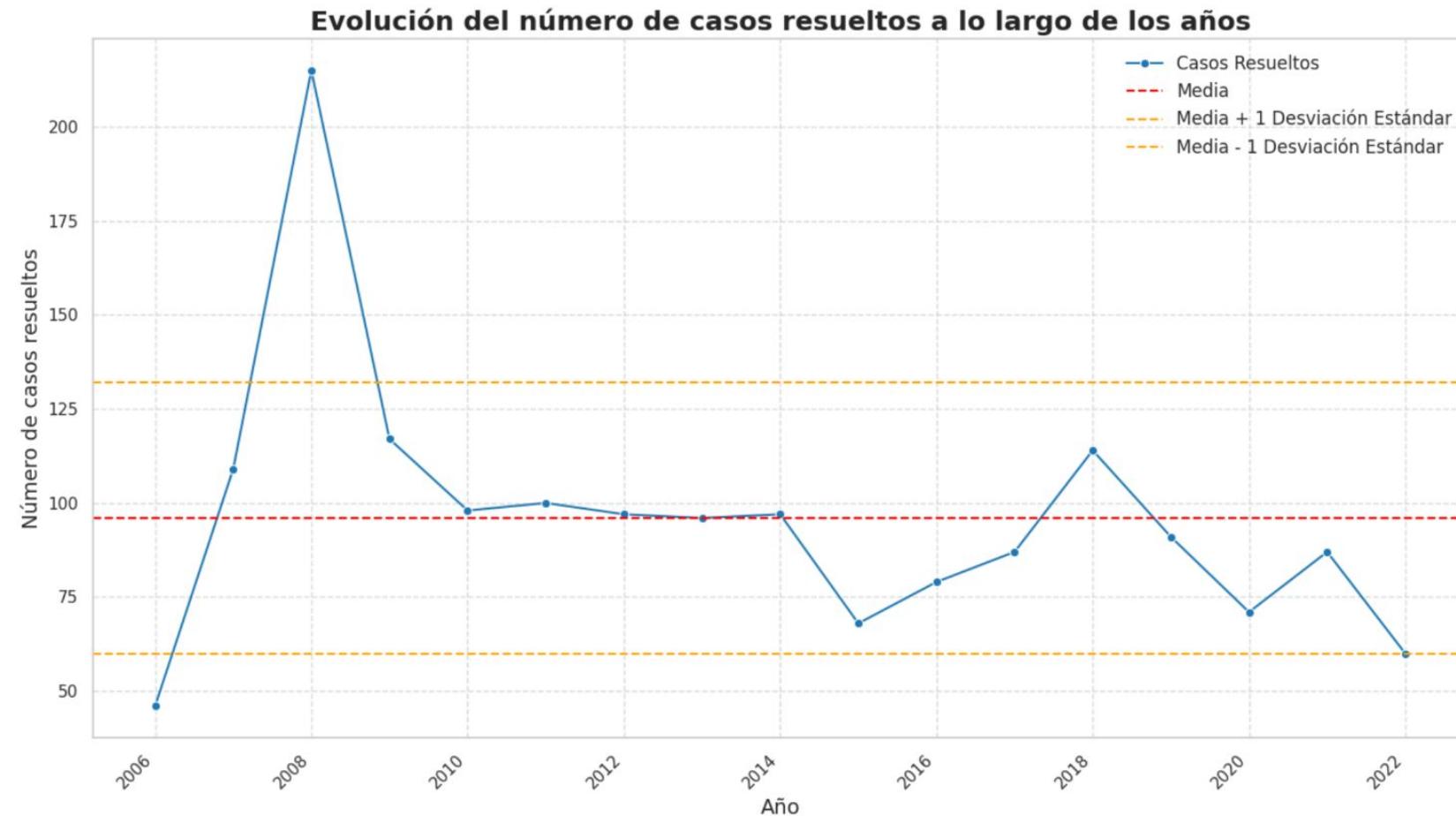
# Añadir grid y ajustar diseño
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

i. Efficiency analysis

4. Chronological evolution of solved cases

Output:



i. Efficiency analysis

4. Chronological evolution of solved cases as compared to pending

Code :

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import auth

# Autenticar en Google Colab
auth.authenticate_user()

# Enlace a tu hoja de cálculo de Google Sheets (asegúrate de que sea compartida públicamente)
gsheet_url = "https://docs.google.com/spreadsheets/d/1a7-hqPUsXDWpTSfEshh4ti2yZWXbGZxC/edit?usp=sharing&ouid=110850547870905770175&rtpof=true&sd=true"

# Cargar la hoja de cálculo en un DataFrame de pandas
datos = pd.read_csv(f"{gsheet_url.replace('/edit?usp=sharing', '/export?format=csv')}")

# Filtrar las filas de "Resueltos" y "Pendientes al finalizar"
total_casos = datos.loc[datos['Anos'].notnull(), ['Anos', 'Resueltos', 'Pendientes al finalizar']]

# Configurar el estilo del gráfico
sns.set(style="whitegrid")

# Configurar el gráfico de líneas en múltiples facetas
plt.figure(figsize=(14, 8))

# Graficar líneas para casos resueltos
sns.lineplot(x='Anos', y='Resueltos', data=total_casos, label='Resueltos', marker='o')

# Graficar líneas para casos pendientes
sns.lineplot(x='Anos', y='Pendientes al finalizar', data=total_casos, label='Pendientes', marker='o')

# Calcular y graficar líneas de media y desviación estándar para casos resueltos
mean_resueltos = total_casos['Resueltos'].mean()
std_dev_resueltos = total_casos['Resueltos'].std()
plt.axhline(mean_resueltos, color='red', linestyle='--', label='Media Resueltos')
plt.axhline(mean_resueltos + std_dev_resueltos, color='orange', linestyle='--', label='Media + 1 Desviación Resueltos')
plt.axhline(mean_resueltos - std_dev_resueltos, color='orange', linestyle='--', label='Media - 1 Desviación Resueltos')

# Calcular y graficar líneas de media y desviación estándar para casos pendientes
mean_pendientes = total_casos['Pendientes al finalizar'].mean()
std_dev_pendientes = total_casos['Pendientes al finalizar'].std()
plt.axhline(mean_pendientes, color='green', linestyle='--', label='Media Pendientes')
plt.axhline(mean_pendientes + std_dev_pendientes, color='yellow', linestyle='--', label='Media + 1 Desviación Pendientes')
plt.axhline(mean_pendientes - std_dev_pendientes, color='yellow', linestyle='--', label='Media - 1 Desviación Pendientes')

# Personalizar el gráfico
plt.title('Evolución del número total de casos resueltos y pendientes a lo largo de los años', fontsize=18, fontweight='bold')
plt.xlabel('Año', fontsize=14)
plt.ylabel('Número de casos', fontsize=14)
plt.xticks(rotation=45, ha='right') # Rotar etiquetas del eje x para mayor legibilidad
plt.legend(fontsize=10)

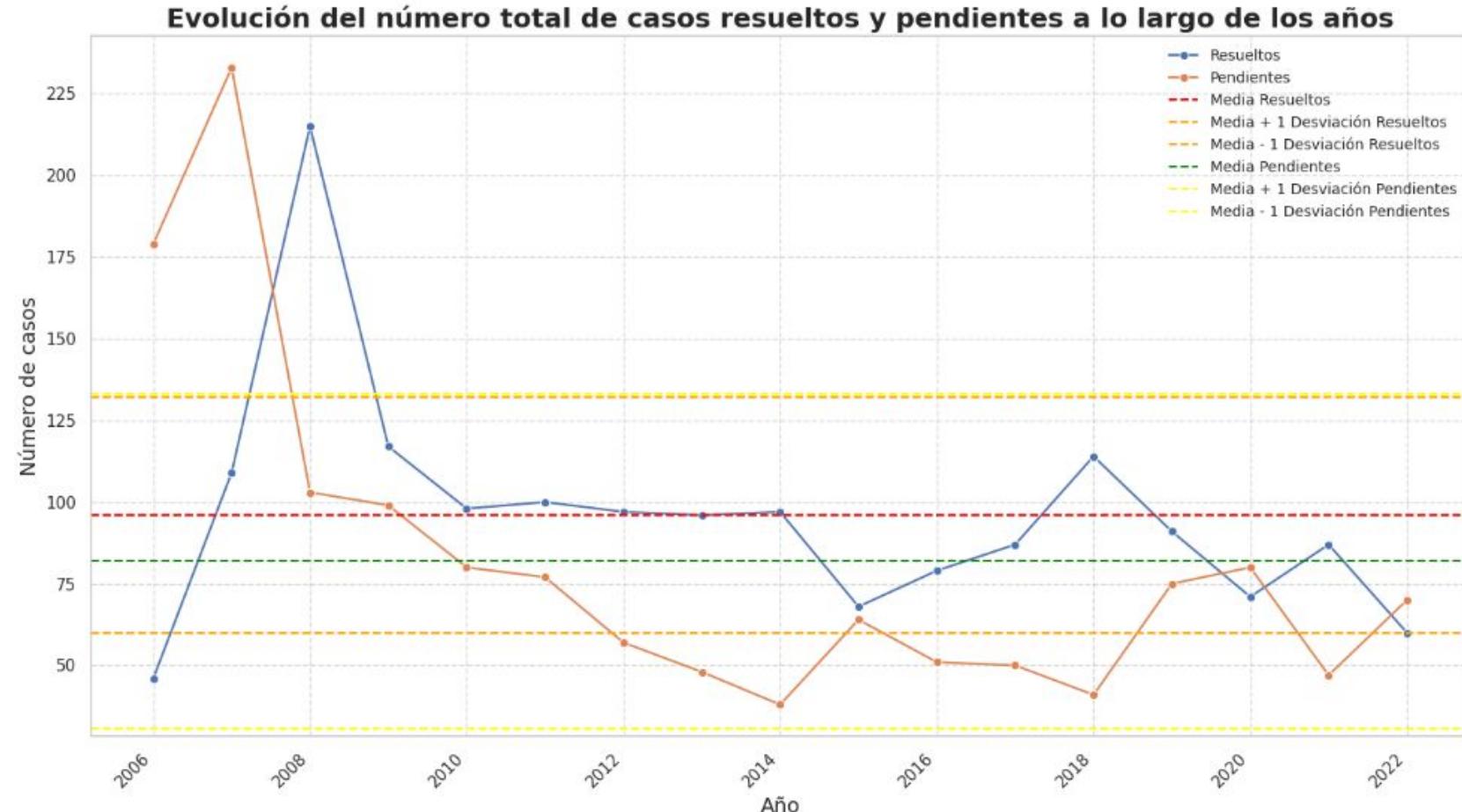
# Añadir grid y ajustar diseño
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()

# Mostrar el gráfico
plt.show()
```

i. Efficiency analysis

4. Chronological evolution of solved cases as compared to pending

Output:



i. Efficiency analysis

5. Percentage of men that directly went to prison

Code :

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import auth

# Autenticar en Google Colab
auth.authenticate_user()

# Enlace a tu hoja de cálculo de Google Sheets (asegúrate de que sea compartida públicamente)
gsheet_url = "https://docs.google.com/spreadsheets/d/1a7-hqPUsXDWpTSfEshh4ti2yZWxbGzxC/edit?usp=sharing&ouid=110850547870905770175&rtpof=true&sd=true"

# Cargar la hoja de cálculo en un DataFrame de pandas
datos = pd.read_csv(f"{gsheet_url.replace('/edit?usp=sharing', '/export?format=csv')}")

# Limpiar valores no finitos en la columna "Anos"
datos['Anos'] = pd.to_numeric(datos['Anos'], errors='coerce')

# Redondear y convertir la columna "Anos" a números enteros
datos['Anos'] = datos['Anos'].round().astype('Int64') # Usamos 'Int64' para permitir NaN

# Crear una nueva columna 'Grupo' para marcar los bloques de años
datos['Grupo'] = datos['Anos'].ffill()

# Agrupar por 'Grupo' y sumar los valores correspondientes
ingresos_por_anio = datos.groupby('Grupo')[['Ingresados. Directamente', 'Ingresados. Procedentes transformacion']].sum().reset_index()

# Configurar el estilo del gráfico
sns.set(style="whitegrid")

# Graficar
fig, ax = plt.subplots(figsize=(14, 8))

# Gráfico de barras apiladas con colores mejorados
ax.bar(ingresos_por_anio['Grupo'], ingresos_por_anio['Ingresados. Directamente'], label='Directos', color="#69B3E2")
ax.bar(ingresos_por_anio['Grupo'], ingresos_por_anio['Ingresados. Procedentes transformacion'], bottom=ingresos_por_anio['Ingresados. Directamente'], label='Transformación', color="#FFA07A")

# Añadir etiquetas y leyenda
ax.set_xlabel('Año', fontsize=14)
ax.set_ylabel('Número de casos', fontsize=14)
ax.set_title('Número de casos ingresados por año', fontsize=18, fontweight='bold')
ax.legend(title='Tipo de ingreso', loc='upper right', bbox_to_anchor=(1.2, 1), fontsize=12)

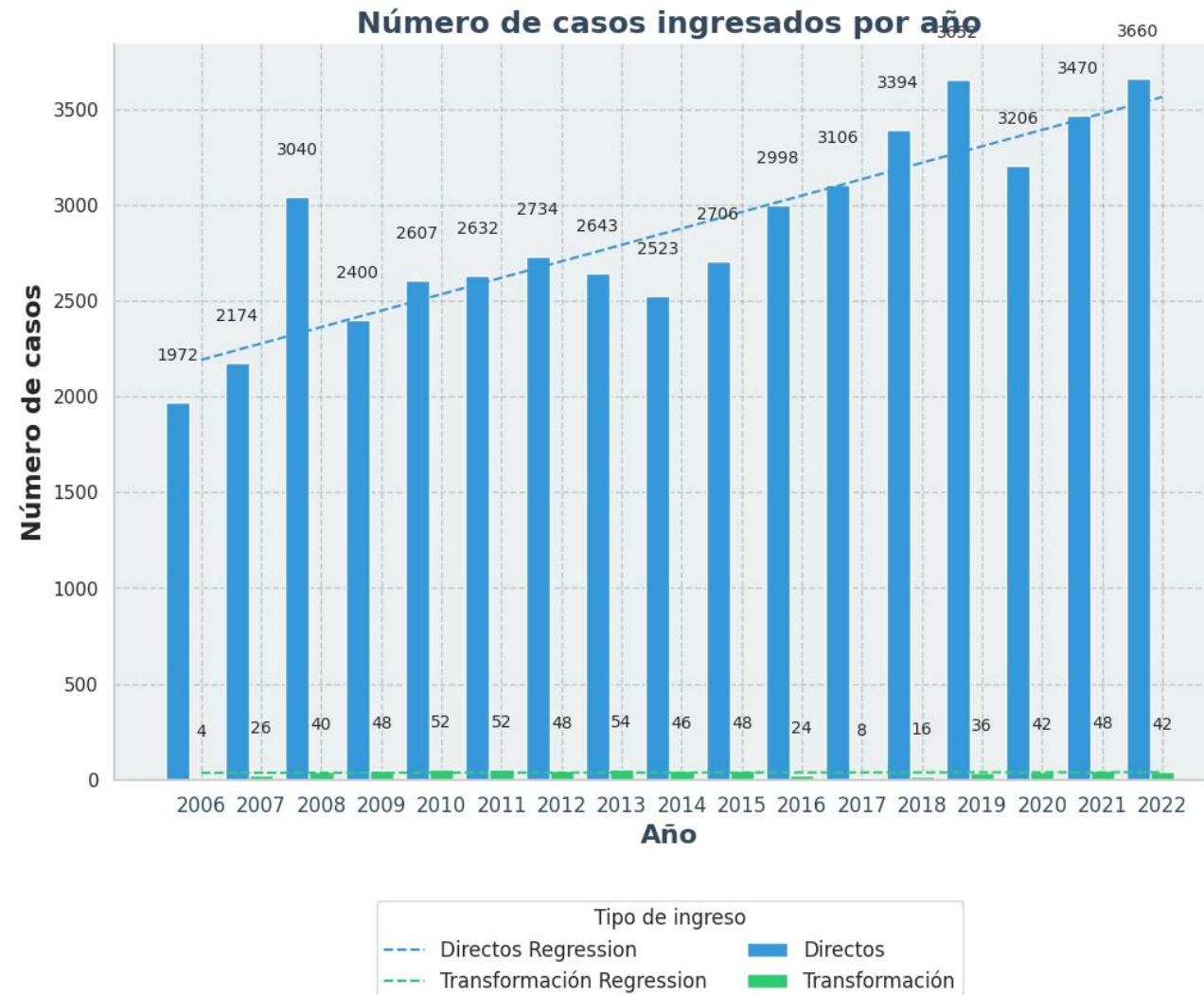
# Añadir una rejilla y ajustar el diseño para evitar solapamientos
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()

# Mostrar la gráfica
plt.show()
```

i. Efficiency analysis

5. Percentage of men that directly went to prison

Output:



ii. Judgement analysis

1. Different procedures

Code:

```
import pandas as pd
from google.colab import auth
import re

# Autenticar en Google Colab
auth.authenticate_user()

# Enlace a tu hoja de cálculo de Google Sheets (asegúrate de que sea compartida públicamente)
gsheet_url = "https://docs.google.com/spreadsheets/d/1ha3bvX2cffUFH45DHbwYpuINDYtPoIZt_8FqnGSHddk/edit?usp=sharing"

# Cargar la hoja de cálculo en un DataFrame de pandas
datos = pd.read_csv(f"{gsheet_url.replace('/edit?usp=sharing', '/export?format=csv')}")

# Obtener la columna 'Summary'
summary_column = datos['Summary']

# Obtener las cuatro palabras siguientes después de "Procedimiento:"
procedimiento_words = summary_column.str.extract(r"Procedimiento: ([^\s]+) ([^\s]+) ([^\s]+) ([^\s]+)", flags=re.IGNORECASE).dropna()

# Crear una lista con las palabras del procedimiento
procedimiento_list = procedimiento_words.apply(lambda x: f"{x[0]} {x[1]} {x[2]} {x[3]}", axis=1).tolist()

# Mostrar la lista de palabras del procedimiento
print(procedimiento_list)
```

ii. Judgement analysis

1. Different procedures

Result:

['Diligencias Urgentes n° 195/15', 'Judici sobre Delitos', 'DILIGENCIAS URGENTES 90/2015 C', 'Diligencias Urgentes n° 175/2015', 'Juicio Inmediato por Delito', 'Diligencias Urgentes/ Juicio rápido', 'Juicio por Delito Leve', 'Diligencias Urgentes/ Juicio rápido', 'DILIGENCIAS URGENTES n° 160/2015', 'DILIGENCIAS URGENTES n° 153/15', 'Diligencias Urgentes n° 139/15', 'Diligencias Urgentes n° 130/15', 'Diligencias Urgentes/ Juicio rápido', 'Diligencias Urgentes n° 127/15', 'Diligencias Urgentes n° 120/2015', 'DILIGENCIAS URGENTES/JUICIO RAPIDO N°', 'Divorcio mutuo acuerdo 17/15', 'Diligencias Urgentes n° 69/15', 'Divorcio n° 49/14 Magistrada-', 'Diligencias Urgentes n° 228/14', 'Modificación Medidas Divorcio mutuo', 'Diligencias Urgentes n° 233/14', 'Diligencias Urgentes n° 221/2014', 'Diligencias Urgentes n° 219/14', 'Diligencias Urgentes n° 216/14', 'Diligencias Urgentes n° 214/14', 'Diligencias Urgentes n° 210/14', 'Diligencias Urgentes n° 202/14', 'Diligencias Urgentes n° 179/14', 'JUICIO DE FALTAS 43/2014', 'Diligencias Urgentes n° 185/14', 'Diligencias Urgentes n° 182/14', 'Diligencias Urgentes n° 169/14', 'Diligencias Urgentes n° 159/14', 'Diligencias Urgentes n° 161/14', 'Diligencias Urgentes n° 153/14', 'Diligencias Urgentes n° 140/14', 'Diligencias Urgentes n° 134/14', 'Diligencias Urgentes n° 133/14', 'Diligencias Urgentes n° 106/14', 'Modificación de Medidas n°', 'Divorcio contencioso n° 23/14', 'Diligencias Urgentes n° 67/14', 'Diligencias Urgentes n° 16/14', 'Diligencias Urgentes n° 15/14', 'Diligencias Urgentes n° 14/14', 'Diligencias Urgentes n° 11/14', 'Guarda y Custodia n°', 'Guarda y Custodia mutuo', 'MODIFICACIÓN DE MEDIDAS N°', 'Diligencias Urgentes/ Juicio rápido', 'Diligencias Urgentes/ Juicio rápido', 'Diligencias Urgentes/ Juicio rápido', 'Diligencias Urgentes n° 165/13']

ii. Judgement analysis

1. Different procedures

Code:

```
# Convertir todas las palabras a minúsculas para evitar problemas de mayúsculas y minúsculas
procedimiento_list_lower = [palabra.lower() for palabra in procedimiento_list]

# Contar la frecuencia de la palabra "urgentes" en la lista (en minúsculas)
frecuencia_urgentes = sum('urgentes' in palabra for palabra in procedimiento_list_lower)

# Mostrar la frecuencia
print(f'La palabra "urgentes" aparece {frecuencia_urgentes} veces en la lista.')

La palabra "urgentes" aparece 42 veces en la lista.

import matplotlib.pyplot as plt
import numpy as np

# Datos de ejemplo (reemplaza esto con tus propios datos)
procedimiento_list = ['Urgentes', 'No Urgentes', 'Urgentes', 'Urgentes', 'No Urgentes']

# Calcular la frecuencia de 'Urgentes'
frecuencia_urgentes = procedimiento_list.count('Urgentes')

# Calcular el porcentaje
porcentaje_urgentes = (frecuencia_urgentes / len(procedimiento_list)) * 100
porcentaje_resto = 100 - porcentaje_urgentes

# Configuración de colores y estilos
colors = plt.cm.Paired(np.arange(2) / 2.0) # Utilizar una paleta de colores más compleja
explode = (0.1, 0) # Separación de la porción "Urgentes" para resaltar

# Configuración del gráfico de pastel
fig, ax = plt.subplots(figsize=(10, 6))
wedges, texts, autotexts = ax.pie([porcentaje_urgentes, porcentaje_resto], labels=['Urgentes', 'Resto'],
                                   autopct=lambda p: '{:.1f}%'.format(p) if p > 0 else '',
                                   colors=colors, startangle=90, explode=explode, wedgeprops=dict(width=0.3, edgecolor='w'))

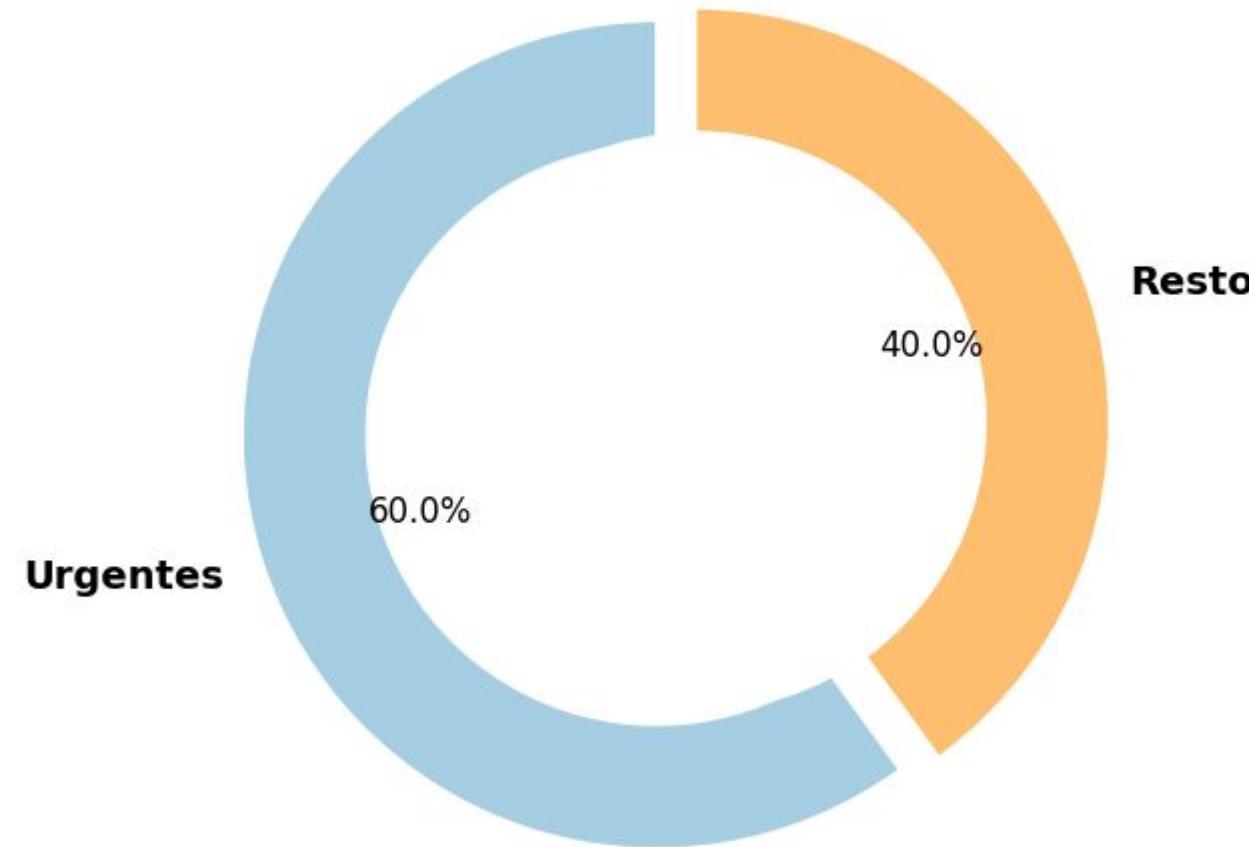
# Personalizar el texto en cada porción
for text, autotext in zip(texts, autotexts):
    text.set_size(14)
    text.set_weight('bold')
    autotext.set_size(12)

# Añadir un círculo blanco en el centro para hacer un donut chart
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# Configuración adicional
plt.title('Porcentaje de "Urgentes" en comparación con el resto de palabras', fontsize=18, fontweight='bold')
plt.axis('equal') # Aspecto circular
plt.show()
```

ii. Judgement analysis
1. Different procedures

Result: Porcentaje de "Urgentes" en comparación con el resto de palabras



ii. Judgement analysis

2. Prosecutor's involvement

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Simular datos (reemplaza esto con tus propios datos)
datos = pd.DataFrame({'Summary': ['Fiscal', 'Resto', 'Fiscal', 'Resto', 'Fiscal']})

# Contar las ocurrencias de la palabra "fiscal" en la columna 'Summary'
frecuencia_fiscal_summary = datos['Summary'].str.lower().str.count("fiscal").sum()

# Calcular el porcentaje
total_filas = datos.shape[0]
porcentaje_fiscal_summary = (frecuencia_fiscal_summary / total_filas) * 100
porcentaje_resto_fiscal_summary = 100 - porcentaje_fiscal_summary

# Configuración de colores y estilos
colors = ['#3498db', '#95a5a6'] # Azul para "Fiscal", Gris para "Resto"
explode = (0.1, 0) # Separación de la porción "Fiscal" para resaltar

# Configuración del gráfico de pastel
fig, ax = plt.subplots(figsize=(10, 6))
wedges, texts, autotexts = ax.pie([porcentaje_fiscal_summary, porcentaje_resto_fiscal_summary],
                                   labels=['Fiscal', 'Resto'],
                                   autopct=lambda p: '{:.1f}%'.format(p) if p > 0 else '',
                                   colors=colors, startangle=90, explode=explode, wedgeprops=dict(width=0.3, edgecolor='w'))

# Personalizar el texto en cada porción
for text, autotext in zip(texts, autotexts):
    text.set_size(14)
    text.set_weight('bold')
    autotext.set_size(12)

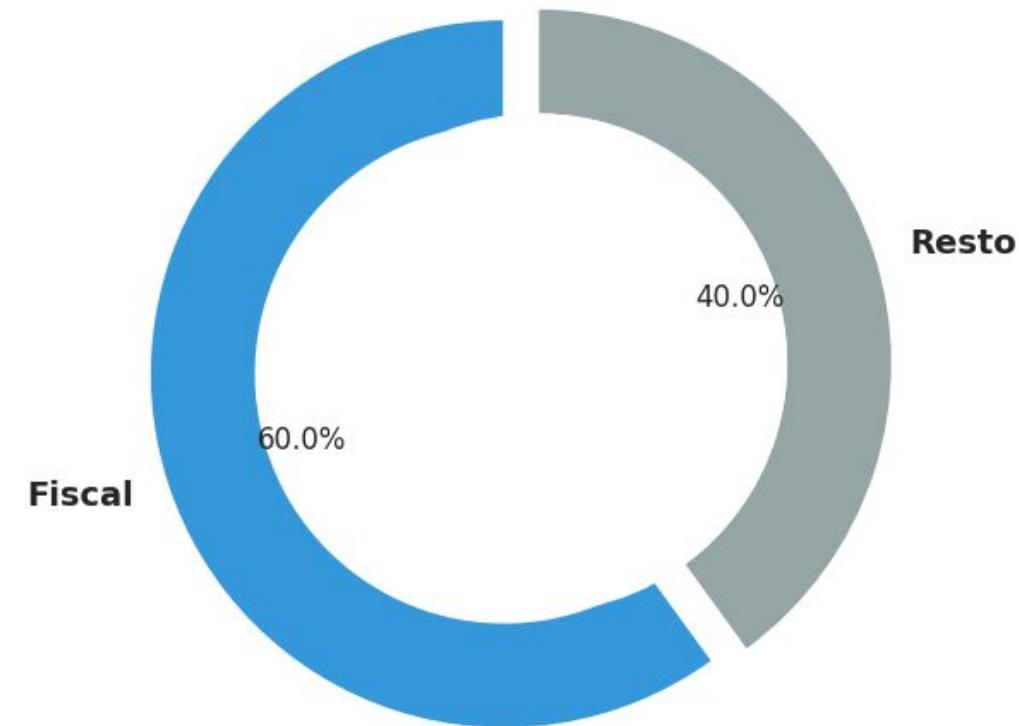
# Añadir un círculo blanco en el centro para hacer un donut chart
centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# Configuración adicional
plt.title('Porcentaje de "Fiscal" en comparación con el resto en la columna Summary', fontsize=18, fontweight='bold')
plt.axis('equal') # Aspecto circular
plt.show()
```

ii. Judgement analysis
2. Prosecutor's involvement

Output:

Porcentaje de "Fiscal" en comparación con el resto en la columna Summary



A wooden gavel with a gold band rests on a dark wooden block. In the background, a brass scale of justice sits on a desk next to several thick books with red and black covers.

THANK YOU!