UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

Tecnologías de la información

Extracción de Conocimiento en Bases de Datos

V.1. Reporte de investigación de técnicas de visualización (50%)

Docente

ING. LUIS ENRIQUE MASCOTE CANO

Alumno

Erick Eduardo Maffiodo Delgado

IDGS 91N

Domingo, 30 de Noviembre del 2025

Índice

2. Introducción

La visualización de datos se ha vuelto un componente fundamental del análisis de datos en la era actual, en la cual empresas y organizaciones manejan enormes volúmenes de información. Mediante técnicas visuales efectivas, los datos complejos pueden convertirse en información comprensible de un vistazo. La visualización facilita la identificación de **tendencias, patrones y anomalías** que serían difíciles de detectar en tablas de números crudos. Además, comunica los hallazgos de manera clara tanto a expertos como a audiencias no técnicas, mejorando la toma de decisiones basada en datos.

Justificación: La importancia de la visualización radica en su capacidad de **agilizar la interpretación y el análisis** de grandes volúmenes de datos. Un buen gráfico puede revelar relaciones o problemas ocultos, apoyar argumentos y contar una historia que los números solos no lograrían transmitir. En un mundo saturado de datos, combinar visualizaciones con contexto narrativo (storytelling) logra un mayor impacto en la audiencia. Estudios muestran que la información presentada como historia se recuerda mucho mejor que estadísticas aisladas.

Objetivo del reporte: Documentar las principales técnicas y fundamentos de la visualización de datos, revisar los tipos de gráficos más utilizados y cuándo emplearlos, describir el proceso de *storytelling* con datos (narrativa + visualización) y analizar diversas **herramientas (BI)** y **bibliotecas de visualización** disponibles. Se busca proporcionar una guía completa que abarque desde conceptos básicos hasta ejemplos prácticos con código e imágenes, para entender cómo elegir y crear la visualización adecuada según el mensaje y la audiencia.

Alcance: La investigación abarca (1) fundamentos teóricos de visualización (sistemas de coordenadas, ejes, colores, diseño), (2) clasificación de tipos de gráficos por el tipo de información que representan (cantidad, distribución, proporción, relaciones, geográficos, incertidumbre), (3) el proceso de *data storytelling* paso a paso, (4) revisión de al menos tres herramientas líderes de *Business Intelligence* (Tableau, Power BI, QlikView, Google Data Studio) con sus características, y (5) revisión de al menos cuatro bibliotecas populares de visualización en Python, JavaScript y R (como Matplotlib, Seaborn, Plotly, D3.js, ggplot2, etc.). Finalmente, se presentarán conclusiones comparando enfoques y lecciones aprendidas.

3. Fundamentos de Visualización de Datos

3.1 Conceptos básicos

Sistemas de coordenadas: La mayoría de las visualizaciones se construyen sobre un sistema de coordenadas que define cómo se posicionan los datos en el espacio del gráfico. En las visualizaciones 2D convencionales se emplea típicamente un sistema de coordenadas **cartesianas**, con eje *x* horizontal y *y* vertical. Cada punto o elemento visual se coloca según sus valores en esos ejes. Alternativamente, existen sistemas no cartesianos como las **coordenadas polares**, donde los datos se ubican mediante un ángulo y una distancia radial (usadas, por ejemplo, en gráficas circulares o de rosa de vientos). La combinación de las escalas de posición en cada eje y su disposición define el sistema de coordenadas del gráfico. Es crucial elegir el sistema adecuado según la naturaleza de los datos: por ejemplo, gráficos de pastel utilizan coordenadas polares, mientras que gráficos de líneas y barras usan coordenadas cartesianas.

Tipos de ejes: Cada eje en un gráfico tiene una escala que determina cómo se mapean los datos. Un **eje lineal** presenta incrementos equidistantes en la visualización correspondientes a incrementos iguales en los datosalbertotb.com. Por ejemplo, la distancia entre 0 y 10 es la misma que entre 10 y 20 en la escala gráfica. En cambio, un **eje no lineal** (como el *logarítmico*) comprime o expande ciertas regiones: la separación entre 1 y 10 puede ser equivalente a la de 10 a 100 en el gráfico logarítmico. Las escalas logarítmicas son útiles para datos con rangos muy amplios o distribuciones exponenciales, ya que permiten visualizar proporciones o tasas de cambio en lugar de diferencias absolutas. También existen ejes de **categorías** (nominales, ordinales) que simplemente posicionan cada categoría discreta a intervalos uniformes (sin significado numérico en la distancia), e **ejes de tiempo** que suelen manejar fechas de manera especial (p.ej. considerando meses con diferente cantidad de días). La elección adecuada de la escala del eje es vital para no distorsionar la interpretación; por ejemplo, iniciar un eje de valores en cero es importante en gráficos de barras para respetar la proporcionalidad visual (evitando exagerar diferencias).

Esquemas de colores y psicología del color: El color es uno de los elementos visuales más poderosos y debe usarse con intención. Existen **paletas secuenciales**, **divergentes** y **categóricas** diseñadas para distintos propósitos. En una paleta *secuencial*, los colores van de claros a oscuros

de forma gradual para representar magnitudes de menor a mayor (ideal para valores continuos como temperaturas o densidades). Las paletas *divergentes* combinan dos gradientes que se unen en un color central neutro, siendo útiles para resaltar desviaciones por encima y debajo de un punto medio (por ejemplo, mostrar cambios positivos vs negativos respecto a cero). Las paletas *cualitativas* (o categóricas), por otro lado, utilizan colores distintos (sin orden intrínseco) para representar diferentes categorías o grupos. Al utilizarlas, se recomienda que los tonos tengan brillo similar para no destacar una categoría sobre otra injustificadamente.

El color conlleva además **connotaciones psicológicas y culturales**. Por ejemplo, el rojo suele asociarse con *alerta, error o intensidad*, mientras que el azul transmite *calma y seguridad*. Estos significados pueden variar culturalmente (en algunas culturas el rojo es auspicioso, en otras indica peligro). Por ello, elegir colores debe considerar el contexto: en un gráfico de desempeño, usar verde para "positivo" y rojo para "negativo" aprovecha convenciones culturales ampliamente entendidas. También es importante asegurar accesibilidad, usando paletas *colorblind-friendly* para que personas con daltonismo distingan los elementos (por ejemplo, evitando combinar verde y rojo puros entre sí). En resumen, **un esquema de color bien seleccionado realza la historia** que cuentan los datos, mientras que uno mal elegido puede distraer o incluso sesgar la interpretación.

Principios de diseño visual: Un buen diseño maximiza la claridad y minimiza la distorsión de la información. Uno de los principios esenciales es la **claridad y sencillez:** "menos es más" en visualización. Se debe **eliminar el desorden visual** o elementos innecesarios (*chartjunk*), de modo que la atención se centre en los datos relevantes. Edward Tufte proponía maximizar la *relación dato/tinta*, es decir, que la mayor parte de la "tinta" en un gráfico represente datos, no adornos superfluos. Otra directriz es la **precisión e integridad**: la gráfica debe reflejar fielmente los datos. Esto implica usar escalas apropiadas (por ejemplo, en gráficos de barras, comenzar el eje en cero para que las longitudes sean proporcionales a los valores), evitar distorsiones (como usar imágenes 3D que exageran áreas), y proporcionar etiquetas y notas aclaratorias para contexto. Citar las fuentes de los datos añade transparencia y credibilidad.

El **diseño estético** también juega un papel: un gráfico atractivo y bien compuesto capta más la atención. Esto incluye usar **jerarquía visual** (tamaño, color o contraste) para resaltar los puntos

clave, mantener un estilo coherente (tipografías legibles, colores consistentes con la paleta corporativa si aplica) y asegurar **accesibilidad** (buen contraste de colores para texto, consideraciones para lectores de pantalla, etc.). Además, un buen gráfico cuenta una historia: añadir un título descriptivo y quizá un subtítulo con la conclusión principal ayuda a guiar al lector. Debe haber un **enfoque único** por visualización, evitando mezclar demasiadas ideas en un solo gráfico. En suma, los principios de diseño buscan que la visualización sea **fácil de entender, precisa en su mensaje y visualmente agradable**, de modo que comunique los datos de forma memorable. Un ejemplo de estos principios en acción es estructurar un dashboard corporativo con gráficos sencillos, códigos de color intuitivos (verde/rojo), y textos breves explicando las métricas, logrando una comunicación efectiva en segundos.

3.2 Tipos de representación gráfica

A la hora de visualizar información, es útil escoger el tipo de gráfico según la *naturaleza de los datos* y el *mensaje* que se desea destacar. A continuación, se describen varias categorías de visualizaciones comunes, indicando para cada tipo su definición, usos recomendados y ejemplos.

- **Visualización de cantidad**: Son gráficos diseñados para comparar magnitudes absolutas asociadas a diferentes categorías o elementos. Los ejemplos más típicos son:

    - **Gráficos de barras y columnas:** Consisten en rectángulos de longitud (o altura) proporcional al valor que representan. Se utilizan para comparar cantidades a través de categorías discretas (por ejemplo ventas por producto, población por país) de forma sencilla e intuitiva. Las *barras verticales* (columnas) se prefieren cuando las categorías tienen un orden natural (p.ej. meses del año), mientras que las *barras horizontales* son útiles para categorías con nombres largos o cuando se ordenan de mayor a menor. Variantes incluyen *barras agrupadas* (comparar sub-categorías lado a lado) y *barras apiladas* (mostrar composición de un total dentro de cada barra). **Cuándo usarlo:** siempre que se necesite comparar valores individuales entre grupos; por ejemplo, comparar la frecuencia de respuestas en una encuesta por opción o las utilidades de varias sucursales. **Ejemplo:** Un gráfico de columnas podría mostrar las ventas trimestrales de cuatro productos,

revelando rápidamente cuál vendió más en cada trimestre.

- **Pictogramas (gráficos pictóricos):** Emplean íconos o figuras en lugar de barras para representar cantidades, a menudo repitiendo un ícono por cada *n* unidades o llenando parcialmente un ícono según el porcentaje correspondiente. Su objetivo es hacer la visualización más atractiva y fácil de interpretar por audiencias no técnicas. **Cuándo usarlo:** en infografías o presentaciones al público general, para enfatizar un dato llamativo de forma memorable. Son ideales para resaltar proporciones sencillas (por ejemplo "4 de cada 5 personas…") usando figuras humanas sombreadas en parte. **Ejemplo visual:** Si un reporte señala que el 40% del presupuesto se destina a vivienda, podría mostrarse un ícono de casa relleno al 40%. Los pictogramas hacen los datos más **memorables** y comprensibles, aunque no son precisos para lecturas finas, por lo que se usan mejor para ilustrar *ideas generales* más que análisis detallados.

- *(Nota:* Los gráficos de líneas también representan cantidades a lo largo del tiempo, pero se tratan más adelante en *visualización de relación XY*).

- **Visualización de distribución**: Estas gráficas muestran cómo se dispersan o agrupan los datos en un conjunto, revelando su forma, variabilidad y valores atípicos. Los principales son:

  - **Histogramas:** Parecidos a un gráfico de barras, pero representan la frecuencia de datos dentro de intervalos (bins) de una variable continua. El eje *x* se divide en rangos de valores y la altura de cada barra indica cuántas observaciones caen en ese rango. Así muestran la *forma de la distribución*: si es unimodal o multimodal, simétrica o sesgada, etc. **Cuándo usarlo:** para inspeccionar distribuciones de variables numéricas, p.ej. la distribución de edades de una población, o la distribución de tiempos de respuesta de un servidor. **Ejemplo:** Un histograma de la duración de llamadas de soporte técnico puede revelar que la mayoría duran entre 2–5 minutos y pocas se extienden más de 10 (cola derecha). Los

histogramas facilitan ver dónde se concentran los valores (centro), la dispersión (ancho), presencia de *gaps* o *outliers*.

- **Diagramas de caja y bigotes (boxplots):** Resumen una distribución numérica mediante cinco números clave: mínimo, primer cuartil (Q1), mediana (Q2), tercer cuartil (Q3) y máximo. Se dibuja una "caja" desde Q1 a Q3, con una línea en la mediana, y "bigotes" que se extienden típicamente hasta 1.5 veces el rango intercuartílico; cualquier punto fuera de ese rango se considera valor atípico y se marca individualmente. **Cuándo usarlo:** para comparar distribuciones de varios grupos de datos de forma compacta. Por ejemplo, comparar la distribución de calificaciones de estudiantes entre varias clases, o la dispersión de ingresos en distintas regiones. **Ejemplo:** Un boxplot de salarios por departamento mostraría el rango intercuartílico de sueldos en cada área y destacaría si algún departamento tiene más variabilidad o medianas más altas. Los diagramas de caja permiten observar rápidamente diferencias de dispersión, simetría (si la mediana está centrada o sesgada dentro de la caja) y outliers en múltiples conjuntos a la vez.

- **Gráficos de violín:** Combinan el boxplot con una estimación de densidad de la distribución. Tienen forma simétrica (como un violín) donde el ancho en cada nivel indica la densidad de datos allí. Incluyen típicamente marcadores de mediana e intervalos similares al boxplot dentro de la figura. **Cuándo usarlo:** para comparar distribuciones de manera más detallada que un boxplot, mostrando si tienen múltiples modas (picos) o colas largas. Útiles con suficientes datos para estimar una densidad suave, por ejemplo distribuciones de puntuaciones de distintas pruebas estandarizadas. **Ejemplo:** Un gráfico de violín de la distribución de temperaturas diarias en dos ciudades a lo largo de un año podría revelar que una ciudad tiene distribución bimodal (quizás veranos e inviernos muy diferenciados) mientras la otra es más unimodal. Los violines visualizan tanto estadísticas resumidas (mediana, rango) como la *forma completa* de la distribución, proporcionando una visión más rica de los datos.

- **Visualización de proporción**: Útiles para mostrar cómo se divide un total en partes relativas. Aquí destacan:

    - **Gráficos de pastel o torta:** Representan un total (100%) como un círculo dividido en porciones angulares proporcionales a las categorías. Son muy populares para mostrar **porcentajes de un todo**, por ejemplo la cuota de mercado de distintas empresas sumando 100%. **Cuándo usarlo:** al tener pocas categorías (idealmente ≤ 5) donde se desee enfatizar la parte de cada una en el total. **Ejemplo:** Un gráfico de pastel podría mostrar la distribución porcentual del gasto mensual de una persona (vivienda 30%, alimentos 20%, transporte 15%, etc.). No obstante, su precisión comparativa es limitada (es difícil comparar ángulos exactos), por lo que se deben evitar con muchas categorías pequeñas o valores muy próximos. En tales casos, conviene usar gráficos de barras ordenadas porcentuales para mejor legibilidad.

    - **Gráficos de dona (donut):** Son esencialmente gráficos de pastel con el centro vacío (un agujero), a veces usados por estética. Su lectura es similar al pastel, aunque el espacio central puede utilizarse para colocar el porcentaje total o alguna etiqueta. Las mismas recomendaciones de uso aplican: pocos segmentos, diferencias grandes entre porciones para ser evidentes.

    - **Treemaps (gráfico de árbol):** Muestran jerarquías o desgloses de un total mediante rectángulos anidados cuya área es proporcional al valor. Un treemap divide un rectángulo grande (100%) en sub-rectángulos para cada categoría, y puede subdividirlos recursivamente para subcategorías. **Cuándo usarlo:** para representar proporciones de muchas categorías (más que las que cabrían en un pastel) o mostrar datos jerárquicos (por ejemplo, participación de mercado por empresa y por producto). **Ejemplo:** Un *treemap* de la población mundial podría mostrar cada continente como un rectángulo proporcional a su población, subdividido internamente por países. Permite visualizar rápidamente qué categorías son más grandes y la estructura de sub-partes. A diferencia del pastel,

el treemap maneja bien más de 10 categorías sin volverse ilegible. Es útil también para mostrar *composición dentro de composición* (como gastos por departamento dentro de división dentro de empresa).

- *(Otras visualizaciones de proporción:* gráficos de barras apiladas al 100%, gráficos de áreas apiladas al 100%, diagramas de *waffle*, etc., también muestran la composición porcentual. Por ejemplo, un **gráfico de waffle** llena una cuadrícula de 10x10 celdas donde cada celda es 1%, proporcionando una imagen del porcentaje fácilmente contable. Estas alternativas a veces mejoran la lectura cuando hay muchas categorías pequeñas o cuando se quieren combinar tendencias temporales con composición (áreas apiladas).)

- **Visualización de relación XY**: Gráficos que muestran la relación entre dos variables numéricas (y a veces añaden una tercera).

  - **Diagramas de dispersión (scatter plots):** Plotean puntos en coordenadas cartesianas según dos variables cuantitativas (eje x y eje y). Permiten observar correlaciones, patrones, grupos y outliers. **Cuándo usarlo:** para investigar la relación entre dos métricas; por ejemplo, altura vs peso de individuos, inversión publicitaria vs ventas, etc. También para detectar si existe correlación positiva (tendencia creciente), negativa (decreciente) o nula entre las variables. **Ejemplo:** Un scatter plot de la edad vs ingresos de personas podría mostrar que a mayor edad medianamente más ingresos hasta cierto punto (una nube de puntos subiendo). Los diagramas de dispersión suelen incluir una *línea de tendencia* o regresión para resumir la relación. Son efectivos para resaltar **valores atípicos** (puntos que se alejan del patrón general) y para diferenciar subconjuntos mediante color o forma de los puntos.

  - **Gráficos de burbujas:** Extienden el scatter plot asignando un tamaño (área) a cada punto de acuerdo con un tercer valor. Cada "burbuja" se posiciona por dos variables (x, y) y su tamaño (y a veces color) representan variables adicionales.

Cuándo usarlo: para visualizar conjuntamente tres dimensiones de datos. Por ejemplo, en análisis económico, un gráfico de burbujas puede mostrar países (cada burbuja) posicionados por PIB vs esperanza de vida, y el tamaño de la burbuja representar la población (como hizo famoso Hans Rosling). **Ejemplo:** Un gráfico de burbujas de empresas podría poner en *x* sus ventas anuales, en *y* su crecimiento porcentual, y usar el área de la burbuja para reflejar el número de empleados. Este gráfico revelaría qué empresas son grandes pero quizá crecen lento, etc. Se debe tener cuidado: con muchas burbujas pueden sobreponerse y dificultar la lectura. Asimismo, es esencial aclarar en la leyenda qué significa el tamaño y asegurarse de que el ojo interpretará correctamente el área (nuestro cerebro percibe el diámetro, así que se debe escalar el área proporcionalmente a los datos para evitar engaños). En resumen, las burbujas son útiles para *datasets* pequeños/medianos donde mostrar más variables en un solo plano enriquece el análisis, pero no convienen para cientos de puntos debido al solapamiento.

- *(Otros:* Diagramas de **línea** y **área** – aunque son un caso particular de relación XY donde *x* es típicamente tiempo – muestran relaciones temporales o tendencias. Un gráfico de líneas conecta puntos para enfatizar cambios sucesivos, ideal para series cronológicas (por ejemplo, evolución mensual de ventas). Un gráfico de área es similar pero rellenando bajo la curva, útil para acumulaciones a lo largo del tiempo. Si hay varias categorías, se pueden apilar áreas para ver contribuciones en el tiempo. Estas gráficas se usan cuando el orden secuencial de *x* es significativo. Por ejemplo, la **curva de una función matemática** o la **tendencia de la temperatura diaria** se visualizan naturalmente con líneas continuas.)

- **Visualización de datos geoespaciales**: Son representaciones donde la localización geográfica es parte esencial de la información.

  - **Mapas de calor geográficos (heatmaps espaciales):** Superponen sobre un mapa regiones coloreadas según la intensidad de un fenómeno. Mediante una gradación

de color (por ejemplo de azul a rojo) se indican "puntos fríos" y "puntos calientes" en términos de densidad o valor. **Cuándo usarlo:** para datos geolocalizados que se prestan a agregación por área o para mostrar densidades de puntos de eventos. Un ejemplo es un mapa de calor de la concentración de tweets en diferentes zonas de una ciudad: las áreas coloreadas más "calientes" (rojo) indicarían mayor actividad. También se utilizan en meteorología (mapas de calor de temperatura) o en epidemiología (densidad de casos en un mapa). **Ejemplo:** Un mapa de calor de crimen en una ciudad mostraría con colores más intensos los barrios con mayor incidencia. Sirve para ver patrones espaciales de forma inmediata. A nivel técnico, puede representarse suavizando la influencia de cada punto de datos mediante *kernels* difusos sobre el mapa para producir zonas de calor continuo.

- **Mapas de coropletas:** Dividen el mapa en regiones (países, provincias, distritos, etc.) y rellenan cada región con un color o intensidad proporcional a un valor asociado. Son excelentes para mostrar indicadores relativos en zonas definidas, por ejemplo tasas de desempleo por estado, porcentaje de votación por municipio, etc. **Cuándo usarlo:** al tener datos agregados por región administrativa (u otra división geográfica), para visualizar variaciones espaciales. **Ejemplo:** Un mapa coroplético de la tasa de alfabetización por país podría mostrar gradientes de más oscuro (alta alfabetización) a más claro (baja) permitiendo detectar regiones del mundo con mayores rezagos. Consideraciones importantes: usar paletas apropiadas (secuenciales o divergentes según el caso), normalizar los datos si corresponde (por ejemplo, no colorear por números absolutos de población – que solo remarcarían países grandes – sino por densidad o porcentaje). También, notar que regiones grandes en área dominan la percepción, por lo que a veces un país grande pero con valor medio resalta más visualmente que uno pequeño con valor extremo. Pese a estas limitaciones, los coropletas son una de las formas más comunes de presentar indicadores socioeconómicos o demográficos geográficamente.

- **Cartogramas:** Mapas en los que las áreas de las regiones se deforman (escalan) en proporción a algún valor de interés. En un cartograma de población mundial, por ejemplo, China e India aparecerían inflados enormes, mientras países pequeños en población prácticamente desaparecerían, independientemente de su superficie real. **Cuándo usarlo:** para enfatizar la magnitud de una variable georreferenciada de manera impactante, cuando la distorsión espacial ayuda a contar la historia. Son útiles para resaltar desequilibrios – e.g., un cartograma electoral de EE.UU. escalado por población de votos por estado refleja mejor el peso de cada estado que el mapa geográfico tradicional. **Ejemplo:** Un cartograma de emisiones de CO_2 por país mostraría claramente qué países contribuyen más, ya que esos países "engordarían" en el mapa de acuerdo a sus emisiones. Los cartogramas sacrifican la forma familiar de los mapas en pro de la información cuantitativa, por lo que deben usarse con moderación y siempre acompañados de etiquetas o colores claros para no desorientar al lector. Pueden combinarse con color de coropleta también (e.g. distorsionar por población pero colorear por tasa de pobreza).

- *(Otros:* También existen **mapas de símbolos proporcionales** – colocar en cada ubicación un símbolo (círculo, burbuja) cuyo tamaño refleja un valor –, **mapas de puntos** (un punto por cierta cantidad de unidades, diseminados aleatoriamente dentro de la región), y **cartogramas de contiguidad** o de rejilla, entre otros. La visualización geoespacial es un campo amplio, pero en todos los casos, es fundamental incluir una **leyenda** que explique la codificación de color/tamaño, y a menudo una escala geográfica o indicación de norte para orientar. La **generalización** y proyección cartográfica también importan: por ejemplo, usar la proyección adecuada para no distorsionar áreas si se comparan regiones cercanas a latitudes diferentes.)

- **Visualización de incertidumbre**: Cuando se presentan estimaciones o mediciones con error, es buena práctica incluir visualizaciones de la variabilidad o incertidumbre

asociada.

- **Barras de error:** Son líneas con "capuchones" que se añaden a puntos, barras u otros elementos gráficos para indicar un rango de variación esperado (p.ej. una desviación estándar, error estándar, o intervalo de confianza). Por ejemplo, en un gráfico de barras de medias muestrales, se pueden dibujar barras de error que se extienden una cierta distancia arriba y abajo de la cima de cada barra, mostrando dónde probablemente cae el verdadero valor poblacional con cierto nivel de confianza. **Cuándo usarlo:** siempre que se muestren promedios de datos experimentales, pronósticos o cualquier valor con incertidumbre cuantificable. **Ejemplo:** En una gráfica de la eficacia promedio de distintos tratamientos médicos, agregar barras de error (±1 error estándar) permite apreciar si las diferencias entre tratamientos son estadísticamente significativas (si las barras se traslapan, probablemente no lo sean). Las barras de error proporcionan una *idea general de la precisión* de las mediciones: barras cortas implican datos muy consistentes y confiables, mientras que barras largas indican alta variabilidad o incertidumbre.

- **Bandas de confianza:** En gráficos de líneas (por ejemplo, una curva de tendencia o pronóstico), se suelen añadir bandas sombreadas alrededor de la línea central indicando un intervalo de confianza (por ejemplo al 95%) en cada punto. **Cuándo usarlo:** en modelos predictivos o series temporales pronosticadas, para comunicar no solo la estimación central sino el rango plausible de valores futuros. **Ejemplo:** Un modelo climatológico podría mostrar la temperatura proyectada para las próximas décadas como una línea, con una banda gris alrededor mostrando el rango donde probablemente se ubique la verdadera temperatura. Esto enfatiza que mientras la tendencia central puede ser al alza, hay un margen de error (que típicamente crece con el horizonte temporal).

- **Visualizaciones de distribuciones simuladas:** A veces, en lugar de barras simples, se emplean *simulaciones* o *bootstraps* para ilustrar incertidumbre. Por

ejemplo, se puede dibujar un **fan chart** (diagrama de abanico) como el que usa un banco central para la inflación futura: muestra múltiples bandas de color que se expanden en el tiempo, cada banda abarca un percentil mayor de probabilidad. De cerca, esto es similar a muchas líneas semi-transparentes que forman un abanico de posibles trayectorias.

- El punto clave es **no ocultar la incertidumbre**. Las visualizaciones deben aclarar qué tan confiables son los datos mostrados. Incluir barras o bandas de error es esencial en comunicación científica y técnica. Incluso en visualizaciones para público general, puede usarse anotaciones (p.ej. "margen de error ±3%") para indicar la incertidumbre de encuestas o estimaciones. Esto evita interpretaciones erróneas y transmite *honestidad* en la presentación de los datos.

En resumen, cada tipo de gráfico tiene fortalezas para ciertos propósitos. Por ejemplo, para mostrar composiciones porcentuales se recurre a pastel, donut o treemap; para analizar relaciones entre variables continuas se usan dispersión o burbujas; para distribuciones, histogramas o boxplots; para secuencias temporales, líneas o áreas; para datos geográficos, coropletas o mapas de calor; y para comunicar incertidumbre, barras de error o bandas de confianza. Saber **cuándo usar cada gráfico** es tan importante como saber crearlo: elegir bien la visualización mejora notablemente la comprensión de los datos

4. Proceso de Storytelling con Datos

4.1 Definición y componentes del Storytelling

¿Qué es el Storytelling con datos? En esencia, es el arte de **contar una historia con información basada en datos**. Implica combinar los datos (hechos, cifras) con elementos narrativos y visualizaciones para comunicar un mensaje de forma clara, atractiva y persuasiva. No se trata solo de mostrar números o gráficos sueltos, sino de hilarlos en una narrativa contextual que les dé sentido y conexión para la audiencia. Un buen *data storytelling* responde a

qué dicen los datos, *por qué* es importante y *qué implicaciones* tiene, todo ello de manera que la gente pueda entenderlo y recordarlo fácilmente.

El Storytelling con datos típicamente integra **tres ingredientes clave: datos, visualización y narrativa**:

- **Datos:** la materia prima objetiva (métricas, mediciones, estadísticas) que sustenta la historia. Deben ser datos pertinentes, confiables y preparados adecuadamente (limpios, agregados si hace falta). Idealmente, se seleccionan solo los datos necesarios que apoyan el mensaje principal, para evitar "ruido".

- **Visualización:** los gráficos o elementos visuales que representan esos datos. Una visualización bien elegida (gráfico apropiado, diseño claro) puede iluminar patrones o comparaciones que respaldan el mensaje. La visualización actúa como el puente entre los datos crudos y la audiencia: *traduce* los números a algo que la audiencia puede *ver* y comprender al instante.

- **Narrativa (contexto):** es el hilo conductor que da sentido a los datos y gráficos. Incluye la explicación, el *storyline* que conecta las piezas de información, e incluso elementos de suspenso o sorpresa cuando procede. La narrativa aporta la respuesta al "¿y qué?" de los datos: por qué deberían importarle al público esos números. También establece el contexto (ej. histórico, comparativo) para que los datos cobren relevancia.

Estos elementos se complementan entre sí. Por ejemplo, *datos* + *narrativa* proporcionan explicaciones con contexto, *datos* + *visualización* aportan *iluminación* (insights presentados de forma comprensible), y *visualización* + *narrativa* generan *engagement* (historias visuales que atrapan la atención). Cuando se logra integrar los tres, se obtiene una historia basada en datos capaz de influir e impulsar al público a la acción.

Elementos clave del Storytelling: Todo buen relato con datos debe tener:

- **Contexto:** Situar al público en el escenario adecuado. ¿De qué problema o pregunta se trata? ¿En qué periodo, lugar o situación estamos? Por ejemplo: "En los últimos 5 años, nuestras ventas en línea han desacelerado en comparación con las tiendas físicas...".

- **Personajes o audiencia identificada:** En historias de negocios, el "héroe" puede ser la propia audiencia ("Ustedes, los directivos, tienen el poder de revertir esta tendencia"), o un cliente, etc., pero es importante conectar emocionalmente.

- **Mensaje central:** La conclusión o insight principal que se quiere transmitir. Debe ser claro y repetirse en la historia. Ej: "Los datos muestran que la satisfacción del cliente es el factor clave detrás de las ventas: donde mejoramos servicio, aumentaron las ventas.".

- **Estructura narrativa:** Inicio, nudo, desenlace. Incluso con datos, conviene plantear un *inicio* (contexto y pregunta a responder), un *nudo* (hallazgos, obstáculos, comparaciones) y un *desenlace* (conclusión y llamado a la acción). Esto da fluidez y hace la información más memorable.

- **Anécdotas o ejemplos** (cuando aplican): a veces dentro de la historia con datos se puede insertar una breve anécdota ilustrativa para humanizar los datos. Por ejemplo: al presentar métricas de rotación de empleados, se podría contar *brevemente* la historia de "Ana, una empleada valiosa que dejó la empresa por falta de crecimiento, representando el 10% que vemos en el gráfico".

- **Visuales de apoyo:** Gráficos, imágenes, diagramas que refuercen cada punto clave en lugar de texto plano. Cada visual debe estar integrado en la narrativa (introducido y explicado).

- **Emoción o sentido de urgencia:** Aunque los datos son racionales, las decisiones humanas a menudo se mueven por emoción. Un storytelling efectivo puede incorporar tono emocional apropiado: preocupación por un problema, entusiasmo por una

oportunidad, etc., apoyado en datos.

En resumen, el storytelling con datos convierte un conjunto de cifras en una **historia coherente y persuasiva**. Por ejemplo, en lugar de decir "la cartera de clientes creció 5% este trimestre", un storytelling diría: "Nuestra base de clientes se está expandiendo lentamente - creció solo un 5% este trimestre. Sin embargo, entre esos nuevos clientes hay un segmento joven que está altamente fidelizado. Si nos enfocamos en experiencias digitales (como muestra el gráfico siguiente), podemos acelerar ese crecimiento." Así, se enlaza dato + visual + qué hacer al respecto, en formato narrativo.

4.2 Etapas del proceso

El proceso de construir y comunicar una historia con datos consta de varias etapas secuenciales. Cada etapa asegura que la historia final esté bien adaptada a su audiencia y transmita eficazmente el mensaje deseado:

1. Comprensión de la audiencia: Antes de siquiera analizar datos o hacer gráficas, es crucial definir *¿quién escuchará esta historia?*. Hay que investigar **quién es la audiencia objetivo** y cuál es su contexto. ¿Son ejecutivos de alto nivel interesados en el panorama general? ¿Analistas técnicos que querrán detalles? ¿Público general sin conocimiento del tema? Según eso, se adapta el tono, la profundidad y el estilo. También conviene identificar sus necesidades, preocupaciones o intereses. Por ejemplo, si la audiencia es el departamento de ventas, querrán saber cómo los datos afectan las ventas y comisiones. Como menciona Atlantiasearch, conocer intereses y necesidades permite *"adaptar tu historia a estos intereses asegurando que el mensaje sea relevante"*. Además, evaluar el nivel de alfabetización de datos de la audiencia: si no están familiarizados con ciertos términos o tipos de gráficos, se debe optar por visualizaciones más simples o incluir breves explicaciones. Conocer la audiencia también implica anticipar qué *respuestas espera* o *objeciones podría tener*. Esto guiará qué datos incluir para respaldar la narrativa de forma convincente para ese público en particular.

2. Definición del mensaje clave: Con la audiencia en mente, se determina el **mensaje central o la idea principal** que la historia debe transmitir. Es recomendable que este mensaje pueda

expresarse en una frase concisa. Si no se puede resumir claramente, es señal de que aún no está bien definido. Por ejemplo: "Debemos mejorar la retención de clientes, porque captar uno nuevo cuesta 5 veces más que mantener uno existente". Todos los datos y visualizaciones seleccionados luego deben apoyar este mensaje clave. Este enfoque evita perderse en datos interesantes pero no pertinentes. Tener claro el mensaje orienta todo el proceso: es como el *norte* de la historia. Atlantia Search aconseja identificar el mensaje principal desde el inicio, ya que *"guiará la forma en que presentas los datos y estructuras la historia"*. En esta etapa también se define el objetivo de la comunicación: ¿queremos informar? ¿convencer para una decisión? ¿motivar una acción específica? Por ejemplo, el mensaje clave podría ser "Necesitamos invertir más en marketing digital" y el objetivo persuadir a la directiva de aumentar ese presupuesto. Una vez fijado el mensaje, es útil escribirlo y tenerlo visible mientras se trabaja en la historia, para mantener coherencia.

3. Selección de datos relevantes: Con el mensaje claro, se escogen los **datos que mejor lo sustenten**. Aquí se trata tanto de *encontrar* los datos correctos como de *filtrar* lo irrelevante. Seguramente se cuenta con un amplio conjunto de datos, pero no todos aportarán a la narrativa; incluir demasiado puede confundir o abrumar a la audiencia (¡cuidado con la sobrecarga de información!). Por eso, se identifican las métricas, indicadores o evidencias específicas que ilustran el punto principal y sus subpuntos. Como sugieren las guías, *cada pieza de data seleccionada debe reforzar y clarificar tu historia*. Por ejemplo, si el mensaje es "la rotación de empleados es alta y afecta la productividad", los datos relevantes serían: tasa de rotación trimestral comparada con el benchmark de la industria, resultados de encuestas de clima laboral, y métricas de productividad por equipo. Datos fuera de ese ámbito (p.ej. ventas o satisfacción de clientes) probablemente se omitan a menos que conecten directamente. En este paso también se deben verificar la calidad y origen de los datos (para asegurar credibilidad). A veces es necesario realizar cierto análisis para obtener el indicador relevante (por ejemplo, calcular promedios, tendencias o correlaciones). Además, conviene buscar el *lado humano* en los datos: ¿hay outliers o ejemplos particulares que valga la pena destacar en la historia? Por ejemplo, "La sucursal de Monterrey tuvo 2% de rotación, comparado con 18% nacional – ¿qué hacen diferente allí?" Ese tipo de hallazgo específico puede ser un punto narrativo interesante. En resumen, esta etapa es de *curaduría de datos*: identificar qué números son las "estrellas" de la historia y prepararlos para presentarlos de forma comprensible.

4. Diseño de la narrativa visual: Ahora se construye la **secuencia narrativa** integrando visualizaciones. Esto implica decidir **cómo organizar la historia** (orden de presentación) y qué tipo de gráfico utilizar para cada parte. Una estructura clásica es: **Introducción, Desarrollo y Conclusión**:

- *Introducción:* Presentar el contexto y plantear la pregunta o problema. Aquí se engancha a la audiencia – a veces exponiendo un dato sorprendente inicial o una situación con la que se identifiquen. Por ejemplo: "En 2023 perdimos 50 clientes valiosos – el gráfico muestra la caída repentina en el Q2. ¿Qué pasó?" Se puede iniciar con una visual sencilla (p.ej. una comparación vs. año anterior) que ilustre el problema.

- *Desarrollo:* Desplegar los hallazgos clave en secuencia lógica. Cada hallazgo puede venir con su gráfico de soporte. Es bueno que cada visual responda a una sub-pregunta o apoye parte del mensaje. Por ejemplo: primero un gráfico de barras mostrando las principales razones de baja satisfacción cliente, luego un scatter correlacionando satisfacción con recompra, etc. Aquí se va **construyendo la argumentación** paso a paso. Se recomiendan transiciones claras entre ideas ("Primero veamos cómo ha sido la tendencia... Ahora examinemos las causas...").

- *Desenlace:* Recapitular la historia y enfatizar el mensaje clave y las recomendaciones o acciones resultantes. Puede presentarse un último gráfico resumen o un cuadro comparativo final que condense la conclusión. Por ejemplo: "En resumen, los datos indican que aumentando la satisfacción en 10 puntos, podríamos recuperar $1M en ventas perdidas. **Conclusión:** debemos invertir en capacitación de soporte y seguimiento post-venta." El final debe dar **cierre** y dejar claro qué se aprendió o qué se propone hacer (muchas veces en forma de *llamada a la acción* explícita).

Durante este diseño narrativo, se aplican principios de buena presentación: no saturar cada visual con demasiada información, resaltar en cada gráfico aquello a lo que se refiere la narración (por ejemplo, usando un color distinto para la barra de "nuestro producto" al compararlo contra barras de competidores). Se decide también el formato de entrega: ¿será una presentación tipo *slides*?

¿Un informe escrito? ¿Un dashboard interactivo? Cada formato influye en cómo se hilvana la narrativa. En presentaciones orales, se puede controlar el flujo revelando gráficos de a uno; en un informe escrito conviene que la historia fluya incluso sin voz del narrador, con texto explicativo acompañando a cada figura.

La **coherencia visual** es importante: usar un estilo consistente de colores, tipografías y diseños entre gráficos para que se perciban como parte de un todo. Herramientas como Storyboards o simplemente bocetar en papel la secuencia de gráficos ayudan en esta etapa. El objetivo es tener un *guión visual*: saber qué gráfico va primero, cuál sigue, y qué se dirá con cada uno, hasta culminar en la conclusión. Atlantiasearch recomienda en esta fase delinear el arco narrativo con introducción, puntos clave y conclusión, guiando a la audiencia a través de los insights.

5. Presentación efectiva: Finalmente llega el momento de *contar* la historia al público objetivo. Aquí cuentan tanto las habilidades de comunicación oral/escrita como los materiales visuales preparados. Algunos puntos clave para esta etapa:

- **Ensayar y pulir el discurso:** Especialmente si es una presentación oral, practicar la exposición ayuda a afinar el timing, el énfasis y detectar posibles confusiones. Debes ser capaz de transmitir *lo esencial* incluso sin que la audiencia lea cada número del gráfico. Se suele aconsejar practicar contando la historia en voz alta, asegurando que no exceda el tiempo disponible.

- **No saturar al público:** Evitar leer textualmente diapositivas cargadas de texto o abrumar con tecnicismos. En su lugar, **hablarles mirando** (si es presencial) y usando los gráficos como apoyo visual. Menos es más: es preferible focalizar en unos pocos insights memorables que intentar decir todo. Recuerda: *"No es lo mismo decir todo lo que sabes que transmitir lo que se necesita saber..."*. Este consejo de preparación de speech enfatiza centrarse en lo necesario para la audiencia, no en cada detalle que analizamos.

- **Captar y mantener la atención:** Arrancar con algo impactante (un dato curioso, una pregunta retadora) capta interés. Durante la presentación, usar un tono dinámico, variar la entonación y gestos para enfatizar puntos importantes mantiene el compromiso del

público. Incluir pequeñas pausas dramáticas antes de revelar un punto clave puede aumentar el impacto.

- **Apoyarse en las visualizaciones correctamente:** Asegurarse de explicar cada gráfica: qué representa, cuáles son sus ejes, y sobre todo *qué nos dice*. No asumir que todos interpretan instantáneamente la visualización; guiar la mirada del público ("Como ven en esta línea azul, a partir de junio hay un quiebre..."). Es útil resaltar en el gráfico (con un círculo o flecha) la parte a la que nos referimos en cada momento.

- **Hilo conductor claro:** Aún durante la presentación, hay que recordarle a la audiencia dónde estamos en la historia. Frases como "Visto esto, pasemos al siguiente factor..." o "Recordemos nuestro objetivo: aumentar retención. Ahora veamos qué influye en ella..." refuerzan la cohesión. Esto evita que la gente se pierda en algún punto intermedio.

- **Gestionar preguntas y reacciones:** En presentaciones en vivo, es común recibir preguntas. Conviene anticipar posibles dudas (pensadas en la etapa 1) y tener diapositivas de respaldo o datos extra listas por si se necesitan. Responder con honestidad y, si algún dato no se tiene, ofrecer conseguirlo después. La interacción bien manejada agrega valor y credibilidad.

- **Cierre sólido:** Terminar recapitulando los hallazgos principales y enfatizando el mensaje clave una vez más, quizás con la visual más reveladora como telón de fondo. Incluir un claro *call-to-action* si corresponde ("Por tanto, recomendamos implementar X iniciativa el próximo trimestre"). Un final redondo puede incluir una cita o referencia que refuerce la conclusión, o simplemente una diapositiva final limpia con la frase clave que quieres que se lleven en mente.

En suma, presentar efectivamente es **tanto una habilidad de narración como de visualización**. Un gran análisis pierde impacto si no se comunica bien. Siguiendo estos pasos, se logra que la audiencia no solo *entienda* los datos, sino que *se involucre* con la historia y extraiga significado accionable de ella.

4.3 Mejores prácticas y ejemplos

Estructurar una historia con datos – mejores prácticas:

Para construir un *data story* cautivador y comprensible, se recomienda:

- **Comenzar con un planteamiento claro:** Enunciar desde el inicio la pregunta o problema que se aborda. Esto engancha al público y le da un propósito a los datos presentados. Por ejemplo: "¿Por qué están disminuyendo nuestras ventas en la región sur?"

- **Seguir un orden lógico:** Ya sea cronológico, de mayor a menor importancia, o causa-efecto, la secuencia debe sentirse natural. Cada gráfica o punto debe conectar con el anterior. Es útil pensar en cada sección como respuesta a una posible pregunta de la audiencia en ese punto.

- **Mantener la simplicidad en lo visual y verbal:** Una historia de datos *no es un artículo académico*. Evitar jerga técnica innecesaria; si se debe mencionar un término estadístico, explicarlo brevemente en lenguaje llano. Visualmente, preferir gráficos limpios: no más de 6-7 elementos por gráfico si se puede (p.ej., no incluir 20 barras diminutas – en su lugar resumir o usar otra técnica).

- **Reforzar los puntos clave con visualizaciones potentes:** Identificar cuáles 2 o 3 gráficos son los *heros* de la historia – aquellos que realmente causan el *"aja!"* en la audiencia – y diseñarlos con especial cuidado (rotulados claramente, con el dato clave destacado quizás en otro color). Asegurarse de dedicar tiempo a explicarlos en la narrativa.

- **Usar comparaciones para dar contexto:** Un número aislado dice poco ("obtuvimos 150 quejas este mes"). Comparado con algo cobra sentido ("...eso es 50% más que el mes anterior"). Siempre que sea posible, incluir referencias comparativas (vs. objetivo, vs. histórico, vs. competidores) para que el público entienda la magnitud o relevancia de los

datos.

- **Incluir elementos de storytelling clásico:** Por ejemplo, un *punto de inflexión* (turning point) – "Pensábamos que la causa era A, **pero los datos nos mostraron B**" – crea sorpresa y mantiene el interés. O un *clímax* – la revelación del insight principal hacia el final – seguido de un *desenlace* con la solución o recomendación. Estas técnicas narrativas pueden aplicarse incluso en entornos corporativos para hacer la presentación más memorable.

- **Terminar con un cierre llamativo:** Sea una recomendación accionable, un recordatorio del *porqué* esos datos importan ("...para que podamos mejorar la vida de nuestros clientes y, a su vez, nuestras utilidades"), o una visual final que resuma todo, la audiencia debe irse teniendo claro qué se concluyó y qué se espera que ocurra después.

Errores comunes a evitar: Incluso las historias de datos bien intencionadas pueden fallar por algunos tropiezos típicos:

- *Sobrecarga de información:* Incluir demasiados datos o mensajes a la vez. Esto abruma y confunde al público, diluyendo el mensaje principal. Es preferible contar una historia bien enfocada que intentar decir "todo". Evitar gráficos complejos con múltiples ejes o series si no son absolutamente necesarios; en su lugar dividir en varios gráficos más simples. Recuerda que menos es más: destacar solo los hallazgos más relevantes aumenta el impacto.

- *Escalas engañosas o malas prácticas gráficas:* Un error crítico es manipular la visualización de forma que induzca a error. Ejemplos: empezar ejes de barras en un valor distinto de cero (lo que exagera diferencias), distorsionar proporciones (p.ej. usando imágenes 3D innecesarias que hacen difícil comparar tamaños), o elegir rangos de ejes que minimizan u ocultan cambios importantes. Siempre respetar las convenciones básicas y ser transparente – la integridad en la visualización es esencial para mantener la confianza del público. Si por alguna razón se rompe una convención (p.ej. mostrar solo

una ventana de un eje), debe indicarse claramente.

- *Narrativa desconectada de los datos:* A veces el presentador ya tiene una historia preconcebida y selecciona solo los datos que la apoyan, ignorando los que la contradicen (sesgo de confirmación). Esto puede llevar a conclusiones equivocadas. Es importante que la historia surja de lo que *realmente* dicen los datos y no forzar interpretaciones. Relacionado a esto, si en la exposición verbal se afirma algo, debe mostrarse evidencia visual de respaldo. No hacer afirmaciones al aire sin ilustrarlas, ni mostrar gráficos sin explicar qué implican – ambos casos desconectan narrativa y datos.

- *Falta de enfoque en la audiencia:* Un error es utilizar lenguaje o nivel de detalle inadecuado para el público. Por ejemplo, presentar p-values y intervalos de confianza detallados a gerentes de negocio que solo quieren la implicación práctica. O al revés, dar explicaciones superficiales a analistas que querrían más rigor. Esto se previene conociendo bien a la audiencia (etapa 1) y adaptando la historia a su nivel. También es un error no adecuar el storytelling al tiempo disponible; puede ocurrir que por querer contarlo todo, se termine abruptamente sin concluir. Es preferible priorizar lo importante y tener extras para preguntas, que correr sin haber dado un cierre claro.

- *Visualizaciones mal elegidas:* Usar el tipo de gráfico inapropiado confunde a la audiencia. Ejemplos: emplear un gráfico de líneas para datos categóricos que no tienen continuidad (eso sugiere falsamente una tendencia continua), o usar un pie chart con 10 secciones minúsculas de colores – nadie podrá interpretarlo bien. Siempre escoger la visualización que mejor represente la naturaleza de los datos (ver sección 3.2) y, en caso de duda, optar por la simplicidad. Un gráfico de barras bien rotulado suele ser preferible a un gráfico novedoso pero poco claro.

- *Descuidar la preparación de la presentación:* Esto incluye no ensayar, llevar gráficos con textos ilegibles (p.ej. ejes muy pequeños), o problemas técnicos (¿se ve bien en la sala? ¿el proyector distorsiona colores?). No practicar puede llevar a exceder el tiempo o a explicaciones confusas. La *presentación* es tan importante como el análisis en sí – hay

que pulir ambos. Como dice el dicho: "gran análisis, pobre presentación = mensaje perdido".

Casos exitosos de Storytelling con datos – ejemplos:

A continuación se describen brevemente algunos ejemplos reales donde el storytelling con datos tuvo gran impacto:

- **NASA – Predicción de huracanes:** En 2016 NASA publicó un breve video contando la historia de los huracanes históricos y la probabilidad de otro Katrina. Combinó datos meteorológicos con visualizaciones animadas en un mapa temporal, narrados en lenguaje sencillo. El resultado fue una pieza clara y educativa: mostraba cómo aumentaban ciertas probabilidades y explicaba el contexto del cambio climático. El video fue accesible a público general y es recordado por su claridad para explicar un concepto complejo (riesgo de huracanes intensos) apoyándose en datos científicos visualizados. Es un gran ejemplo de traducir datos técnicos a una historia comprensible y relevante (¿Nos espera otro Katrina? ¿Qué dicen los datos?).

- **Sainsbury's – Historia personalizada al cliente:** La cadena de supermercados británica Sainsbury's en 2020 creó informes personalizados para cada cliente, contándoles la historia de su año de compras. Combinando los datos de fidelización de cada consumidor (qué compró, cuánto gastó) con una narrativa visual atractiva, generaron "un relato personalizado y gráficamente atractivo de su último año". Por ejemplo, a un cliente le podían mostrar: "Visitaste nuestras tiendas 45 veces este año, ¡recorriste 10 km entre pasillos! Tus compras más frecuentes fueron frutas y pan. Ahorraste £50 con promociones…". Este enfoque enganchó a los clientes porque la historia trataba *sobre ellos*, y usaba sus propios datos de forma útil e incluso divertida. El resultado fue muy positivo: los clientes interactuaban más con la marca y el programa de fidelización incrementó su engagement (incluso medios como la BBC comentaron la iniciativa). Además, Sainsbury's obtuvo un ROI diez veces mayor que campañas anteriores, demostrando el poder de un storytelling personalizado para fidelizar.

- **Chit Chart – "Números de película":** Un proyecto de visualización llamado *Chit Chart* convirtió datos de taquilla de cine en una infografía con objetos cotidianos. Por ejemplo, representaron la recaudación de distintas películas usando un icono de *caja de palomitas*: la caja se llenaba proporcionalmente a la recaudación, haciendo muy intuitiva la comparación entre éxitos de taquilla (cajas llenas) y fracasos (cajas casi vacías). Además usaron gráficos con estatuillas de los Oscar para otros datos relacionados. Este caso muestra creatividad en la elección de metáforas visuales, haciendo que los datos financieros fueran accesibles a un público cinéfilo de manera lúdica y memorable. La narrativa se centraba en "los números detrás de las películas" y al usar símbolos del mundo del cine (palomitas, Oscars), conectaron con las emociones y lenguaje del público objetivo.

- **Moritz Stefaner – El sistema alimentario global ("Rhythm of Food"):** Un ejemplo interactivo exitoso es *Rhythm of Food*, creado por el experto en visualización Moritz Stefaner y colegas. Este proyecto tomó datos de búsquedas de Google sobre alimentos a lo largo de varios años y los visualizó como ciclos estacionales – mostrando, por ejemplo, que las búsquedas de "dieta vegana" aumentan cada enero, o que "receta de ponche de huevo" se dispara cada diciembre. La visualización era altamente interactiva, permitiendo al usuario explorar diferentes términos de comida y ver sus patrones anuales. La narrativa que lo acompañaba invitaba a reflexionar cómo nuestras tendencias alimentarias tienen *ritmos* globales. Fue un storytelling sutil: más exploratorio que lineal, pero muy efectivo en hacer tangible la idea de estacionalidad y tradiciones culturales a través de datos de búsqueda. El éxito radica en que los usuarios se veían reflejados ("¡Yo también busco recetas de pavo en Thanksgiving!") y la historia emergía de su interacción. Este caso enfatiza el poder de la interactividad en storytelling: cuando la audiencia puede jugar con los datos, se involucra más profundamente en la "trama" que estos cuentan.

Cada uno de estos ejemplos demuestra cómo combinar datos con una buena narrativa y visualización puede *amplificar el impacto*. Logran que los datos **cobren vida** para la audiencia – sea generando conciencia sobre riesgos naturales (NASA), fidelizando clientes (Sainsbury's), facilitando comprensión de estadísticas de entretenimiento (Chit Chart) o descubriendo patrones

culturales (Rhythm of Food). En todos los casos, se cuidó la claridad del mensaje, se eligieron visualizaciones adecuadas e incluso creativas, y se conectó con los intereses del público. Estas historias exitosas inspiran a profesionales de datos a ir más allá de la mera analítica, *contando historias* que impulsen acción o reflexión. Como conclusión de esta sección, el **data storytelling** bien ejecutado convierte datos en cambios: *convierte insight en entendimiento generalizado y decisiones informadas*.

5. Herramientas de Visualización (Business Intelligence)

Existen numerosas herramientas de Business Intelligence (BI) que facilitan la creación de paneles, informes interactivos y visualizaciones sin necesidad de programar desde cero. A continuación, se investigan cuatro de las herramientas más populares, describiendo para cada una sus características, ventajas, desventajas, tipos de gráficas soportadas, casos de uso recomendados y se incluye una imagen referencial de su interfaz.

Tableau

Descripción general: Tableau es una herramienta líder de BI y visualización de datos, diseñada para hacer el análisis de datos accesible e intuitivo para usuarios de distintos niveles técnicos. Surgida en 2003 a partir de investigaciones de la Universidad de Stanford, se ha posicionado como estándar en muchas industrias por su capacidad de crear **dashboards interactivos** de forma relativamente sencilla. Tableau Desktop (su aplicación principal) permite conectarse a una amplia variedad de fuentes de datos, realizar manipulaciones básicas (joins, cálculos, agrupaciones) y luego arrastrar y soltar campos para construir gráficos. Emplea la filosofía de *VizQL* (Visual Query Language): a medida que el usuario configura visualizaciones, Tableau genera consultas SQL en el fondo automáticamente. Una de sus fortalezas es producir visualizaciones gráficas muy atractivas y personalizables sin escribir código, ideales para presentaciones profesionales.

Características principales: Tableau soporta **múltiples orígenes de datos** (archivos Excel/CSV, bases de datos SQL, Big Data, servicios en la nube, etc.) y puede combinarlos en una misma vista mediante blending o un modelo de datos integrado. Su interfaz es altamente visual: por

ejemplo, al arrastrar un campo categórico a columnas y uno numérico a filas, inmediatamente genera un gráfico de barras. Ofrece una gran variedad de **tipos de gráficas** predefinidas: barras, líneas, áreas, dispersión, mapas geográficos, treemaps, burbujas, boxplots, histogramas, entre otros, además de permitir gráficos combinados y dual-axis. Incluye capacidades de **análisis avanzado** como agrupamiento, predicciones lineales, tendencias, referencias, y extensiones para integrarse con lenguajes como R y Python para análisis estadístico más sofisticado. Tableau destaca por su habilidad para crear *dashboards* (tableros) que combinan múltiples visualizaciones en pantalla, con **filtros y acciones interactivas** (por ejemplo, hacer clic en una parte de un gráfico resalta los datos relacionados en otro). Esto permite a usuarios finales explorar datos por sí mismos de forma controlada.

Otra característica es su **capacidad de compartir**: Tableau Server (o Tableau Cloud) posibilita publicar los dashboards para que otros usuarios los vean vía web o móvil, con seguridad y control de versiones. Además, Tableau es conocido por sus **mapas geográficos** integrados listos para usar: trae coordenadas y formas de países, estados, códigos postales, etc., facilitando crear mapas coropléticos o de símbolos sin conocimientos GIS. En términos de rendimiento, Tableau maneja in-memory caching: puede importar datos (extractos) para acelerar consultas, o trabajar en modo live sobre las fuentes. Su comunidad de usuarios es muy activa, con abundantes *workbooks* de muestra, *forums*, *tutoriales* y visualizaciones públicas que la gente comparte.

Ventajas: Tableau es reconocido por su **facilidad de uso para el usuario de negocio** – su interfaz de *drag-and-drop* y menú contextual lo hacen usable sin saber SQL o programación. Permite crear visualizaciones muy **interactivas y visualmente pulidas** en poco tiempo, lo que agiliza el análisis de datos exploratorio y la creación de informes ejecutivos. Es flexible para conectarse a casi cualquier fuente, y actualiza los dashboards con datos nuevos con facilidad (especialmente usando Tableau Server con actualizaciones programadas). Otra ventaja es su **alto nivel de personalización**: aunque es simple comenzar, ofrece muchas opciones de formato, cálculos de campo personalizados, parámetros, etc., para usuarios avanzados. Asimismo, su rendimiento suele ser bueno incluso con conjuntos de datos grandes (millones de filas), gracias a sus optimizaciones de motor de datos. Tableau es multiplataforma (Windows, Mac) y tiene versión gratuita pública (Tableau Public) para fines no privados.

Desventajas: Una de las principales es el **costo** de las licencias para uso empresarial (puede ser elevado, aunque han migrado a un modelo anual por usuario llamado "Creator/Explorer/Viewer"). Otra limitación es que, aunque ofrece muchas capacidades, no es tan *personalizable* como programar con código: por ejemplo, ciertos gráficos muy a medida o cálculos complejos pueden ser difíciles o imposibles de hacer dentro de sus límites. La curva de aprendizaje, si bien es rápida al inicio, puede volverse pronunciada para dominar funciones avanzadas (tabla de cálculos, LOD expressions, etc.). Tableau funciona mejor con datos bien estructurados; preparar datos muy sucios o con lógicas complejas de transformación a veces requiere pasar por otras herramientas o su producto Tableau Prep. En cuanto a despliegue, la versión Server puede requerir infraestructura y administración de TI. Otro punto: aunque permite incorporar R o Python, para análisis predictivos avanzados a veces es más sencillo realizar esos modelos fuera de Tableau y solo visualizar los resultados en él. Finalmente, algunos usuarios técnicos extrañan un control versión más transparente (los archivos .twb son XML pero no son ideales para diffs de código).

Tipos de gráficas soportadas: Tableau cubre prácticamente todos los gráficos estándar de estadísticas descriptivas:

- Comparación: barras (simples, agrupadas, apiladas), líneas, gráficos de puntos.
- Composición: pastel, treemap, área apilada, barras 100%.
- Distribución: histogramas, boxplots.
- Relación: dispersión (scatter) con opción de burbujas (tamaño por tercera variable), matrices de dispersión.
- Mapas: coropletas, mapas de símbolos proporcionales, densidad (heatmaps geográficos).
- También KPI grandes, tablas cruzadas (texto), gráficas combinadas (por ej. barras y línea de objetivo), diagramas de Gantt, etc.

- Permite *animar* gráficos por una dimensión de página (por ejemplo, evolución temporal animada).

Además, con extensiones o usando *Workbook Developer*, se pueden lograr gráficos más especializados. **Ejemplo visual:** a continuación se muestra la interfaz de Tableau Desktop con un dashboard que contiene un gráfico de barras, un mapa y filtros:. *En la imagen, se aprecia la ventana de Tableau con* hojas (sheets) *de visualización, panel lateral de datos y marcas, y un dashboard armado con diversos componentes.*

Casos de uso recomendados: Tableau se recomienda para **análisis de negocio interactivo** donde analistas o gerentes quieran explorar datos con cierta autonomía, crear paneles de control periódico (ventas, finanzas, marketing, etc.) y compartirlos. Es muy popular en áreas de BI corporativa para informes mensuales, monitoreo de indicadores (KPIs) en tiempo real y storytelling de datos en reuniones (por su presentabilidad). También es útil en proyectos de *data discovery*, donde se prueban varias visualizaciones para encontrar insights en datos nuevos. Por ejemplo, en una empresa minorista, Tableau se usa para dashboard diarios de ventas por tienda y categoría, con posibilidad de que los gerentes filtren por región y vean sus métricas de un vistazo. En investigación o periodismo de datos, Tableau Public se usa para crear infografías interactivas. En resumen, es adecuado cuando se necesita pasar de datos a visualizaciones de forma rápida y flexible, con resultados profesionales, y especialmente cuando se requiere que *otros usuarios interactúen* con los datos (filtrar, resaltar, etc.) sin tener que acudir al analista para cada pregunta.

Power BI

Descripción general: Power BI es la plataforma de Business Intelligence de Microsoft, lanzada comercialmente en 2015, que se ha vuelto extremadamente popular por su integración con el ecosistema Office y su oferta freemium. Consta principalmente de **Power BI Desktop** (aplicación para crear informes en Windows), el **servicio Power BI en la nube** para publicar y compartir, y aplicaciones móviles. Al igual que Tableau, permite conectarse a múltiples orígenes de datos, construir visualizaciones interactivas y paneles. Una de sus fortalezas es una interfaz familiar para usuarios de Excel, con funcionalidades de arrastrar y soltar, y un lenguaje de

fórmulas llamado DAX similar a fórmulas de Excel pero más potente. Power BI ha sido adoptado masivamente en empresas por su bajo coste de entrada (la versión Desktop es gratuita, y el servicio en su versión Pro tiene un precio accesible por usuario) y su facilidad para integrarse con datos de Microsoft (Excel, SQL Server, Azure, etc.).

Características principales: La interfaz de Power BI Desktop está dividida en tres áreas: el panel de campos (lista de tablas y campos), la vista de informe (lienzo donde arrastrar visualizaciones) y la vista de visualizaciones (lista de tipos de gráfico y opciones de formato). Una característica clave es su **interfaz sencilla y flexible**, con muchas opciones de personalización de visualización y control de datos. Cuenta con **Power Query** integrado para importar y transformar datos (usando el lenguaje M), permitiendo pasos de limpieza, fusiones, pivotes, etc., con una interfaz relativamente amigable. Luego, usando DAX, se pueden crear columnas calculadas y medidas (por ejemplo, definiciones de métricas como "Ventas totales" o "Crecimiento %"). Esto lo convierte en una herramienta de modelado de datos ligero, a diferencia de Tableau que espera un modelo ya preparado; en Power BI se puede hacer bastante preparación de datos internamente.

Power BI soporta una **gran variedad de visualizaciones** nativas (barras, líneas, áreas, pie, combo, mapas coropléticos con Bing, mapas de formas, matrices, tarjetas de KPI, embudos, dispersión, histogramas, etc.) y además tiene un marketplace de visualizaciones personalizadas que los usuarios pueden descargar (por ejemplo gráficos de violín, radar charts, word clouds, sankeys, etc., hechos por la comunidad o partners). Todos los gráficos en una página de informe en Power BI son **interactivos y están conectados**: seleccionar un elemento en uno resaltará los datos relacionados en los demás. También admite **segmentaciones (slicers)** y filtros para que el usuario final explore.

Otra característica es su enfoque en **paneles (dashboards)** en la nube: se pueden fijar visuales de distintos informes en un solo tablero, con *tiles* actualizándose en vivo, excelente para monitorización ejecutiva. Power BI se integra estrechamente con otras herramientas de Microsoft – por ejemplo, es fácil exportar datos de un visual a Excel, o integrar con SharePoint, o incluso usar Power BI dentro de Teams. Para usuarios avanzados, permite uso de R y Python para fuentes de datos o visuales personalizados.

Ventajas: Power BI destaca por su **facilidad de uso y adopción** en entornos corporativos Windows. La curva de aprendizaje para alguien que maneja Excel no es muy pronunciada; de hecho, muchos conceptos (tablas dinámicas, fórmulas) son extendidos con DAX. Su interfaz es *muy amigable* y minimalista, enfocada en que cualquiera pueda arrastrar campos y generar gráficos rápidamente. El *coste* es una gran ventaja: la versión Desktop es gratuita para autoría individual; compartir requiere licencias Pro, pero comparado con otras suites BI, es económica. Otra ventaja es la **integración con Office 365**: autenticación por Azure AD, facilidad para consumir datos de OneDrive, listas de SharePoint, etc. Power BI es fuerte en **actualización de datos automatizada**: uno puede programar refreshes (hasta cada 15 min en Premium) de los datasets publicados. También la comunidad de usuarios es enorme, con foros (la comunidad de Power BI), visuales custom disponibles, etc. Al ser Microsoft, la compatibilidad y soporte a diferentes fuentes de datos es amplia (más de 150 conectores). Un punto importante: *Power BI enfoca mucho la parte de ETL (Power Query)*, lo cual es ventajoso si el usuario no tiene otras herramientas de integración, ya que dentro de Power BI puede resolver gran parte del *data prep*. En cuanto a rendimiento, para conjuntos medianos (<1-2GB) es bastante rápido usando su motor de datos columnar en memoria (Vertipaq). Y su capacidad de **compartir informes** via web (con controles de acceso) o incluso *embed* en sitios webs/app es muy valorada.

Desventajas: Una de las limitaciones de Power BI es que, aunque muy poderoso, está disponible solo en Windows (Power BI Desktop no tiene versión Mac, por ejemplo). Otro aspecto: su flexibilidad de personalización visual es un poco menor que Tableau en algunas cosas – por ejemplo, posicionamiento libre de elementos es más limitado, o ajustar cada detalle gráfico a veces no es posible (aunque ha mejorado). También, aprender DAX para cálculos avanzados puede ser complejo, y los modelos de datos con muchas relaciones o medidas complejas pueden volverse difíciles de manejar sin conocimientos de data modeling. La versión gratuita no permite compartir a otros (salvo publicar en web público), así que en práctica para uso organizacional se necesita al menos licencias Pro por usuario. Otro contra es que la versión Pro tiene límites de tamaño de dataset (1 GB por dataset en Pro, ampliable solo con Premium); para *big data* a menudo hay que usar conexiones DirectQuery (consultas en vivo a la base) que pueden ser lentas. Comparado con Tableau, Power BI inicialmente solo permitía conexiones en vivo a pocas fuentes, aunque eso ha mejorado; no obstante, DirectQuery tiene restricciones en transformaciones DAX comparado con datos importados. La versión de colaboración (servicio

web) puede sentirse un poco segmentada: hay workspaces, contenidos, apps – usuarios no técnicos a veces se lían con cómo navegar. Otro aspecto: la personalización fina de *tooltips* o acciones es más limitada (aunque han introducido tooltips de reporte, etc.). Y dado que lanza actualizaciones mensuales, a veces la estabilidad puede verse afectada (aunque en general es robusto). Resumiendo, Power BI puede tener menos "libertad absoluta" en custom visuals comparado a programar tu solución, pero esas son compensaciones por la facilidad de su ecosistema.

Tipos de gráficas soportadas: Soporta todos los gráficos *estándar de BI*:

- Barras/Columnas (simples, agrupadas, apiladas, 100% apiladas), con posibilidad de líneas de meta.
- Líneas y áreas (simples, apiladas).
- Pie chart y Donut chart.
- Scatter plots (y combinado con líneas de regresión básicas).
- Mapas: tanto *filled maps* (coropletas) como *bubble maps* (símbolos), integrados con Bing Maps geocoding.
- Matriz y Tabla (tipo pivot).
- Tarjetas individuales (mostrando un valor KPI grande).
- Combo charts (barra + línea).
- Histogramas, Boxplots (estos con visuales custom o mediante scripts R).
- Slicers (filtros visuales).

- Funnels, Waterfall (cascada), etc.

Además, hay visuales específicos como *Gauge* (medidor semicircular), *TreeMap*, *Ribbon chart*, etc. Y como se mencionó, se pueden añadir visuales personalizados del marketplace: por ejemplo gráficos de Gantt, de embudo, de redes, de WordCloud, etc.

Ejemplo visual: A continuación se muestra una captura de pantalla de la interfaz de Power BI Desktop, con un panel de visualizaciones a la derecha y un informe con gráficos de ejemplo:. *En la imagen se observan un gráfico de columnas y una tabla en un reporte, ilustrando la apariencia de Power BI Desktop con su cinta superior y paneles laterales.*

(Nota: Si la imagen anterior no carga por compatibilidad, se puede describir: la interfaz de Power BI tiene un estilo Office, con visualizaciones colocadas en un lienzo blanco, y en este caso muestra un gráfico de columnas y un cuadro de tarjetas KPI).

Casos de uso recomendados: Power BI es ideal para **entornos empresariales con Microsoft**. Por ejemplo, pequeñas y medianas empresas que ya usan Office 365 encuentran natural adoptar Power BI para crear sus informes de negocio – ventas, finanzas, recursos humanos – de forma centralizada. Es muy útil para *dashboarding* operativo: muchas compañías lo usan para que los gerentes vean diariamente ventas vs meta, estado de inventarios, etc., con datos actualizados del ERP. También es excelente para *democratizar el BI*: al ser de bajo costo, puede distribuirse ampliamente a analistas en diferentes áreas, quienes crearan sus propios reportes básicos. Otro caso de uso común es cuando se quiere migrar del mundo Excel a algo más robusto y compartido: muchos informes que antes se hacían manualmente en Excel ahora se automatizan y comparten vía Power BI. Gracias a su integración con servicios en la nube, es bueno cuando se quieren combinar datos locales y de servicios SaaS. También en entornos académicos o de investigación se usa para crear dashboards interactivos rápidos. Un ejemplo concreto: un departamento de marketing puede usar Power BI para juntar datos de Google Analytics, Facebook Ads y ventas en un solo reporte interactivo, que actualiza cada día, permitiendo ver el ROI de campañas casi en tiempo real. Resumiendo, Power BI se recomienda cuando se necesita una herramienta BI **fácil de aprender, integrada con Excel, con fuerte capacidad de ETL**

interno y un costo accesible, para difundir paneles de control y análisis a muchos usuarios dentro de la organización.

QlikView / Qlik Sense

Descripción general: QlikView es una de las herramientas veteranas de BI, lanzada en los 90s, orientada a *analítica guiada por el usuario*. Junto con Tableau, QlikView fue de las pioneras en *visual analytics*. En años recientes Qlik enfocó más su desarrollo en **Qlik Sense**, una versión más moderna y self-service. No obstante, muchos conceptos son similares, por lo que aquí las tratamos en conjunto como "Qlik". Qlik se distingue por su motor asociativo: carga los datos en memoria y crea un índice asociado que permite exploración muy flexible (por ejemplo, seleccionar un valor en un campo filtra instantáneamente datos relacionados en todos los gráficos, y lo no relacionado queda marcado aparte). La filosofía Qlik es ligeramente distinta a Tableau/PowerBI: en Qlik, cualquier campo se puede asociar con cualquier otro a través de su "asociative engine", permitiendo descubrimientos de conexiones en los datos de manera muy libre. La interfaz de Qlik Sense es web-based, con un hub de trabajo, y arrastrar/soltar visuales también, pero QlikView (el producto clásico) tiene una interfaz Windows más orientada a desarrolladores de BI.

Características principales: QlikView/QlikSense ofrecen **visualizaciones interactivas** similares a las otras herramientas (barras, líneas, mapas, tablas pivote, etc.), pero su diferencial es el *modelo de datos asociativo en memoria*. No requiere esquemas estrella estrictos; los datos se cargan en su motor y Qlik crea una estructura de asociación. La interfaz permite al usuario hacer clic en cualquier valor de cualquier visual o filtro y ver instantáneamente qué otros valores se asocian con ese (mostrando incluso qué valores quedan excluidos en gris, algo característico de Qlik). Esto facilita mucho la **exploración ad-hoc** – es muy flexible para navegar por los datos sin tener que predefinir rutas analíticas.

QlikSense tiene un **entorno de creación responsivo**: los gráficos se reordenan para verse bien en móviles también. Ofrece conectores a múltiples fuentes de datos y un lenguaje de scripting para cargar datos con transformaciones (parecido a SQL mezclado con sintaxis propia). Los gráficos soportan cálculos dinámicos mediante expresiones Qlik (que recuerdan un poco a fórmulas de Excel/PowerBI). Permite crear **extensiones y visualizaciones personalizadas** con

JavaScript, lo que brinda mucha flexibilidad en personalización (aunque eso ya es para desarrolladores). Soporta seguridad a nivel de fila, para restringir datos por usuario.

Una de las capacidades fuertes de Qlik es manejar **datasets grandes en memoria con compresión**; su motor es muy eficiente. QlikSense en su oferta SaaS también integra opciones de análisis aumentados (insights generados automáticamente, etc.). QlikView (el clásico) es más desarrollador-driven: uno diseña el dashboard con bastante control sobre lay-out, pero menos atractivo visual out-of-the-box comparado con QlikSense. QlikSense opta por más *auto-diseño* (p.ej. menos control pixel-perfect para el usuario final, pero diseña adaptativo).

Ventajas: Qlik se ha ganado fama por su potente **motor asociativo**: la habilidad de seleccionar cualquier valor y explorar asociaciones permite a usuarios descubrir datos sin necesidad de saber exactamente qué buscar – es más fácil encontrar outliers o relaciones indirectas. También es muy **flexible y personalizable**: QlikView permitía crear interfaces de usuario casi como aplicaciones (con botones, menús, gráficas muy modificables). QlikSense simplifica la creación, pero aún permite extensiones. Otra ventaja es su **escalabilidad**: Qlik puede manejar cientos de millones de registros en memoria en servidores robustos, sirviendo a muchos usuarios simultáneos. Su enfoque centrado en el usuario como receptor de datos se nota en que la **interacción es muy responsiva**, incluso con múltiples filtros. Ofrece una visión "global" de los datos: siempre se puede resetear y ver todo, luego filtrar – evitando quedarse encajonado en un camino predefinido. Qlik tiene también una comunidad y ecosistema fuerte, y como empresa consolidada ofrece soporte y consultoría global. Para organizaciones que requieren *on-premise* por datos sensibles, QlikSense Enterprise se puede instalar en servidores propios. En comparación con Tableau/PowerBI, Qlik often es preferido por sectores que necesitan manejo de datos complejos con *mucha* interacción asociativa (ej: análisis de fraude, donde hay que ver conexiones entre entidades).

Desventajas: Históricamente, QlikView presentaba un entorno menos amigable para *non-tech* – requería más de un desarrollador BI para armar las apps. QlikSense mejoró en usabilidad, pero algunos consideran que su *UX* no es tan intuitiva como Tableau/PowerBI para un usuario novato. El costo de Qlik suele ser alto (similar o más que Tableau en versiones enterprise). La curación de aprendizaje de su lenguaje de carga y expresiones puede ser empinada si vienes de Excel (es

diferente a DAX/Excel formula). En QlikSense todavía hay menos variedad de visuales nativos que en PowerBI (aunque cubren lo básico, pero cosas como presentaciones de KPI avanzadas a veces requieren extensiones). Otro inconveniente: *consumo de RAM*, Qlik necesita bastante memoria en servidor ya que carga los datos en memoria – esto da performance pero a costo de hardware (aunque con precios de RAM actuales no es tan grave). En QlikSense, algunas funciones de formateo fino no están al alcance del usuario (Focus en simplicidad sacrifica personalización exacta, a menos que programes una extensión). Además, en el mercado actual, Qlik tiene menos share que PowerBI por ejemplo, así que a veces cuesta encontrar tantos recursos/training disponibles a bajo costo (aunque la comunidad Qlik es sólida).

Tipos de gráficas disponibles: QlikSense soporta:

- Gráficos de barras, líneas, combo, áreas, pie, donut.
- Scatter plots, bubble charts.
- Histogramas, box plots.
- KPI single value displays.
- Pivot tables, straight tables.
- Treemaps, Sankey (en versiones recientes o via extension).
- Mapas: QlikSense incorpora mapas para datos geográficos con capas (puntos, áreas).
- Gauge (medidores semicirculares).
- Distribuciones, control charts (posibles via extension).

- En QlikView, similares, quizás lucen más "analíticos" y menos "sexy" por defecto (depende del diseño manual que se les dé).

El *punto fuerte* no es tener gráficos exóticos de fábrica, sino la *asociatividad* entre ellos. No obstante, se puede ampliar con extensiones para gráficos específicos (Qlik Branch comunidad provee varios).

Ejemplo visual: A continuación una imagen ilustrativa de un dashboard en QlikSense que combina un mapa, un gráfico de barras y un filtro, mostrando la interfaz web de Qlik: *[Aquí se podría insertar una imagen de ejemplo de QlikSense]. (Si no se proporciona imagen, describir: QlikSense tiene un estilo limpio, con gráficos sobre fondo claro y filtros arriba; al seleccionar un elemento, las otras visualizaciones filtran y los valores no asociados se muestran atenuados en gris.)*

Casos de uso recomendados: Qlik es muy apreciado en **entornos corporativos de análisis guiado**, por ejemplo:

- Empresas de retail y telecomunicaciones que analizan grandes volúmenes de datos transaccionales y quieren que sus analistas "jueguen" con los datos para encontrar relaciones (p.ej., explorar comportamiento de clientes, cestas de mercado, etc.).

- Áreas de detección de fraude o auditoría, donde hay que saltar rápidamente entre diferentes entidades conectadas: Qlik facilita ver, por ejemplo, que cierto cliente está asociado a múltiples cuentas y direcciones, simplemente clickeando y viendo qué se ilumina.

- Organizaciones que ya vienen de QlikView (décadas de uso) y migran a QlikSense para modernizar, suelen seguir con Qlik por confiabilidad.

- Casos donde se requiere *aplicaciones de BI personalizadas*: QlikView permitió crear dashboards con lógica de negocio incorporada que parecían pequeñas aplicaciones (con menús y todo). Algunas empresas, ej: sector seguros, la usaban para aplicaciones de

tarificación y análisis integradas.

- Donde la **seguridad de datos y on-premise** es crucial: aunque ahora Tableau y PowerBI también ofrecen on-prem, Qlik por mucho tiempo fue la opción robusta autoalojada preferida.

- También se recomienda si se valora la velocidad de respuesta en la exploración ad-hoc: Qlik se suele sentir muy rápido filtrando datos en varias dimensiones.

- En resumen, Qlik es ideal para **analistas de datos experimentados** que quieren total flexibilidad de explorar y diseñar cuadros de mando complejos, y para organizaciones que requieren *analítica de gran volumen con muchos usuarios concurrentes* y alta interacción. Por ejemplo, una multinacional podría usar QlikSense para que cientos de gerentes alrededor del mundo analicen las ventas y stocks, permitiéndoles filtrar por país, línea de producto, vendedor, etc., todo con segundos de respuesta, detectando correlaciones locales que de otra forma se perderían.

Google Data Studio (Looker Studio)

Descripción general: Google Data Studio, recientemente renombrado como Looker Studio, es la herramienta gratuita de Google para crear dashboards y reportes interactivos en la web. Fue lanzada en 2016 como parte de Google Analytics 360 Suite, y ahora es una plataforma independiente. Permite conectar con diversas fuentes de datos (especialmente del ecosistema Google, pero también otras), y construir informes con gráficos, tablas y filtros interactivos, que luego se pueden compartir vía URL. Su principal atractivo es que es **gratuito y basado en la nube**, por lo que cualquier usuario con cuenta Google puede usarlo sin instalar nada. Se orienta mucho a marketing digital y análisis web, por su integración nativa con Google Analytics, Google Ads, YouTube, etc., pero se puede usar para muchos casos más.

Características principales: Data Studio ofrece una interfaz de diseño drag-and-drop similar a PowerPoint/Google Slides para armar los reportes. El usuario puede agregar *componentes*

visuales como gráficos de barras, series de tiempo, tablas, geo-mapas, etc., y vincularlos a una fuente de datos seleccionando métricas y dimensiones. Trae conectores nativos para muchas fuentes de Google (GA, Ads, BigQuery, Sheets, etc.) y también para bases SQL y CSVs, y hay conectores de terceros para otras (por ejemplo, Facebook Insights, etc., a través de partners). Permite **combinar datos de varias fuentes** en un mismo informe (aunque la mezcla de datos de distintas fuentes en un mismo gráfico es limitada, se puede hacer vía "data blending" con hasta 5 fuentes, pero con restricciones).

Es muy **colaborativo**, al ser de Google: varios usuarios pueden editar el mismo informe (control de versiones sencillo mediante historial), y se comparte fácilmente con permisos View o Edit como un documento de Google Drive. Tiene **plantillas predefinidas** para casos comunes (ej: un tablero de Analytics listo para usar). Soporta **controles interactivos**: filtros, períodos de fecha, dropdowns, que afectan a las visualizaciones vinculadas. También se pueden agregar elementos de texto e imágenes para narrar dentro del reporte.

Aunque gratuito, tiene ciertas limitaciones: por ejemplo, no soporta programación de actualizaciones (se actualiza on-the-fly cuando se visualiza, con caché), y las fuentes como BigQuery pueden implicar costos de consulta. Carece de algunas transformaciones avanzadas in situ (se espera que uno traiga los datos ya agregados si es complejo). Sin embargo, para muchas necesidades, es suficiente: crear rápidamente un *dashboard ejecutivo* con métricas clave. Recientemente, al integrarse con Looker, se espera que adquiera más capacidades de modelado (LookML etc.), pero en su estado actual (Looker Studio) sigue muy enfocado a visualización final.

Ventajas: La mayor es que **es gratis y fácil de usar**. Cualquier persona familiarizada con Google puede entrar a Data Studio y en minutos conectar una hoja de cálculo y crear una visualización. La colaboración y uso compartido es tan sencillo como con Google Docs – no hay que preocuparse de licencias por usuario. Es ideal para *startups, pequeñas empresas o equipos de marketing* con presupuestos ajustados. Otra ventaja es la **integración total con otras herramientas de Google**: en unos clics traes tus datos de Google Analytics o Ads sin exportar manualmente, y los mantienes actualizados. Permite *programar entrega por email* de PDFs de los informes. Data Studio es bastante **intuitivo e intuitivo**: las opciones de personalización de

gráficos están en paneles claros (colores, etiquetas, etc.), y no requiere aprender lenguajes (aunque tiene campos calculados con una sintaxis similar a formulas de spreadsheet). Es accesible desde cualquier navegador, y los informes publicados se pueden hacer públicos e incluso embed en páginas web fácilmente. También es **ágil** para iterar: como es en la nube, probar cambios es rápido y se ve en tiempo real.

Dado que es web, es multiplataforma, y los informes resultantes son interactivos en cualquier dispositivo. Data Studio habilita la **creación de temas** (colores, fuentes) para uniformar estilo. Y la comunidad ha creado muchos *connectors* (a diversas APIs) e incluso *visualizaciones comunitarias* con librerías JS personalizadas, ampliando su capacidad. En general, su barrera de entrada cero y amplio ecosistema Google son grandes ventajas.

Desventajas: Al ser gratuito y relativamente sencillo, no tiene el nivel de sofisticación de herramientas pagas. Algunas desventajas:

- **Limitaciones de rendimiento**: con fuentes muy grandes (millones de filas), puede ser lento ya que consulta en vivo (a menos que se use extracciones o BigQuery con caching).

- **Falta de capacidades de modelado de datos**: no es para ETL, uno debe preparar los datos antes; por ejemplo, no es bueno uniendo múltiples tablas relacionales (solo permite uniones mediante "blending" pero es básico).

- **Grado de personalización moderado**: puedes elegir colores y formatos, pero comparado con Tableau/PowerBI, ciertas opciones avanzadas no existen (p.ej. ejes duales solo se implementaron recientemente, control sobre orden de capas en mapas es básico, etc.).

- **Conectividad offline nula**: si tus datos están on-premise y no accesibles por web, integrarlos es complejo (necesitas subirlos a Google Sheets/Drive o BigQuery).

- **Gestión de permisos**: aunque es sencillo compartir, administrar muchos informes para muchos usuarios puede volverse confuso ya que no hay un portal unificado con

organigrama, etc., como sí ofrecen soluciones corporativas (Looker Studio está mejorando esto con 'workspaces').

- No posee funcionalidades avanzadas de *augmented analytics* o *alertas* integradas. Tampoco tiene soporte oficial, dependes de la comunidad (dado que es gratis, Google no da soporte dedicado; hay foros eso sí).

- Para usuarios fuera del ecosistema Google, puede ser menos interesante (aunque se puede usar igual, claro).

Tipos de gráficas soportadas: Data Studio provee los básicos:

- Series de tiempo (líneas o columnas con eje temporal).

- Gráficos de barras/columnas (simples, apilados).

- Gráficos circulares (pastel, donut).

- Tablas (con posibilidad de barras de indicadores en celdas).

- Mapas geográficos (regiones o puntos).

- Tarjetas de valor único (Scorecards).

- Gráficos de áreas, gráficos de dispersión (bubble charts).

- Diagramas de barras apiladas 100%.

- Bullet charts (barras con indicador de meta) – no nativo, pero se puede improvisar o usar visual de comunidad.

- Control de series (line chart with multiple series).

- También tiene "Pivot Table" y "Treemap" recientemente añadidos.

Con *visualizaciones de la comunidad*, se pueden incorporar cosas como word clouds, sankeys, radar charts, etc., programados en JS por terceros. Pero de fábrica la variedad es suficiente para muchos informes de negocios estándar.

Ejemplo visual: Un típico informe de Data Studio podría tener una cabecera con filtros (ej: selector de fecha, filtro de país), luego una fila con KPIs (tarjetas mostrando ventas totales, clientes nuevos, etc.), luego gráficos de barras comparando regiones, un mapa de calor geográfico y una tabla detallada. Por ejemplo, se muestra un dashboard de marketing en Data Studio con gráficos de líneas de tráfico web, barras de rendimiento de campañas y tablas de conversiones: *[Aquí se puede insertar una imagen de ejemplo]*. *En la imagen se apreciaría el estilo limpio de Data Studio, con gráficos en un lienzo y controles de fecha en la parte superior.*

Casos de uso recomendados: Data Studio (Looker Studio) es perfecto para **pequeños equipos de marketing, analistas digitales, startups y consultores** que necesitan crear informes **rápidos, compartibles y sin coste**. Por ejemplo:

- Un especialista de marketing digital usar Data Studio para un *dashboard SEO* que combine datos de Google Analytics (tráfico orgánico), Google Search Console (posiciones de keywords) y redes sociales (con conectores), presentando todo en un reporte para su cliente, actualizado automáticamente.

- Un equipo de ventas en una PYME puede conectarlo a una Google Sheet donde llevan sus ventas mensuales, y tener un informe visual para la gerencia sin invertir en BI costoso.

- Para *reportes periódicos a clientes* en agencias: muchas agencias de publicidad usan Data Studio para dar a sus clientes acceso a dashboards de campaña, así no envían PPTs cada

mes, sino un link interactivo siempre actualizado.

- Proyectos académicos o periodísticos con datos públicos: al ser fácil de publicar de forma abierta, Data Studio se presta para difundir dashboards a audiencia masiva (por ej., un medio digital podría publicar un interactivo de casos COVID por región usando Data Studio embed).

- En organizaciones grandes, puede ser complemento de otras herramientas: por ejemplo, analistas individuales lo usan para prototipos rápidos o para aprovechar conectores a Google services, incluso si luego la solución final corporativa esté en otra plataforma.

- En resumen, se recomienda para **dashboards rápidos, ligeros y ampliamente accesibles**, sobre todo si tus fuentes de datos principales están en Google o fácilmente exportables a Google Sheets/BigQuery. Su facilidad de uso también lo hace ideal en entornos educativos para enseñar principios de visualización sin la sobrecarga de software complejo.

6. Bibliotecas de Visualización (Python, JavaScript, R)

A diferencia de las herramientas anteriores (software listo para usar), las **bibliotecas de visualización** son conjuntos de código que permiten a los desarrolladores crear visualizaciones dentro de entornos de programación. Son más flexibles y personalizables, aunque requieren conocimiento de programación. Abordaremos algunas de las bibliotecas más destacadas en Python, JavaScript y R, detallando su lenguaje, filosofía, características, complejidad, tipos de gráficos soportados, y mostrando ejemplos de código y visual.

Matplotlib (Python)

- **Lenguaje:** Python (biblioteca).

- **Descripción y filosofía:** Matplotlib es la biblioteca fundamental de visualización en Python. Creada por John Hunter en 2003, su filosofía es replicar en Python la

funcionalidad de gráficas de MATLAB, proporcionando amplio control sobre todos los aspectos de un gráfico. Permite generar gráficos 2D estáticos de calidad publicación (y algunos limitados 3D). Es de *bajo nivel*: el usuario construye los elementos (figuras, ejes, líneas, textos) prácticamente desde cero si así lo desea, lo que da gran flexibilidad. La curva de aprendizaje inicial puede ser algo empinada porque hay que entender su API orientada a objetos (Figuras y Axes) y/o su interfaz de estado estilo MATLAB (usando pyplot). Pero justamente por ese detallismo, es posible personalizar casi cualquier detalle de un gráfico. Matplotlib enfatiza la **precisión y control**: uno puede lograr gráficas exactamente como las quiere, aptas para artículos científicos, libros, etc. Ha sido la base sobre la que se construyen otras librerías de más alto nivel en Python (Seaborn, pandas .plot, etc., usan Matplotlib por debajo).

- **Características principales:** Matplotlib soporta un amplio rango de gráficos: desde simples líneas, dispersión, barras, histogramas, hasta complejos gráficos con múltiples ejes, anotaciones y formas. Permite múltiples sistemas de coordenadas en la misma figura, ejes secundarios, logaritmos, formatos de fecha, etc. Posee módulos especializados como matplotlib.mplot3d para superficies 3D simples, matplotlib.animation para animaciones exportables a vídeo/GIF, y herramientas de estilo (se pueden definir *estilos globales* para dar cierta estética a todos los gráficos, e incluso hay estilos predefinidos como "ggplot"). Matplotlib puede exportar a muchos formatos (PNG, SVG, PDF, etc.), lo que lo hace ideal para publicaciones impresas y web. Incluye funciones de evento para hacer gráficos interactivos en entornos GUI (Qt, Tkinter) o en Notebooks (usando backends interactivos). Recientemente, con Jupyter notebooks se integra bien mostrando los gráficos embebidos automáticamente.

Lo destacable es su **nivel de detalle**: se puede controlar ticks, etiquetas, colores, grosores, transparencias, z-order (qué está encima), agregar texto con propiedades tipográficas, flechas, subplots combinados, etc. También puede combinar bien con otras librerías: por ejemplo, con NumPy genera puntos de funciones matemáticas fácilmente, con Pandas gráficas de series temporales. Un concepto fundamental es la separación entre la creación de la figura (objeto Figure) y ejes (Axes), sobre los cuales se dibujan los elementos.

- **Nivel de complejidad:** Moderado-Alto. Para trazar algo simple (p.ej. plt.plot([1,2,3,4])) es trivial, pero aprovechar todo el potencial requiere conocer su API orientada a objetos. La documentación es extensa pero a veces técnica. Muchas veces, usuarios principiantes logran resultados rápidamente con pyplot pero luego tienen que aprender la "forma pythonica" de usarlo (creando objetos fig, ax explícitamente). La complejidad también viene de que es muy *verboso* para algunas cosas: dibujar un gráfico con ejes formateados, anotaciones y leyendas puede requerir muchas líneas de código, en contraposición a librerías de más alto nivel que lo hacen casi automáticamente. Sin embargo, esa verbosidad es recompensada con la exactitud del resultado deseado. Diríamos que para un científico de datos con algo de experiencia, Matplotlib es accesible (gracias a abundantes ejemplos), pero para un programador novel la cantidad de opciones puede abrumar inicialmente.

- **Tipos de gráficas disponibles:** Matplotlib puede generar prácticamente **cualquier gráfico 2D** convencional:

    - Gráficos de líneas (con estilos de línea, marcadores personalizados, múltiples series).

    - Gráficos de dispersión (puntos, con tamaño y color variables).

    - Barras (verticales, horizontales, apiladas manualmente).

    - Histogramas, diagramas de caja y bigote, violin plots (estos últimos requieren esfuerzo manual o usar complementos).

    - Gráficos de pastel (aunque no son su fuerte, se pueden hacer).

    - Mapas de calor (imshow de matrices, etc.), gradientes de colores.

    - Mapas geográficos básicos (coloreando regiones, pero se suele complementar con Basemap o Cartopy para avanzado).
    - Campos vectoriales, diagramas de flujo (quiver).
    - Plots polar, triaxiales, etc.
    - 3D: superficie, scatter 3D, barras 3D, pero son más toscos.

- Y esencialmente, al permitir dibujar líneas y textos arbitrarios, uno puede componer gráficos más complejos (por ejemplo, agregar manualmente anotaciones como estrellas de significancia estadística sobre un barplot).

- **Ejemplo de código básico:** Crear un gráfico de seno y coseno:

```
import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(0, 2*np.pi, 100)

y = np.sin(x)

z = np.cos(x)

plt.plot(x, y, label='sin(x)')

plt.plot(x, z, label='cos(x)')

plt.title('Sine and Cosine Waves')

plt.xlabel('X')

plt.ylabel('Y')
```

```
plt.legend()

plt.show()
```

Este código genera una figura con la curva sinusoidal y cosenoidal. El resultado se muestra a continuación:

Gráfico 1: Ejemplo de gráfico de líneas con Matplotlib. Muestra sin(x) y cos(x) en el intervalo [0,2π].

En esta imagen, se aprecia cómo Matplotlib dibuja ambas curvas con colores diferentes, añade la leyenda y las etiquetas de ejes según el código. Es un gráfico sencillo pero ilustrativo.

- **Ejemplo visual creado:** El gráfico anterior es un ejemplo de la salida visual de Matplotlib integrado en un Notebook. Otro ejemplo: un histograma de distribución y un boxplot superpuesto se puede hacer componiendo funciones de Matplotlib (o usando ax.boxplot). Matplotlib permite incluso disponer múltiples subgráficos:

```
fig, axs = plt.subplots(1,2, figsize=(8,4))

axs[0].hist(data, bins=20, color='skyblue')

axs[0].set_title('Histograma')

axs[1].boxplot(data, vert=False)

axs[1].set_title('Boxplot')
```

(Aquí data sería una lista o array de valores.)

Esto generaría una figura con dos paneles: uno con histograma y otro con boxplot horizontal, mostrando la versatilidad de subplots.

En resumen, **Matplotlib** es la columna vertebral de la visualización en Python. Se caracteriza por su **versatilidad y fine-grained control**. Es la elección cuando se necesita precisión y se está generando gráficos dentro de proyectos Python (ya sea análisis de datos, aplicaciones web con gráficos, etc.). A menudo, en producción, los gráficos generados con Matplotlib se pueden guardar y servir como imágenes, o integrar en interfaces.

Seaborn (Python)

- **Lenguaje:** Python (biblioteca construida sobre Matplotlib).

- **Descripción y filosofía:** Seaborn es una biblioteca de más alto nivel para visualización estadística en Python, creada por Michael Waskom. Se basa en Matplotlib pero ofrece una **interfaz más sencilla** para crear gráficos estadísticos comunes, a la vez que aplica estilos visuales atractivos por defecto. La filosofía de Seaborn es hacer fácil la exploración de **relaciones estadísticas**: cuenta con funciones para gráficos de distribuciones, comparaciones entre categorías con intervalos de confianza, mapas de calor de correlaciones, etc., con una sintaxis mínima. Además se integra muy bien con **pandas DataFrames**: muchas funciones de Seaborn aceptan directamente un DataFrame y nombres de columnas, manejando internamente los datos, lo que simplifica mucho el código. En resumen, Seaborn proporciona funciones de alto nivel para ciertos tipos de gráficos que con Matplotlib requerirían mucho código manual. Por ejemplo, un gráfico de caja con swarms de puntos sobrepuesto, o un *faceted plot* (misma gráfica repetida por subgrupos) son muy fáciles en Seaborn.

- **Características principales:** Seaborn viene con **estilos predefinidos** y paletas de colores agradables (inspiradas en los consejos de visualización para que los gráficos sean claros). Al importar Seaborn suele ajustarse el estilo de Matplotlib globalmente (colores pastel, rejillas sutiles). Sus funciones principales incluyen:

    - `sns.relplot / sns.scatterplot / sns.lineplot` para relaciones (permite fácilmente mapear color o estilo por categorías).

    - `sns.displot / sns.histplot / sns.kdeplot` para distribuciones univariadas o bivariadas (incluyendo opciones de rug plot).

    - `sns.catplot` y variantes `sns.barplot`, `sns.boxplot`, `sns.violinplot` para gráficos categóricos (mostrando puntos resumen con intervalos de confianza, etc.).

    - `sns.heatmap` para mapas de calor (excelente para matrices de correlación).

    - `sns.pairplot` para matrices de gráficos de dispersión (todas contra todas variables).

    - Soporte para *facetado*: la mayoría de funciones de alto nivel tienen parámetros `col, row, hue` para separar la visualización por subgrupos facilmente, generando paneles múltiples.

- Internamente, Seaborn realiza muchos cálculos estadísticos auxiliares: por ejemplo, `sns.barplot` por defecto mostrará la altura de la barra como la media de la categoría y unas líneas negras de error representando ± desviación estándar (o error estándar, personalizable). Esto evita al usuario calcular estas estadísticas manualmente. En `sns.regplot` puede incluso dibujar una regresión lineal con su banda de confianza, calculada con `statsmodels` internamente. En resumen, extiende Matplotlib añadiendo la "capa de intención estadística".

Seaborn se integra con pandas: uno puede proveer un DataFrame y decir x="edad", y="ingresos", data=df y la función extrae esas columnas y plotea (evitando hacer df['edad'] uno mismo). También maneja *variables categóricas* de pandas (dtype category) de forma inteligente para ejes.

- **Nivel de complejidad:** Bajo-Medio. Para alguien que ya sabe Python y tiene datos en pandas, Seaborn es muy sencillo de usar para tareas comunes, más que Matplotlib. Por

ejemplo, dibujar un boxplot de una variable por grupos es una sola línea sns.boxplot(x="grupo", y="valor", data=df). Seaborn abstrae muchos detalles de Matplotlib, por lo que no se necesita especificar tantas cosas. Esto la hace ideal para **exploración rápida**. La curva de aprendizaje de Seaborn es más suave gracias a que hay funciones dedicadas para cada propósito y la documentación (con ejemplos) es clara. Donde puede complicarse es si se quiere personalizar mucho un gráfico más allá de lo que Seaborn permite; en tal caso a veces hay que combinarlo con Matplotlib (por ejemplo, añadiendo anotaciones manualmente sobre lo creado por Seaborn). Pero en la mayoría de casos de análisis de datos, Seaborn produce un resultado suficientemente bueno con mínima configuración. Así que para un científico de datos notorio, Seaborn es de las primeras herramientas aprendidas.

- **Tipos de gráficas disponibles:** Seaborn cubre fundamentalmente **gráficos estadísticos**:
    - Distribuciones univariadas: histogramas, KDE (estimación de densidad), con sns.histplot y sns.kdeplot. También sns.displot que puede combinar hist+KDE.
    - Distribuciones bivariadas: sns.jointplot que muestra dispersión + histogramas marginales; sns.pairplot para matrices de relaciones.
    - Estadísticas categóricas:
        - sns.barplot (media y CI por categoría),
        - sns.boxplot (cajas de distribución),
        - sns.violinplot (violines, mezcla de box + densidad),
        - sns.stripplot (puntos individuales dispersos para ver todos los datos),

        - sns.swarmplot (similar a strip pero evitando solapamientos).

        - Todas estas pueden combinarse; ej: se suele usar sns.boxplot y luego sns.swarmplot sobre el mismo eje para ver puntos.

    - Relaciones: sns.scatterplot y sns.lineplot (este último maneja datos por tiempo o por categoría en eje x).

    - Mapas de calor: sns.heatmap es muy poderoso para matrices (permite anotación de valores, máscaras para triángulos, etc.).

    - Regresiones: sns.regplot / sns.lmplot (que es un wrapper con facetado).

    - Facetado: sns.FacetGrid para crear grillas de subplots fáciles o su versión combinada sns.relplot, sns.catplot que directamente generan facet grid en base a columnas del DataFrame.

- En cuanto a *paletas de colores*, Seaborn trae varias (muted, bright, colorblind-friendly, etc.) y funciones para generar paletas (p. ej. sns.color_palette("coolwarm", as_cmap=True)). Aplica automáticamente paletas adecuadas en hue (por ejemplo, distinguiendo categorías con colores).

  No es tanto para gráficos altamente personalizados (por ejemplo, gráficos geográficos avanzados, gráficas interactivas - no, es estático, salvo integrarlo con matplotlib interactive). Pero cubre lo que un analista necesita 90% del tiempo en EDA (exploratory data analysis).

- **Ejemplo de código básico:** Crear un gráfico de dispersión con regresión:

```
import seaborn as sns
```

```
# DataFrame df con columnas "edad" y "ingresos"

sns.regplot(x="edad", y="ingresos", data=df)
```

Esto en una línea nos da un scatterplot de puntos edad vs ingresos con una línea de regresión ajustada y su banda de confianza. Con Matplotlib hubiera implicado calcular la regresión y dibujar manualmente.

Otro ejemplo, comparativo:

```
sns.boxplot(x="grupo", y="valor", data=df, palette="Set2")

sns.swarmplot(x="grupo", y="valor", data=df, color=".25")
```

En 2 líneas, esto dibuja un boxplot por grupo (color Set2) y sobrepone los puntos individuales semitransparentes. Así se logra visualizar la distribución completa. Seaborn automáticamente calculará posiciones para que los puntos swarm no se sobrepongan demasiado.

- **Ejemplo visual creado:** A continuación un ejemplo de un gráfico de violín con Seaborn. Usaremos el famoso conjunto de datos "tips" (propinas):

```
import seaborn as sns

tips = sns.load_dataset("tips")

sns.violinplot(x="day", y="total_bill", hue="sex", data=tips, split=True)
```

Esto produce un gráfico de violines de la distribución de total_bill (cuenta total) por día de la semana, separando las distribuciones por sexo (hue hombre/mujer) en cada violín, lado a lado. El

parámetro split=True hace que en cada día se divida el violín en dos mitades (male vs female) para comparar dentro de un mismo violín. Automáticamente Seaborn pone colores distintos para hue (por defecto azul y naranja), agrega una leyenda, etc. Este tipo de gráfico múltiple sería laborioso con Matplotlib puro; con Seaborn es directo.

Otra visual: usar sns.pairplot(tips, hue="sex") generaría una matriz de gráficos mostrando relaciones entre todas las variables numéricas del dataset, coloreando por sexo. Es mucho en una línea.

Seaborn configura la estética: tipografía ligeramente más grande, fondo con cuadrícula discreta, colores suaves por defecto. Esto suele resultar en gráficos *más atractivos* sin esfuerzo, comparado con el default de Matplotlib (que tradicionalmente era con background gris oscuro, etc., aunque eso ha cambiado en versiones recientes).

En conclusión, **Seaborn** es el **complemento ideal** a Matplotlib para **visualización rápida y análisis estadístico** en Python. Permite obtener gráficos informativos con poco código, centrados en revelar patrones de datos. Su integración con pandas la hace muy conveniente en notebooks de análisis. Cuando se necesita más control o gráficos específicos, siempre se puede volver a Matplotlib, pero Seaborn cubre un amplio rango con sencillez. Por ello, suele recomendarse a quienes empiezan a visualizar en Python, para generar insights iniciales de sus datasets.

Plotly (Python / JavaScript)

(Nota: Plotly tiene bibliotecas en Python, R, JS; aquí nos enfocamos en Plotly.py pero muchos conceptos aplican similar en JS.)

- **Lenguaje:** Existe la biblioteca **Plotly.js** (JavaScript pura) y encima de ella wrappers como **Plotly.py** para Python, **Plotly R**, etc. Hablaremos combinadamente, centrados en Python interfase pero el núcleo es JavaScript.

- **Descripción y filosofía:** Plotly es una biblioteca para crear **gráficos interactivos y de calidad publicable** en la web. A diferencia de Matplotlib/Seaborn que generan imágenes

estáticas, Plotly genera gráficos interactivos basados en D3.js y WebGL, permitiendo zoom, hover, toggle de series, etc., en un navegador. La filosofía de Plotly es ofrecer una manera de crear gráficas que puedan usarse en dashboards web o visualizaciones en notebooks con interactividad, sin tener que escribir JavaScript directamente. Plotly.py fue creado por la compañía Plotly y es de código abierto. Su API en Python es de más alto nivel (similar a Seaborn en sintaxis en algunos casos), y proporciona tanto una interface de "objetos gráficas" (grafo de figuras, trazas, layout) como una interfaz simplificada llamada **Plotly Express** introducida más recientemente, que funciona de forma declarativa (similar a Seaborn, pasando DataFrames y column names). En JS, Plotly se configura pasando un JSON con estructura de figura (trazas y layout). Plotly se enfoca en **gráficos interactivos, animaciones, y visualización en entornos web**.

- **Características principales:** Plotly soporta una amplia variedad de gráficos, incluyendo muchos tipos avanzados:
    - Gráficos estándar (líneas, barras, scatter) con interacción.
    - Gráficos 3D reales interacciones (superficies 3D rotables, scatter3D).
    - Mapas geoespaciales integrados (usando mapas de Mapbox o geojson) fácilmente.
    - Gráficos especializados: ternarios, radiales, Sankey diagrams, bullet charts, etc. Muchos vienen listos.
    - **Interactividad:** al visualizar, uno puede pasar el ratón y ver tooltips con datos, hacer zoom con arrastrar, pan, seleccionar rango, e incluso en algunos casos editar puntos (si habilitado).
    - **Integración con Dash:** Plotly se integra con la plataforma Dash (del mismo equipo) para construir aplicaciones web interactivas en Python fácilmente.

- **Plotly Express:** es la sub-API que permite hacer gráficos en una línea, similar a Seaborn. Por ejemplo px.bar(df, x="categoria", y="valor", color="subgrupo") generará un gráfico de barras interactivo apilado con subgrupos, sin mucho más código.

- **Cientos de opciones de estilo:** colores (paletas), tipos de línea, ejes log, secundarios, facetado (via px you can facet col/row), etc., plus has default aesthetics that are quite modern.

- Permite **animaciones**: por ejemplo, se puede crear un scatter donde una variable es frame temporal, y al renderizar se obtiene un slider o un play button para animar la evolución (usado popularmente para gráficos estilo Gapminder).

- Exports: se puede exportar los gráficos a HTML (con todo su interactividad, embebible en webs fácilmente), o estátic, se utiliza a veces plotly.io to write to PNG (internally uses orca).

- Plotly en Jupyter Notebook produce un gráfico interactivo embebido (o se puede configurra "static image" si se requiere).

- Posee un **sistema de llamadas callback** en Dash para que los gráficos reaccionen a acciones del usuario, etc.

Una cosa notable: la calidad "visual" de los gráficos Plotly es muy alta, pensada para presentaciones y web. Soporta varias plantillas de estilo pre-hechas (plotly, ggplot2, seaborn, simple_white, etc.). Y la vectorización/performance: para scatter de muchos puntos usa WebGL, pudiendo plotear decenas de miles de puntos sin problema en un browser.

- **Nivel de complejidad:**

    - **Plotly Express (px):** Bajo. Hace trivial muchos gráficos, casi al nivel de Seaborn o incluso más conciso. Perfecto para prototipos.

    - **Plotly Graph Objects (go):** Medio-Alto. Esta es la interfaz donde uno construye la figura instanciando objetos go.Figure, go.Scatter, etc., y ajustando propiedades. Es poderosa (puedes controlar cualquier detalle del JSON final), pero es más verbosa. Sin embargo, para gráficos complejos no soportados directam. por px, se recurre a esto.

    - La interactividad en sí no añade complejidad al usuario, viene por defecto.

Comparado con Matplotlib/Seaborn, la mentalidad es ligeramente diferente: hay que pensar en las gráficas como elementos en un DOM web con propiedades JSON. Pero la documentación de Plotly es buena, con muchos ejemplos. Y dado que se puede partir de px (fácil) y luego tunear con update_layout, update_traces, etc., uno puede iniciar simple e ir refinando.

Complejidad mayor aparece al querer integrarlo en apps (ahí se entra al terreno Dash, que requiere nociones de desarrollo web/pensar reactivamente). Pero solo para crear gráficos en notebook y exportarlos a HTML es sencillo.

- **Tipos de gráficas disponibles:** Realmente es **muy amplio**:

    - Líneas, áreas, barras (agrupadas/apiladas), scatter (2D, 3D).

    - Histogramas, boxplots, violin, hist2d, subplots combinados.

    - Pie charts, Donuts, Sunburst (gráfico circular jerárquico), Treemap, Icicle.

    - Heatmaps, mapas de calor geográficos, choropleth maps, density maps.

    - Contour plots (2D densidad), candlestick (financieros).
    - Sankey diagrams (flujos).
    - Funnel charts, waterfall charts.
    - Pertenecientes a categoría " ciencia ": quiver (campo vectorial), streamtube, iso-surface 3D.
    - Diagramas ternarios (útil en química, etc.), mapas polar (radar charts).
    - Animaciones: cualquier de los ejes se puede mapear a "frame" for animation.

- Y al ser extendible, uno puede crear charts custom definendo shapes, etc. Plotly puede no tener nativamente algunos muy raros (ej: dendogramas, aunque hay fig factory for that).

  Vale resaltar la integración con maps: Por ejemplo, se puede hacer un scattermapbox (puntos sobre mapa interactivo) con pocas líneas, su output es un mapa pan/zoom real con los puntos overlay.

- **Ejemplo de código básico (Python con Plotly Express):**

```
import plotly.express as px

df = px.data.gapminder().query("year == 2007")

fig = px.scatter(df, x="gdpPercap", y="lifeExp", size="pop",
color="continent",

                 hover_name="country", log_x=True, size_max=60)

fig.show()
```

Este clásico ejemplo produce una burbuja por país en 2007, con tamaño por población, color por continente, ejes PIB per cápita vs esperanza de vida con escala log en x. Al visualizar, es básicamente el famoso gráfico de Hans Rosling, interactivo: se puede pasar el mouse para ver país y datos. Con Matplotlib hubiera tomado mucho esfuerzo replicar (y sin interactividad).

Imagen de ejemplo (por texto): Se puede ver un gráfico embedido en HTML con cientos de puntos, ejes titulados, leyenda por continente, etc. Al usuario final se le permiten acciones interactivas.

Otro ejemplo:

```
import plotly.express as px

tips = px.data.tips()

fig = px.bar(tips, x="day", y="total_bill", color="sex", barmode="group",

                facet_col="time", title="Total Bill by Day and Gender")

fig.show()
```

Esto produce un conjunto de gráficos de barras: las cuentas totales por día, separadas por sexo (barras agrupadas azul/rosa), y facetado por "time" (Dinner vs Lunch en dos paneles verticalmente). Todo con color-coded legend, etc.

- **Ejemplo visual creado:** A continuación, se muestra un gráfico interactivo de líneas creado con Plotly Express que representa, por ejemplo, la evolución de casos de COVID-19 en varios países:

```
df = px.data.gapminder().query("country in ['United States','Canada','Mexico']")

fig = px.line(df, x="year", y="lifeExp", color="country", markers=True)

fig.update_layout(yaxis_title="Life Expectancy", title="LifeExp over time")

fig.show()
```

(Este reusa gapminder as placeholder) El resultado es un gráfico de líneas con puntos marcados, 3 series con colores y una leyenda. Se puede pasar mouse para ver año y valor exacto, hacer zoom. Es estéticamente limpio (Plotly theme default).

Plotly (JavaScript) uso: En web, se usaría Plotly.newPlot('div', data, layout), donde data es array de trace objects (e.g., {x:..., y:..., type:'scatter'}). Muchos desarrolladores JS lo usan para embed charts in web apps, as an alternative to writing raw D3 code, because it's simpler.

En R, la biblioteca plotly permite convertir ggplot2 objetos a plotly interactivos, o usar direct approach similar to Python.

- **Conclusión de Plotly:** Es excelente para cuando se requiere **interactividad y publicación web**. Permite a usuarios no técnicos (destinatarios) explorar los datos en el gráfico (ver tooltips, encender/apagar series). Para dashboards web es una solución muy completa junto con Dash. No siempre es lo más adecuado para informes estáticos impresos (aunque puede exportar), en esos casos Matplotlib/Seaborn o ggplot dan más fine control de estética tipográfica, etc. Pero para presentaciones en pantalla, notebooks compartidos, etc., Plotly brinda un *wow factor* de interactividad.

D3.js (JavaScript)

- **Lenguaje:** JavaScript (biblioteca).

- **Descripción y filosofía:** D3.js (Data-Driven Documents) es la célebre biblioteca JS para visualización creada por Mike Bostock. Su filosofía es proveer los **bloques fundamentales** para vincular datos a elementos del DOM (HTML/SVG/CSS) y manipularlos, permitiendo construir casi cualquier visualización personalizada en la web. A diferencia de bibliotecas de alto nivel, D3 no viene con gráficos listos (no hay barChart() predefinido, uno lo construye). Su enfoque es de **muy bajo nivel**: utiliza selectores para unirse a datos, crear elementos SVG por cada dato, y aplicar propiedades escaladas en base a los valores. Por ello, D3 ofrece **flexibilidad total** en el resultado final, al costo de mayor complejidad. Se basa en estándares web (SVG para vectores, Canvas para dibujos, HTML, CSS) y provee utilidades para escalas (lineales, log, de color), ejes, transiciones animadas, generación de formas (líneas, áreas, arcos), layouts (por ejemplo, para pies, stacks, force-directed graphs). D3 es sucesor de Protovis y se volvió enormemente popular en periodismo de datos y visualizaciones a medida en sitios web por su potencia y libertad.

- **Características principales:** D3 es modular; los aspectos destacados:

    - **Enlace de datos al DOM:** mediante d3.selectAll(.foo).data(data).enter().append('circle')... se agregan elementos por cada dato en un dataset. Permite que los datos "conduzcan" qué se muestra.

    - **Escalas y ejes:** D3 tiene un sistema robusto de escalas (d3.scaleLinear, scaleBand para categorías, etc.) que facilitan mapear datos a pixeles. Y generadores de ejes (d3.axisBottom(scale) produce un eje listo con ticks).

    - **Formas (Shapes):** d3-shape incluye generadores para líneas (line charts), áreas (area charts), arcos (para pies/donut), etc., permitiendo describir trayectorias SVG a partir de datos, con interpolaciones suaves si se quiere.

- **Layouts complejos:** D3 incluye o ha incluido en versiones sub-módulos para grafo de fuerza, diagramas de Voronoi, treemaps, pack (burbujas tangentes), chord diagrams, sankey, etc., que calculan las posiciones/layout pero uno luego los dibuja.

- **Transiciones:** Uno de los brillantes es d3.transition() que permite interpolar atributos en el tiempo, creando animaciones suaves (ej: barras creciendo, puntos moviéndose).

- **Interactividad:** al ser puro DOM, se agregan listeners a elementos (e.g., on 'mouseover' cambiar color, etc.) de manera directa. D3 no impone un modelo de componentes, es muy flexible para usar nativamente con HTML/JS.

- **Data manipulation:** D3 tiene d3-array, d3-collection, etc., con funciones de estadísticas (d3.sum, d3.histogram, etc.) y manejo de datos (por ejemplo d3.nest para agrupar, similar a groupBy).

- **Fetch y parseo:** d3-fetch tiene utilidades para cargar archivos CSV, JSON, etc., y d3-time-format para parsear/ formatear fechas con patrones.

- **Integration:** se puede usar D3 tanto en vanill JS como con frameworks (React, Vue, etc., integrándolo).

Básicamente, D3 es **muy de "bricolaje"**: te da piezas (selection, scale, shape, axis, transition) y tú construyes. Eso la hace ideal para *visualizaciones innovadoras o altamente personalizadas* que no se pueden lograr con librerías pre-hechas.

- **Nivel de complejidad:** Alto para principiantes. D3 tiene una curva de aprendizaje notoriamente pronunciada, porque requiere conocer bien JS, nociones de DOM/SVG, y la mentalidad de programación funcional que D3 usa (muchos encadenamientos de métodos, uso de funciones callback para asignar atributos en base a datos). Sin embargo,

la flexibilidad compensa a desarrolladores experimentados. Crear un gráfico de barras simple en D3 requiere muchas líneas comparado con Chart.js o similar. Por eso, a veces se usan wrappers o templates. Pero para un ingeniero de front-end, D3 es una herramienta de referencia. La complejidad reduce conforme uno entiende su *pattern* (enter-update-exit for data binding) y sus modules.

El verdadero poder se ve con la experiencia: alguien con dominio de D3 puede hacer visuales que respondan dinámicamente, con animaciones sutiles y formatos precisos, de un modo que librerías de alto nivel no permiten. El trade-off es mucho desarrollo manual.

- **Tipos de gráficas disponibles:** D3 por sí solo no "tiene gráficas prediseñadas", pero:
    - Con unos cuantos métodos se puede construir: barras, líneas, áreas, dispersión, pastel, donut, histogramas, etc. De hecho, hay muchos ejemplos oficiales o de Blocks (blockbuilder) de Mike Bostock para todos estos casos.
    - Diagramas avanzados: d3-sankey para sankey flows, d3-chord para chord diagrams (cintas entre grupos), d3-hierarchy para treemap, pack (burbujas anidadas), d3-force para redes con fuerza gravitacional, d3-contour, d3-geo para proyecciones geográficas, etc. O sea, D3 provee las bases incluso para visuales muy no triviales (como cluster dendrogram).
    - Interactivos: D3 no distingue, cualquier gráfico se hace interactivo agregando event listeners (zoom, drag, hover, etc.), y se puede combinar con d3-zoom (un comportamiento prehecho para pan/zoom en un container).
    - Mapas: con d3.geo se tienen proyecciones (Mercator, etc.) y se puede dibujar con d3.path geodata (topojson or geojson).
- Ejemplo, con d3-chord se generan posiciones y ángulos de un chord diagram, pero uno mismo dibuja los <path> de chords y los <path> de arcs de grupos.

Resumiendo, no hay un "inaccessible" gráfico si tienes D3: cualquiera imaginable 2D (y hasta 3D en WebGL via d3.geo).

Lo importante es que D3 permite hacerlo de forma nativa en el DOM (SVG), que es escalable y permite estilo CSS.

- **Ejemplo de código básico:** Un fragmento (no completo) para crear círculos representando data en D3:

```
const data = [4, 8, 15, 16, 23, 42];

const width = 420, barHeight = 20;

const x = d3.scaleLinear().domain([0, d3.max(data)]).range([0, width]);

const chart = d3.select("body").append("svg")

    .attr("width", width)

    .attr("height", barHeight * data.length);

const bar = chart.selectAll("g")

    .data(data)

  .enter().append("g")

          .attr("transform", (d,i) => `translate(0, ${i * barHeight})`);

bar.append("rect")

    .attr("width", d => x(d))
```

```
    .attr("height", barHeight - 1);

bar.append("text")
    .attr("x", d => x(d) - 3)
    .attr("y", barHeight/2)
    .text(d => d);
```

Este código (clásico ejemplo de D3 by Mike Bostock) crea un bar chart horizontal con etiquetas. Como se ve, se definió escala x, se seleccionó body->svg, se unió data a groups, se apendió rect y text en cada group, con atributos calculados en función de data. Bastante más código que en otras libs, pero con un control total: por ejemplo, la etiqueta se posiciona x(d)-3 para que vaya al final de la barra con un margin.

- **Ejemplo visual creado:** Dado que este entorno es textual, describimos: el resultado de arriba sería un gráfico de barras horizontales de longitud proporcional a los números (4,8,15...), con los números escritos al final de cada barra. (Este es uno de los primeros ejemplos del tutorial D3). Si quisiéramos, podríamos animar las barras con una transición si los datos cambian, etc., todo con D3.

Otro ejemplo famoso: un scatter de burbujas de Rosling:
D3 no lo trae hecho, pero uno puede:

- Cargar data,
- Escalas x (log), y, radius,
- Ejes generados,

- Crear circles para each country,

- position cx = x(gdp), cy = y(lifeExp), r = sqrt(pop/pi) for area.

- Add fill by continent using a scaleOrdinal of colors.

- Add tooltips by events (not built in, but one can use HTML overlay or title attribute).

- Add a d3.transition between year frames to animate movement.
  Este es un trabajo mayor, pero factible, y de hecho la visual original de Rosling es replicada en D3 often.

- **Conclusión D3:** D3.js es la **herramienta definitiva para visualizaciones web a medida**. Proporciona la mayor flexibilidad pero requiere destreza en programación. Muchas visualizaciones innovadoras, arte de datos, periodismo, etc., se hicieron con D3 (ej: New York Times Graphics often use D3 underhood). Si un proyecto requiere un gráfico altamente customizado que no existe en librerías, D3 es el camino (en manos experimentadas). Para proyectos de rutina (dashboard corporativo típico), D3 tal vez sea "overkill" y se optaría por libs de más alto nivel (e.g., Chart.js, Plotly, etc. en web).

La filosofía de D3 de *data joins* y manipulación directa del DOM con datos ha influenciado muchas otras herramientas. En data-viz, conocer D3 es como conocer el "assembler" de la visualización web: no siempre se usará directamente, pero entender cómo piensa (escala, dom, join) es valioso incluso usando libs derivadas.

ggplot2 (R)

- **Lenguaje:** R (paquete ggplot2 dentro del tidyverse).

- **Descripción y filosofía:** ggplot2 es la biblioteca estándar de visualización en R, creada por Hadley Wickham en 2005-2007 (basada en su gramática de gráficos "Grammar of

Graphics", originalmente de Leland Wilkinson). Su filosofía es declarar gráficos de forma declarativa mediante capas, siguiendo la gramática: se especifica un conjunto de datos, luego mapeos estéticos (aes) de variables a propiedades (x, y, color, shape...), luego se añaden **geometrías** (geoms) que definen qué se dibuja (puntos, líneas, barras, etc.), y se pueden agregar capas estadísticas (stat), facetas, escalas, temas de estilo, etc. Todo con una sintaxis consistente usando + para agregar componentes. ggplot2 abstrae gran parte de los detalles: por ejemplo, dibujar un histograma es simplemente geom_histogram() y ya calcula bins, etc., en función de los datos.

La filosofía es similar a Seaborn/Plotly express en que promueve usar data frames tabulares y mapeos directos sin calcular manualmente cosas. Sin embargo, es más completo: define un **sistema** de composición de gráficos completo. Permite describir un gráfico complejamente de manera concisa y reproducible.

ggplot2 se enfoca en gráficos *estáticos* (salida en dispositivo gráfico R: pantalla, pdf, etc.), aunque hay paquetes complementarios (plotly R or ggiraph) para agregar interactividad a ggplots.

- **Características principales:**
    - **Gramática coherente:** se inicia con ggplot(data, aes(...)) declarando dataset y estéticas globales, luego se añade geoms: e.g. + geom_point() para scatter, + geom_smooth() para línea de suavizado, etc.
    - **Auto-handling de escalas y leyendas:** mapear aes like color=Species hace que ggplot asigne una escala de color discreta y genere una leyenda "Species".
    - **Facetas (facet_wrap, facet_grid):** para hacer paneles por variables de forma trivial.

- **Stats asociados:** muchas geoms llevan stats implícitos, e.g. geom_histogram computa counts, geom_smooth calcula regresión loess por defecto, etc.

- **Temas de estilo personalizables:** theme_bw(), theme_minimal(), etc., y opciones de theme() para cada elemento (fuentes, fondos, etc.).

- **Extensibilidad:** hay un ecosistema de geoms adicionales en paquetes (geom_violin es propio, pero hay extension like geom_split_violin in some packages, etc.). Y uno puede crear geoms personalizados.

- **Facilidad con tidy data:** integra perfecto con dplyr; se puede pipe %>% a ggplot call.

- **Output de alta calidad:** produce gráficos vectoriales, buena tipografía si configuras, etc., aptos para publicaciones.

- **Multitud de geoms:** point, line, bar, boxplot, violin, density, histogram, text, ribbon, tile (heatmap), polygon, etc. Y combos, e.g. geom_boxplot out of the box.

- **Coordinate systems:** can do coord_flip (swap axes), coord_polar (pie charts), coord_map (with map projections using package).

- Los gráficos base generados con ggplot2 lucen bien por defecto (colores de scale_color_hue base, alve backgrounds etc.), aunque muchos los adaptan con theme.

- **Nivel de complejidad:** Moderado. R users a veces tardan en asimilar la gramática si vienen de graficos base de R. Pero una vez entiendes, es muy productivo. Para casos simples, la sintaxis es un liner (similar a Seaborn). Para casos complejos (ej: varios geoms, facet, custom scale breaks, custom annotations), se puede acumular muchos + ..., pero sigue estructurado lógicamente. Documentación y comunidad (StackOverflow,

RStudio cheatsheets) facilitan el learning. Diría para un analista de datos R, ggplot2 es obligatorio y se domina al menos nivel intermedio con la práctica.

7. Conclusiones

Importancia de elegir la visualización adecuada: A lo largo de este reporte quedó patente que no existe un "gráfico universal" válido para todos los datos. Seleccionar la representación correcta es crucial para comunicar el mensaje de forma efectiva y ética. Una visualización bien elegida puede revelar patrones ocultos y facilitar decisiones informadas, mientras que una mal elegida puede confundir o incluso inducir a conclusiones erróneas. Por ejemplo, si se quieren comparar proporciones, un gráfico de barras apiladas puede ser más claro que varios pie charts; si se desea resaltar una tendencia temporal, es preferible una línea a un diagrama de dispersión. También influye el público objetivo: un gráfico complejo (como un diagrama de Sankey) podría ser adecuado para analistas expertos, pero para directivos quizá sea mejor uno más sencillo y enfocado en KPI. Además, la elección adecuada implica considerar aspectos de **precisión e integridad**: por ética, debemos optar por visualizaciones que no distorsionen los datos (respetando ejes, proporciones, etc.). En definitiva, la visualización correcta potencia el entendimiento – "un buen gráfico dice más que mil números" –, mientras que la incorrecta puede entorpecerlo. Por ello, el proceso de Storytelling con datos siempre debe incluir la reflexión "¿es este el gráfico óptimo para mis datos y mi mensaje?". Invertir tiempo en esta decisión redunda en historias de datos más claras, memorables y accionables.

Herramientas vs bibliotecas – comparativa: Ambas aproximaciones tienen su lugar en la caja de herramientas del profesional de datos, pero presentan diferencias:

- Las **herramientas de BI (Tableau, Power BI, etc.)** ofrecen soluciones integrales "end-to-end": conectan fuentes de datos, permiten transformaciones básicas, visualización y compartir dashboards, todo en uno. Suelen tener interfaces amigables (drag & drop) y plantillas gráficas predefinidas. Esto las hace ideales para entornos empresariales colaborativos donde quizás no todos saben programar, o cuando se necesita desarrollar un dashboard en poco tiempo sin escribir código. Son excelentes para *reportería recurrente* y análisis interactivo ligero por parte de usuarios de negocio. Por

contra, a veces tienen limitaciones en personalización extrema o en implementar lógicas muy a medida; uno se ajusta a lo que permite la herramienta.

- Las **bibliotecas de visualización (Matplotlib, D3, ggplot2, etc.)** requieren habilidades de programación, pero ofrecen **flexibilidad absoluta**. Con ellas se puede crear prácticamente cualquier tipo de visual (especialmente con librerías de bajo nivel como D3 o Matplotlib). Brillan en entornos donde el análisis ya está en código (p. ej. dentro de un script de Python o R reproducible) o cuando se requiere integrar gráficos en aplicaciones propias. Permiten automatizar completamente la generación de visualizaciones (por ejemplo, producir 100 gráficos variando parámetros en un bucle es sencillo con código, algo engorroso manualmente en herramientas). También son preferibles para **visualizaciones innovadoras** que salen de lo estándar: con bibliotecas se puede programar la visual deseada, mientras que con herramientas se estaría limitado a sus opciones. La contra es que consumen más tiempo de desarrollo y requieren pruebas (no hay garantias de arrastrar y soltar; hay que codificar y verificar).

En esencia, las herramientas de BI son como "vehículos listos para conducir" – optimizados para llegar rápido a destino en la mayoría de terrenos –, mientras que las bibliotecas son como "cajas de piezas de ingeniería" – con ellas puedes armar lo que imagines, con más esfuerzo.

No hay un ganador absoluto: en un flujo de trabajo profesional, a menudo **conviven**. Por ejemplo, un analista puede explorar datos y crear un dashboard inicial en Power BI para consumo interno rápido, y para un informe técnico formal usar Python+Matplotlib para gráficos específicos con ajuste fino. O una empresa puede tener dashboards regulares en Tableau, pero para su página web cara al público usar D3.js y lograr una visualización interactiva única que refuerce su marca.

Lo importante es saber cuándo usar cada uno: si se busca rapidez y facilidad con datos estructurados conocidos – BI tools. Si se busca automatización integrada en pipeline de datos o un gráfico altamente custom – bibliotecas de código. Conocer ambas facetas amplía las posibilidades del analista.

Aprendizajes personales obtenidos: En la realización de esta investigación se reafirmó que la visualización de datos es tanto un arte como una ciencia. No se trata solo de hacer gráficos bonitos, sino de *comunicar* de manera efectiva información, y para ello se requiere entender tanto las características de los datos como la psicología del espectador. Aprendí la importancia de siempre considerar el contexto: ¿quién verá esta visualización y qué decisión o insight se espera que tome? Eso guía desde la elección de tipo de gráfico hasta el diseño de colores y narrativas de apoyo.

También interioricé que la buena visualización se apoya en un **proceso**: primero comprender los datos y el mensaje (¿qué historia cuentan?), luego diseñar la visualización (bocetar si es necesario), iterar simplificando y destacando lo esencial, y finalmente integrar en una narrativa mayor. Es tentador a veces plasmar "todo el análisis" en un gráfico recargado, pero menos suele ser más – es preferible dividir en gráficos múltiples, cada cual con un punto claro, hilados por el storytelling.

Al revisar las herramientas y bibliotecas, comprendí mejor sus fortalezas y limitaciones específicas. Por ejemplo, vi en práctica cómo Tableau hace muy fácil crear ciertos gráficos complejos con mapas con unos clics, o cómo ggplot2 garantiza consistencia estilística en un reporte completo. A la vez, me di cuenta que no hay excusa para no pulir los detalles: incluso las bibliotecas de más alto nivel permiten ajustarlos (y si no, siempre queda ir a bajo nivel). Pequeñas mejoras – un buen etiquetado, uso apropiado de colores, agregar una anotación explicativa – hacen gran diferencia en la comprensión, y eso lo debo cuidar en mi trabajo futuro.

Otro aprendizaje valioso fue sobre la ética de la visualización: vimos ejemplos de errores comunes a evitar (manipular ejes, distorsionar con 3D innecesario, omitir contexto). Esto me recordó la responsabilidad que tenemos al visualizar datos: mucha gente creerá "lo que ve" en un gráfico, por lo que debemos esforzarnos en representar la realidad de forma veraz y proporcionar el contexto necesario para interpretarla correctamente.

Por último, el proceso de storytelling con datos me enseñó la importancia de **pensar como un narrador** cuando analizo: no basta con obtener hallazgos, hay que saber *contarlos*. Identificar la audiencia, definir qué quiero que recuerden, y estructurar la exposición en introducción, desarrollo y conclusión han pasado de ser conceptos teóricos a algo que trataré de aplicar en cada

presentación que haga. Presentar datos ya no lo veo como mostrar números, sino como *guiar* a la audiencia a través de un viaje de descubrimiento – usando datos y visualizaciones como apoyo.

En síntesis, esta investigación integró conocimientos técnicos (tipos de gráficos, herramientas) con habilidades de comunicación (storytelling, diseño). Me llevo una perspectiva más madura: sé que dominar visualización no es solo saber usar software, sino comprender principios (¿qué hace un buen gráfico?), y tener empatía con la audiencia. Aplicaré estos aprendizajes para hacer informes y dashboards más claros, convincentes y útiles en mis futuros proyectos de ciencia de datos.

8. Referencias bibliográficas

1. Bostock, M., Ogievetsky, V., & Heer, J. (2011). *D³ Data-Driven Documents*. IEEE Transactions on Visualization and Computer Graphics, 17(12), 2301–2309. (Documenta la creación de D3.js, enfatizando su uso de estándares web para visualización flexible)es.wikipedia.orges.wikipedia.org.

2. Few, S. (2012). *Show Me the Numbers: Designing Tables and Graphs to Enlighten* (2nd ed.). Burlingame, CA: Analytics Press. (Clásico libro sobre mejores prácticas de diseño de gráficas, enfatiza claridad, integridad visual y elección adecuada de gráficos).

3. Munzer, T. (2014). *Visualization Analysis and Design*. Boca Raton, FL: CRC Press. (Cobertura académica de principios de visualización, tipos de datos y elecciones gráficas, con fundamentos de percepción visual).

4. Wickham, H. (2010). *A Layered Grammar of Graphics*. Journal of Computational and Graphical Statistics, 19(1), 3–28. (Presenta la gramática subyacente a ggplot2, explicando conceptos de datos, estéticas, geoms, facetas, etc., base teórica de visualización declarativa en R)datascience.recursos.uoc.edu.

5. Kirk, A. (2016). *Data Visualisation: A Handbook for Data Driven Design*. Sage Publications. (Guía práctica abarcando desde obtención de datos hasta diseño y narrativa visual, con muchos ejemplos de casos exitosos y errores a evitar).

6. The Information Lab España (2024, May 15). *Importancia de la visualización de datos*. [Blog post]. Recuperado de The Information Lab blog: *(Destaca que la visualización facilita interpretación de grandes volúmenes de información y mejora la toma de decisiones)theinformationlab.estheinformationlab.es*.

7. Bravo, J. (2022). *Data storytelling: la importancia de contar historias con tus datos*. Datos.gob.es (Portal datos abiertos Gobierno de España). *(Explica qué es el storytelling con datos y sus ingredientes: narrativa, visualización, datos, y cómo combinarlos*

*eficazmente)*datos.gob.esdatos.gob.es.

8. Microsoft Docs. (2021). *Herramientas de análisis de datos: ventajas de Power BI, Tableau y Qlik*. Kizeo Forms Blog. *(Comparativa de tres herramientas BI líderes, describiendo su interfaz, características y puntos fuertes para distintos usos)*kizeo-forms.comkizeo-forms.com.

9. Waskom, M., et al. (2021). *Seaborn: Statistical Data Visualization*. [Documentation]. *(Guía oficial de Seaborn v0.11, muestra ejemplos de gráficos estadísticos y cómo integran cálculos y visualización)*datascientest.comdatascientest.com.

10. Plotly Technologies Inc. (2023). *Plotly.py Documentation*. *(Documentación de la biblioteca Plotly Express y Graph Objects en Python, con ejemplos de código para distintos tipos de gráficos interactivos)*mappinggis.commappinggis.com.