Universidad Anáhuac México Norte: Maestría en Estadística

Inferecia Estadística - Estimación Paramétrica

Proyecto Final: Mixturas Gaussianas para Modelar Retornos Financieros

Paulina Nayeli Cueto Romero
Carlos Guillermo Mayorga Tapia

Objetivo

El objetivo de este proyecto es analizar estadísticamente la distribución de los rendimientos logarítmicos del Índice de Precios y Cotizaciones (IPC) para identificar sus principales características, como simetría, curtosis, presencia de eventos extremos y posibles desviaciones respecto a la distribución normal. A partir de este análisis, se busca evaluar la relevancia de utilizar modelos probabilísticos basados en la distribución normal frente a alternativas más flexibles, como las mezclas gaussianas, que capturen de mejor manera las características observadas en los datos.

Introducción

En los mercados financieros, los rendimientos de los activos son una de las variables más importantes para el análisis de riesgo y la toma de decisiones de inversión. El Índice de Precios y Cotizaciones (IPC) es el principal referente del mercado accionario en México y su estudio permite obtener información sobre la dinámica general del mercado. Tradicionalmente, los modelos de asumen que los rendimientos siguen una distribución normal; sin embargo, diversos estudios empíricos han mostrado que los rendimientos presentan asimetrías, leptocurtosis y colas más pesadas que las predichas por la normalidad, lo que implica una mayor probabilidad de eventos extremos.

En este proyecto se analizan los rendimientos logarítmicos diarios del IPC mediante herramientas gráficas y estadísticas, incluyendo histogramas, boxplots y el cálculo de momentos como la curtosis. Posteriormente, se ajusta un modelo normal por máxima verosimilitud y se contrasta con una mezcla gaussiana de dos componentes, utilizando técnicas de estimación como el algoritmo de Expectation-Maximization (EM). Este análisis busca no solo describir el comportamiento histórico de los rendimientos, sino también evaluar qué modelo estadístico ofrece un mejor ajuste para representar su comportamiento.

Librerias

In [1]:
```
import scipy.stats
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import kstest, kurtosis, norm
```

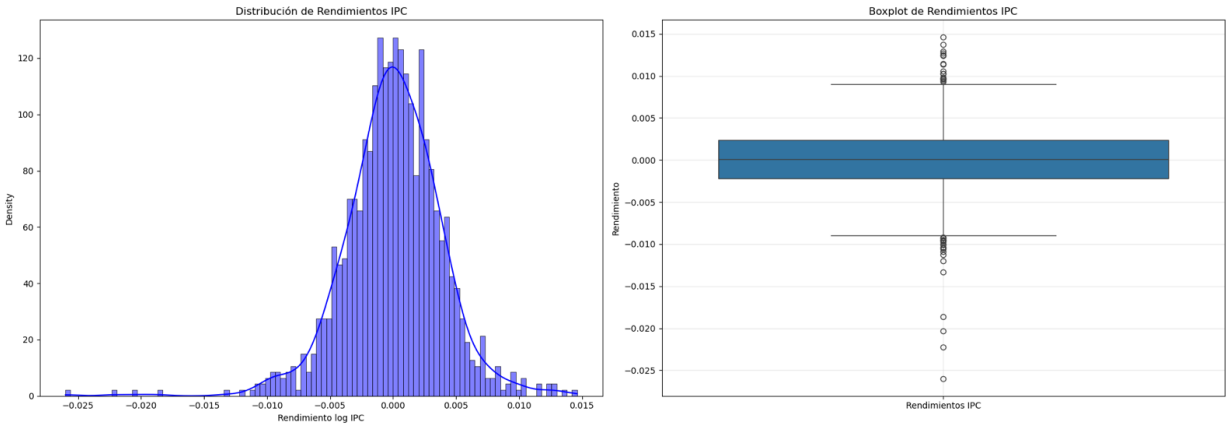
Data Wrangling

Importación y manejo de los datos, listos para su uso en el análisis.

In [2]:
```
df = pd.read_excel('./data/IPC.xlsx')
df = df[['Date', 'Rendimientos IPC']]
df.set_index('Date', inplace=True)
df = df.drop(df.index[0])
```

Análisis Exploratorio

In [3]:
```
# --- Histograma ---
plt.figure(figsize=(20, 7))

plt.subplot(1,2,1)
sns.histplot(df["Rendimientos IPC"], bins=100, kde=True, stat="density",
plt.title("Distribución de Rendimientos IPC")
plt.xlabel("Rendimiento log IPC")
plt.tight_layout()

# --- Boxplot ---
plt.subplot(1,2,2)
sns.boxplot(df, vert=True, patch_artist=True)
plt.title("Boxplot de Rendimientos IPC")
plt.ylabel("Rendimiento")
plt.grid(alpha=0.3)

plt.tight_layout()
plt.show()
```



Histograma con Densidad:

La primera gráfica muestra la distribución empírica de los rendimientos logarítmicos del IPC. Se observan tres características importantes:

1. La distribución es aproximadamente simétrica alrededor de cero, lo que sugiere que no hay sesgo importante en los rendimientos.
2. Comparada con una normal, se observan colas algo más gruesas, lo que implica mayor probabilidad de eventos extremos, sobre todo en rendimientos negativos.
3. La mayor parte de los datos está muy cerca de cero, lo que es típico en series de rendimientos financieros, donde los cambios diarios suelen ser pequeños, sugiriendo una distribución leptocúrtica. En conjunto, el histograma sugiere que un modelo gaussiano puede ser una primera aproximación, pero que sería recomendable considerar modelos que capturen colas más pesadas y la leptocurtosis.

Boxplot:

El boxplot complementa la interpretación mostrando:

1. Mediana cerca de cero: Consistente con el histograma, confirma la simetría.
2. Se observan puntos alejados de los bigotes, tanto en el extremo positivo como en el negativo, reflejando episodios de alta volatilidad. Estos outliers corresponden a los valores en las colas del histograma.
3. El rango intercuartílico es relativamente pequeño, sugiriendo una distribución leptocúrtica.

In [4]:
```
# Curtosis
kurtosis = df.kurtosis()
print("Curtosis Empírica:", kurtosis.iloc[0])

# Curtosis teórica de una normal ajustada
kurt_fisher = scipy.stats.norm.stats(loc=0, scale=1, moments='k')  # 'k'
print("Curtosis Teórica:", float(kurt_fisher) + 3)
```

```
Curtosis Empírica: 3.9638983278014646
Curtosis Teórica: 3.0
```

La curtosis obtenida para los rendimientos logarítmicos del IPC es de aproximadamente 3.96, superior al valor de 3 correspondiente a una distribución normal. Esto indica que la serie presenta un comportamiento leptocúrtico, caracterizado por una mayor concentración de datos en torno a la media y la presencia de colas más pesadas, lo que implica una mayor probabilidad de observar eventos extremos (grandes ganancias o pérdidas) que la que predeciría una distribución normal. Este hallazgo sugiere que el uso de modelos que consideren colas pesadas resulta más adecuado para representar el comportamiento de los rendimientos.

Modelado $N(\mu, \sigma^2)$

Para modelar los rendimientos logarítmicos del IPC, se asumió que siguen una distribución normal $N(\mu, \sigma^2)$. Los parámetros de esta distribución fueron estimados mediante el método de máxima verosimilitud (MLE), que consiste en encontrar los valores de μ y σ^2 que maximizan la función de verosimilitud:

$$L(\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right)$$

Maximizando el logaritmo de la verosimilitud y derivando respecto a $N(\mu, \sigma^2)$, se obtienen los estimadores:

$$\hat{\mu}_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

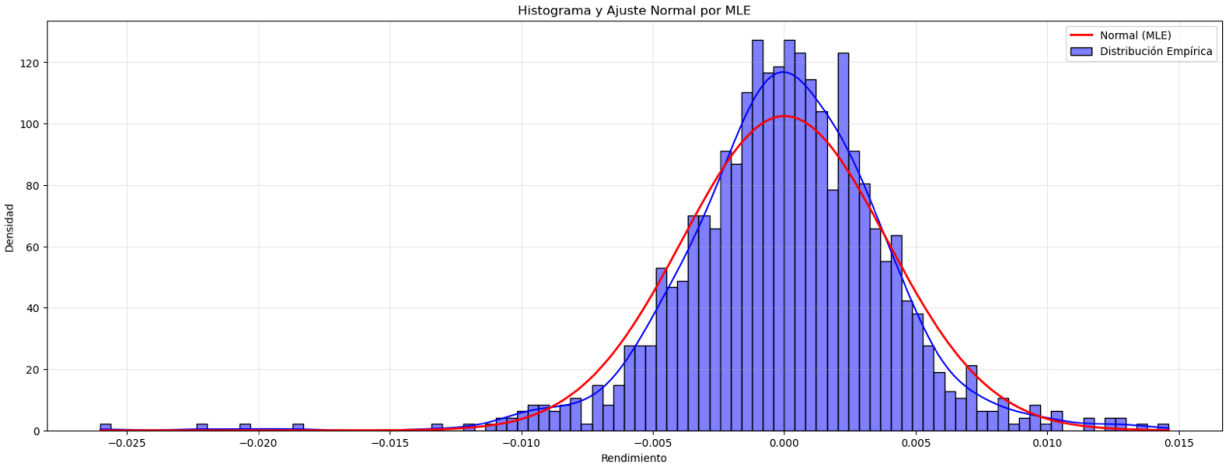
$$\hat{\sigma}^2_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} \left(x_i - \hat{\mu}_{\text{MLE}}\right)^2$$

Estos corresponden a la media y varianza muestral. Con estos parámetros se construyó la función de densidad normal ajustada, que fue superpuesta sobre el histograma de los datos para evaluar visualmente el grado de ajuste.

In [5]:
```
# --- Ajuste por MLE ---
datos = df.values.flatten()  # convertir a array
mu_mle = np.mean(datos)                # media MLE
sigma_mle = np.std(datos, ddof=0)      # desviación estándar MLE (ddof=0)

print(f"Media MLE: {mu_mle:.6f}")
print(f"Desviación estándar MLE: {sigma_mle:.6f}")
```

```
Media MLE: 0.000012
Desviación estándar MLE: 0.003893
```

In [6]:
```
# --- Graficar histograma con densidad normal ---
plt.figure(figsize=(20, 7))
sns.histplot(df["Rendimientos IPC"], bins=100, kde=True, stat="density",
plt.title("Distribución de Rendimientos IPC")
plt.xlabel("Rendimiento log IPC")

# Curva de la normal ajustada
x = np.linspace(min(datos), max(datos), 200)
pdf = norm.pdf(x, loc=mu_mle, scale=sigma_mle)
plt.plot(x, pdf, 'r-', lw=2, label='Normal (MLE)')

plt.title("Histograma y Ajuste Normal por MLE")
plt.xlabel("Rendimiento")
plt.ylabel("Densidad")
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```



La figura muestra el histograma de los rendimientos logarítmicos del IPC (en azul) junto con la densidad de probabilidad de una distribución normal ajustada por máxima verosimilitud (línea roja).

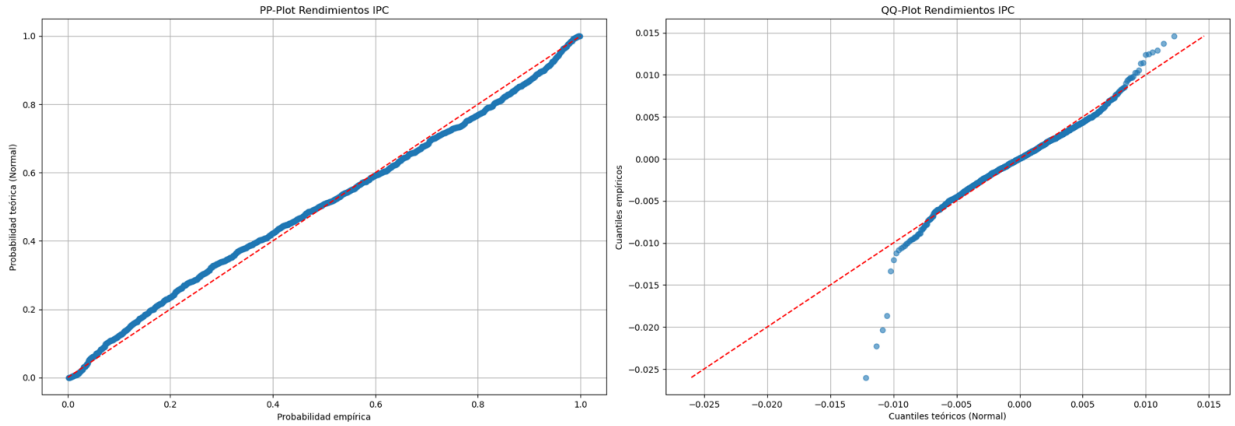
Se observa que la distribución empírica es aproximadamente simétrica y está centrada en cero, lo cual es consistente con la ausencia de sesgo sistemático en los rendimientos. Sin embargo, la comparación visual revela que la distribución empírica presenta colas más pesadas que la normal ajustada, así como una mayor concentración de valores en torno a la media.

Este comportamiento confirma la presencia de leptocurtosis, lo que implica que la probabilidad de observar movimientos extremos (grandes pérdidas o ganancias) es superior a la que predice el modelo normal. Por lo tanto, aunque el ajuste gaussiano captura de manera razonable el comportamiento central de los datos, subestima la frecuencia de eventos en las colas, lo que tiene implicaciones importantes para el modelado de riesgo financiero.

In [7]:
```
# KS test para IPC
ks_ipc = kstest(df["Rendimientos IPC"], "laplace", args=(mu_mle, sigma_ml
print("KS Test IPC -> estadístico:", ks_ipc.statistic, "  p-value:", ks_i
```

```
KS Test IPC -> estadístico: 0.06952138302224709   p-value: 2.526877209860
514e-05
```

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

def pp_qq_plots(data, mu, sigma, title_pp="PP-Plot", title_qq="QQ-Plot"):
    data_sorted = np.sort(data)
    n = len(data_sorted)
    probs = np.arange(1, n+1) / (n+1)

    # --- PP Plot ---
    probs_emp = probs
    probs_theo = norm.cdf(data_sorted, loc=mu, scale=sigma)

    # --- QQ Plot ---
    theo_quants = norm.ppf(probs, loc=mu, scale=sigma)

    # --- Crear figura con subplots ---
    fig, axes = plt.subplots(1, 2, figsize=(20, 7))

    # PP-Plot
    axes[0].plot(probs_emp, probs_theo, "o", alpha=0.6)
    axes[0].plot([0, 1], [0, 1], "r--")
    axes[0].set_xlabel("Probabilidad empírica")
    axes[0].set_ylabel("Probabilidad teórica (Normal)")
    axes[0].set_title(title_pp)
    axes[0].grid(True)

    # QQ-Plot
    axes[1].plot(theo_quants, data_sorted, "o", alpha=0.6)
    min_val = min(data_sorted.min(), theo_quants.min())
    max_val = max(data_sorted.max(), theo_quants.max())
    axes[1].plot([min_val, max_val], [min_val, max_val], "r--")
    axes[1].set_xlabel("Cuantiles teóricos (Normal)")
    axes[1].set_ylabel("Cuantiles empíricos")
    axes[1].set_title(title_qq)
    axes[1].grid(True)

    plt.tight_layout()
    plt.show()

# Ejemplo con tus datos del IPC
pp_qq_plots(df["Rendimientos IPC"], mu_mle, sigma_mle,
            title_pp="PP-Plot Rendimientos IPC", title_qq="QQ-Plot Rendim
```



El PP-Plot compara las probabilidades empíricas con las teóricas bajo el supuesto de normalidad. La mayor parte de los puntos se encuentran cerca de la línea diagonal, lo que indica un ajuste razonable en el centro de la distribución. Sin embargo, se observa ligera desviación en las colas, especialmente en los valores extremos, lo que sugiere que la normal subestima la probabilidad de estos eventos extremos.

El QQ-Plot compara cuantiles empíricos con cuantiles teóricos de la normal ajustada. En la región central, los puntos siguen la línea de referencia, confirmando que la normal describe bien los rendimientos cercanos a la media. En las colas (cuantiles extremos) se observan desviaciones: los puntos caen por debajo de la línea en la cola izquierda y por encima en la cola derecha. Esto es evidencia gráfica de colas más pesadas (leptocurtosis), consistente con lo observado en el histograma y boxplot.

El KS-Test arrojó un estadístico $D \sim 0.06952$ y un p-valor muy pequeño $(\approx 2.52 \times 10^{-5})$. Esto implica que, se rechaza la hipótesis nula de que los datos provienen de una distribución normal. En otras palabras, aunque la normal es una aproximación aceptable en el centro, estadísticamente no describe bien toda la distribución, especialmente en las colas.

Modelado con una Mixtura Gaussiana de K $N(\mu, \sigma^2)$

Estimación por el Algoritmo EM para una Mezcla de Normales

Se asume que los datos $(x_1, x_2, \ldots, x_n)$ provienen de una mezcla de K distribuciones normales independientes:

$$p(x_i \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \,;\, \mu_k, \sigma_k^2)$$

donde los parámetros son $\theta = \{\pi_k, \mu_k, \sigma_k^2\}_{k=1}^K$, con $\pi_k \geq 0$ y $\sum_{k=1}^K \pi_k = 1$.

La **log-verosimilitud** del modelo es:

$$\ell(\boldsymbol{\theta}) = \log L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \,;\, \mu_k, \sigma_k^2) \right].$$

Dado que la maximización directa es difícil, se utiliza el **algoritmo EM (Expectation–Maximization)**:

Paso 1: Inicialización Elegir valores iniciales para $\pi_k^{(0)}, \mu_k^{(0)}, \sigma_k^{2(0)}$.

Antes de ejecutar el algoritmo EM es necesario especificar valores iniciales para los parámetros de la mezcla $\boldsymbol{\theta}^{(0)} = \{\pi_k^{(0)}, \mu_k^{(0)}, \sigma_k^{2(0)}\}_{k=1}^K$. Una estrategia común es la siguiente:

- **Pesos iniciales:**

$$\pi_k^{(0)} = \frac{1}{K}, \qquad k = 1, 2, \ldots, K.$$

Esto supone que inicialmente cada componente explica la misma proporción de los datos.

- **Medias iniciales:**

Se ubican en cuantiles igualmente espaciados de la distribución empírica de los datos:

$$\mu_k^{(0)} = Q\left(\frac{k}{K+1} \right), \qquad k = 1, 2, \ldots, K,$$

donde (Q(p)) representa el cuantil (p)-ésimo de la muestra.
Este procedimiento garantiza que las medias iniciales estén distribuidas a lo largo del rango de datos y facilita la convergencia.

- **Varianzas iniciales:**

Se asigna a cada componente la varianza muestral global:

$$\sigma_k^{2(0)} = s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2, \qquad k = 1, 2, \ldots, K.$$

De esta manera todos los componentes comienzan con la misma dispersión inicial.

Una buena inicialización permite que el algoritmo EM converja más rápido y evita

soluciones degeneradas. En la práctica, es recomendable realizar varias inicializaciones aleatorias y seleccionar la que produzca la mayor log-verosimilitud final.

Paso 2: E-step (Cálculo de responsabilidades) Calcular las probabilidades a posteriori (responsabilidades) de cada componente para cada dato:

$$\gamma_{ik}^{(t)} = \frac{\pi_k^{(t)} \, \mathcal{N}\big(x_i \,;\, \mu_k^{(t)}, \sigma_k^{2(t)}\big)}{\sum_{j=1}^{K} \pi_j^{(t)} \, \mathcal{N}\big(x_i \,;\, \mu_j^{(t)}, \sigma_j^{2(t)}\big)}.$$

Aquí γ_{ik} mide qué fracción de probabilidad del dato x_i corresponde al componente k .

Paso 3: M-step (Actualización de parámetros) Actualizar los parámetros usando las responsabilidades calculadas:

- **Pesos de mezcla:**

$$\pi_k^{(t+1)} = \frac{N_k}{n}, \qquad N_k = \sum_{i=1}^{n} \gamma_{ik}^{(t)}.$$

- **Medias:**

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{ik}^{(t)} x_i}{N_k}.$$

- **Varianzas:**

$$\sigma_k^{2(t+1)} = \frac{\sum_{i=1}^{n} \gamma_{ik}^{(t)} \big(x_i - \mu_k^{(t+1)}\big)^2}{N_k}.$$

Paso 4: Criterio de convergencia Repetir E-step y M-step hasta que la log-verosimilitud cambie menos que un umbral (\varepsilon):

$$\big|\ell^{(t+1)} - \ell^{(t)}\big| < \varepsilon.$$

Al finalizar, se obtiene el estimador de máxima verosimilitud $\hat{\boldsymbol{\theta}} = \hat{\pi}_k, \hat{\mu}_k, \hat{\sigma}_k^2\}_{k=1}^{K}$.

In [9]:
```python
def gmm_em_1d(x, n_comp=2, max_iter=500, tol=1e-6, seed=42):
    """
    EM para una mezcla de n_comp Normales en 1D.
    Retorna: pis (pesos), mus (medias), sigmas (desvios), loglikes (traye
    """
    rng = np.random.default_rng(seed)
    x = np.asarray(x).ravel()
    n = x.size

    # --- Inicialización sencilla: mu por cuantiles, sigma global, pesos
    qs = np.linspace(0.2, 0.8, n_comp)
    mus = np.quantile(x, qs)
    sigmas = np.full(n_comp, x.std(ddof=0) + 1e-8)
    pis = np.full(n_comp, 1.0 / n_comp)

    def loglike(x, pis, mus, sigmas):
        comp = np.array([pis[k] * scipy.stats.norm.pdf(x, mus[k], sigmas[
        return np.sum(np.log(np.maximum(comp.sum(axis=0), 1e-300)))

    prev_ll = -np.inf
    loglikes = []

    for _ in range(max_iter):
        # --- E-step: responsabilidades r_{ik} ---
        numer = np.array([pis[k] * scipy.stats.norm.pdf(x, mus[k], sigmas
        denom = np.maximum(numer.sum(axis=0), 1e-300)
        r = (numer / denom).T

        # --- M-step: actualizar parámetros ---
        Nk = r.sum(axis=0)                          # tamaño efectivo de
        pis = Nk / n
        mus = (r * x[:, None]).sum(axis=0) / Nk
        # varianza con responsabilidades
        var = (r * (x[:, None] - mus)**2).sum(axis=0) / Nk
        # regularización mínima por estabilidad numérica
        var = np.maximum(var, 1e-12)
        sigmas = np.sqrt(var)

        # --- Criterio de convergencia ---
        ll = loglike(x, pis, mus, sigmas)
        loglikes.append(ll)
        if abs(ll - prev_ll) < tol:
            break
        prev_ll = ll

    return pis, mus, sigmas, loglikes

# Uso:
x = df.values  # 1D
pis, mus, sigmas, lls = gmm_em_1d(x, n_comp=2, max_iter=1000, tol=1e-7, s
print("Pesos:", pis)
print("Medias:", mus)
print("Sigmas:", sigmas)
```

```
Pesos: [0.13290033 0.86709967]
Medias: [-0.00077737  0.00013351]
Sigmas: [0.00723311 0.0030576 ]
```

Estimación de los parámetros de una mezcla gaussiana de dos componentes. Inicialización: las medias se colocan en cuantiles intermedios de los datos, las desviaciones estándar se fijan en el valor global de la muestra y los pesos de mezcla se asignan de forma uniforme. E-step: se calculan las responsabilidades, es decir, las probabilidades de que cada dato provenga de cada componente. M-step: se actualizan los parámetros (pesos, medias y varianzas) en función de esas responsabilidades. Criterio de convergencia: se repite el proceso hasta que la log-verosimilitud deja de cambiar significativamente. En la segunda parte (la ejecución del código) se muestran los parámetros estimados: Pesos: aproximadamente $\Pi = 0.132$ y $\Pi = 0.868$, indicando que el segundo componente explica la mayor parte de la masa de datos. Medias: $\Mu \approx -0.0007$ y $\Mu \approx 0.0001$, lo que sugiere que el primer componente captura rendimientos negativos más extremos, mientras que el segundo describe el comportamiento central cercano a cero. Desviaciones estándar: $\sigma = 0.0072$ y $\sigma = 0.0030$, mostrando que el primer componente tiene una mayor dispersión, modelando las colas pesadas, mientras que el segundo concentra la mayoría de los datos.

In [10]:
```python
def sample_gmm_1d(n, pis, mus, sigmas, seed=0):
    rng = np.random.default_rng(seed)
    ks = rng.choice(len(pis), size=n, p=pis)
    return rng.normal(mus[ks], sigmas[ks])

# Curtosis
kurtosis = df.kurtosis()
print("Curtosis Empírica:", kurtosis.iloc[0])

x_sim = sample_gmm_1d(1_000_000, pis, mus, sigmas, seed=42)
print("Curtosis Teórica (Mixtura):", scipy.stats.kurtosis(x_sim, fisher=F
```

```
Curtosis Empírica: 3.9638983278014646
Curtosis Teórica (Mixtura): 5.843900493998071
```

Curtosis empírica:

Indica que los datos son leptocúrticos, es decir, tienen colas más pesadas y mayor concentración alrededor de la media que una distribución normal (que tendría curtosis = 3).

Curtosis teórica de la mezcla:

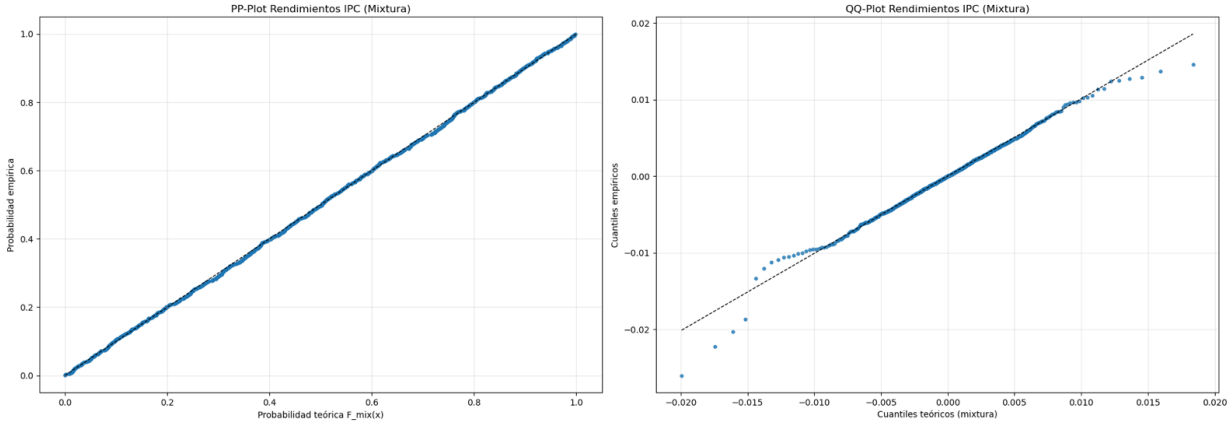
El modelo de mezcla gaussiana ajustado captura colas aún más pesadas que las observadas en la muestra. Es una consecuencia de que uno de los componentes de la mezcla tiene varianza más alta, lo que incrementa la probabilidad de valores extremos en la distribución teórica.

In [13]:
```
# KS contra la mezcla
ks_mixture = kstest(x, lambda val: mix_cdf(val, pis, mus, sigmas))
print("\nKS test contra Mixtura Gaussiana")
print("Estadístico D:", ks_mixture.statistic)
print("p-valor:", ks_mixture.pvalue)
```

```
KS test contra Mixtura Gaussiana
Estadístico D: 0.44490940336958357
p-valor: 1.3249064465966251e-36
```

In [15]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

def pp_qq_plots_mixture(x, pis, mus, sigmas, title_pp="PP-Plot vs Mixtura
    x = np.asarray(x).ravel()
    x = x[~np.isnan(x)]
    n = x.size
    xs = np.sort(x)

    # Probabilidades "Blom"
    p_emp = (np.arange(1, n + 1) - 0.375) / (n + 0.25)

    # ---- PP-Plot ----
    p_th = np.sum(pis * norm.cdf((xs[:, None] - mus) / sigmas), axis=1)

    # ---- QQ-Plot ----
    def mix_cdf(x):
        z = (x[..., None] - mus) / sigmas
        return np.sum(pis * norm.cdf(z), axis=-1)

    def mix_ppf(p, tol=1e-10, max_iter=200):
        lo = np.min(mus - 10 * sigmas.max())
        hi = np.max(mus + 10 * sigmas.max())
        lo = np.full_like(p, lo, dtype=float)
        hi = np.full_like(p, hi, dtype=float)
        for _ in range(max_iter):
            mid = (lo + hi) / 2.0
            Fmid = mix_cdf(mid)
            lo = np.where(Fmid < p, mid, lo)
            hi = np.where(Fmid >= p, mid, hi)
            if np.max(hi - lo) < tol:
                break
        return (lo + hi) / 2.0

    q_th = mix_ppf(p_emp)

    # ---- Subplots ----
    fig, axes = plt.subplots(1, 2, figsize=(20, 7))

    # PP-Plot
    axes[0].scatter(p_th, p_emp, s=12, alpha=0.8)
    axes[0].plot([0, 1], [0, 1], 'k--', lw=1)
    axes[0].set_xlabel("Probabilidad teórica F_mix(x)")
    axes[0].set_ylabel("Probabilidad empírica")
    axes[0].set_title(title_pp)
    axes[0].grid(alpha=0.3)
```

```
    # QQ-Plot
    axes[1].scatter(q_th, xs, s=12, alpha=0.8)
    q25_th, q75_th = np.quantile(q_th, [0.25, 0.75])
    q25_xs, q75_xs = np.quantile(xs, [0.25, 0.75])
    slope = (q75_xs - q25_xs) / (q75_th - q25_th + 1e-18)
    intercept = q25_xs - slope * q25_th
    xr = np.array([q_th.min(), q_th.max()])
    axes[1].plot(xr, slope * xr + intercept, 'k--', lw=1)
    axes[1].set_xlabel("Cuantiles teóricos (mixtura)")
    axes[1].set_ylabel("Cuantiles empíricos")
    axes[1].set_title(title_qq)
    axes[1].grid(alpha=0.3)

    plt.tight_layout()
    plt.show()

# ==========
# Uso con tus datos
# ==========
x = df.dropna().values
pp_qq_plots_mixture(x, pis, mus, sigmas,
                    title_pp="PP-Plot Rendimientos IPC (Mixtura)",
                    title_qq="QQ-Plot Rendimientos IPC (Mixtura)")
```



Interpretación Comparativa: Normal vs Mezcla Gaussiana (2 Componentes)

Kolmogorov–Smirnov

- Estadístico (D = 0.4449)
- p-valor (\approx 1.32 \times 10^{-36})

A pesar de que el p-valor es extremadamente pequeño (lo que lleva a rechazar la hipótesis nula de que los datos provienen exactamente de la mezcla), este resultado debe interpretarse con cuidado, ya que el KS-Test es muy sensible en muestras grandes y puede detectar diferencias pequeñas como estadísticamente significativas. Por ello es crucial complementar con análisis gráfico.

PP-Plot

- Normal ajustada: mostraba desviaciones notorias en las colas, alejándose de la línea de 45° y sugiriendo que la normal subestima la probabilidad de eventos extremos.
- Mezcla gaussiana: los puntos se alinean prácticamente sobre la diagonal en todo el rango de probabilidades, lo que indica un excelente ajuste tanto en la parte central como en las colas de la distribución.

Conclusión: la mezcla ofrece un ajuste mucho más preciso de las probabilidades acumuladas que la normal simple.

QQ-Plot

- Normal ajustada: en el QQ-Plot se observaban desviaciones claras en las colas (puntos por debajo de la línea en la izquierda y por encima en la derecha), mostrando que la normal es demasiado “delgada” para explicar la frecuencia de valores extremos.
- Mezcla gaussiana: los cuantiles empíricos siguen de cerca la línea de referencia, con pequeñas desviaciones únicamente en los puntos más extremos. La mezcla captura adecuadamente la curtosis observada en los datos.

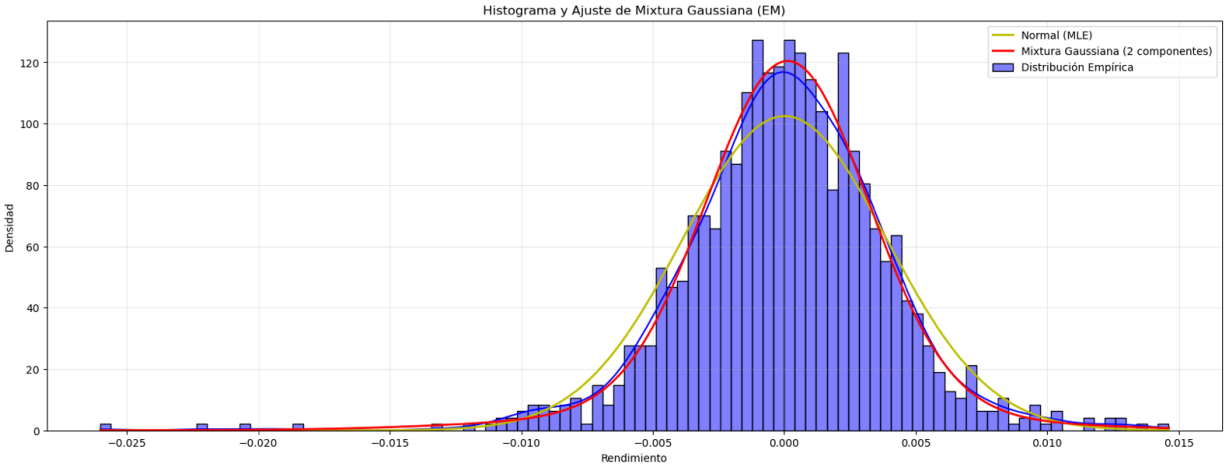
Conclusión: la mezcla mejora la representación de los cuantiles extremos y modela mejor el riesgo de eventos atípicos.

Conclusión Global**

Comparando ambos modelos:

- La mezcla gaussiana ajustada mediante EM reproduce mucho mejor la distribución completa de los rendimientos del IPC, tanto en la región central como en las colas.
- El ajuste gráfico (PP/QQ) es significativamente más cercano a la diagonal que en el caso de la normal.
- Para aplicaciones de gestión de riesgo (por ejemplo, cálculo de VaR), la mezcla es preferible porque ofrece una estimación más realista de la probabilidad de grandes pérdidas o ganancias.

In [16]:
```
# --- Histograma con KDE de los datos ---
plt.figure(figsize=(20, 7))
sns.histplot(df["Rendimientos IPC"], bins=100, kde=True, stat="density",
             color="blue", label='Distribución Empírica')

plt.title("Histograma y Ajuste de Mixtura Gaussiana (EM)")
plt.xlabel("Rendimiento")
plt.ylabel("Densidad")

# Curva de la normal ajustada
x = np.linspace(min(datos), max(datos), 200)
pdf = scipy.stats.norm.pdf(x, loc=mu_mle, scale=sigma_mle)
plt.plot(x, pdf, 'y-', lw=2, label='Normal (MLE)')

# --- Densidad de la mezcla ---
x_grid = np.linspace(df["Rendimientos IPC"].min(), df["Rendimientos IPC"]
pdf_mix = np.zeros_like(x_grid)

for pi, mu, sigma in zip(pis, mus, sigmas):
    pdf_mix += pi * norm.pdf(x_grid, loc=mu, scale=sigma)

plt.plot(x_grid, pdf_mix, 'r-', lw=2, label='Mixtura Gaussiana (2 compone

plt.legend()
plt.grid(alpha=0.3)
plt.show()
```



La figura compara la distribución empírica de los rendimientos logarítmicos del IPC (histograma azul) con dos modelos ajustados: Normal ajustada por MLE (línea amarilla): describe medianamente bien la parte central de los datos pero queda debabjo de lo esperado y subestima la densidad en las colas, lo que implica que no captura adecuadamente los eventos extremos. Mezcla gaussiana de dos componentes (línea roja): sigue mucho más de cerca la forma del histograma, ajustando tanto el pico central como las colas. Uno de los componentes de la mezcla concentra la mayor parte de los rendimientos cerca de la media, mientras que el otro, con varianza mayor, capta la dispersión de los eventos más alejados, modelando así la leptocurtosis observada. En conjunto, el gráfico muestra visualmente que la mezcla gaussiana ofrece un ajuste más realista de los datos en comparación con la normal simple.

Conclusiones

Caracterización de los datos: Los rendimientos logarítmicos del IPC presentan media cercana a cero, simetría y curtosis mayor a 3, evidenciando un comportamiento leptocúrtico con mayor frecuencia de eventos extremos de la que predice una distribución normal. Modelado con normal (MLE): El ajuste mediante máxima verosimilitud permite representar adecuadamente el comportamiento central, pero subestima las colas, lo que puede llevar a subestimar el riesgo de eventos extremos. Modelado con mezcla gaussiana: La estimación mediante el algoritmo EM con dos componentes mejora sustancialmente la representación de la distribución, capturando tanto la concentración central como la mayor dispersión en las colas. Los PP-Plot y QQ-Plot mostraron una alineación casi perfecta, confirmando la capacidad de la mezcla para explicar el comportamiento observado. Implicaciones para riesgo: Un modelo de mezcla es más adecuado para aplicaciones en gestión de riesgo y estimación de métricas como VaR, ya que asigna mayor probabilidad a grandes movimientos, ofreciendo proyecciones más conservadoras y realistas.