UNIVERSIDAD NACIONAL AGRARIA LA MOLINA - ESCUELA DE POSGRADO

DOCTORADO EN INGENIERÍA Y CIENCIAS AMBIENTALES

DISEÑO Y ANÁLISIS DE EXPERIMENTOS EN INGENIERÍA Y CIENCIAS AMBIENTALES

Actividad: Trabajo final encargado de teoría

Docente:

Ph.D. Christian René Encina Zelada

Presenta:

José Augusto Zevallos Ruiz

Lima – Perú

10 de diciembre del 2024

I. INTRODUCCIÓN

La región de Piura, en la costa norte de Perú, enfrenta desafíos significativos asociados a eventos de precipitaciones extremas vinculados al fenómeno El Niño. Estudios previos han demostrado una relación consistente entre las precipitaciones intensas y las anomalías positivas en la temperatura de la superficie del mar (SST), que favorecen la formación de sistemas convectivos al modificar las condiciones en la capa límite planetaria (Tapley & Waylen, 1990; Waylen & Caviedes, 1986). Durante estos eventos, las precipitaciones acumuladas en Piura pueden ser hasta treinta veces superiores a lo normal, resultado de incursiones anómalas hacia el polo de la Corriente del Niño que generan inestabilidad atmosférica extendida y condiciones propicias para el ascenso del aire, especialmente sobre pendientes montañosas cercanas (Takahashi, 2004). Estos fenómenos meteorológicos incrementan significativamente el riesgo de inundaciones, lo que subraya la necesidad de herramientas predictivas robustas para la gestión del riesgo.

En el contexto del cambio climático, el aumento en la frecuencia e intensidad de eventos extremos refuerza la urgencia de desarrollar modelos predictivos eficaces que apoyen la planificación y la mitigación de riesgos. Tradicionalmente, se han empleado modelos hidráulicos basados en las ecuaciones de aguas poco profundas, los cuales han sido aplicados para simular inundaciones históricas en Piura, como las asociadas al evento El Niño de 2017, y proyectar escenarios futuros de períodos de retorno hasta 500 años (Alvarez et al., 2021; Muñoz et al., 2022). Aunque útiles, estas metodologías tienden a enfocarse en evaluaciones posteriores a los eventos, dejando aspectos como el llenado de datos faltantes sin explorar completamente.

Por otro lado, los modelos hidrológicos han mostrado avances en la predicción a corto plazo gracias a la integración de datos satelitales, mejorando la precisión en eventos de inundaciones en regiones montañosas como los Andes tropicales (Llauca et al., 2023). Asimismo, sistemas globales como el GloFAS han demostrado ser efectivos para pronosticar inundaciones en cuencas con grandes áreas de captación, alcanzando una precisión del 65% en ríos peruanos (Bischiniotis et al., 2019). Sin embargo, estas herramientas suelen depender de conjuntos de datos completos, lo

que representa un desafío en áreas donde las estaciones de monitoreo presentan discontinuidades significativas en sus registros.

En este contexto, las técnicas de regresión múltiple ofrecen una alternativa prometedora para abordar el problema del llenado de datos faltantes, particularmente en zonas con baja densidad de estaciones meteorológicas. La regresión múltiple permite relacionar las precipitaciones en estaciones cercanas mediante la integración de variables predictoras, como la elevación, latitud y longitud, logrando estimaciones precisas a nivel local (Naoum et al., 2004). Por ejemplo, estudios realizados en la isla de Creta han demostrado cómo la incorporación de estas variables mejora significativamente las predicciones de precipitación en terrenos complejos y con limitadas estaciones de monitoreo.

Además, la aplicación de modelos de regresión múltiple ha sido exitosa en diferentes regiones, como en la isla de Creta, donde se lograron estimaciones robustas de la precipitación media anual al incluir parámetros orográficos y geográficos (Naoum et al., 2004). Estas metodologías no solo capturan la variabilidad espacial de las precipitaciones, sino que también son aplicables en diferentes escalas, desde cuencas hasta áreas más extensas, lo que resalta su versatilidad.

El presente estudio tiene como objetivo principal utilizar modelos de regresión múltiple para llenar vacíos en datos de precipitación diaria en la cuenca del río Piura. Para ello, se utilizarán datos de cinco estaciones meteorológicas: Chusis, Chalaco, Huamarca, Huancabamba y Miraflores, aprovechando la información de las estaciones vecinas para realizar las estimaciones. Este enfoque busca mejorar la disponibilidad de datos en la región, lo que es esencial para la modelación hidrológica y la planificación de recursos hídricos.

La metodología planteada permitirá no solo abordar la problemática de datos faltantes, sino también generar insumos que puedan integrarse a modelos hidrológicos e hidráulicos para la gestión del riesgo de inundaciones en Piura. Así, se busca contribuir al fortalecimiento de herramientas de predicción, esenciales para mitigar los impactos de los eventos extremos que afectan recurrentemente a esta región.

II. MARCO TEÓRICO

Regresión lineal múltiple

La regresión múltiple es una técnica estadística utilizada para modelar la relación entre una variable dependiente (Y) y varias variables independientes $(X_1, X_2, \dots, X_k)$. Su objetivo es predecir Y o explicar su variabilidad basándose en las variables predictoras. La ecuación general de un modelo de regresión múltiple es:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \epsilon$$

Donde:

- Y: Variable dependiente.
- $X_1, X_2, \dots, X_k$: Variables independientes o predictoras.
- β_0: Intercepto del modelo.
- $\beta_1, \beta_2, \dots, \beta_x$Coeficientes de regresión que representan el cambio esperado en Y por unidad de cambio en cada X_j, manteniendo las demás constantes.
- ϵ: Término de error o residual, que captura la variabilidad no explicada por el modelo.

Forma Matricial del Modelo

El modelo de regresión múltiple puede expresarse en términos matriciales como:

$$Y = X\beta + \epsilon$$

Donde:

- Y es el vector de observaciones de la variable independiente $(n\ x\ 1)$.

$$\mathbf{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix}$$

- X es la matriz de diseño $(n\ x\ (k+1))$, que incluye una columna de unos para el termino de intercepto y las observaciones de las variables independientes

$$\mathbf{X} = \begin{bmatrix} 1 & X_{11} & X_{12} & \cdots & X_{1k} \\ 1 & X_{21} & X_{22} & \cdots & X_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & X_{n2} & \cdots & X_{nk} \end{bmatrix}$$

- β es el vector de coeficientes $((k+1)\ x\ 1)$:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}$$

- ϵ es el vector de errores $(n\ x\ 1)$:

$$\epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ . \\ . \\ \epsilon_n \end{bmatrix}$$

Estimación de los Coeficientes

Los coeficientes del modelo se estiman minimizando la suma de los cuadrados de los errores $(\epsilon'\epsilon)$, siguiendo el método de los mínimos cuadrados ordinarios (OLS). La estimación matricial de β se obtiene como:

$$\hat{\beta} = (X'X)^{-1}X'Y$$

Donde:

- X' es la transpuesta de la matriz de diseño.
- $(X'X)^{-1}$es la matriz inversa del producto $X'X$.

- $X'Y$ es el producto de la transpuesta de X con Y.

El vector $\hat{\beta}$ contiene los coeficientes estimados del modelo

Prueba Kolmogórov-Smirnov

La prueba de Kolmogórov-Smirnov es una prueba no paramétrica que evalúa si una muestra sigue una distribución específica. Es utilizada para verificar la normalidad de los datos en un análisis ANOVA. La estadística de prueba DDD para esta prueba es:

$$D = \sup |F_n(x) - F(x)|$$

Donde:

- $F_n(x)$ es la función de distribución empírica de la muestra,
- $F(x)$ es la función de distribución acumulada teórica.

Si el valor de D es suficientemente grande, se rechaza la hipótesis nula de que los datos siguen la distribución especificada.

Prueba de Durbin-Watson

La prueba de Durbin-Watson es una prueba estadística utilizada para detectar la autocorrelación en los residuos de un modelo de regresión. Su estadística de prueba es:

$$d = \frac{\sum_{t=2}^{n}(e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2}$$

Donde:

- e_t son los residuos,
- n es el número de observaciones

Un valor cercano a 2 indica que no hay autocorrelación en los residuos, mientras que valores alejados de 2 sugieren la presencia de autocorrelación.

Error cuadrático medio

Es un estimador que mide el promedio de los errores al cuadrado. El error es la diferencia entre el valor estimado por un modelo y el valor real medido de la variable. Esta diferencia se debe a que el modelo generalmente no captura toda la información necesaria para reproducir la realidad o a la existencia de errores de medición aleatoria (ver Ecuación).

$$ECM = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2$$

III. METODOLOGÍA

4.1. Metodología empleada

El flujograma representa el proceso de análisis y modelado de datos faltantes mediante regresión múltiple, comenzando con la carga de datos y configuración del entorno de trabajo en herramientas como R o Python. A continuación, se preparan y verifican los datos, identificando columnas con valores faltantes y filtrando estaciones según un umbral predefinido de datos incompletos. Luego, el flujo pasa a iterar sobre los datos faltantes, entrenar modelos de regresión múltiple y realizar predicciones para llenar los vacíos, calculando el RMSE como métrica de desempeño. En cada iteración, se evalúa si hubo mejora; si no, el proceso finaliza y retorna los datos completados, las exclusiones y los errores calculados. Finalmente, se generan gráficos para comparar series originales y completadas, así como un análisis de correlación entre estaciones mediante un mapa de calor.

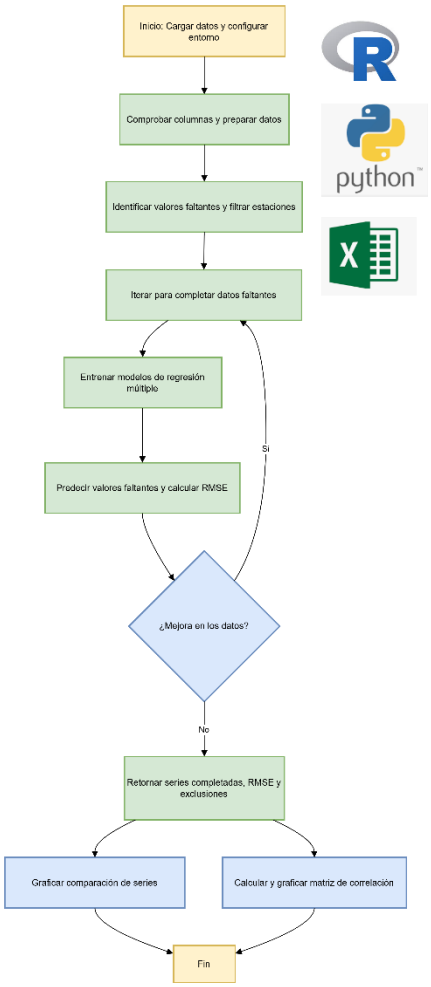

Figura 1 Flujograma metodológico empleado en el presente trabajo

4.3. Datos

El conjunto de datos registra las precipitaciones mensuales acumuladas en cinco estaciones meteorológicas (Chusis, Chalaco, Huamarca, Huancabamba y Miraflores) desde enero de 1980 hasta diciembre de 2017. Cada fila representa el total mensual de precipitaciones en milímetros (mm) para cada estación, con una columna adicional que indica la fecha correspondiente al último día del mes.

Los datos muestran variaciones significativas entre estaciones y períodos, con valores mínimos de 0 mm durante meses secos y máximos superiores a 700 mm en eventos extremos, como durante fenómenos de El Niño. También se observan valores faltantes (NA), que son relevantes para análisis de imputación o modelado. Este conjunto de datos es clave para estudiar la estacionalidad y los patrones de lluvia en la región, así como para desarrollar modelos predictivos relacionados con eventos climáticos.

Tabla 1 Fragmento de los datos de precipitación mensual registrados en cinco estaciones meteorológicas de la región de Piura.

Fecha	chusis	chalaco	huamarca	huancabamba	miraflores
1980-01-31	0	70.4	38.7	23.5	0.2
1980-02-29	0	108.7	139	35.9	2.5
1980-03-31	6.4	121.6	116.9	51.3	13.7
1980-04-30	7.6	178	129.1		35.2
1980-05-31	0	42.5	17.9	22.1	0.3
1980-06-30	0	0	2.3	8.7	0
1980-07-31	0	0	0.1	2.9	0
1980-08-31	0	0	1.1	2.8	0
1980-09-30	0	0	0	11.2	0
1980-10-31	5.2	36.7	62.2	49.9	0.4
1980-11-30	4.1	40	33.3	65.6	5.6

IV. RESULTADOS Y DISCUSIONES

Prueba Kolmogórov-Smirnov

Para el test de normalidad se tiene $F_n(x)$ = distribución normal muestral, $F(x)$ = distribución teórica Weibull.

De la siguiente expresión se obtuvo que:

$$D = \sup |F_n(x) - F(x)| = 0.263$$

El valor de $D_{critic\acute{o}}$:

$$D_{critic\acute{o}} = \frac{1.36}{\sqrt{N}} = 0.0942$$

Como el valor de D supera al $D_{critic\acute{o}}$, se concluye que los residuos no cumplen el criterio de normalidad para la estación Chusis.

Prueba de Durbin-Watson

El valor obtenido para el estadístico de Durbin-Watson (0.1086) al completar los datos de la estación Chusis evidencia una alta autocorrelación positiva en los residuos del modelo de regresión múltiple utilizado. Esto indica que los errores no son independientes, lo que sugiere que el modelo no está capturando adecuadamente la estructura temporal de los datos. La autocorrelación positiva puede afectar la precisión y la validez de las estimaciones generadas, subrayando la necesidad de ajustes en la metodología. Para abordar este problema, sería recomendable explorar alternativas como la incorporación de modelos autorregresivos (ARIMA) o métodos de regresión que incluyan componentes temporales explícitos. Además, la revisión de los predictores utilizados y la consideración de transformaciones en los datos podrían ayudar a mitigar estos efectos y mejorar la calidad de los resultados obtenidos.

$$d = \frac{\sum_{t=2}^{N}(e_t - e_{t-1})^2}{\sum_{t=1}^{N} e_t^2} = 0.108$$

Matriz de correlaciones

La imagen muestra una matriz de correlación entre las estaciones meteorológicas Chusis, Chalaco, Huamarca, Huancabamba y Miraflores, con valores que oscilan entre -1 (correlación negativa perfecta) y 1 (correlación positiva perfecta). Las correlaciones más altas se observan entre Chalaco y Huamarca (0.94) y entre Miraflores y Chusis (0.87), indicando que estas estaciones tienen patrones similares de precipitación. Por otro lado, las correlaciones más bajas se encuentran entre Huancabamba y Chusis (0.13) y entre Huancabamba y Miraflores (0.22), lo que sugiere diferencias significativas en los patrones de lluvia entre estas estaciones. El mapa utiliza una escala de colores donde el rojo indica correlación positiva fuerte, el azul muestra correlación baja o negativa, y el blanco representa valores intermedios.

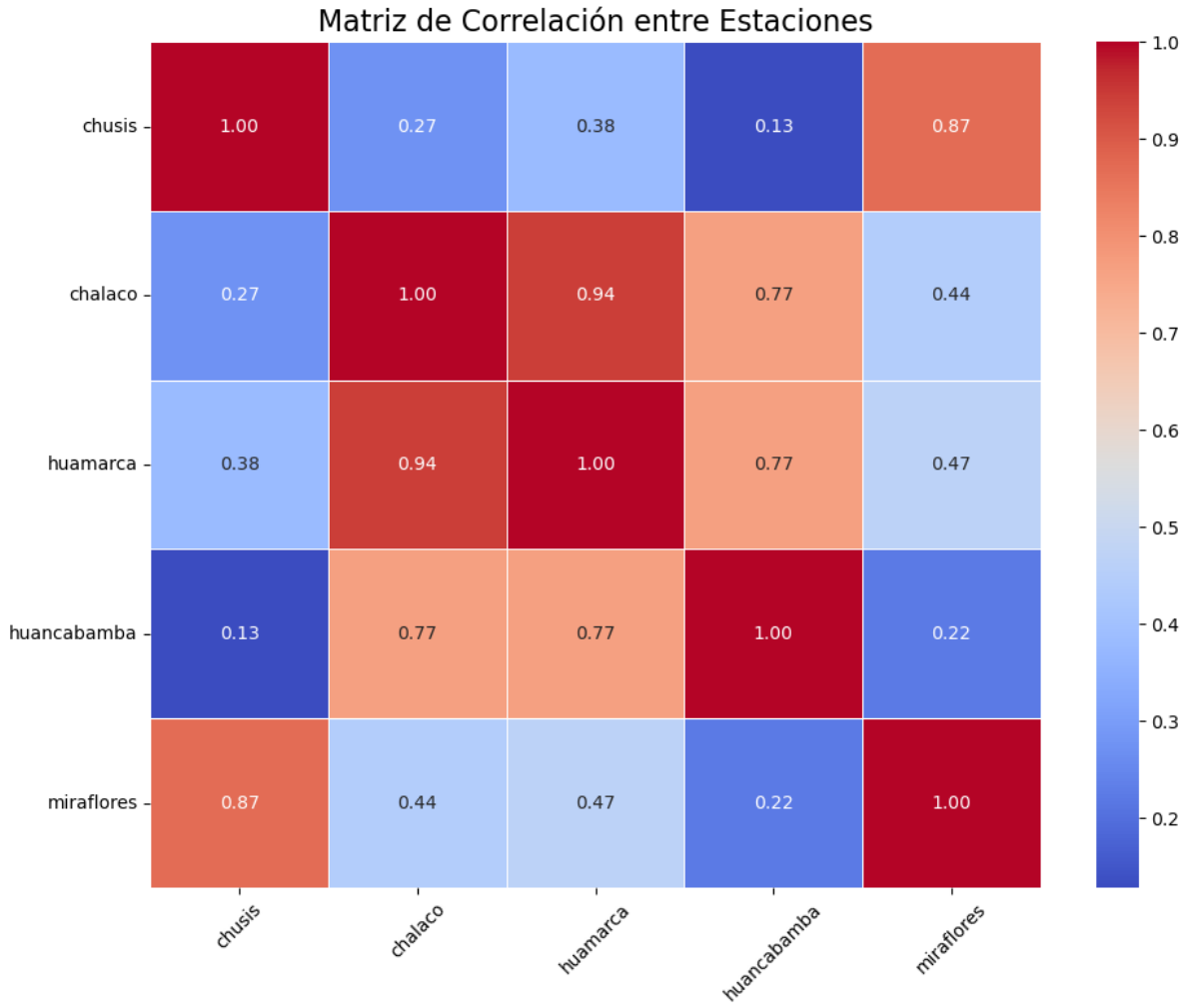

Figura 2 Matriz de correlación entre las precipitaciones de las estaciones meteorológicas estudiadas

Errores cuadráticos medios

La imagen muestra un gráfico de barras que representa el error cuadrático medio (RMSE) para diferentes estaciones meteorológicas: Chalaco, Miraflores, Huancabamba y Huamarca. El RMSE es una métrica que mide la diferencia promedio entre los valores observados y los valores predichos, con valores más bajos indicando mejor precisión en el modelo. La estación Miraflores tiene el mayor RMSE, superando los 70, lo que sugiere mayores discrepancias en las predicciones. Por otro lado, Huancabamba presenta el RMSE más bajo, cercano a 20, indicando una mayor precisión en los cálculos para esta estación. Chalaco y Huamarca tienen valores intermedios, alrededor de 30. Este gráfico es útil para evaluar la calidad del modelo de imputación de datos en cada estación.

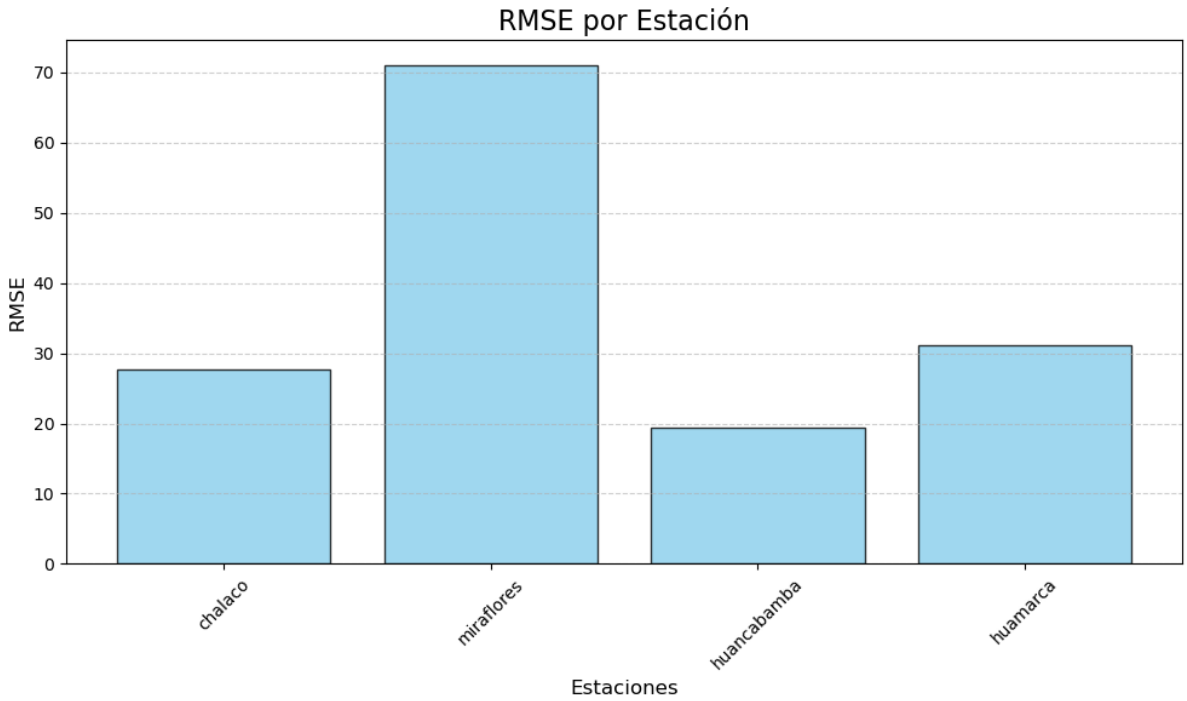

Figura 3 Gráfico de barras mostrando los valores de RMSE por cada estación meteorológica.

Series completadas

El gráfico muestra una comparación entre la serie de datos originales y la serie completada para la estación meteorológica "Chalaco". En el eje vertical se representan los valores de precipitación acumulada (en milímetros), mientras que en el eje horizontal se encuentran las fechas de observación. La serie original está representada con una línea azul continua, mientras que la serie completada se muestra con una línea naranja discontinua.

Se observa que ambas series coinciden en gran medida, especialmente en los períodos sin datos faltantes. En los intervalos con valores faltantes en la serie original, la serie completada ofrece estimaciones que siguen la tendencia general de los datos históricos. Esto sugiere que el modelo de imputación utilizado logró reconstruir los valores faltantes de manera consistente con el comportamiento de las precipitaciones en esta estación.

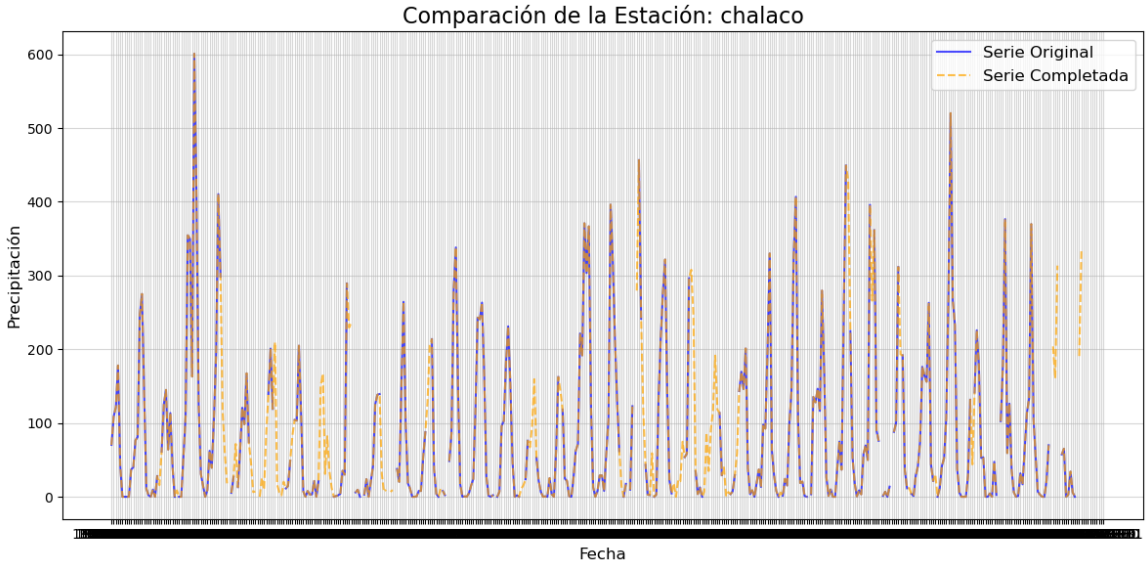

Figura 4 Gráfico comparativo de la serie original y la completada para la estación Chalaco

V. CONCLUSIONES Y RECOMENDACIONES

El presente estudio aplicó modelos de regresión múltiple para completar datos faltantes en registros de precipitación mensual en la cuenca del río Piura, utilizando información de cinco estaciones meteorológicas: Chusis, Chalaco, Huamarca, Huancabamba y Miraflores. Los resultados muestran que esta metodología es efectiva para estimar valores faltantes, especialmente cuando existe una correlación significativa entre las estaciones. La implementación permitió mejorar la disponibilidad de datos, lo cual es esencial para estudios hidrológicos y la gestión de recursos hídricos en la región.

Sin embargo, es importante reconocer que la regresión múltiple es solo una de varias técnicas disponibles para el llenado de datos faltantes. Para robustecer los resultados y reducir posibles sesgos, es recomendable explorar métodos adicionales como la imputación múltiple, modelos basados en series temporales, aprendizaje automático y técnicas geoestadísticas. Además, se requiere un control de calidad más riguroso de los datos, incluyendo la detección y corrección de outliers, análisis de consistencia interna y validación cruzada de los modelos utilizados.

Finalmente, futuros trabajos deberían enfocarse en el análisis de tendencias a largo plazo y la evaluación de la consistencia de los datos completados. Integrar estos datos en modelos hidrológicos e hidráulicos permitirá mejorar las predicciones de eventos extremos y apoyar la planificación estratégica para mitigar los impactos

asociados al fenómeno El Niño. Esto contribuirá significativamente a la resiliencia de la región frente a eventos climáticos adversos y al desarrollo sostenible de las comunidades locales.

VI. REFERENCIAS BIBLIOGRÁFICAS

Alvarez, G., Moreno, A., Guzmán, E., & Santos, S. (2021). Numerical Simulation of Hydrodynamic Conditions in Rivers Facing Extreme Events Due to the “El Niño” Phenomenon. *Smart Innovation, Systems and Technologies*, *202*, 235–244. https://doi.org/10.1007/978-3-030-57566-3_23

Bischiniotis, K., van den Hurk, B., Zsoter, E., Coughlan de Perez, E., Grillakis, M., & Aerts, J. C. J. H. (2019). Evaluation of a global ensemble flood prediction system in Peru. *Hydrological Sciences Journal*, *64*(10), 1171–1189. https://doi.org/10.1080/02626667.2019.1617868

Llauca, H., Arestegui, M., & Lavado-Casimiro, W. (2023). Constraining Flood Forecasting Uncertainties through Streamflow Data Assimilation in the Tropical Andes of Peru: Case of the Vilcanota River Basin. *Water*, *15*(22). https://doi.org/10.3390/w15223944

Muñoz, D. F., Yin, D., Bakhtyar, R., Moftakhari, H., Xue, Z., Mandli, K., & Ferreira, C. (2022). Inter-Model Comparison of Delft3D-FM and 2D HEC-RAS for Total Water Level Prediction in Coastal to Inland Transition Zones. *Journal of the American Water Resources Association*, *58*(1), 34–49. https://doi.org/10.1111/1752-1688.12952

Naoum, S., Asce, M., & Tsanis, I. K. (2004). Orographic Precipitation Modeling with Multiple Linear Regression. *Journal of Hydrologic Engineering*, *9*, 79–102. https://doi.org/10.1061/ASCE1084-069920049:279

Takahashi, K. (2004). The atmospheric circulation associated with extreme rainfall events in Piura, Peru, during the 1997-1998 and 2002 El Niño events. *Annales Geophysicae*, *22*(11), 3917–3926. https://doi.org/https://doi.org/10.5194/angeo-22-3917-2004

Tapley, T. D., & Waylen, P. R. (1990). Spatial variability of annual precipitation and the variable nature of ENSO in western Peru. *Hydrological Sciences Journal*, *35*(4), 429–446. https://doi.org/10.1080/02626669009492444

Waylen, P. R., & Caviedes, C. N. (1986). El Niño and annual floods on the north Peruvian littoral. *Journal of Hydrology*, *89*(3–4), 141–156. https://doi.org/https://doi.org/10.1016/0022-1694(86)90141-8

VII. ANEXOS

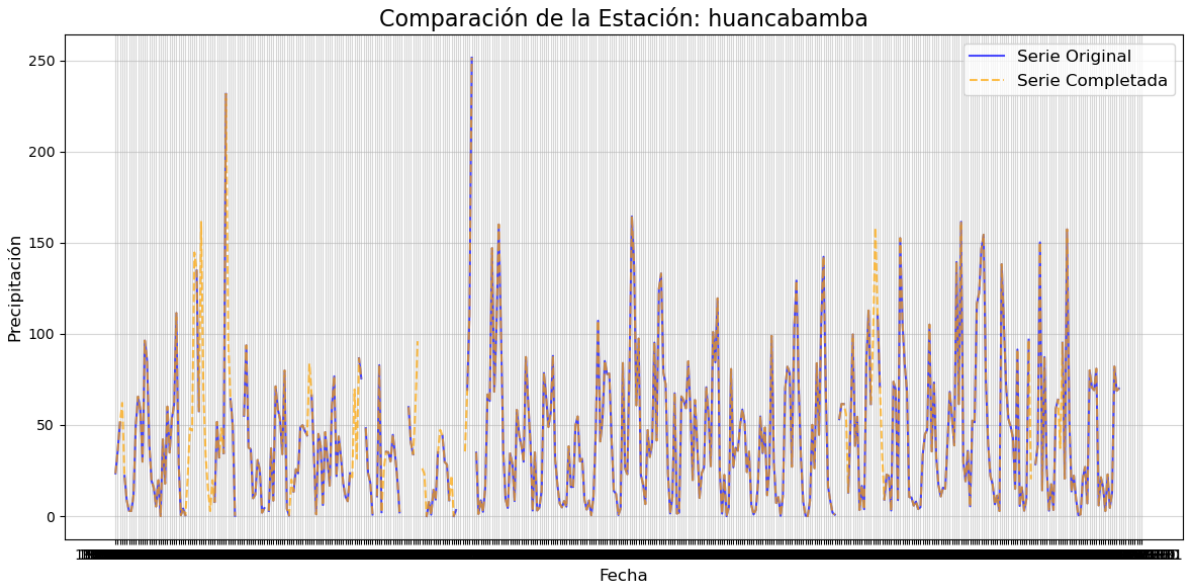

Figura 5 Gráfico comparativo de la serie original y la completada para la estación Huacabamba

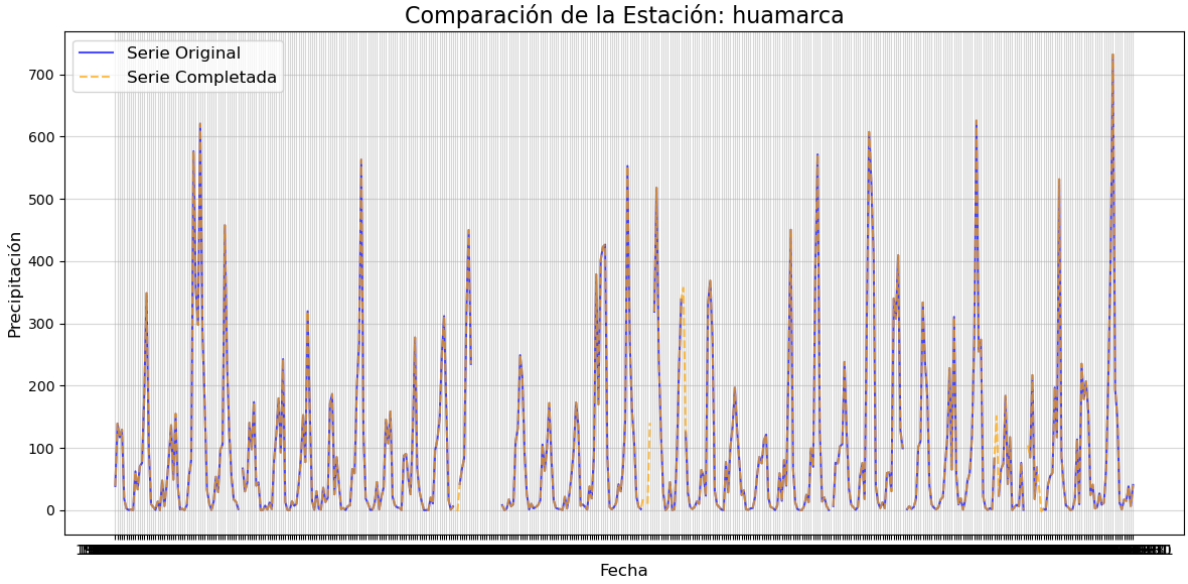

Figura 6 Gráfico comparativo de la serie original y la completada para la estación Huarmaca

Código en Python

```
#!/usr/bin/env python
# coding: utf-8

# In[44]:


import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt

def completar_datos_regresion_multiple(data_xts):
    # Guardar una copia de la serie original
    serie_original = data_xts.copy()

    # Asegurarse de que los datos sean numéricos y excluir la columna de
fechas
    if 'Fecha' in data_xts.columns:
        data_xts = data_xts.set_index('Fecha')
    data_xts = data_xts.astype(float)

    # Variables de seguimiento
    prev_na_count = float('inf')
    estaciones_excluidas = []
    iteration = 0
    errores_rmse = {}  # Para guardar el RMSE de cada estación

    while True:
        iteration += 1

        # Identificar valores faltantes
        nas_por_estacion = data_xts.isna().sum()
        porcentaje_nas = nas_por_estacion / len(data_xts) * 100

        # Filtrar estaciones válidas
        estaciones_validas = porcentaje_nas[porcentaje_nas <=
30].index.tolist()
        estaciones_invalidas = porcentaje_nas[porcentaje_nas >
30].index.tolist()

        # Registrar estaciones excluidas
        estaciones_excluidas.extend(estaciones_invalidas)

        if not estaciones_validas:
            print("No quedan estaciones con menos del 30% de datos
faltantes.")
            break

        # Ordenar estaciones por número de valores faltantes
        estaciones_ordenadas =
nas_por_estacion[estaciones_validas].sort_values(ascending=False).index.to
list()

        # Verificar mejora
        total_nas = nas_por_estacion.sum()
        if total_nas == 0 or total_nas >= prev_na_count:
```

```
            print("No hay más mejoras o todos los NAs han sido
completados.")
            break

        prev_na_count = total_nas

        # Completar datos faltantes por estación
        for columna_a_completar in estaciones_ordenadas:
            if nas_por_estacion[columna_a_completar] > 0:
                # Seleccionar predictores
                predictores = [col for col in estaciones_validas if col !=
columna_a_completar]
                if not predictores:
                    continue

                # Crear conjuntos de entrenamiento
                data_entrenamiento =
data_xts.dropna(subset=[columna_a_completar])
                if data_entrenamiento.empty:
                    continue

                X = data_entrenamiento[predictores]
                y = data_entrenamiento[columna_a_completar]

                # Eliminar valores problemáticos
                mask = ~X.isna().any(axis=1)
                X = X[mask]
                y = y[mask]

                if X.empty or y.empty:
                    continue

                # Dividir los datos en entrenamiento (80%) y prueba (20%)
                X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

                # Modelo de regresión múltiple
                modelo_multiple = LinearRegression()
                try:
                    modelo_multiple.fit(X_train, y_train)

                    # Calcular predicciones en el conjunto de prueba y el
RMSE
                    y_pred = modelo_multiple.predict(X_test)
                    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
                    errores_rmse[columna_a_completar] = rmse
                except ValueError as e:
                    print(f"Error al ajustar modelo para
{columna_a_completar}: {e}")
                    continue

                # Predecir valores faltantes
                data_faltantes =
data_xts[data_xts[columna_a_completar].isna()]
                if not data_faltantes.empty:
                    X_faltantes = data_faltantes[predictores]
                    X_faltantes = X_faltantes.dropna(axis=0, how='any')  #
Asegurarse de no predecir con NaN
                    if not X_faltantes.empty:
                        predicciones =
modelo_multiple.predict(X_faltantes)
```

```
                        data_xts.loc[X_faltantes.index,
columna_a_completar] = predicciones

        print(f"Iteración {iteration} - Total NAs restantes: {total_nas}")
        if iteration > 5:
            break

    # Retornar datos completados, serie original, estaciones excluidas y
errores RMSE
    return {
        "SerieOriginal": serie_original,
        "SerieCompletada": data_xts,
        "EstacionesExcluidas": list(set(estaciones_excluidas)),
        "ErroresRMSE": errores_rmse
    }

# Cargar los datos y ejecutar la función
gauges = pd.read_csv('datos_meses.csv')
resultado = completar_datos_regresion_multiple(data_xts=gauges)

# Imprimir resultados
print("Estaciones Excluidas:", resultado["EstacionesExcluidas"])
print("Errores RMSE por estación:", resultado["ErroresRMSE"])
print("Serie Original:")
print(resultado["SerieOriginal"].head())
print("Serie Completada:")
print(resultado["SerieCompletada"].head())


# In[48]:


def graficar_rmse_por_estacion(errores_rmse):
    """
    Grafica en barras los RMSE por cada estación.

    Args:
    - errores_rmse (dict): Diccionario con los nombres de las estaciones
como claves y sus RMSE como valores.
    """
    # Verificar que haya datos para graficar
    if not errores_rmse:
        print("No hay valores de RMSE para graficar.")
        return

    # Crear el gráfico de barras
    estaciones = list(errores_rmse.keys())
    valores_rmse = list(errores_rmse.values())

    plt.figure(figsize=(10, 6))
    plt.bar(estaciones, valores_rmse, color='skyblue', edgecolor='black',
alpha=0.8)
    plt.title("RMSE por Estación", fontsize=16)
    plt.xlabel("Estaciones", fontsize=12)
    plt.ylabel("RMSE", fontsize=12)
    plt.xticks(rotation=45, fontsize=10)
    plt.grid(axis='y', linestyle='--', alpha=0.6)

    # Mostrar el gráfico
    plt.tight_layout()
    plt.show()
```

```
# In[49]:


# Obtener los RMSE del resultado
errores_rmse = resultado["ErroresRMSE"]

# Graficar los RMSE por estación
graficar_rmse_por_estacion(errores_rmse)


# In[45]:


import matplotlib.pyplot as plt

def graficar_comparacion(resultado, estacion):
    """
    Grafica y compara las series originales y completadas para una
estación específica.

    Args:
    - resultado (dict): Resultado de la función
`completar_datos_regresion_multiple`,
                        debe contener 'SerieOriginal' y 'SerieCompletada'.
    - estacion (str): Nombre de la estación a graficar.
    """
    # Verificar que la estación exista en los datos
    if estacion not in resultado["SerieOriginal"].columns:
        print(f"La estación '{estacion}' no existe en los datos.")
        return

    # Extraer las series original y completada
    serie_original = resultado['SerieOriginal'][estacion]
    serie_completada = resultado['SerieCompletada'][estacion]

    # Crear la gráfica
    plt.figure(figsize=(12, 6))
    plt.plot(serie_original, label="Serie Original", color="blue",
alpha=0.7)
    plt.plot(serie_completada, label="Serie Completada", color="orange",
linestyle="--", alpha=0.7)

    # Añadir detalles
    plt.title(f"Comparación de la Estación: {estacion}", fontsize=16)
    plt.xlabel("Fecha", fontsize=12)
    plt.ylabel("Precipitación", fontsize=12)
    plt.legend(fontsize=12)
    plt.grid(alpha=0.5)
    plt.tight_layout()

    # Mostrar la gráfica
    plt.show()


# In[41]:


# Graficar la comparación para la estación 'chusis'
graficar_comparacion(resultado, 'huancabamba')
```

```
# In[34]:


# Graficar la comparación para la estación 'chusis'
graficar_comparacion(resultado, 'chalaco')


# In[38]:


# Graficar la comparación para la estación 'chusis'
graficar_comparacion(resultado, 'huamarca')


# In[46]:


def calcular_matriz_correlacion(data):
    """
    Calcula la matriz de correlación entre las estaciones.

    Args:
    - data (DataFrame): DataFrame con los datos de las estaciones.

    Returns:
    - matriz_correlacion (DataFrame): Matriz de correlación entre las
estaciones.
    """
    # Calcular la matriz de correlación
    matriz_correlacion = data.corr()
    return matriz_correlacion


def graficar_matriz_correlacion(matriz_correlacion):
    """
    Grafica la matriz de correlación utilizando un mapa de calor.

    Args:
    - matriz_correlacion (DataFrame): Matriz de correlación entre las
estaciones.
    """
    # Crear el mapa de calor
    plt.figure(figsize=(10, 8))
    sns.heatmap(matriz_correlacion, annot=True, cmap="coolwarm",
fmt=".2f", linewidths=0.5)
    plt.title("Matriz de Correlación entre Estaciones", fontsize=16)
    plt.xticks(rotation=45, fontsize=10)
    plt.yticks(rotation=0, fontsize=10)
    plt.tight_layout()
    plt.show()


# In[47]:


matriz_correlacion =
calcular_matriz_correlacion(resultado["SerieCompletada"])

# Mostrar la matriz de correlación
```

```
print(matriz_correlacion)

# Graficar la matriz de correlación
graficar_matriz_correlacion(matriz_correlacion)
```

Código en R

```
setwd("D:/Proyectos_GitHub/Diseno_experimental/trabajo_individual2")
library(ggplot2)
library(reshape2)

completar_datos_regresion_multiple <- function(data_xts) {
  # Guardar una copia de la serie original
  serie_original <- data_xts

  # Asegurarse de que los datos sean numéricos y excluir la columna de
fechas
  if ("Fecha" %in% colnames(data_xts)) {
    rownames(data_xts) <- data_xts$Fecha
    data_xts <- data_xts[, -which(colnames(data_xts) == "Fecha")]
  }
  data_xts <- as.data.frame(lapply(data_xts, as.numeric))

  # Variables de seguimiento
  prev_na_count <- Inf
  estaciones_excluidas <- c()
  iteration <- 0
  errores_rmse <- list()

  repeat {
    iteration <- iteration + 1

    # Identificar valores faltantes
    nas_por_estacion <- colSums(is.na(data_xts))
    porcentaje_nas <- (nas_por_estacion / nrow(data_xts)) * 100

    # Filtrar estaciones válidas
    estaciones_validas <- names(porcentaje_nas[porcentaje_nas <= 30])
    estaciones_invalidas <- names(porcentaje_nas[porcentaje_nas > 30])

    # Registrar estaciones excluidas
    estaciones_excluidas <- unique(c(estaciones_excluidas,
estaciones_invalidas))

    if (length(estaciones_validas) == 0) {
      cat("No quedan estaciones con menos del 30% de datos faltantes.\n")
      break
    }

    # Ordenar estaciones por número de valores faltantes
    estaciones_ordenadas <- sort(nas_por_estacion[estaciones_validas],
decreasing = TRUE)

    # Verificar mejora
    total_nas <- sum(nas_por_estacion)
    if (total_nas == 0 || total_nas >= prev_na_count) {
      cat("No hay más mejoras o todos los NAs han sido completados.\n")
      break
    }

    prev_na_count <- total_nas

    # Completar datos faltantes por estación
    for (columna_a_completar in names(estaciones_ordenadas)) {
      if (nas_por_estacion[columna_a_completar] > 0) {
        # Seleccionar predictores
```

```
        predictores <- setdiff(estaciones_validas, columna_a_completar)
        if (length(predictores) == 0) next

        # Crear conjuntos de entrenamiento
        data_entrenamiento <- data_xts[complete.cases(data_xts[,
c(columna_a_completar, predictores)]), ]
        if (nrow(data_entrenamiento) == 0) next

        X <- as.matrix(data_entrenamiento[, predictores])
        y <- data_entrenamiento[, columna_a_completar]

        # Dividir los datos en entrenamiento (80%) y prueba (20%)
        set.seed(42)
        train_indices <- sample(1:nrow(X), size = 0.8 * nrow(X))
        X_train <- X[train_indices, ]
        X_test <- X[-train_indices, ]
        y_train <- y[train_indices]
        y_test <- y[-train_indices]

        # Modelo de regresión múltiple
        modelo_multiple <- lm(y_train ~ ., data =
as.data.frame(cbind(y_train, X_train)))
        tryCatch({
          # Calcular predicciones en el conjunto de prueba y el RMSE
          y_pred <- predict(modelo_multiple, newdata =
as.data.frame(X_test))
          rmse <- sqrt(mean((y_test - y_pred)^2))
          errores_rmse[[columna_a_completar]] <- rmse
        }, error = function(e) {
          cat(sprintf("Error al ajustar modelo para %s: %s\n",
columna_a_completar, e$message))
          next
        })

        # Predecir valores faltantes
        data_faltantes <- data_xts[is.na(data_xts[, columna_a_completar]),
]
        if (nrow(data_faltantes) > 0) {
          X_faltantes <- as.matrix(data_faltantes[, predictores])
          if (any(is.na(X_faltantes))) next
          predicciones <- predict(modelo_multiple, newdata =
as.data.frame(X_faltantes))
          data_xts[rownames(data_faltantes), columna_a_completar] <-
predicciones
        }
      }
    }

    cat(sprintf("Iteración %d - Total NAs restantes: %d\n", iteration,
total_nas))
    if (iteration > 5) break
  }

  # Retornar datos completados, serie original, estaciones excluidas y
errores RMSE
  list(
    SerieOriginal = serie_original,
    SerieCompletada = data_xts,
    EstacionesExcluidas = estaciones_excluidas,
    ErroresRMSE = errores_rmse
  )
```

```
}

# Cargar los datos y ejecutar la función
data_xts <- read.csv("datos_meses.csv")
resultado <- completar_datos_regresion_multiple(data_xts)

# Imprimir resultados
print(resultado$EstacionesExcluidas)
print(resultado$ErroresRMSE)
head(resultado$SerieOriginal)
head(resultado$SerieCompletada)

graficar_rmse_por_estacion <- function(errores_rmse) {
  # Verificar que haya datos para graficar
  if (length(errores_rmse) == 0) {
    cat("No hay valores de RMSE para graficar.\n")
    return(NULL)
  }

  # Convertir los datos a un data frame para ggplot2
  data_rmse <- data.frame(
    Estacion = names(errores_rmse),
    RMSE = as.numeric(errores_rmse)
  )

  # Crear el gráfico de barras
  ggplot(data_rmse, aes(x = Estacion, y = RMSE)) +
    geom_bar(stat = "identity", fill = "skyblue", color = "black", alpha =
0.8) +
    labs(
      title = "RMSE por Estación",
      x = "Estaciones",
      y = "RMSE"
    ) +
    theme_minimal() +
    theme(
      axis.text.x = element_text(angle = 45, hjust = 1, size = 10),
      plot.title = element_text(size = 16, face = "bold"),
      axis.title = element_text(size = 12),
      panel.grid.major.y = element_line(linetype = "dashed", color =
"gray", size = 0.5)
    )
}


graficar_rmse_por_estacion(errores_rmse = resultado$ErroresRMSE)


# Calcular la matriz de correlación
calcular_matriz_correlacion <- function(data) {
  # Asegurarse de que todas las columnas sean numéricas
  data <- data[, sapply(data, is.numeric)]

  if (ncol(data) == 0) {
    stop("No hay columnas numéricas en los datos.")
  }

  # Eliminar columnas con demasiados valores faltantes
  data <- data[, colSums(!is.na(data)) > 1]

  if (ncol(data) == 0) {
```

```
    stop("No hay columnas suficientes con datos para calcular la
correlación.")
  }

  # Calcular la matriz de correlación
  matriz_correlacion <- cor(data, use = "complete.obs")
  return(matriz_correlacion)
}

# Graficar la matriz de correlación
graficar_matriz_correlacion <- function(matriz_correlacion) {

  # Convertir la matriz de correlación a un formato largo para ggplot2
  matriz_larga <- melt(matriz_correlacion)

  # Crear el mapa de calor
  ggplot(matriz_larga, aes(x = Var1, y = Var2, fill = value)) +
    geom_tile(color = "white") +
    scale_fill_gradient2(low = "blue", mid = "white", high = "red",
midpoint = 0, limit = c(-1, 1), space = "Lab") +
    labs(title = "Matriz de Correlación entre Estaciones", x =
"Estaciones", y = "Estaciones", fill = "Correlación") +
    theme_minimal() +
    theme(
      axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1),
      plot.title = element_text(size = 16, face = "bold")
    )
}

matriz_correlacion <-
calcular_matriz_correlacion(resultado$SerieCompletada)
print(matriz_correlacion)

graficar_matriz_correlacion(matriz_correlacion)

graficar_comparacion <- function(resultado, estacion) {
  # Verificar que la estación exista en los datos
  if (!(estacion %in% colnames(resultado$SerieOriginal))) {
    cat(sprintf("La estación '%s' no existe en los datos.\n", estacion))
    return(NULL)
  }

  # Extraer las series original y completada
  serie_original <- resultado$SerieOriginal[, estacion, drop = FALSE]
  serie_completada <- resultado$SerieCompletada[, estacion, drop = FALSE]

  # Combinar ambas series en un solo data frame para ggplot2
  data_grafico <- data.frame(
    Fecha = resultado$SerieOriginal$Fecha,
    Original = as.numeric(serie_original[, 1]),
    Completada = as.numeric(serie_completada[, 1])
  )

  # Pasar a formato largo para ggplot2
  data_grafico <- reshape2::melt(data_grafico, id.vars = "Fecha",
variable.name = "Tipo", value.name = "Valor")
  data_grafico$Fecha <- as.Date(data_grafico$Fecha)  # Asegurar que la
fecha sea de tipo Date

  # Crear el gráfico
```

```
  ggplot(data_grafico, aes(x = Fecha, y = Valor, color = Tipo, linetype =
Tipo)) +
    geom_line(size = 1) +
    labs(
      title = sprintf("Comparación de la Estación: %s", estacion),
      x = "Fecha",
      y = "Precipitación",
      color = "Serie",
      linetype = "Serie"
    ) +
    scale_color_manual(values = c("blue", "orange")) +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 16, face = "bold"),
      axis.title = element_text(size = 12),
      legend.title = element_text(size = 12),
      legend.text = element_text(size = 10)
    )
}

graficar_comparacion(resultado, "chalaco")
graficar_comparacion(resultado, "huamarca")
graficar_comparacion(resultado, "huancabamba")
```