FUNDAMENTOS DE PYTHON PARA LA CIENCIA DE DATOS

Separata Académica

MÓDULO 5

Visualización de Datos: Comunicando Insights con Matplotlib y Seaborn

Nivel: Básico-Intermedio

1. PRESENTACIÓN

La visualización de datos es una de las competencias más valoradas dentro del perfil del analista de datos y el científico de datos. Transformar tablas de números en gráficos claros y significativos permite que los hallazgos de un análisis sean comprensibles no solo para especialistas técnicos, sino también para equipos de negocio, directivos y cualquier persona que necesite tomar decisiones basadas en datos.

Este módulo aborda dos de las librerías más utilizadas en el ecosistema Python para la ciencia de datos: Matplotlib y Seaborn. Matplotlib es la librería base de visualización en Python, ofrece control total sobre cada elemento de una figura; Seaborn se construye sobre Matplotlib y proporciona una interfaz de alto nivel que facilita la creación de gráficos estadísticos con estética moderna y código más conciso (Hunter, 2007; Waskom, 2021).

Los ejemplos y ejercicios de esta separata utilizan dos conjuntos de datos reales del entorno empresarial peruano:

Dataset	Archivo	Descripción
ventas.csv	ventas.csv	1 año de transacciones de una empresa tecnológica: productos, categorías, vendedores, regiones, canales y métodos de pago (2024).
rrhh.csv	rrhh.csv	Información de empleados: edad, ciudad, departamento, salario y estado civil.

Al finalizar este módulo, el participante habrá construido al menos 10 tipos de gráficos distintos, aplicará buenas prácticas de personalización visual y estará en condiciones de comunicar resultados de un análisis exploratorio de datos (EDA) de forma profesional.

2. OBJETIVO

2.1 Objetivo General

Desarrollar la capacidad del participante para crear, personalizar e interpretar visualizaciones de datos en Python, utilizando las librerías Matplotlib y Seaborn sobre conjuntos de datos reales, con el fin de comunicar hallazgos de manera efectiva dentro de un flujo de trabajo de ciencia de datos.

2.2 Objetivos Específicos

- Comprender la arquitectura de figuras y ejes en Matplotlib para construir gráficos básicos y compuestos.
- Aplicar opciones de personalización visual (colores, etiquetas, leyendas, anotaciones) para producir gráficos de calidad profesional.
- Utilizar Seaborn para generar gráficos estadísticos de distribución, relación y categorías con mínimas líneas de código.
- Seleccionar el tipo de gráfico adecuado según la naturaleza de los datos y el mensaje que se desea transmitir.
- Integrar visualizaciones en un flujo EDA sobre los datasets ventas.csv y rrhh.csv.

Relación con el Sílabo

Este módulo corresponde al contenido del Módulo 5 del sílabo del curso:
"VISUALIZACIÓN DE DATOS: comunicando insights con Matplotlib y Seaborn".
Competencias previas requeridas: Módulos 1 a 4 (Python básico, NumPy y Pandas).

3. REALIZAR GRÁFICOS CON MATPLOTLIB

3.1 ¿Qué es Matplotlib?

Matplotlib es la librería de visualización más utilizada en Python. Fue creada por John D. Hunter en 2003 y actualmente es mantenida por la comunidad open source. Su módulo principal es matplotlib.pyplot, que ofrece una interfaz similar a MATLAB para la generación de gráficos (Hunter, 2007).

Instalación: Si no está instalada en el entorno, se puede instalar con:

```
pip install matplotlib
```

Importación estándar:

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

3.2 Arquitectura de una Figura en Matplotlib

Para aprovechar al máximo Matplotlib es fundamental comprender su jerarquía de objetos:

Objeto	Descripción	Ejemplo de uso
Figure	Contenedor principal (lienzo completo)	fig = plt.figure()
Axes	Área donde se dibuja el gráfico	ax = fig.add_subplot(1,1,1)
Axis	Ejes X e Y de cada Axes	ax.xaxis, ax.yaxis
Artist	Todo elemento visual (líneas, texto, etc.)	ax.set_title('Título')

Existen dos formas de trabajar con Matplotlib:

- Interfaz pyplot (stateful): más sencilla, adecuada para gráficos rápidos.
- Interfaz orientada a objetos (OO): más explícita y recomendada para proyectos profesionales (Matplotlib Development Team, 2024).

3.3 Gráfico de Líneas — Tendencia de ventas mensuales (ventas.csv)

El gráfico de líneas es ideal para visualizar datos ordenados en el tiempo (series temporales). En el siguiente ejemplo se agrupa el total de ventas por mes usando el dataset ventas.csv.

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

# Carga del dataset
df_ventas = pd.read_csv('ventas.csv')

# Convertir columna fecha a tipo datetime
df_ventas['fecha'] = pd.to_datetime(df_ventas['fecha'])

# Extraer mes y agrupar
df_ventas['mes'] = df_ventas['fecha'].dt.to_period('M')
ventas_mes = df_ventas.groupby('mes')['total'].sum().reset_index()
ventas_mes['mes'] = ventas_mes['mes'].astype(str)

# Crear gráfico
fig, ax = plt.subplots(figsize=(12, 5))
ax.plot(ventas_mes['mes'], ventas_mes['total'],
        marker='o', linewidth=2, color='steelblue',
        markerfacecolor='white', markeredgewidth=2)

ax.set_title('Ventas Totales Mensuales - 2024', fontsize=14, fontweight='bold')
ax.set_xlabel('Mes')
ax.set_ylabel('Total de Ventas (S/.)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Nota Pedagógica

marker='o' → Añade un marcador circular en cada punto de dato.
tight_layout() → Ajusta automáticamente los márgenes para evitar recortes.
figsize=(12, 5) → Dimensiones en pulgadas del lienzo (ancho, alto).

3.4 Gráfico de Barras — Ventas por Región (ventas.csv)

El gráfico de barras permite comparar magnitudes entre categorías. En el siguiente ejemplo se visualiza el total de ventas por región.

```
# Datos
ventas_region =
df_ventas.groupby('region')['total'].sum().sort_values(ascending=False)


fig, ax = plt.subplots(figsize=(10, 5))
barras = ax.bar(ventas_region.index, ventas_region.values,
                color='steelblue', edgecolor='white', linewidth=0.8)

# Añadir etiquetas de valor sobre cada barra
for barra in barras:
    height = barra.get_height()
    ax.text(barra.get_x() + barra.get_width() / 2., height + 1000,
            f'S/. {height:,.0f}', ha='center', va='bottom', fontsize=9)

ax.set_title('Ventas Totales por Región - 2024', fontsize=13, fontweight='bold')
ax.set_xlabel('Región')
ax.set_ylabel('Total de Ventas (S/.)')
plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```

3.5 Histograma — Distribución de Salarios (rrhh.csv)

El histograma muestra la distribución de frecuencias de una variable numérica continua. Es especialmente útil en el análisis exploratorio para detectar asimetría, valores atípicos o concentraciones.

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

df_rrhh = pd.read_csv('rrhh.csv')

fig, ax = plt.subplots(figsize=(9, 5))
ax.hist(df_rrhh['salario'].dropna(), bins=15,
        color='teal', edgecolor='white', linewidth=0.6)

ax.set_title('Distribución de Salarios de Empleados', fontsize=13,
fontweight='bold')
ax.set_xlabel('Salario (S/.)')
ax.set_ylabel('Frecuencia')
ax.axvline(df_rrhh['salario'].mean(), color='red',
           linestyle='--', linewidth=1.5, label='Media')
ax.legend()
plt.tight_layout()
plt.show()
```

¿Por qué dropna()?

El método dropna() elimina los valores nulos (NaN) antes de graficar.
En el dataset rrhh.csv, la columna 'edad' contiene algunos valores nulos que también se podrían manejar de esta forma en otros análisis.

3.6 Gráfico de Dispersión (Scatter) — Cantidad vs. Total de Venta (ventas.csv)

El gráfico de dispersión permite explorar relaciones entre dos variables numéricas continuas. En el siguiente ejemplo se examina si existe correlación entre la cantidad de unidades vendidas y el monto total.

```
fig, ax = plt.subplots(figsize=(8, 5))
ax.scatter(df_ventas['cantidad'], df_ventas['total'],
           alpha=0.4, color='darkorange', edgecolors='white', s=50)

ax.set_title('Cantidad de Unidades vs. Total de Venta', fontsize=13,
fontweight='bold')
ax.set_xlabel('Cantidad (unidades)')
ax.set_ylabel('Total de Venta (S/.)')
plt.tight_layout()
plt.show()
```

3.7 Gráfico de Pastel (Pie) — Distribución por Canal de Venta (ventas.csv)

El gráfico de pastel es adecuado para mostrar proporciones de un total cuando las categorías son pocas (idealmente no más de 6). En este caso se visualiza la participación de cada canal de venta.

```
canal_counts = df_ventas['canal'].value_counts()
print(canal_counts)

fig, ax = plt.subplots(figsize=(7, 6))
ax.pie(canal_counts.values,
       labels=canal_counts.index,
       autopct='%1.1f%%',
       startangle=140,
       colors=['#2E75B6', '#ED7D31', '#A9D18E'])

ax.set_title('Distribución de Ventas por Canal', fontsize=13, fontweight='bold')
plt.tight_layout()
plt.show()
```

4. PERSONALIZACIÓN DE GRÁFICOS

La personalización es lo que diferencia un gráfico exploratorio rápido de una visualización lista para presentación. Matplotlib ofrece un nivel de control muy detallado sobre cada elemento visual de una figura (Matplotlib Development Team, 2024).

4.1 Elementos Personalizables

Elemento	Método / Propiedad	Ejemplo
Título	ax.set_title()	ax.set_title('Ventas 2024', fontsize=14)
Etiquetas de ejes	ax.set_xlabel() / set_ylabel()	ax.set_xlabel('Mes', fontsize=11)
Rango de ejes	ax.set_xlim() / set_ylim()	ax.set_ylim(0, 50000)
Cuadrícula	ax.grid()	ax.grid(True, linestyle='--', alpha=0.5)
Leyenda	ax.legend()	ax.legend(loc='upper right')
Colores	color= / palette=	color='#2E75B6' o color='steelblue'
Tamaño de fuente	fontsize=	fontsize=12
Rotación de ticks	plt.xticks(rotation=)	plt.xticks(rotation=45)
Anotaciones	ax.annotate()	ax.annotate('Pico', xy=(6, 45000))
Estilo global	plt.style.use()	plt.style.use('seaborn-v0_8-whitegrid')

4.2 Estilos Disponibles en Matplotlib

Matplotlib incluye estilos predefinidos que modifican la apariencia global de todos los gráficos. Para listar los estilos disponibles:

```
import matplotlib.pyplot as plt
print(plt.style.available)
```

Algunos estilos recomendados para presentaciones académicas y ejecutivas:

- 'seaborn-v0_8-whitegrid' — Fondo blanco con cuadrícula gris clara.
- 'ggplot' — Estilo inspirado en el popular paquete ggplot2 de R.
- 'bmh' — Estilo bayesiano con colores suaves y agradables.
- 'fivethirtyeight' — Inspirado en la web de análisis FiveThirtyEight.

4.3 Ejemplo Completo — Gráfico de Barras Personalizado (ventas.csv)

El siguiente ejemplo integra múltiples técnicas de personalización en un único gráfico de barras con los datos de ventas por departamento del dataset rrhh.csv.

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

df_rrhh = pd.read_csv('rrhh.csv')

# Calcular salario promedio por departamento
sal_depto =
df_rrhh.groupby('departamento')['salario'].mean().sort_values(ascending=False)

# Aplicar estilo global
plt.style.use('seaborn-v0_8-whitegrid')

fig, ax = plt.subplots(figsize=(10, 6))

colores = ['#1F5C99' if s >= sal_depto.mean() else '#AEC6E8' for s in
sal_depto.values]

barras = ax.barh(sal_depto.index, sal_depto.values,
                 color=colores, edgecolor='white', height=0.6)

# Etiquetas de valor al lado de cada barra
for barra in barras:
    width = barra.get_width()
    ax.text(width + 50, barra.get_y() + barra.get_height() / 2,
            f'S/. {width:,.0f}', va='center', fontsize=10)

# Línea de promedio general
ax.axvline(sal_depto.mean(), color='red', linestyle='--',
           linewidth=1.5, label=f'Promedio: S/. {sal_depto.mean():,.0f}')

# Personalización
ax.set_title('Salario Promedio por Departamento', fontsize=14,
             fontweight='bold', pad=15)
ax.set_xlabel('Salario Promedio (S/.)', fontsize=11)
ax.set_ylabel('Departamento', fontsize=11)
ax.legend(fontsize=10)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)

plt.tight_layout()
plt.savefig('salario_departamento.png', dpi=150, bbox_inches='tight')
plt.show()
```

Buenas Prácticas de Visualización

1. Eliminar elementos innecesarios (spines, grillas excesivas) → menos es más.
2. Usar colores con significado: resaltar lo importante, atenuar lo secundario.
3. Siempre etiquetar los ejes con unidades (S/., unidades, %, etc.).
4. Incluir título descriptivo que comunique el hallazgo principal.
5. Exportar con dpi >= 150 para presentaciones de calidad.

4.4 Subplots — Múltiples Gráficos en una Figura

Para comparar múltiples perspectivas en una sola figura se utilizan subplots. La función plt.subplots(filas, columnas) devuelve una cuadrícula de Axes:

```
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Subplot 1: Ventas por canal
canal_total = df_ventas.groupby('canal')['total'].sum()
axes[0].bar(canal_total.index, canal_total.values,
color=['#2E75B6','#ED7D31','#A9D18E'])
axes[0].set_title('Ventas por Canal')
axes[0].set_ylabel('Total (S/.)')

# Subplot 2: Ventas por método de pago
pago_total = df_ventas.groupby('metodo_pago')['total'].sum().sort_values()
axes[1].barh(pago_total.index, pago_total.values, color='teal')
axes[1].set_title('Ventas por Método de Pago')
axes[1].set_xlabel('Total (S/.)')

plt.suptitle('Análisis de Canales y Métodos de Pago - 2024',
             fontsize=14, fontweight='bold', y=1.02)
plt.tight_layout()
plt.show()
```

5. GRÁFICOS CON SEABORN

5.1 ¿Qué es Seaborn?

Seaborn es una librería de visualización estadística construida sobre Matplotlib. Fue creada por Michael Waskom y publicada originalmente en 2012. Ofrece una API de alto nivel que permite crear gráficos estadísticos complejos con código más conciso, incluyendo automáticamente estimaciones de densidad, intervalos de confianza y paletas de colores optimizadas para la comunicación de datos (Waskom, 2021).

Instalación:

```
pip install seaborn
```

Importación estándar:

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

5.2 Configuración del Tema

Seaborn permite configurar el tema global mediante sns.set_theme(), que afecta todos los gráficos posteriores:

```
# Opciones de estilo: darkgrid, whitegrid, dark, white, ticks
sns.set_theme(style='whitegrid', palette='muted', font_scale=1.1)
```

5.3 Gráficos de Distribución

Los gráficos de distribución permiten explorar la forma y dispersión de una variable numérica.

5.3.1 histplot — Histograma con KDE (rrhh.csv)

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

df_rrhh = pd.read_csv('rrhh.csv')
sns.set_theme(style='whitegrid')

fig, ax = plt.subplots(figsize=(9, 5))
sns.histplot(data=df_rrhh, x='salario', kde=True,
             bins=15, color='steelblue', ax=ax)

ax.set_title('Distribución de Salarios (con estimación KDE)', fontsize=13)
ax.set_xlabel('Salario (S/.)')
ax.set_ylabel('Frecuencia')
plt.tight_layout()
plt.show()
```

¿Qué es KDE?

KDE (Kernel Density Estimation) es una estimación de densidad de probabilidad.
La curva suavizada sobre el histograma indica cómo se distribuye la variable.
Es más informativa que el histograma solo cuando los datos son continuos.
Referencia: Silverman, B.W. (1986). Density Estimation for Statistics and Data Analysis.

5.3.2 boxplot — Distribución por grupos (rrhh.csv)

El boxplot (diagrama de caja) muestra la mediana, cuartiles y valores atípicos de una distribución. Es especialmente útil para comparar grupos.

```
fig, ax = plt.subplots(figsize=(10, 5))
sns.boxplot(data=df_rrhh, x='departamento', y='salario',
            palette='Set2', ax=ax)

ax.set_title('Distribución Salarial por Departamento', fontsize=13)
ax.set_xlabel('Departamento')
ax.set_ylabel('Salario (S/.)')
plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```

5.3.3 violinplot — Distribución con forma de densidad (rrhh.csv)

El violinplot combina el boxplot con la estimación KDE, mostrando la forma completa de la distribución a ambos lados del eje.

```
fig, ax = plt.subplots(figsize=(10, 5))
sns.violinplot(data=df_rrhh, x='estado_civil', y='salario',
               palette='pastel', inner='quartile', ax=ax)

ax.set_title('Salario según Estado Civil', fontsize=13)
ax.set_xlabel('Estado Civil')
ax.set_ylabel('Salario (S/.)')
plt.tight_layout()
plt.show()
```

5.4 Gráficos de Relaciones

Permiten explorar la asociación entre dos o más variables numéricas.

5.4.1 scatterplot — Dispersión con agrupación por color (ventas.csv)

```
df_ventas = pd.read_csv('ventas.csv')

fig, ax = plt.subplots(figsize=(9, 6))
sns.scatterplot(data=df_ventas, x='cantidad', y='total',
                hue='canal', style='canal',
                palette='Set1', alpha=0.7, ax=ax)

ax.set_title('Relación: Cantidad vs. Total de Venta por Canal', fontsize=13)
ax.set_xlabel('Cantidad (unidades)')
ax.set_ylabel('Total (S/.)')
ax.legend(title='Canal')
plt.tight_layout()
plt.show()
```

5.4.2 heatmap — Mapa de calor de correlaciones (ventas.csv)

El mapa de calor es una representación matricial de valores numéricos mediante colores. Es ampliamente utilizado para visualizar matrices de correlación en el EDA.

```
# Seleccionar variables numéricas
numericas = df_ventas[['cantidad', 'precio_unit', 'descuento_pct',
                       'descuento_s', 'total']]

correlacion = numericas.corr()

fig, ax = plt.subplots(figsize=(8, 6))
sns.heatmap(correlacion,
            annot=True,
            fmt='.2f',
            cmap='coolwarm',
            center=0,
            linewidths=0.5,
            ax=ax)

ax.set_title('Mapa de Correlación – Variables Numéricas de Ventas', fontsize=13)
plt.tight_layout()
plt.show()
```

Interpretación del Heatmap

Valores cercanos a +1 → correlación positiva fuerte (ambas variables suben juntas).
Valores cercanos a -1 → correlación negativa fuerte (una sube, la otra baja).
Valores cercanos a 0 → sin correlación lineal aparente.
En ventas.csv se espera alta correlación entre precio_unit y total.

5.5 Gráficos de Categorías

Permiten comparar distribuciones o agregaciones entre grupos categóricos.

5.5.1 barplot — Media de ventas por región (ventas.csv)

```
fig, ax = plt.subplots(figsize=(10, 5))
sns.barplot(data=df_ventas, x='region', y='total',
            estimator='mean', palette='Blues_d',
            errorbar='sd', ax=ax)

ax.set_title('Venta Promedio por Región (con desviación estándar)', fontsize=13)
ax.set_xlabel('Región')
ax.set_ylabel('Total Promedio (S/.)')
plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```

5.5.2 countplot — Conteo de ventas por método de pago (ventas.csv)

```
fig, ax = plt.subplots(figsize=(9, 5))
sns.countplot(data=df_ventas, x='metodo_pago',
              order=df_ventas['metodo_pago'].value_counts().index,
              palette='Set2', ax=ax)

ax.set_title('Número de Transacciones por Método de Pago', fontsize=13)
ax.set_xlabel('Método de Pago')
ax.set_ylabel('Cantidad de Transacciones')
plt.xticks(rotation=20)
plt.tight_layout()
plt.show()
```

5.5.3 FacetGrid — Ventas por región y canal (ventas.csv)

FacetGrid permite crear múltiples subplots automáticamente segmentando los datos por una variable categórica. Es una herramienta poderosa para el análisis multivariado.

```
g = sns.FacetGrid(df_ventas, col='canal', height=4, aspect=1.2)
g.map_dataframe(sns.histplot, x='total', bins=15, color='steelblue')
g.set_titles(col_template='Canal: {col_name}')
g.set_axis_labels('Total Venta (S/.)', 'Frecuencia')
g.figure.suptitle('Distribución de Ventas por Canal de Venta',
                  y=1.02, fontsize=13, fontweight='bold')
plt.tight_layout()
plt.show()
```

5.6 pairplot — Matriz de Dispersión Multivariada (ventas.csv)

El pairplot genera una matriz de gráficos de dispersión para todas las combinaciones posibles de variables numéricas, incluyendo histogramas en la diagonal. Es ideal para obtener una visión panorámica de las relaciones en el dataset.

```
# Seleccionar muestra para mayor velocidad
muestra = df_ventas[['cantidad', 'precio_unit', 'total', 'canal']].sample(200,
random_state=42)

sns.pairplot(muestra, hue='canal', palette='Set1',
             diag_kind='kde', plot_kws={'alpha': 0.5})

plt.suptitle('Pairplot: Variables Numéricas por Canal de Venta',
             y=1.01, fontsize=13, fontweight='bold')
plt.show()
```

5.7 Comparativa: ¿Cuándo usar Matplotlib o Seaborn?

Criterio	Matplotlib	Seaborn
Nivel de abstracción	Bajo (control total)	Alto (más automático)
Curva de aprendizaje	Mayor	Menor
Gráficos estadísticos	Requiere más código	Incluidos nativamente (KDE, IC)
Personalización fina	Excelente	Buena (vía Matplotlib subyacente)
Rendimiento	Estable en grandes fig.	Similar (usa Matplotlib)
Uso recomendado	Gráficos de ingeniería, líneas, subplots complejos	EDA, gráficos estadísticos, presentaciones

En la práctica profesional ambas librerías se utilizan de forma complementaria: Seaborn para el análisis exploratorio y Matplotlib para los ajustes finos de presentación.

EJERCICIOS PROPUESTOS

Los siguientes ejercicios están diseñados para consolidar las competencias desarrolladas en este módulo. Se recomienda resolverlos en orden, ya que la complejidad incrementa progresivamente.

Ejercicios con ventas.csv

Ejercicio 1 — Gráfico de líneas por semestre

Crea un gráfico de líneas que muestre la evolución del total de ventas por semana durante el año 2024.
Utiliza df['fecha'].dt.isocalendar().week para obtener el número de semana.
Personaliza: título, etiquetas de ejes, color de línea, y agrega una línea de promedio anual.

Ejercicio 2 — Barras apiladas por región y canal

Construye un gráfico de barras apiladas que muestre para cada región el total de ventas desglosado por canal (Tienda física, E-commerce, Corporativo).
Sugerencia: usa pivot_table() de Pandas para preparar los datos.

Ejercicio 3 — Heatmap mensual por vendedor

Crea un heatmap donde las filas sean los 8 vendedores y las columnas sean los 12 meses del año.
El valor de cada celda debe ser el total de ventas del vendedor en ese mes.
Usa sns.heatmap() con annot=True y fmt='.0f'.

Ejercicio 4 — Subplots: Top 5 productos

En una figura con 1 fila y 2 columnas:
- Subplot izq.: Barras horizontales con los 5 productos de mayor total de ventas.
- Subplot der.: Barras horizontales con los 5 productos de menor total de ventas.

Asegúrate de usar plt.suptitle() para añadir un título general a la figura.

Ejercicio 5 — Análisis de descuentos

Usando sns.boxplot(), compara la distribución del total de venta para cada nivel de descuento (0%, 5%, 10%, 15%).
¿Los descuentos mayores generan tickets de venta más altos?
Añade una interpretación escrita en una celda Markdown debajo del gráfico.

Ejercicios con rrhh.csv

Ejercicio 6 — Distribución de edades

Crea un histograma con KDE de la columna 'edad'.
Añade líneas verticales para la media y la mediana con diferentes colores.
Incluye una leyenda que identifique cada línea.

Ejercicio 7 — Violinplot por ciudad

Construye un violinplot que compare la distribución salarial por ciudad.
Personaliza la paleta de colores y elimina los spines superior y derecho.
¿En qué ciudad hay mayor dispersión salarial?

Ejercicio 8 — Gráfico de barras con facetas

Usando FacetGrid, crea una cuadrícula de gráficos donde cada panel corresponda a un departamento y muestre la distribución del salario.
Analiza si los departamentos de Sistemas y Finanzas tienen distribuciones distintas.

6. CONCLUSIONES

1. Matplotlib es la librería base de visualización en Python, ofrece control total sobre cada elemento gráfico pero requiere más líneas de código para resultados estéticamente elaborados. Es la opción predilecta para gráficos de ingeniería, exportación de alta precisión y subplots complejos.

2. Seaborn simplifica significativamente la creación de gráficos estadísticos al operar sobre la interfaz de objetos de Matplotlib. Sus funciones como histplot, boxplot, heatmap o pairplot permiten extraer información distribucional y relacional con mínimo esfuerzo de codificación (Waskom, 2021).

3. La selección del tipo de gráfico adecuado es una decisión analítica, no meramente estética: el gráfico de líneas comunica evolución temporal; el de barras compara categorías; el histograma revela distribuciones; el scatter detecta correlaciones; el heatmap sintetiza matrices de relación.

4. Las buenas prácticas de personalización (eliminación de elementos innecesarios, uso de color con significado, etiquetado claro de ejes y unidades) son fundamentales para que una visualización sea honesta, comprensible y de calidad profesional.

5. El flujo EDA estándar —carga, exploración, limpieza, visualización y comunicación— encuentra en Matplotlib y Seaborn sus herramientas de visualización más representativas dentro del ecosistema Python para la ciencia de datos.

6. El uso conjunto de los datasets ventas.csv y rrhh.csv demuestra que visualizaciones bien diseñadas pueden revelar patrones de negocio relevantes: tendencias estacionales de ventas, disparidades salariales por área, efectividad de canales de distribución, entre otros hallazgos accionables para la toma de decisiones.

7. REFERENCIAS BIBLIOGRÁFICAS

Las referencias se presentan conforme a la norma APA 7.ª edición (American Psychological Association, 2020).

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Matplotlib Development Team. (2024). Matplotlib 3.9 documentation. https://matplotlib.org/stable/

McKinney, W. (2022). Python for data analysis: Data wrangling with pandas, NumPy, and Jupyter (3.ª ed.). O'Reilly Media. https://wesmckinney.com/book/

NumPy Development Team. (2024). NumPy documentation. https://numpy.org/doc/

pandas Development Team. (2024). pandas documentation. https://pandas.pydata.org/docs/

Seaborn Development Team. (2024). Seaborn: Statistical data visualization. https://seaborn.pydata.org/

Silverman, B. W. (1986). Density estimation for statistics and data analysis. Chapman & Hall/CRC.

Tufte, E. R. (2001). The visual display of quantitative information (2.ª ed.). Graphics Press.

VanderPlas, J. (2023). Python data science handbook: Essential tools for working with data. O'Reilly Media. https://jakevdp.github.io/PythonDataScienceHandbook/

Waskom, M. L. (2021). seaborn: Statistical data visualization. Journal of Open Source Software, 6(60), 3021. https://doi.org/10.21105/joss.03021