```
!git clone https://github.com/PP-URC/pp2025_1.git
!pip install -U kaleido
!ls pp2025_1/data
```

```
Cloning into 'pp2025_1'...
remote: Enumerating objects: 78, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (59/59), done.
remote: Total 78 (delta 22), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (78/78), 17.22 MiB | 11.95 MiB/s, done.
Resolving deltas: 100% (22/22), done.
Collecting kaleido
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl.metadata (15 kB)
Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 5.5 MB/s eta 0:00:00
Installing collected packages: kaleido
Successfully installed kaleido-0.2.1
delitosXLSX           lines_colors.csv  stations.csv
delitos_xlsx_links.txt  metro_lines.py
```

```
import csv
import os
import requests
import random
from datetime import datetime
import numpy as np
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import holidays
import openpyxl
from scipy.optimize import curve_fit
import kaleido

stations_path = "/content/pp2025_1/data/stations.csv"
lines_colors_path = "pp2025_1/data/lines_colors.csv"

with open(stations_path) as stations_f:
    st_reader = csv.reader(stations_f)
    next(st_reader)
    stations_dict = {}
    for entry in st_reader:
        station_name = entry[1]
        stations_dict.setdefault(station_name, dict())
        stations_dict[station_name]["lat"] = entry[2]
        stations_dict[station_name]["lon"] = entry[3]
        #stations_dict[station_name].setdefault("lines", list()).append(entry[4])
#pprint(stations_dict)


with open(lines_colors_path) as colors_f:
    colors_reader = csv.reader(colors_f)
    next(colors_reader)
    line_colors_dict = dict()
```

```python
    for line_color_data in colors_reader:
        line_colors_dict[f"Linea {line_color_data[0]}"] = line_color_data[2]
```

```python
metro_lines = {
    "Linea 1": [
        "Observatorio", "Tacubaya", "Juanacatlán", "Chapultepec", "Sevilla",
        "Insurgentes", "Cuauhtémoc", "Balderas", "Salto del Agua", "Isabel la Católica",
        "Pino Suárez", "Merced", "Candelaria", "San Lázaro", "Moctezuma", "Balbuena",
        "Boulevard Pto. Aéreo", "Gómez Farías", "Zaragoza", "Pantitlán"
    ],
    "Linea 2": [
        "Cuatro Caminos", "Panteones", "Tacuba", "Cuitláhuac", "Popotla", "Colegio Militar",
        "Normal", "San Cosme", "Revolución", "Hidalgo", "Bellas Artes", "Allende",
        "Zócalo", "Pino Suárez", "San Antonio Abad", "Chabacano", "Viaducto", "Xola",
        "Villa de Cortés", "Nativitas", "Portales", "Ermita", "General Anaya", "Tasqueña"
    ],
    "Linea 3": [
        "Indios Verdes", "Deportivo 18 de Marzo", "Potrero", "La Raza", "Tlatelolco", "Guerrero",
        "Hidalgo", "Juárez", "Balderas", "Niños Héroes", "Hospital General", "Centro Médico",
        "Etiopía / Plaza de la Transparencia", "Eugenia", "División del Norte", "Zapata",
        "Coyoacán", "Viveros / Derechos Humanos", "Miguel Ángel de Quevedo", "Copilco", "Universidad"
    ],
    "Linea 4": [
        "Martín Carrera", "Talismán", "Bondojito", "Consulado", "Canal del Norte", "Morelos", "Candelaria",
        "Fray Servando", "Jamaica", "Santa Anita"
    ],
    "Linea 5": [
        "Pantitlán", "Hangares", "Terminal Aérea", "Oceanía", "Aragón", "Eduardo Molina",
        "Consulado", "Valle Gómez", "Misterios", "La Raza", "Autobuses del Norte",
        "Instituto del Petróleo", "Politécnico"
    ],
    "Linea 6": [
        "El Rosario", "Tezozómoc", "Azcapotzalco", "Ferrería", "Norte 45",
        "Vallejo", "Instituto del Petróleo", "Lindavista", "Deportivo 18 de Marzo",
        "La Villa-Basílica", "Martín Carrera"
    ],
    "Linea 7": [
        "El Rosario", "Aquiles Serdán", "Camarones", "Refinería", "Tacuba", "San Joaquín",
        "Polanco", "Auditorio", "Constituyentes", "Tacubaya", "San Pedro de los Pinos",
        "San Antonio", "Mixcoac", "Barranca del Muerto"
    ],
    "Linea 8": [
        "Garibaldi", "Bellas Artes", "San Juan de Letrán", "Salto del Agua", "Doctores",
        "Obrera", "Chabacano", "La Viga", "Santa Anita", "Coyuya", "Iztacalco", "Apatlaco",
        "Aculco", "Escuadrón 201", "Atlalilco", "Iztapalapa", "Cerro de la Estrella",
        "UAM-I", "Constitución de 1917"
    ],
    "Linea 9": [
        "Tacubaya", "Patriotismo", "Chilpancingo", "Centro Médico", "Lázaro Cárdenas", "Chabacano",
        "Jamaica", "Mixiuhca", "Velódromo", "Ciudad Deportiva", "Puebla", "Pantitlán"
    ],
    "Linea A": [
        "Pantitlán", "Agrícola Oriental", "Canal de San Juan", "Tepalcates", "Guelatao",
        "Peñón Viejo", "Acatitla", "Santa Marta", "Los Reyes", "La Paz"
    ],
    "Linea B": [
        "Buenavista", "Guerrero", "Garibaldi", "Lagunilla", "Tepito", "Morelos",
        "San Lázaro", "Ricardo Flores Magón", "Romero Rubio", "Oceanía", "Deportivo Oceanía",
        "Bosque de Aragón", "Villa de Aragón", "Nezahualcóyotl", "Impulsora", "Río de los Remedios",
        "Múzquiz", "Ecatepec", "Olímpica", "Plaza Aragón", "Ciudad Azteca"
    ],
    "Linea 12": [
        "Mixcoac", "Insurgentes Sur", "Hospital 20 de Noviembre", "Zapata", "Parque de los Venados",
        "Eje Central", "Ermita", "Mexicaltzingo", "Atlalilco", "Culhuacán", "San Andrés Tomatlán",
        "Lomas Estrella", "Calle 11", "Periférico Oriente", "Tezonco", "Olivos", "Nopalera",
        "Zapotitlán", "Tlaltenco", "Tláhuac"
    ]
}
```

```
%%capture
%%bash

rm /etc/resolv.conf
echo "nameserver 8.8.8.8" > /etc/resolv.conf
echo "nameserver 8.8.4.4" >> /etc/resolv.conf
cat /etc/resolv.conf
```

```
#https://www.fgjcdmx.gob.mx/transparencia/incidencia-delictiva

report_links = [
    "https://www.fgjcdmx.gob.mx/storage/app/media/Transparencia/Incidencia%20Delictiva%202020/1A%20DELITOS%20DE%20ALTO%20IMPACTO%202020.xlsx",
    "https://www.fgjcdmx.gob.mx/storage/app/media/Transparencia/Incidencia%20delictiva%202021/1A%20DELITOS%20DE%20ALTO%20IMPACTO%202021.xlsx",
    "https://www.fgjcdmx.gob.mx/storage/app/media/Transparencia/Incidencia%20Delictiva%202022/1A%20DELITOS%20DE%20ALTO%20IMPACTO%202022.xlsx",
    "https://www.fgjcdmx.gob.mx/storage/app/media/Transparencia/Incidencia%20Delictiva%202023/diciembre/1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx",
    "https://www.fgjcdmx.gob.mx/storage/app/media/Transparencia/Incidencia%20Delictiva%202024/Diciembre/1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx",
    "https://www.fgjcdmx.gob.mx/storage/app/media/Transparencia/Incidencia%20Delictiva%202025/Marzo/1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx"
]
try:
    reports_path = "/content/reportsXLSX"
    os.makedirs(reports_path, exist_ok=True)

    for ind, url in enumerate(report_links):
        filename = os.path.basename(url)
        filepath = os.path.join(reports_path, f"(report{ind}){filename}")
        print(f"Downloading: {filename}")
        response = requests.get(url)
        with open(filepath, "wb") as f:
            f.write(response.content)
except Exception as e:
    print(e)
    reports_path = "/content/pp2025_1/data/delitosXLSX"
```

```
Downloading: 1A%20DELITOS%20DE%20ALTO%20IMPACTO%202020.xlsx
Downloading: 1A%20DELITOS%20DE%20ALTO%20IMPACTO%202021.xlsx
Downloading: 1A%20DELITOS%20DE%20ALTO%20IMPACTO%202022.xlsx
Downloading: 1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx
Downloading: 1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx
Downloading: 1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx
```

```
dfs = []
for filename in os.listdir(reports_path):
    file_path = os.path.join(reports_path, filename)
    if os.path.isfile(file_path):
        print(f"Processing file: {filename}")
        print(file_path)
        wb = openpyxl.load_workbook(filename=file_path)

        df = pd.DataFrame(wb.worksheets[0].iter_rows(values_only=True))
        dfs.append(df)
df = pd.concat(dfs)
df.drop(0, inplace=True)
df = df.iloc[:, :-1].copy()
df.columns = df.iloc[0]
df = df.drop(1).copy()
```

```
Processing file: (report3)1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx
/content/reportsXLSX/(report3)1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx
Processing file: (report2)1A%20DELITOS%20DE%20ALTO%20IMPACTO%202022.xlsx
/content/reportsXLSX/(report2)1A%20DELITOS%20DE%20ALTO%20IMPACTO%202022.xlsx
Processing file: (report0)1A%20DELITOS%20DE%20ALTO%20IMPACTO%202020.xlsx
/content/reportsXLSX/(report0)1A%20DELITOS%20DE%20ALTO%20IMPACTO%202020.xlsx
Processing file: (report4)1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx
/content/reportsXLSX/(report4)1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx
Processing file: (report1)1A%20DELITOS%20DE%20ALTO%20IMPACTO%202021.xlsx
/content/reportsXLSX/(report1)1A%20DELITOS%20DE%20ALTO%20IMPACTO%202021.xlsx
Processing file: (report5)1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx
/content/reportsXLSX/(report5)1A%20DELITOS%20DE%20ALTO%20IMPACTO.xlsx
```

```
df_m = df[df['MODALIDAD - DELITO'].str.contains("metro", case=False)].copy()
df_m['FECHA DE LOS HECHOS'] = pd.to_datetime(df_m['FECHA DE LOS HECHOS'], dayfirst=True)
df_m['HORA DE LOS HECHOS'] = pd.to_datetime(df_m['HORA DE LOS HECHOS'], dayfirst=True)
df_m.loc[:,"month"] = df_m["FECHA DE LOS HECHOS"].dt.month
df_m.loc[:,"year"] = df_m["FECHA DE LOS HECHOS"].dt.year
df_m.loc[:,"day"] = df_m["FECHA DE LOS HECHOS"].dt.day
df_m.loc[:,"dayofweek"] = df_m["FECHA DE LOS HECHOS"].dt.dayofweek
df_m["Hour"] = df_m['HORA DE LOS HECHOS'].dt.hour
df_m = df_m[df_m.year >= 2020].copy()
```

```
<ipython-input-7-937f23d1b574>:3: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To ensure parsing is consistent and as-expected, ple
  df_m['HORA DE LOS HECHOS'] = pd.to_datetime(df_m['HORA DE LOS HECHOS'], dayfirst=True)
```

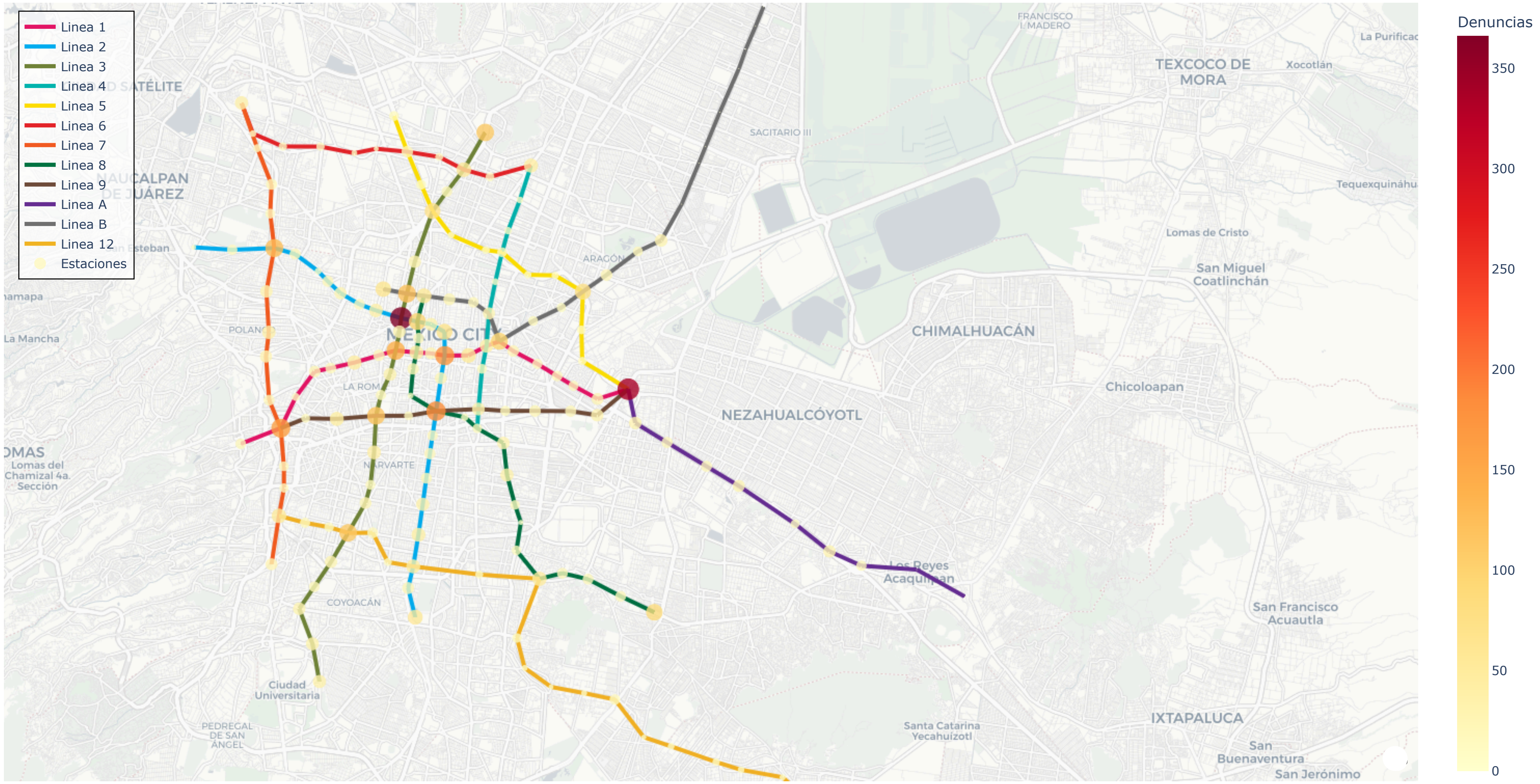
```
df_stations = pd.DataFrame([st, dat["lon"], dat["lat"]] for st, dat in stations_dict.items())
df_stations.columns = ["Station", 'Longitude', 'Latitude']
print(df_stations.dtypes)
df_stations['Longitude'] = pd.to_numeric(df_stations['Longitude'], errors='coerce')
df_stations['Latitude'] = pd.to_numeric(df_stations['Latitude'], errors='coerce')
df_m['Longitude'] = pd.to_numeric(df_m['COORD X'], errors='coerce')
df_m['Latitude'] = pd.to_numeric(df_m['COORD Y'], errors='coerce')
print(df_stations.dtypes)
```

```
Station      object
Longitude    object
Latitude     object
dtype: object
Station       object
Longitude    float64
Latitude     float64
```

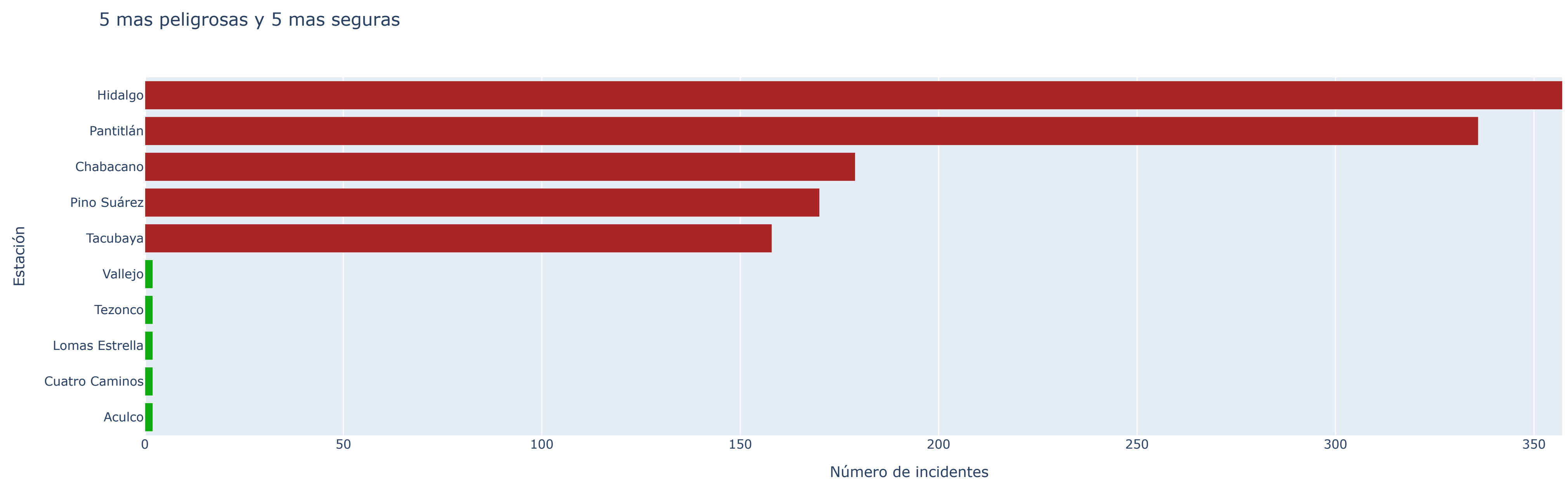
```
dtype: object
```

```
stations_coords = df_stations[['Latitude', 'Longitude']].to_numpy()
df_m = df_m.dropna(subset=["Longitude", "Latitude"]).copy()
# Incident coords as a 2D array: shape (num_incidents, 2)
incidents_coords = df_m[['Latitude', 'Longitude']].to_numpy()
# Expand dimensions to broadcast subtraction
# incidents_coords[:, None, :] shape: (num_incidents, 1, 2)
# stations_coords[None, :, :] shape: (1, num_stations, 2)
diff = incidents_coords[:, None, :] - stations_coords[None, :, :]  # shape: (num_incidents, num_stations, 2)

# Euclidean distance
dists = np.sqrt(np.sum(diff ** 2, axis=2))  # shape: (num_incidents, num_stations)
closest_station_indices = np.argmin(dists, axis=1)
df_m['closest_station'] = df_stations.iloc[closest_station_indices]['Station'].values
df_m['distance_to_station'] = dists[np.arange(len(df_m)), closest_station_indices]
```

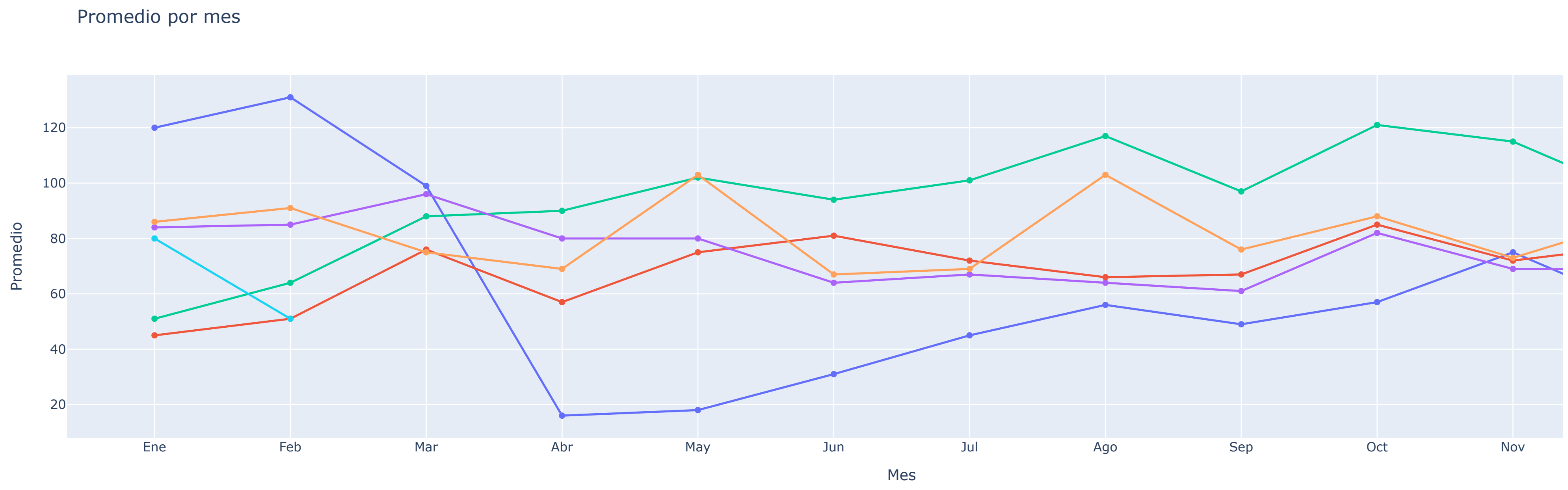
```
crimes_df = df_m.dropna(subset=["Longitude", "Latitude"]).groupby("closest_station").count().iloc[:,0]
crimes_df = crimes_df.reset_index()
crimes_df.columns = ["Station", "Crimes"]
df_st_crimes = pd.merge(df_stations, crimes_df, on="Station", how="left").fillna({"Crimes": 0})
df_st_crimes["Crimes"] = df_st_crimes["Crimes"].astype(int)
df_st_crimes.Crimes = df_st_crimes.Crimes.astype(int)

fig = go.Figure()
for line, stations in metro_lines.items():
    line_color = line_colors_dict[line]
    lons = []
    lats = []

    for start_station, end_station in zip(stations[:-1], stations[1:]):
        start_coords = stations_dict[start_station]
        end_coords = stations_dict[end_station]

        lons += [start_coords['lon'], end_coords['lon'], None]
        lats += [start_coords['lat'], end_coords['lat'], None]


    fig.add_trace(go.Scattermapbox(
        mode="lines",
        lon=lons,
        lat=lats,
        line=dict(width=4, color=line_color),
        hoverinfo='text',
        name=line,
        showlegend=True
    ))
marker_sizes = np.log1p(df_st_crimes["Crimes"])
marker_sizes = (marker_sizes - marker_sizes.min()) / (marker_sizes.max() - marker_sizes.min())

# Scale to reasonable marker sizes: 8 (min) to 25 (max)
marker_sizes = marker_sizes * 20 + 2
fig.add_trace(go.Scattermapbox(
    mode="markers",
    lon=df_st_crimes["Longitude"],
    lat=df_st_crimes["Latitude"],
    marker=dict(
        size=marker_sizes,
        color=df_st_crimes["Crimes"],
        colorscale="YlOrRd",
        showscale=True,
        colorbar=dict(title="Denuncias"),
        opacity=0.8
    ),
    text = df_st_crimes["Station"] + "<br>Denuncias: " + df_st_crimes["Crimes"].astype(str),
    hoverinfo="text",
    name="Estaciones"
))

fig.update_layout(
    mapbox_style="carto-positron",
    mapbox_zoom=11,
    mapbox_center={"lat": 19.43, "lon": -99.13},
    height=800,
    margin=dict(l=10, r=10, t=40, b=10),
    coloraxis_colorbar=dict(x=0.95, y=0.5, len=0.75, title="Delitos"),
    legend=dict(x=0.01, y=0.99, bgcolor='rgba(255,255,255,0.7)', bordercolor='black', borderwidth=1)
)

fig.show()

if not os.path.exists("images"):
    os.mkdir("images")
fig.write_image("images/map.png")
```



```
most_dangerous = crimes_df.sort_values(by="Crimes", ascending=False).head(5)
safest = crimes_df.sort_values(by="Crimes", ascending=True).head(5)
combined = pd.concat([
    most_dangerous.assign(Category="Mas peligrosas"),
    safest.assign(Category="Seguras")
])

fig = px.bar(
    combined,
    x="Crimes",
    y="Station",
    color="Category",
    orientation="h",
    title="5 mas peligrosas y 5 mas seguras",
    labels={"Crimes": "Número de incidentes", "Station": "Estación"},
    color_discrete_map={
        "Mas peligrosas": "#AA2626",  # Use color previously for "Seguras"
        "Seguras": "#10AA10"
    }
)
fig.update_layout(yaxis={'categoryorder':'total ascending'})
fig.show()
fig.write_image("images/top5.png")
```



```
monthly_data = df_m[["year", "month", "ID"]].groupby(["year", "month"]).count()
monthly_data = monthly_data.rename(columns={'ID':'count'})
monthly_pivot = monthly_data['count'].unstack(level=0)
fig = go.Figure()

# Loop through each year (column) to add a line
for year in monthly_pivot.columns:
    fig.add_trace(go.Scatter(
        x=monthly_pivot.index,
        y=monthly_pivot[year],
        mode='lines+markers',
        name=str(year)  # Custom legend label
    ))

# Customize layout
fig.update_layout(
    title="Promedio por mes",
    xaxis_title="Mes",
    yaxis_title="Promedio",
    xaxis=dict(tickmode='array', tickvals=list(range(1, 13)), ticktext=[
        'Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun',
        'Jul', 'Ago', 'Sep', 'Oct', 'Nov', 'Dic'
    ]),
    legend_title="Año"
)

fig.show()
fig.write_image("images/promedio_año_mes.png")
```

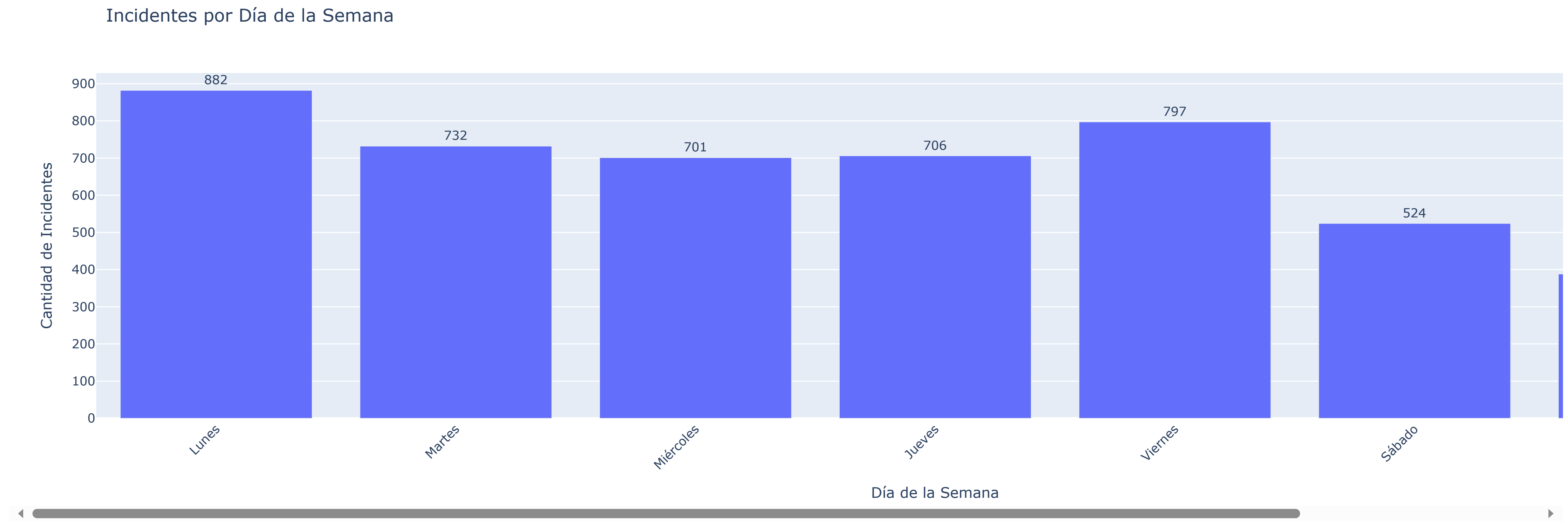


```
week_df = df_m.groupby("dayofweek").count().iloc[:, 1].copy()
week_df.index = week_df.index.map({
    0: "Lunes",
    1: "Martes",
    2: "Miércoles",
    3: "Jueves",
    4: "Viernes",
    5: "Sábado",
    6: "Domingo"
})
```

```
week_df
```

```
                ID_CI
dayofweek
   Lunes         882
  Martes         732
Miércoles        701
  Jueves         706
 Viernes         797
  Sábado         524
 Domingo         388

dtype: int64
```

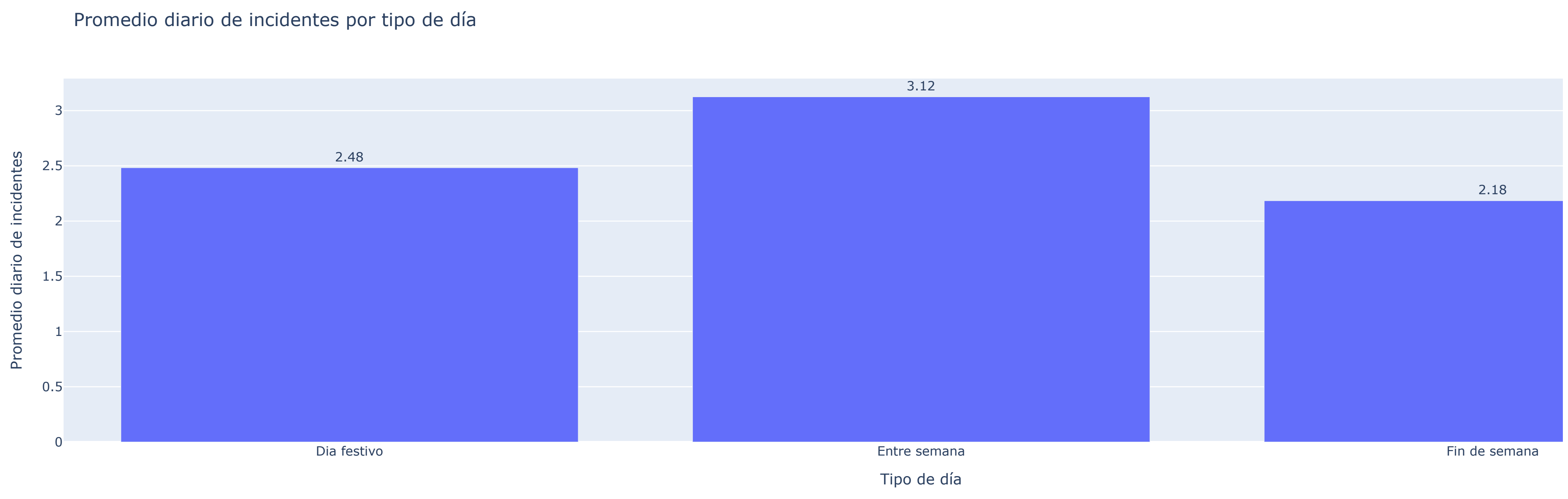
```python
week_df = week_df.reset_index()
week_df.columns = ['Día', 'Cantidad']
fig = px.bar(week_df, x='Día', y='Cantidad',
             title='Incidentes por Día de la Semana',
             labels={'Día': 'Día de la Semana', 'Cantidad': 'Cantidad de Incidentes'},
             text='Cantidad')

fig.update_traces(textposition='outside')
fig.update_layout(xaxis_tickangle=-45)
fig.show()
fig.write_image("images/semana.png")
```



```
years = df_m['FECHA DE LOS HECHOS'].dt.year.unique().tolist()
mx_holidays = holidays.Mexico(years=years)

# Add a boolean column if the date is a holiday


df_m['is_holiday'] = df_m['FECHA DE LOS HECHOS'].dt.date.isin(mx_holidays)
```

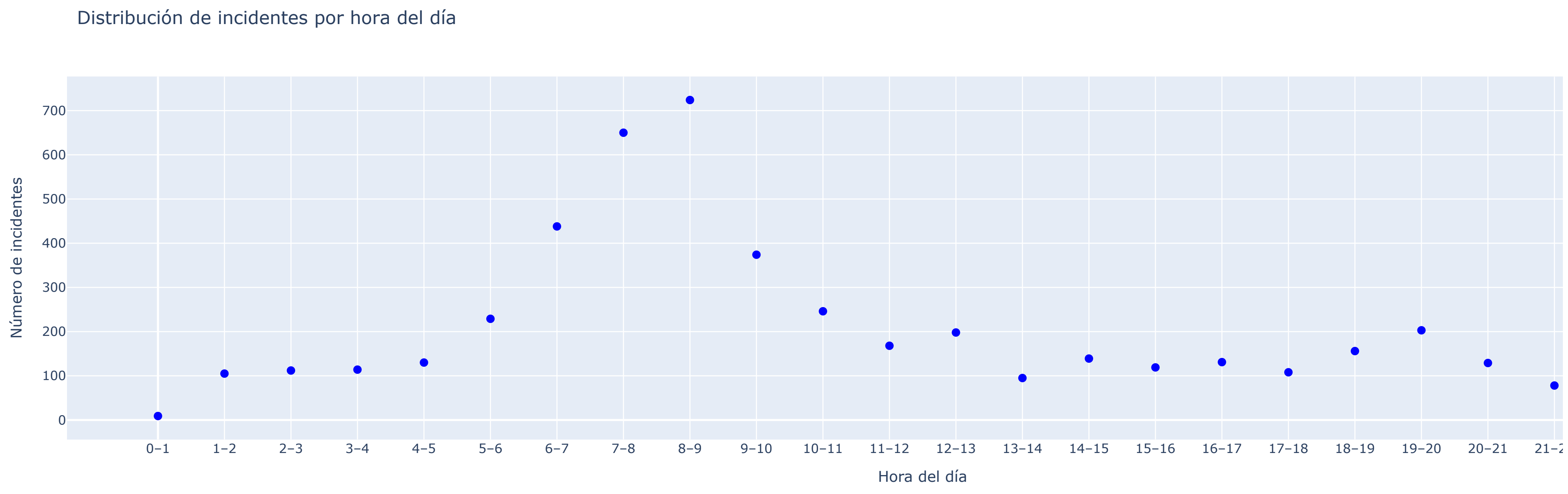
```
def classify_day(row):
    if row['is_holiday']:
        return 'Dia festivo'
    elif row['dayofweek'] in [5, 6]:  # Saturday=5, Sunday=6
        return 'Fin de semana'
    else:
        return 'Entre semana'

df_m['day_type'] = df_m.apply(classify_day, axis=1)

daily_counts = df_m.groupby(['FECHA DE LOS HECHOS', 'day_type']).size().reset_index(name='count')

# Calculate average incidents per day_type
avg_incidents = daily_counts.groupby('day_type')['count'].mean().reset_index()
```

```
fig = px.bar(
    avg_incidents,
    x='day_type',
    y='count',
    text='count',
    title="Promedio diario de incidentes por tipo de día"
)

fig.update_traces(texttemplate='%{text:.2f}', textposition='outside')
fig.update_layout(yaxis_title="Promedio diario de incidentes",
                  xaxis_title="Tipo de día")
fig.show()
fig.write_image("images/festivo_entresemana.png")
```

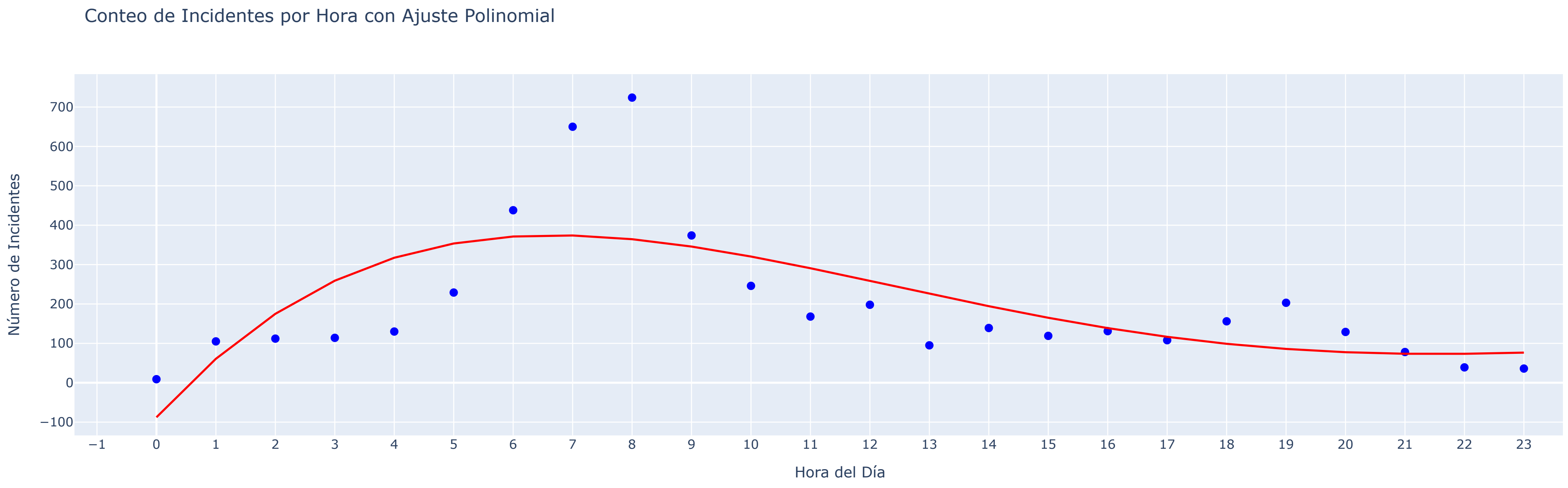


```
hourly_counts = df_m["Hour"].value_counts().reindex(range(24), fill_value=0).sort_index()

# 3. Define custom tick labels
tick_labels = [f"{h}-{(h+1)%24}" for h in range(24)]

# 4. Plot manually
fig = go.Figure(
    data=go.Scatter(
        x=list(range(24)),
        y=hourly_counts.values,
        mode="markers",
        marker=dict(color="blue", size=8),
    )
)

```

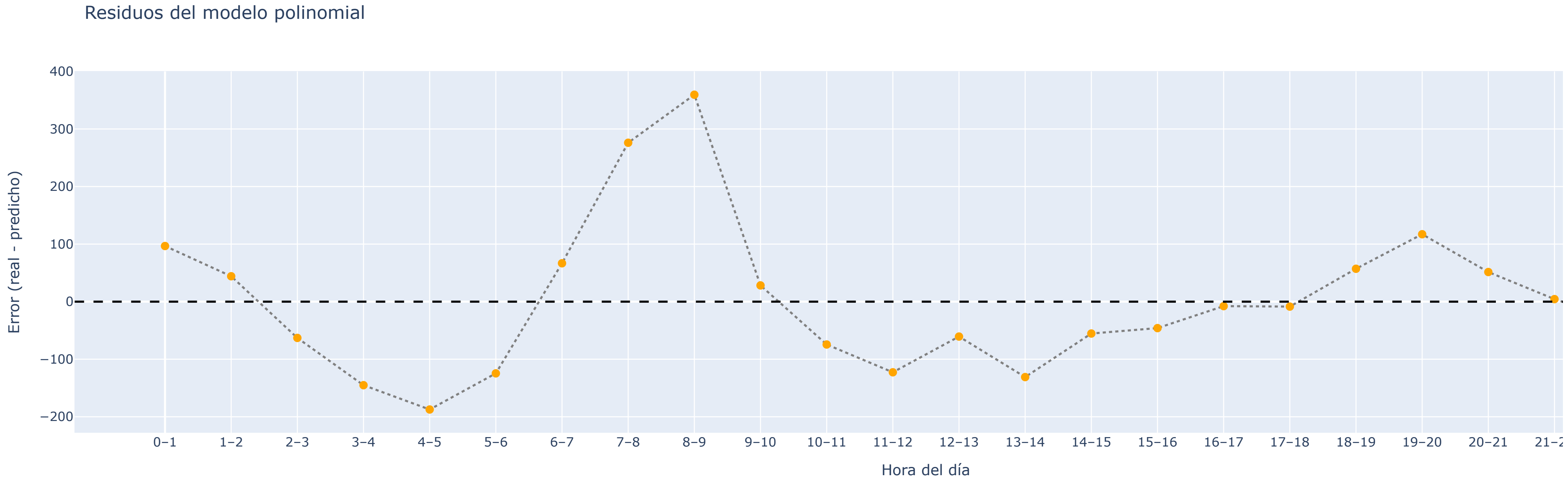
```python
fig.update_layout(
    title="Distribución de incidentes por hora del día",
    xaxis=dict(
        tickmode="array",
        tickvals=list(range(24)),
        ticktext=tick_labels,
        title="Hora del día"
    ),
    yaxis=dict(title="Número de incidentes"),
    bargap=0.1
)

fig.show()
fig.write_image("images/por_hora.png")
```



```python


# Count incidents per hour
counts_by_hour = df_m.groupby("Hour").size().reindex(range(24), fill_value=0)
x = counts_by_hour.index.values  # hours 0 to 23
y = counts_by_hour.values        # incident counts


poly_fit = np.polynomial.polynomial.Polynomial.fit(x, y, 4)


#x_smooth = np.linspace(0, 23, 100)
y_pred = poly_fit(x)
y_pred = poly_fit(x)

# Create plot
fig = go.Figure()

# Original data points as scatter
fig.add_trace(go.Scatter(
    x=x,
    y=y,
    mode='markers',
    name='Datos Reales',
    marker=dict(color='blue', size=8)
))

# Polynomial fit curve
fig.add_trace(go.Scatter(
    x=x,
    y=y_pred,
    mode='lines',
    name='Ajuste Polinomial Grado 4',
    line=dict(color='red')
))

fig.update_layout(
    title='Conteo de Incidentes por Hora con Ajuste Polinomial',
    xaxis_title='Hora del Día',
    yaxis_title='Número de Incidentes',
    xaxis=dict(tickmode='linear', dtick=1)
)

fig.show()

residuals = y - y_pred
ss_res = np.sum(residuals**2)
ss_tot = np.sum((y - np.mean(y))**2)
r_squared = 1 - (ss_res / ss_tot)
rmse = np.sqrt(np.mean(residuals**2))

fig_res = go.Figure()

fig_res.add_trace(go.Scatter(
    x=x,
    y=residuals,
    mode="markers+lines",
    name="Residuos",
    marker=dict(color="orange", size=8),
    line=dict(dash="dot", color="gray")
))

fig_res.add_hline(y=0, line=dict(color="black", dash="dash"))

fig_res.update_layout(
    title="Residuos del modelo polinomial",
    xaxis_title="Hora del día",
    yaxis_title="Error (real - predicho)",
    xaxis=dict(
        tickmode='array',
        tickvals=list(range(24)),
        ticktext=tick_labels
    ),
    showlegend=False
)

fig_res.show()

pio.write_image(fig, 'images/polinomial.png', scale=1, width=1500, height=500)
pio.write_image(fig_res, 'images/residuals_pol.png', scale=1, width=1500, height=500)

print(f"R²:   {r_squared:.4f}")
print(f"RMSE: {rmse:.2f}")
```



Conteo de Incidentes por Hora con Ajuste Polinomial
Número de Incidentes
Hora del Día

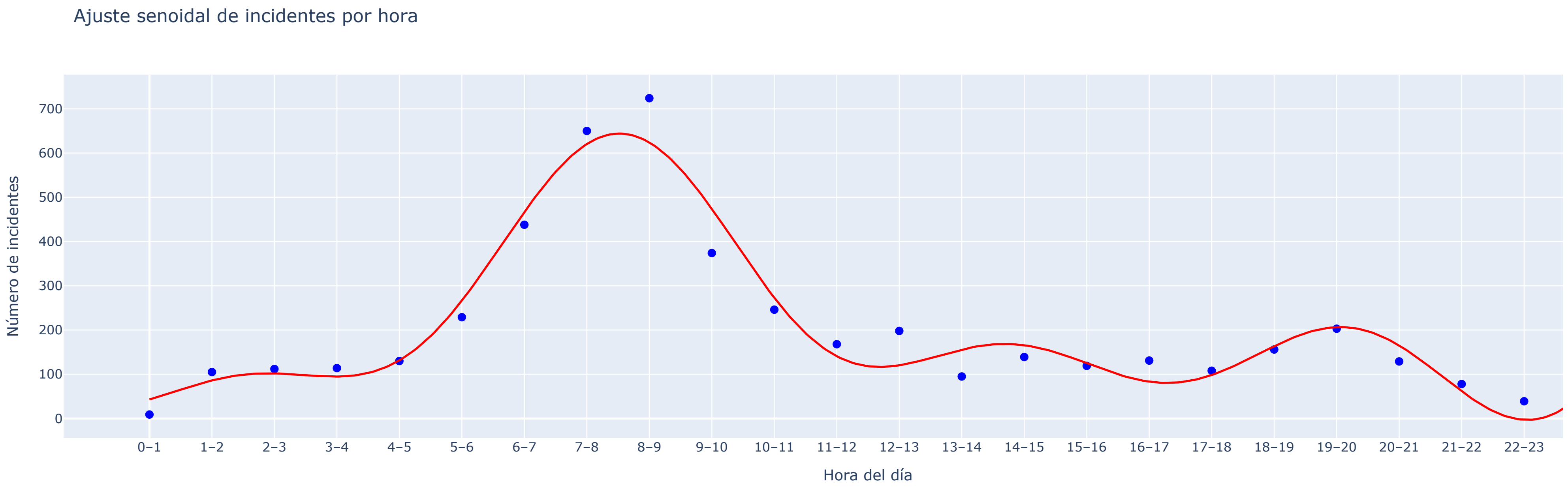


Residuos del modelo polinomial
Error (real - predicho)
Hora del día

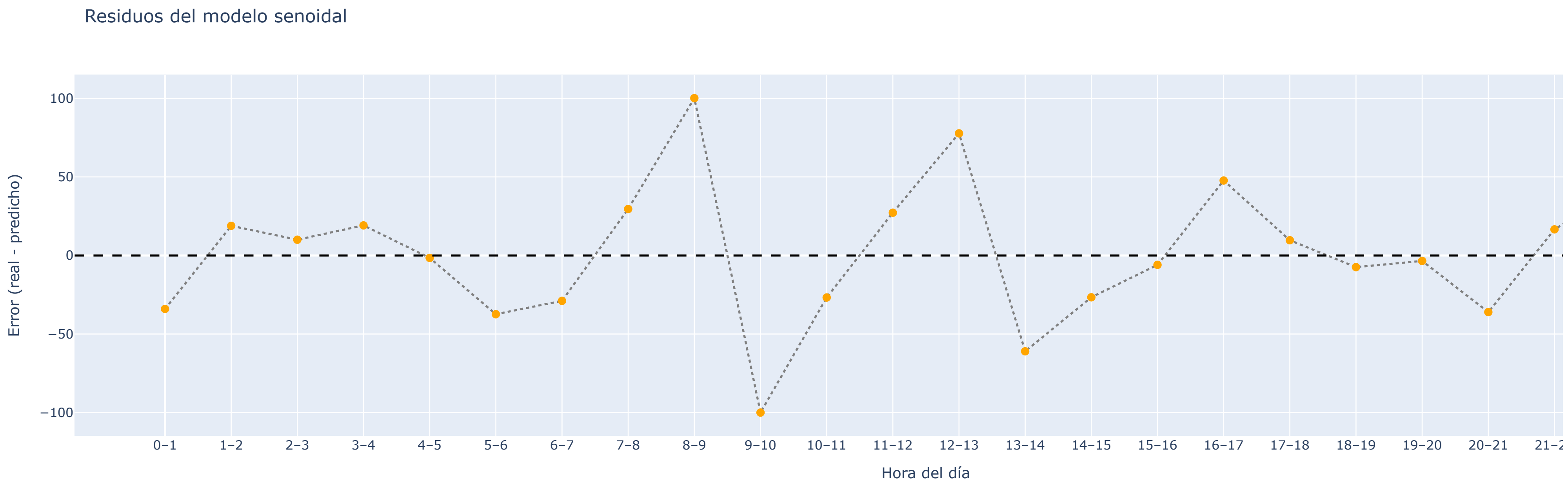
```
R²:   0.5058
```

```
RMSE: 123.65
```

```python
x = np.array(list(range(24)))                  # Hours
y = hourly_counts.values                       # Incident counts

# Sine function
def triple_sine(x, a1, b1, c1, a2, b2, c2, a3, b3, c3, d):
    return (
        a1 * np.sin(b1 * x + c1) +
        a2 * np.sin(b2 * x + c2) +
        a3 * np.sin(b3 * x + c3) +
        d
    )

# Initial guess: amplitudes, frequencies, phases, offset
guess = [
    10, 2*np.pi/24, 0,   # 1st term: daily
    5, 4*np.pi/24, 0,    # 2nd term: 2 cycles per day
    3, 6*np.pi/24, 0,    # 3rd term: 3 cycles per day
    np.mean(y)           # offset
]

# Fit model
popt, _ = curve_fit(triple_sine, x, y, p0=guess)

# Predict
y_pred = triple_sine(x, *popt)

# Residuals
residuals = y - y_pred

x_fit = np.linspace(0, 23, 500)
y_fit = triple_sine(x_fit, *popt)

fig = go.Figure()

fig.add_trace(go.Scatter(
    x=x,
    y=y,
    mode='markers',
    name='Datos Reales',
    marker=dict(color='blue', size=8)
))

fig.add_trace(go.Scatter(
    x=x_fit,
    y=y_fit,
    mode='lines',
    name='Ajuste senoidal',
    line=dict(color='red')
))

fig.update_layout(
    title="Ajuste senoidal de incidentes por hora",
    xaxis_title="Hora del día",
    yaxis_title="Número de incidentes",
    xaxis=dict(tickmode='array', ticktext=tick_labels, tickvals=list(range(24))),
    bargap=0.1
)

fig.show()


fig_res = go.Figure()

fig_res.add_trace(go.Scatter(
    x=x,
    y=residuals,
    mode="markers+lines",
    name="Residuos",
    marker=dict(color="orange", size=8),
    line=dict(dash="dot", color="gray")
))

fig_res.add_hline(y=0, line=dict(color="black", dash="dash"))

fig_res.update_layout(
    title="Residuos del modelo senoidal",
    xaxis_title="Hora del día",
    yaxis_title="Error (real - predicho)",
    xaxis=dict(
        tickmode='array',
        tickvals=list(range(24)),
        ticktext=tick_labels
    ),
    showlegend=False
)

fig_res.show()


pio.write_image(fig, 'images/sinoidal.png', scale=1, width=1500, height=500)
pio.write_image(fig_res, 'images/residuals_sen.png', scale=1, width=1500, height=500)

ss_res = np.sum(residuals**2)
ss_tot = np.sum((y - np.mean(y))**2)
r_squared = 1 - (ss_res / ss_tot)
rmse = np.sqrt(np.mean(residuals**2))

print(f"R²:   {r_squared:.4f}")
print(f"RMSE: {rmse:.2f}")
```



Ajuste senoidal de incidentes por hora
Número de incidentes
700
600
500
400
300
200
100
0
0–1
1–2
2–3
3–4
4–5
5–6
6–7
7–8
8–9
9–10
10–11
11–12
12–13
13–14
14–15
15–16
16–17
17–18
18–19
19–20
20–21
21–22
22–23
Hora del día



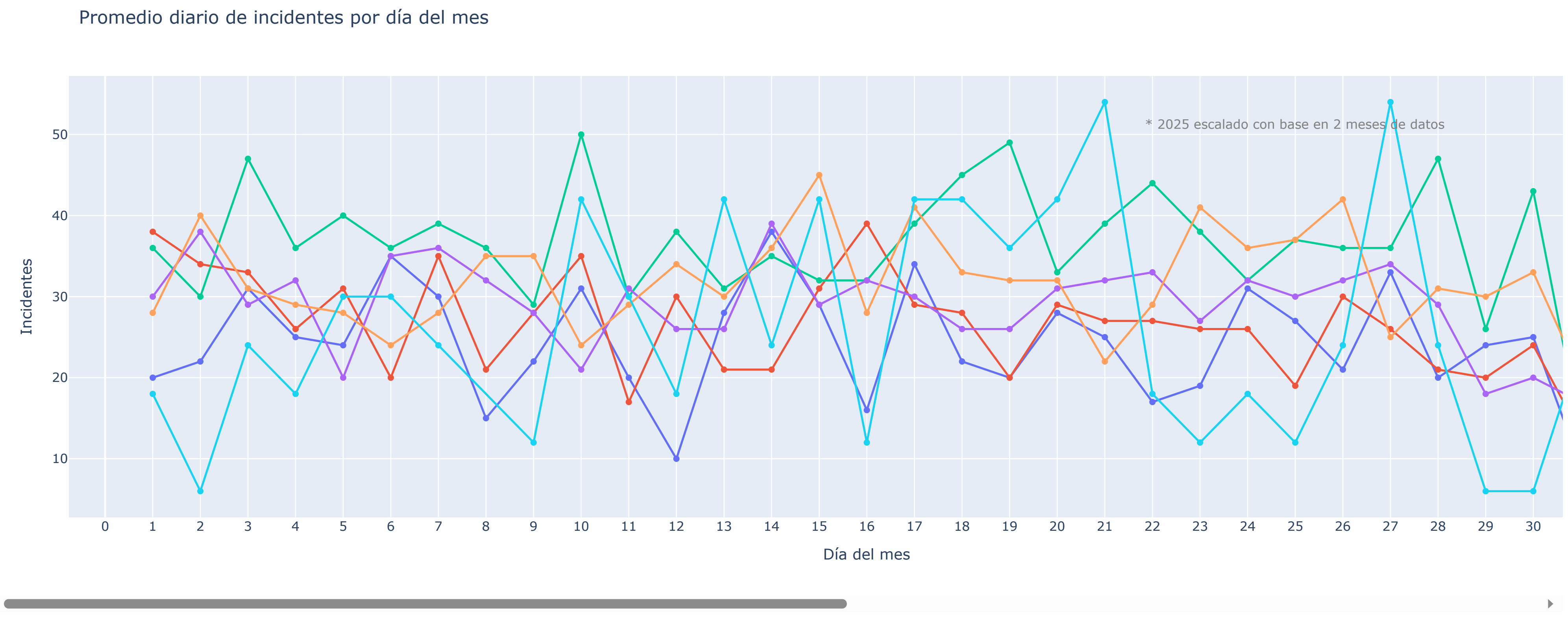
Residuos del modelo senoidal
Error (real - predicho)
100
50
0
−50
−100
0–1
1–2
2–3
3–4
4–5
5–6
6–7
7–8
8–9
9–10
10–11
11–12
12–13
13–14
14–15
15–16
16–17
17–18
18–19
19–20
20–21
Hora del día

```
R²:   0.9411
```

```
RMSE: 42.67
```

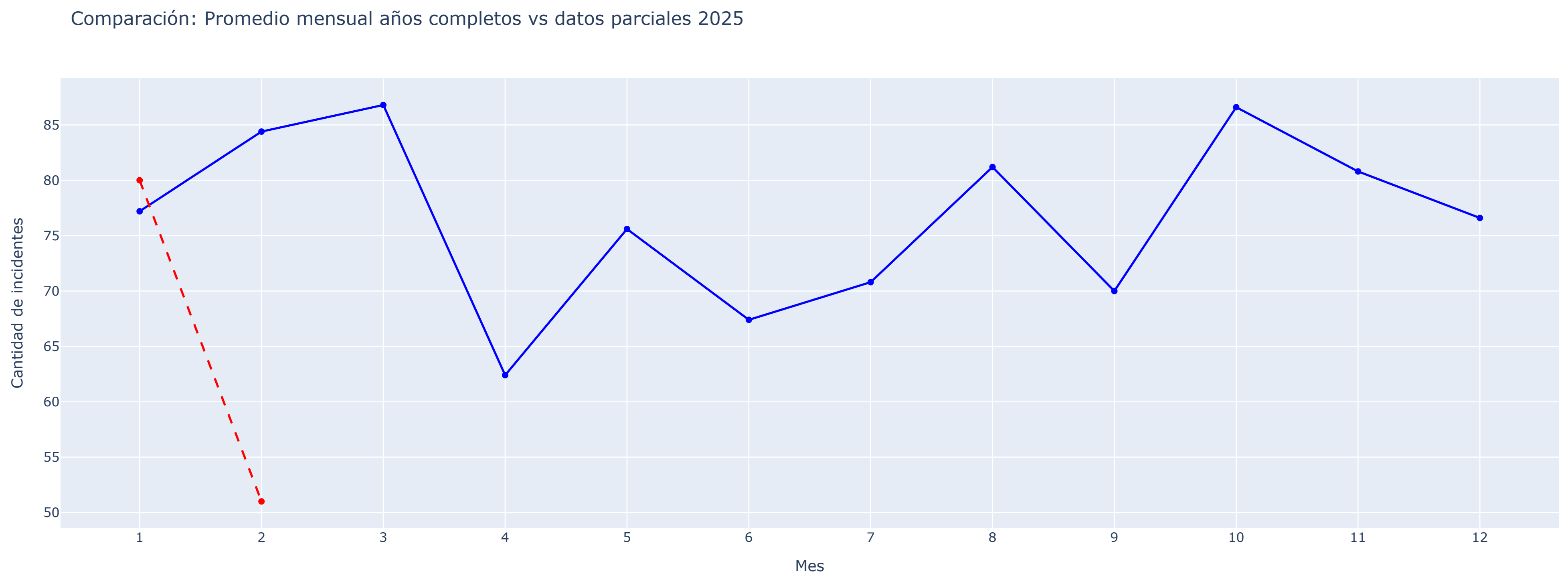
```python
df_full_years = df_m[df_m['year'] != 2025]
df_2025 = df_m[df_m['year'] == 2025]

# ---- FULL YEARS ----
# Calculate average incidents per day-of-month across years (e.g., 1st, 2nd, ..., 31st)
daily_avg_full = (
    df_full_years.groupby(["year", "day"])
    .size()
    .reset_index(name="count")
)

# ---- 2025 (SCALED) ----
# Count incidents per day-of-month in 2025
daily_total_2025 = (
    df_2025.groupby("day")
    .size()
    .reset_index(name="count")
)

# Scale 2025 data: assume similar pattern all year
months_2025 = df_2025["month"].nunique()
scaling_factor = 12 / months_2025  # e.g., if 2 months -> scale by 6
daily_total_2025["count"] = daily_total_2025["count"] * scaling_factor
daily_total_2025["year"] = "2025 (estimado)"


combined_df = pd.concat([daily_avg_full, daily_total_2025], ignore_index=True)

# ---- PLOT ----
fig = px.line(
    combined_df,
    x="day",
    y="count",
    color="year",
    markers=True,
    title="Promedio diario de incidentes por día del mes",
    labels={"day": "Día del mes", "count": "Incidentes", "year": "Año"}
)

fig.update_layout(
    xaxis=dict(tickmode="linear", tick0=1, dtick=1),
    height=600,
    legend_title_text="Año",
    annotations=[
        dict(
            x=25, y=combined_df["count"].max() * 0.95,
            text="* 2025 escalado con base en 2 meses de datos",
            showarrow=False,
            font=dict(size=12, color="gray")
        )
    ]
)

```

```
fig.show()
fig.write_image("images/promedio_dia_pormes.png")
```



```
df_others = df_m[df_m["year"] != 2025]
df_2025 = df_m[df_m["year"] == 2025]

# Calculate average monthly counts for full years (excluding 2025)
monthly_counts_others = df_others.groupby(["year", "month"]).size().reset_index(name="count")
avg_per_month_others = monthly_counts_others.groupby("month")["count"].mean().reset_index(name="avg_count")

# Get 2025 monthly counts (only actual data, no averaging needed)
monthly_counts_2025 = df_2025.groupby("month").size().reset_index(name="count")

# Create figure
fig = go.Figure()

# Add average per month trace for other years (line)
fig.add_trace(go.Scatter(
    x=avg_per_month_others["month"],
    y=avg_per_month_others["avg_count"],
    mode='lines+markers',
    name='Promedio mensual (Años completos)',
    line=dict(color='blue')
```

```python
))

# Add 2025 actual monthly counts trace (markers and dashed line)
fig.add_trace(go.Scatter(
    x=monthly_counts_2025["month"],
    y=monthly_counts_2025["count"],
    mode='lines+markers',
    name='Datos 2025 (Parciales)',
    line=dict(color='red', dash='dash')
))

# Update layout
fig.update_layout(
    title="Comparación: Promedio mensual años completos vs datos parciales 2025",
    xaxis_title="Mes",
    yaxis_title="Cantidad de incidentes",
    xaxis=dict(tickmode='linear', tick0=1, dtick=1),
    height=600
)

fig.show()
fig.write_image("images/promedio_mens.png")
```



```
zonas = ['vagón', 'andén', 'pasillo', 'taquilla', 'acceso']
estaciones = ['Pantitlán', 'Pino Suárez', 'Hidalgo', 'Centro Médico', 'Chabacano', 'Indios Verdes']


```