PEC2: Estudio de técnicas de visualización de datos

Nombre y apellidos: Desiderio Martí Alcaraz
DNI: 52769480M
Correo: dmartial@uoc.edu
Github: https://github.com/desimartiout/vd-pec2

1. Heatmap

Técnica de visualización

- **Nombre**: Heatmap o Mapa de Calor
- **Origen**: Los mapas de calor se originaron en 2D, mostrando los valores en una matriz de datos. Los valores más grandes se representaban en pequeños cuadrados (píxeles) grises o negros y los valores más pequeños en otros cuadrados más claros. El diseñador de software Cormac Kinney registró el término «mapa de calor» en 1991 para representar una muestra de información del mercado financiero en 2D. (fuente https://es.wikipedia.org/wiki/Mapa_de_calor)
- **Descripción**: Un mapa de calor es una representación visual de datos en forma de una matriz en la que los valores individuales están representados como colores. Se utiliza para visualizar la intensidad de una variable en función de otras 2 variables. Los colores más oscuros suelen indicar valores más altos y los mas claros valores más bajos (en función del contexto puede ser al contrario). Existen dos categorías fundamentales de mapas de calor: el mapa de calor de análisis de grupos y el mapa de calor espacial.
- **Ejemplos de Aplicación**:
  - **Ciencia de Datos**: Exploración de correlaciones en un conjunto de datos.
  - **Biología**: Visualización de la expresión genética.

Tipos de datos a representar

- **Tipo de Datos Representados**: Se utiliza para visualizar normalmente una variable cuantitativa y representar la relación entre dos variables continuas o discretas
- **Estructura de Datos**: Es una **matriz bidimensional** donde cada celda de la matriz representa un valor en la intersección de una fila y una columna.
- **Limitaciones**:
  - **Tamaño del Conjunto de Datos**: Los mapas de calor son más efectivos con conjuntos de datos de tamaño moderado a grande para revelar patrones significativos. También hay que tener cuidado con los volúmenes de datos de las variables dependientes en el caso de que sean valores discretos (como es en nuestro caso)

Origen de datos elegido

- **Repositorio de datos elegido**: The Home of the U.S. Government's Open Data
- **Url**: https://data.gov/ (https://data.gov/)
- **Dataset elegido**: Crime Data from 2020 to Present - City of Los Angeles
- **Descripción**: Este conjunto de datos refleja incidentes delictivos en la ciudad de Los Ángeles que se remontan a 2020. Estos datos se transcriben de informes de delitos originales que están escritos a máquina en papel.
- **Url info dataset**: https://catalog.data.gov/dataset/crime-data-from-2020-to-present (https://catalog.data.gov/dataset/crime-data-from-2020-to-present)
- **Url dataset**: https://data.lacity.org/api/views/2nrs-mtv8/rows.csv?accessType=DOWNLOAD (https://data.lacity.org/api/views/2nrs-mtv8/rows.csv?accessType=DOWNLOAD)
- **Publicador**: data.lacity.org
- **Licencia**: http://creativecommons.org/publicdomain/zero/1.0/legalcode (http://creativecommons.org/publicdomain/zero/1.0/legalcode)

En el siguiente código se obtienen los datos del dataset y se muestra una serie de información de utilidad para conocer por ejemplo el número de registros, número de columnas y otros datos similares de utilidad a la hora de conocer el dataset y poder procesar los datos del mismo.

In [1]:
```python
import os
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Ruta al archivo CSV
archivo_csv = os.path.join('datasets', 'Crime_Data_from_2020_to_Present.csv')

# Cargar datos desde el archivo CSV
df_original = pd.read_csv(archivo_csv, sep=',')
```

Transformaciónes realizadas

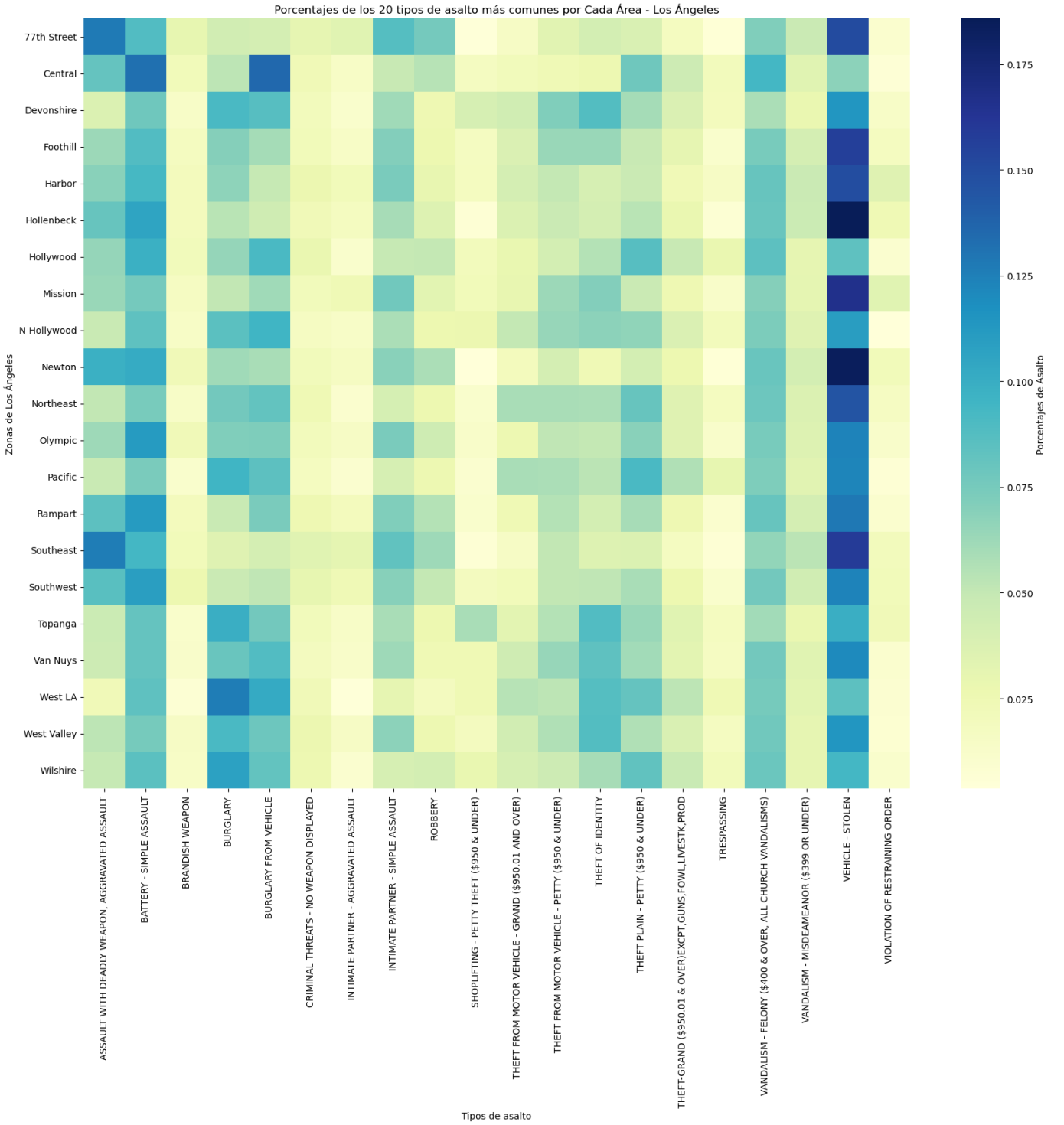
Como el dataset es muy grande y tiene muchos tipos de asaltos nos vamos a centrar en los 20 tipos de asalto más comunes. Para ello primero calculamos los tipos de asalto más comunes y filtramos aquellos registros dentro de nuestro dataset que tienen ese tipo de asalto. Con esto conseguimos tener un mapa de calor con menos registros y más manejable.

In [2]:
```python
# Contar la frecuencia de cada tipo de producto
frecuencia_tipos = df_original['Crm Cd Desc'].value_counts()
# print(frecuencia_tipos)

# Seleccionar los 20 tipos de productos más comunes
tipos_mas_comunes = frecuencia_tipos.head(20).index
# print(tipos_mas_comunes)

# Filtrar el DataFrame para quedarte solo con los 20 tipos de productos más comunes
df_top_20 = df_original[df_original['Crm Cd Desc'].isin(tipos_mas_comunes)]

df_top_20.to_csv('./datasets/Crime_Data_from_2020_to_Present_reducido_top20.csv', index=False)
# print(df_top_20.head())
# print(df_top_20.describe())
# print(df_top_20.shape)
# print(df_top_20.columns)
```

PD: se ha reducido el dataset a 399000 registros por la limitación de Github a no soportar ficheros de más de 100MB

Representación de los datos

In [3]:
```
probabilidad_asalto = df_top_20.groupby(['AREA NAME', 'Crm Cd Desc']).size() / df_top_20.groupby('AREA NAME'
probabilidad_asalto = probabilidad_asalto.unstack().fillna(0)

plt.figure(figsize=(20, 15))
ax = sns.heatmap(probabilidad_asalto, cmap='YlGnBu', annot=False, cbar_kws={'label': 'Porcentajes de Asalto'

plt.title('Porcentajes de los 20 tipos de asalto más comunes por Cada Área - Los Ángeles')

ax.set_xlabel("Tipos de asalto")
ax.set_ylabel("Zonas de Los Ángeles")

plt.show()
```



Comentarios sobre la representación

En este gráfico podemos observar de una forma muy rápida varias conclusiones:

- Que el robo de vehículos es el tipo de asalto más común casi en todas las áreas.
- Que hay áreas dentro de los ángeles donde prácticamente no se producen asaltos (Wilshire) y otras donde se producen más asaltos.

- Podemos hacer un mapa de las áreas donde se producen más asaltos y donde se producen menos así como los tipos de asaltos mñas comunes y menos comunes.

2. Sunburst chart

Técnica de visualización

- **Nombre**: Sunburst Chart (Gráfico Sunburst).
- **Origen**: Los gráficos Sunburst son una evolución de los gráficos de anillo y se han popularizado en la visualización de datos jerarquicos de forma interactiva.
- **Descripción**: Un gráfico Sunburst es una representación visual de datos jerárquicos que se organiza en anillos concéntricos. Cada anillo representa un nivel jerárquico, y las secciones en cada anillo se dividen proporcionalmente según los valores que representan. La estructura se asemeja a los anillos de un árbol, y la forma general del gráfico se parece a un sol (sunburst en inglés), de ahí su nombre. Estos gráficos son particularmente útiles para mostrar la descomposición de un todo en sus partes.
- **Ejemplos de Aplicación**:
  - **Organización Empresarial**: Visualización de la estructura jerárquica de una empresa.
  - **Desglose de Gastos**: Representación de cómo se distribuyen los gastos en un presupuesto.

Tipos de datos a representar

- **Tipo de Datos Representados**: Datos jerárquicos. Muestra la relación entre un elemento principal y sus subelementos.
- **Estructura de Datos**: La estructura de datos se organiza jerárquicamente. Cada nivel jerárquico se representa en un anillo, y las secciones de cada anillo representan las subcategorías o subdivisiones de la categoría superior.
- **Limitaciones**:
  - **Complejidad**: Puede volverse difícil de interpretar con muchas capas o niveles.
  - **Espacio**: Requiere espacio suficiente para mostrar todos los niveles de manera clara.

Origen de datos elegido

- **Repositorio de datos elegido**: Portal datos abiertos del Gobierno de España
- **Url**: https://datos.gob.es/es (https://datos.gob.es/es)
- **Dataset elegido**: Ocupados por situación profesional, sexo y sector económico, por comunidad autónoma. EPA (Identificador API: 4018)
- **Descripción**: Tabla de INEbase Ocupados por situación profesional, sexo y sector económico, por comunidad autónoma. Trimestral. Comunidades y Ciudades Autónomas. Encuesta de Población Activa (EPA)
- **Url info dataset**: (https://datos.gob.es/es/catalogo/ea0010587-ocupados-por-situacion-profesional-sexo-y-sector-economico-por-comunidad-autonoma-epa-identificador-api-4018)
- **Url dataset**: https://www.ine.es/jaxiT3/files/t/csv_bdsc/4018.csv (https://www.ine.es/jaxiT3/files/t/csv_bdsc/4018.csv)
- **Publicador**: Instituto Nacional de Estadística (MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL)
- **Licencia**: https://www.ine.es/aviso_legal (https://www.ine.es/aviso_legal)

En el siguiente código se obtienen los datos del dataset y se muestra una serie de información de utilidad para conocer por ejemplo el número de registros, número de columnas y otros datos similares de utilidad a la hora de conocer el dataset y poder procesar los datos del mismo.

In [4]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import os

# Ruta al archivo CSV
archivo_csv = os.path.join('datasets', '4018.csv')

# Cargar datos desde el archivo CSV
df = pd.read_csv(archivo_csv, sep=';')
# print(df.head())
# print(df.describe())
# print(df.shape)
# print(df.columns)
```

Transformaciónes realizadas

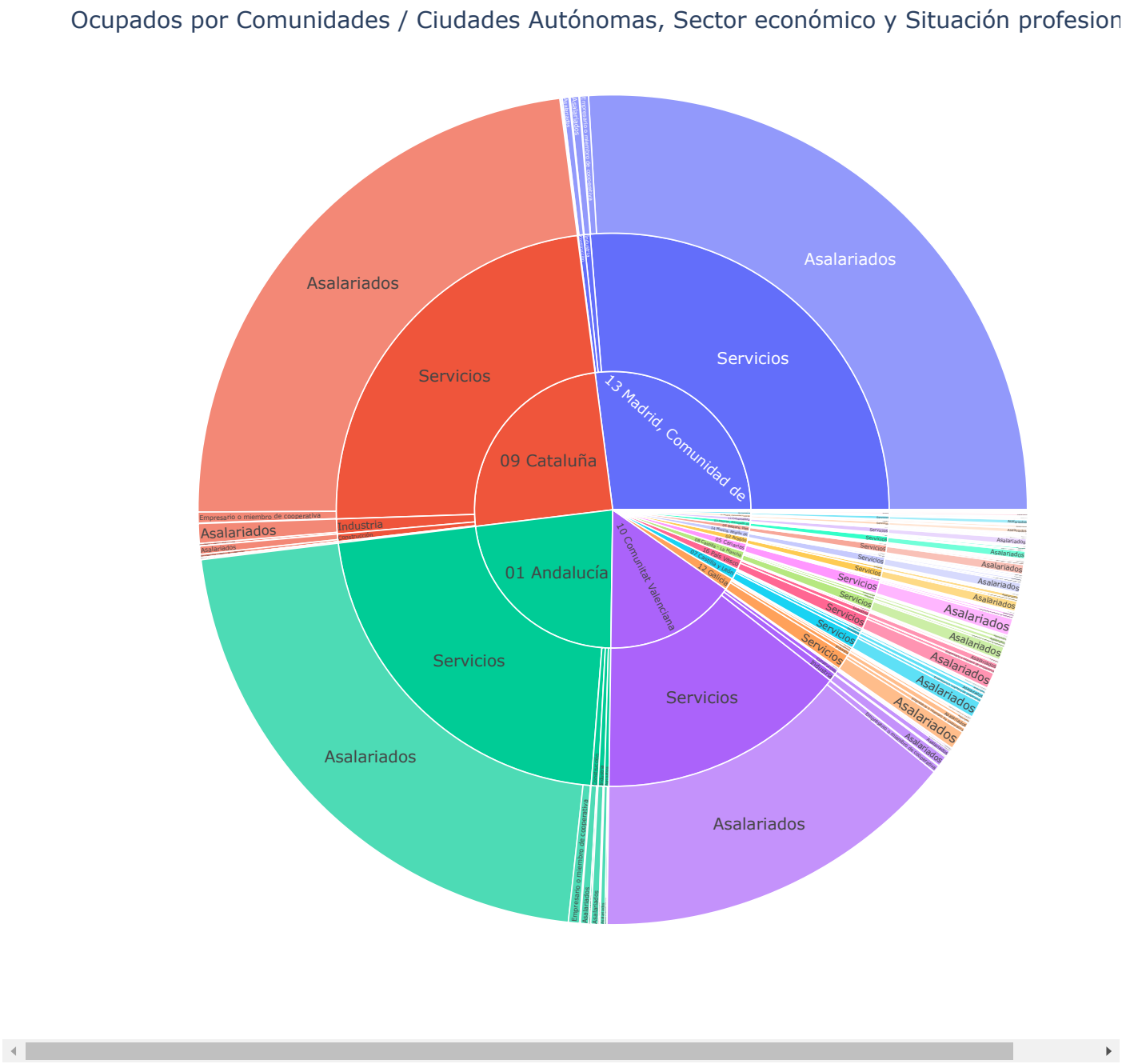
Como hay mucha información de totales que no nos aporta para el estudio que vamos a realizar se han realizado estas transformaciones:

- Eliminación de registros sin valores en columna Total
- Conversión de columna Total a numérica
- Eliminamos registros que muestran totales nacionales, de sectores y situaciones.
- Filtramos solo los registros del periodo 2008T4 (Trimestre 4 del 2008)

In [5]:
```
# Cambiamos los valores .. por NA
# df['Total'] = df['Total'].replace('..', pd.NA)
df['Total'] = df['Total'].replace({',': '.', '\.': ''}, regex=True).replace('..', pd.NA)

# Eliminamos las filas con valores NaN en la columna 'Total'
df = df.dropna(subset=['Total'])
# print(df.head())

# Convertimos la columna 'Total' a tipo numérico
df['Total_numeric'] = pd.to_numeric(df['Total'], errors='coerce')
# df['Total_numeric'] = pd.to_numeric(df['Total'].replace('..', pd.NA), errors='coerce')

# Eliminamos los totales de todas las columnas a visualizar
df_filtrado = df[(df['Comunidades y Ciudades Autónomas'] != 'Total Nacional') & (df['Sector económico'] != '
# print(df_filtrado.head())

# Filtramos solo por el periodo concreto y para ambos sexos
df_filtrado = df_filtrado[(df_filtrado['Periodo'] == '2008T4') & (df_filtrado['Sexo'] == 'Ambos sexos')]
# print(df_filtrado.head())
```

Representación de los datos

In [6]:
```
# Crear el gráfico sunburst
fig = px.sunburst(
    df_filtrado,
    path=['Comunidades y Ciudades Autónomas', 'Sector económico', 'Situación profesional'],
    values='Total_numeric',
    title='Ocupados por Comunidades / Ciudades Autónomas, Sector económico y Situación profesional (2008T4)',
    width=900,
    height=800
)

# Mostrar el gráfico
fig.show()
```



Comentarios sobre la representación

En este gráfico podemos observar de una forma muy rápida varias conclusiones sobre el periodo 2008T4:

- Que la comunicadad con más ocupados en ese periodo fué Madrid, seguida por Cataluña, Andalucia y Comundad Valenciana.
- Que mayoritariamente el sector servicios es el que más ocupa a empleados, con muchísima diferencia con respecto al resto de sectores.
- Que la mayoría de empleados se hace a través de contrados asalariados.

Con un gráfico interactivo podríamos hacer drill down para ver mejor los detalles de comunidades concretas seleccionando las

3. Horizon graph

Técnica de visualización

- **Nombre**: Horizon Plot o Gráfico de Horizonte.
- **Origen**: La técnica de Horizon Plot se originó en la visualización de series temporales y fue introducida por primera vez por Heer y Agrawala en 2009.
- **Descripción**: Un Horizon Plot es una forma de visualizar series temporales comprimiendo la información en una visualización más compacta. Se divide el eje vertical en múltiples bandas (horizontes), cada una representando una porción de los datos. Cada banda se colorea para indicar la dirección y la magnitud del cambio en los datos en ese período de tiempo. El objetivo es proporcionar una visión general de la variabilidad y tendencia temporal de los datos.
- **Ejemplos de Aplicación**:
  - **Finanzas**: Seguimiento de la variación de precios de acciones a lo largo del tiempo.
  - **Meteorología**: Visualización de patrones climáticos a lo largo de los años.
    >

Tipos de datos a representar

- **Tipo de Datos Representados** : Principalmente datos cuantitativos y temporales. Adecuado para series temporales continuas.
- **Estructura de Datos** : Se requieren datos temporales unidimensionales. Cada punto de datos debe tener una marca de tiempo asociada.
- **Limitaciones** :
  - **Cantidad de Datos** : A medida que aumenta el número de series la visualización puede volverse saturada y difícil de interpretar.
  - **Requiere Datos Temporales** : Es más eficaz para datos temporales y puede no ser tan útil para otros tipos de datos.

Origen de datos elegido

- **Repositorio de datos elegido**: Portal de Datos Abiertos de la Generalitat Valenciana
- **Url**: https://portaldadesobertes.gva.es/es (https://portaldadesobertes.gva.es/es)
- **Dataset elegido**: COVID-19 Serie de casos con PDIA positiva en la Comunitat Valenciana, según fecha en la que el laboratorio notifica el diagnóstico
- **Descripción**: Información del número de casos de COVID-19 en la Comunidad Valenciana, tanto información agrupada por provincia como detallada a nivel de departamentos de salud.
- **Url info dataset**: https://dadesobertes.gva.es/es/dataset/covid-19-series-casos-pdia-positiva/resource/cb50e7d2-0c0e-46b8-a359-a0fa35998577 (https://dadesobertes.gva.es/es/dataset/covid-19-series-casos-pdia-positiva/resource/cb50e7d2-0c0e-46b8-a359-a0fa35998577)
- **Url dataset**: https://dadesobertes.gva.es/dataset/ce195af2-39ec-4f44-bb77-b14235519b0d/resource/cb50e7d2-0c0e-46b8-a359-a0fa35998577/download/covid-19-serie-de-casos-con-pdia-positiva-en-la-comunitat-valenciana.csv (https://dadesobertes.gva.es/dataset/ce195af2-39ec-4f44-bb77-b14235519b0d/resource/cb50e7d2-0c0e-46b8-a359-a0fa35998577/download/covid-19-serie-de-casos-con-pdia-positiva-en-la-comunitat-valenciana.csv)
- **Publicador**: Generalitat Valenciana
- **Licencia**: http://www.opendefinition.org/licenses/cc-by (http://www.opendefinition.org/licenses/cc-by)

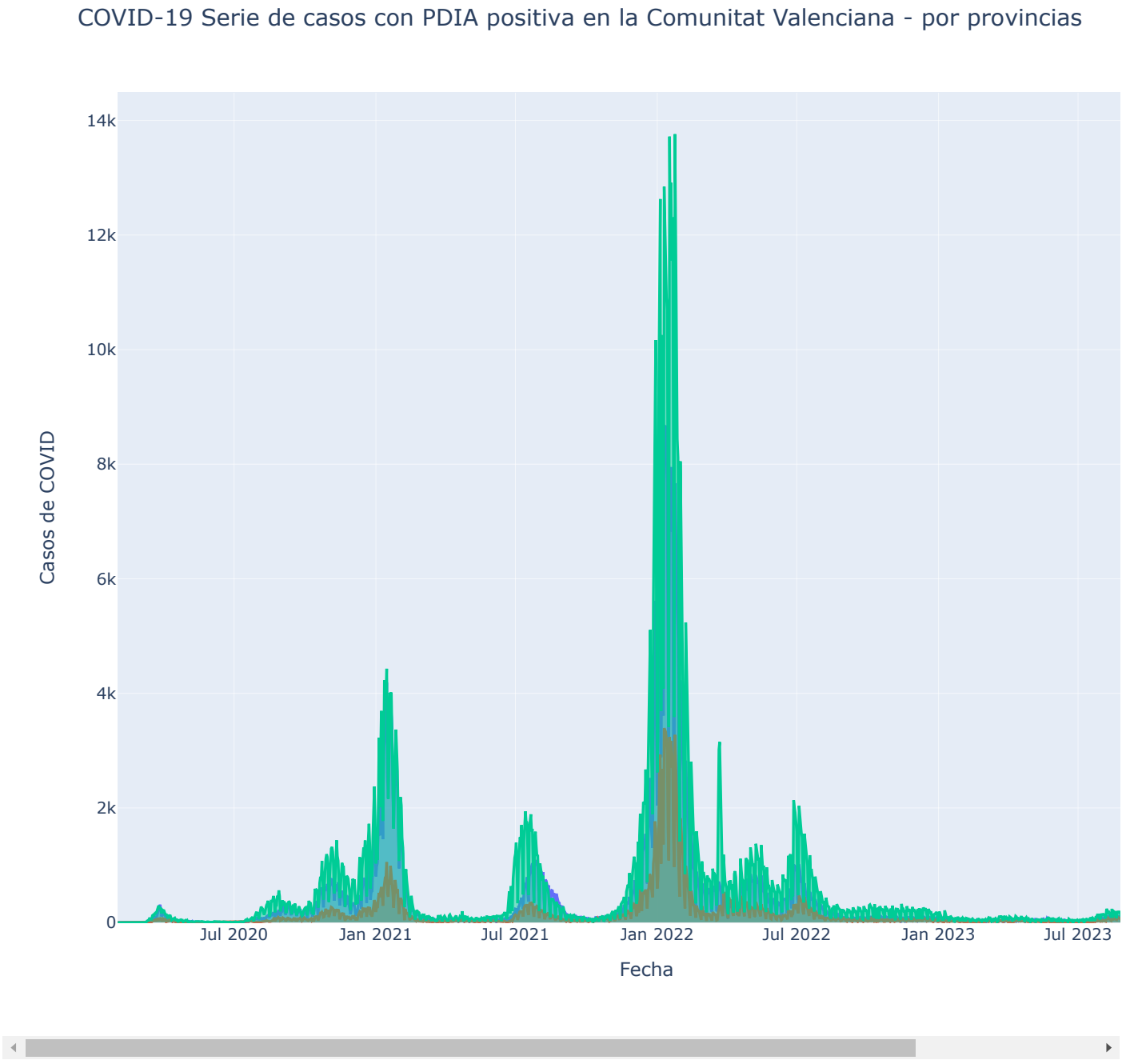
En el siguiente código se obtienen los datos del dataset y se muestra una serie de información de utilidad para conocer por ejemplo el número de registros, número de columnas y otros datos similares de utilidad a la hora de conocer el dataset y poder procesar los datos del mismo.

In [7]:

```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import os

# Constantes para facilitar encontrar las columnas que vamos a estudiar
COLUMNAS = ['Data diagnòstic laboratori/fecha diagnóstico laboratorio', 'C.Valenciana', 'Homes/Hombres','Don
COL_FECHA = 0
COL_ALICANTE = 4
COL_CASTELLON = 5
COL_VALENCIA = 6

# Ruta al archivo CSV
archivo_csv = os.path.join('datasets', 'covid-19-serie-de-casos-con-pdia-positiva-en-la-comunitat-valenciana

# Cargar datos desde el archivo CSV
df = pd.read_csv(archivo_csv, sep=';')
# print(df.head())
# print(df.describe())
# print(df.shape)
# print(df.columns)
```

Representación de los datos

In [8]:
```
# Crear el gráfico de horizonte con plotly
fig = go.Figure()

fig.add_trace(go.Scatter(x=df[COLUMNAS[0]], y=df[COLUMNAS[COL_ALICANTE]], fill='tozeroy', name='Alicante'))
fig.add_trace(go.Scatter(x=df[COLUMNAS[0]], y=df[COLUMNAS[COL_CASTELLON]], fill='tozeroy', name='Castellón'))
fig.add_trace(go.Scatter(x=df[COLUMNAS[0]], y=df[COLUMNAS[COL_VALENCIA]], fill='tozeroy', name='Valencia'))

# Personalizar el diseño
fig.update_layout(
    title='COVID-19 Serie de casos con PDIA positiva en la Comunitat Valenciana - por provincias',
    xaxis_title='Fecha',
    yaxis_title='Casos de COVID',
    width=1000,
    height=800
)

# Mostrar el gráfico
fig.show()
```



Comentarios sobre la representación

En la gráfica se puede ver la evolución en el tiempo de casos detectados de COVID en 3 series superpuestas, cada serie informa los valores agrupados de los centros de salud de cada provincia. Este gráfico se podría mejorar haciendo que la selección de datos fuera interactiva para facilitar la selección de series, identificando los datos incluso a nivel de departamento de salud. No se han incluído los datos a nivel de centro de salud porque en este tipo de gráfino no tendría ningún sentido porque sería imposible idneitifcar cada uno de los valores de las series.