UNIVERSIDAD DE ANTIOQUIA

Desarrollo de una herramienta computacional para la estimación automatizada de la confiabilidad: Aplicación a repuestos, sistemas y equipos a partir de datos históricos de fallas en una empresa del sector cárnico

Adrián David Cogollo Sáez

Informe de práctica presentado para optar al título de Ingeniero Mecánico

Modalidad de Práctica

Semestre de Industria o Práctica Empresarial

Asesores

Interno: Juan Carlos Orrego Barrera, Ingeniero Mecánico

Externo: José Andrés Marín Sierra, Ingeniero Mecánico

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería Mecánica

Medellín, Antioquia, Colombia

2026

Dedicatoria

Dedicado a mi mamá Nelly, a mi papá Gustavo y a mi hermana Adriana, gracias por su amor incondicional.

Agradecimientos

Agradezco de manera especial a mi asesor Juan Carlos, por su orientación, acompañamiento y valiosos aportes durante el desarrollo de este proyecto. A mi cuñado Elkin, por sus consejos, su disposición y el apoyo brindado en momentos clave. A mis amigos Michael y Felipe, gracias por su respaldo, ánimo y compañía cuando más lo necesité. Finalmente, extiendo mi agradecimiento a todas aquellas personas que, de una u otra manera, contribuyeron con su tiempo, conocimientos y apoyo para hacer posible la culminación de este proyecto. A todos, gracias infinitas.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

BI	Business Intelligence
CDF	Cumulative Distribution Function (Función de distribución acumulada)
ERP	Enterprise Resource Planning
KPI	Key Performance Indicators (Indicadores clave de desempeño)
MTBF	Mean Time Between Failures (Tiempo Medio Entre Fallas)
MTTR	Mean Time To Repair (Tiempo Medio de Reparación)
PCC	Punto Crítico de Control
PDF	Probability Density Function (Función de densidad de probabilidad)
PMO	Plant Maintenance Optimization (Optimización del Mantenimiento Planeado)
RCM	Reliability-Centered Maintenance (Mantenimiento Centrado en la Confiabilidad)
SAP	Systems, Applications and Products in Data Processing
TPM	Total Productive Maintenance (Mantenimiento Productivo Total)
TTF	Time To Failure (Tiempo hasta la falla)

RESUMEN

La confiabilidad de los activos es un elemento fundamental en la gestión del mantenimiento industrial, sin embargo, la información histórica almacenada en sistemas empresariales como SAP suele ser poco explotada desde un enfoque estadístico. En este contexto, el presente trabajo tiene como objetivo desarrollar una herramienta computacional para la estimación automatizada de la confiabilidad de repuestos, sistemas y equipos, a partir de datos históricos de una empresa del sector cárnico. La metodología empleada se basa en un enfoque cuantitativo y analítico que integra la extracción y depuración de datos de SAP, la estimación indirecta de los tiempos hasta la falla mediante el análisis de movimientos de mercancía y el modelamiento estadístico del comportamiento de falla utilizando Python y R. La selección de los modelos probabilísticos se realiza de forma objetiva mediante pruebas de bondad de ajuste. Los resultados se integran en un dashboard interactivo desarrollado en Power BI, que permite analizar la confiabilidad a diferentes niveles jerárquicos. Los resultados evidencian comportamientos de falla asociados principalmente a mecanismos de desgaste y envejecimiento, lo que respalda la aplicación de estrategias de mantenimiento preventivo y predictivo. En conclusión, la herramienta desarrollada aporta información analítica útil para la toma de decisiones en mantenimiento.

Palabras clave — **Confiabilidad, mantenimiento, tiempos hasta la falla, análisis estadístico, Power BI.**

ABSTRACT

Asset reliability is a key aspect of industrial maintenance management; however, historical data stored in enterprise systems such as SAP are often underutilized. This work aims to develop a computational tool for the automated estimation of the reliability of spare parts, systems, and equipment using historical data from a meat processing company. The methodology follows a quantitative approach that integrates data extraction from SAP, indirect estimation of time to failure through material movement analysis, and statistical modeling using Python and R. Probability models are selected through goodness-of-fit tests, and the results are integrated into an interactive Power BI dashboard for hierarchical reliability analysis. The results show failure behaviors mainly associated with wear and aging mechanisms, supporting preventive and predictive maintenance strategies. In conclusion, the proposed tool provides actionable analytical information to support maintenance decision-making.

Keywords **— Reliability, maintenance management, time to failure, statistical modeling, Power BI.**

I. INTRODUCCIÓN

En el contexto industrial actual, la gestión del mantenimiento enfrenta el desafío de garantizar altos niveles de confiabilidad, disponibilidad y seguridad en los activos productivos, al tiempo que se optimizan los recursos técnicos y económicos. En efecto, la creciente complejidad de los sistemas, junto con la presión por mejorar la eficiencia operativa, ha impulsado la necesidad de incorporar enfoques analíticos y basados en datos que permitan comprender de manera objetiva el comportamiento de falla de equipos y repuestos. Así pues, la ingeniería de confiabilidad se consolida como un pilar fundamental para sustentar la toma de decisiones en mantenimiento preventivo, predictivo y estratégico.

Ahora bien, uno de los principales obstáculos para la aplicación efectiva de estos enfoques radica en la limitada explotación de la información histórica de mantenimiento, la cual, si bien se encuentra almacenada en sistemas empresariales como SAP ERP, suele utilizarse principalmente con fines operativos y administrativos. En cambio, el análisis estadístico profundo de los tiempos hasta la falla y su modelamiento probabilístico no siempre se realiza de forma sistemática. Dado que los reportes estándar de estos sistemas son, en su mayoría, estáticos y descriptivos, resulta complejo identificar patrones de falla, evaluar tendencias de confiabilidad o anticipar comportamientos futuros de los activos.

Teniendo en cuenta lo anterior, surge la necesidad de desarrollar herramientas que transformen los datos históricos de mantenimiento en información analítica accionable, integrando técnicas estadísticas de confiabilidad con plataformas modernas de visualización. En este sentido, el presente proyecto se orienta a cerrar la brecha existente entre los datos operativos y el análisis avanzado, mediante el desarrollo de una aplicación que permita calcular la confiabilidad de repuestos y equipos a partir de los TTF, empleando modelos estadísticos validados mediante pruebas de bondad de ajuste. De igual manera, se busca que los resultados obtenidos puedan ser interpretados de forma clara y dinámica por medio de dashboards interactivos, facilitando su uso por parte del personal de mantenimiento.

Por consiguiente, el objetivo central de este trabajo consiste en desarrollar una solución computacional que integre el análisis estadístico de confiabilidad con herramientas de visualización, utilizando Python y R para el procesamiento y validación de los datos, y Power BI para la presentación de los resultados. Acto seguido, se plantea como objetivos específicos la

automatización del cálculo de TTF, la selección objetiva de distribuciones de probabilidad, la generación de curvas de supervivencia y la construcción de indicadores clave que permitan evaluar el desempeño de los activos desde una perspectiva probabilística.

En cuanto al planteamiento del problema, este proyecto busca responder interrogantes como: ¿qué modelos estadísticos describen mejor el comportamiento de falla de los repuestos analizados?, ¿cómo se distribuye la confiabilidad a lo largo del tiempo?, y ¿de qué manera esta información puede apoyar la priorización de acciones de mantenimiento? En efecto, abordar estas preguntas permite no solo caracterizar el comportamiento histórico de los equipos, sino también anticipar escenarios futuros y evaluar el impacto de distintas estrategias de mantenimiento.

II. OBJETIVOS

A. Objetivo general

Desarrollar una aplicación que permita calcular la confiabilidad de repuestos y equipos a partir de los tiempos hasta la falla (TTF), utilizando análisis estadístico y pruebas de bondad de ajuste mediante R, integrando los resultados en un archivo exportable que pueda visualizarse mediante un dashboard interactivo en Power BI.

B. Objetivos específicos

- Diseñar una herramienta de software en Python con integración en R para ejecutar análisis de distribución estadística sobre datos de fallas de repuestos y equipos.
- Automatizar el proceso de validación, cálculo de TTF y selección de la mejor distribución de probabilidad, mediante comparación de p-values en pruebas de bondad de ajuste.
- Generar una tabla de confiabilidad y curvas de supervivencia a partir de la función de distribución que mejor se ajuste a los datos.
- Exportar los resultados en un archivo de Excel estructurado, apto para su integración con un dashboard de visualización de KPIs de confiabilidad.
- Diseñar un dashboard interactivo en Power BI que permita visualizar las curvas de supervivencia, métricas de confiabilidad y otros indicadores relevantes de forma gráfica y dinámica.

III. MARCO TEÓRICO

A. Mantenimiento Centrado en la Confiabilidad (RCM)

El Mantenimiento Centrado en la Confiabilidad (RCM, por sus siglas en inglés) es una metodología sistemática y estructurada que permite definir los requerimientos de mantenimiento necesarios para asegurar que los activos cumplan sus funciones dentro de su contexto operativo. De acuerdo con Silva y Orrego [1], el RCM consiste en analizar funciones, identificar posibles fallas, evaluar sus causas, estudiar sus efectos y determinar las acciones de mantenimiento requeridas para garantizar la operación de los activos, a su vez, aplicarlo conjuntamente con metodologías como TPM y PMO "permite satisfacer los requerimientos de la organización en la medida justa y de manera adecuada", asegurando eficiencia técnica y operativa.

El origen del RCM se remonta a la industria aeronáutica, cuando los ingenieros de United Airlines, Stanley Nowlan y Howard Heap desarrollaron un proceso sistemático para reducir la alta tasa de accidentalidad y optimizar las labores de mantenimiento. Este trabajo culminó en el informe Reliability-Centered Maintenance, que dio nombre y estructura formal a la metodología. En dicho documento, Nowlan y Heap definieron el RCM como un proceso lógico diseñado para identificar las acciones de mantenimiento más efectivas, con el fin de garantizar que los sistemas continúen desempeñando sus funciones deseadas dentro de su entorno operacional específico [2].

Este enfoque representó una evolución sustancial respecto a los métodos tradicionales de mantenimiento preventivo, basados principalmente en intervalos de tiempo fijos ("hard-time"). Si bien estos métodos resultaban eficaces en sistemas simples, se tornaron ineficientes en equipos modernos debido a su creciente complejidad tecnológica y a la variabilidad de los modos de falla [2]. Nowlan y Heap demostraron que muchas fallas no dependen directamente del tiempo ni de la edad del componente, sino de condiciones operativas, cargas y factores ambientales. Por ello, aplicar mantenimientos periódicos sin considerar la función real del activo puede generar sobrecostos, fallas inducidas y pérdida de disponibilidad.

El análisis comienza con la identificación de las funciones que el sistema debe cumplir y los estándares de desempeño esperados; continúa con la determinación de las fallas funcionales y sus modos de falla asociados; y culmina con la evaluación de las consecuencias, clasificándolas en categorías de seguridad, operacionales, no operacionales y ocultas [2]. Este proceso analítico

permite priorizar los recursos de mantenimiento según la criticidad del componente y el impacto real sobre la operación y la seguridad.

Desde una perspectiva contemporánea, Moubray [3] amplió el alcance del RCM al considerarlo una metodología que busca no solo conservar la maquinaria en funcionamiento, sino preservar las funciones que el usuario requiere dentro del contexto operativo. En su planteamiento, el RCM se sustenta en siete preguntas fundamentales que orientan la toma de decisiones respecto a las estrategias de mantenimiento más efectivas. Dichas preguntas abordan las funciones del activo, los modos y causas de falla, las consecuencias derivadas y las tareas proactivas destinadas a evitar o detectar la pérdida de funcionalidad. A través de este esquema lógico, el RCM se convierte en una herramienta que integra el análisis técnico con la gestión del riesgo, facilitando la asignación óptima de recursos y priorizando las intervenciones de acuerdo con la criticidad funcional.

Por su parte, Smith y Hinchcliffe [4], en su obra RCM: Gateway to World Class Maintenance, sostienen que esta metodología constituye un elemento esencial para alcanzar niveles de mantenimiento de clase mundial (World Class Maintenance, WCM). Su implementación ha demostrado reducciones significativas en los costos de mantenimiento correctivo y en los tiempos de inactividad, además de fortalecer la cultura de análisis, retroalimentación y mejora continua. Estudios citados por los autores en el sector eléctrico reportaron periodos de recuperación de la inversión en RCM inferiores a un año, gracias a los beneficios indirectos obtenidos en rediseño, operación y gestión de activos.

Smith y Hinchcliffe también explican que el RCM se consolida como la vía más efectiva para maximizar la confiabilidad, disponibilidad y desempeño de los equipos, orientando las estrategias de mantenimiento hacia el mayor retorno de inversión (ROI) y promoviendo una cultura de excelencia operacional sustentada en el análisis técnico y la mejora continua. Su aplicación práctica permite a las organizaciones transitar desde un mantenimiento reactivo hacia un modelo predictivo y estratégico, alineado con los principios del mantenimiento de clase mundial [4].

B. Indicadores de desempeño en mantenimiento

Con respecto a los indicadores clave de desempeño (KPI), estos constituyen herramientas esenciales para evaluar la eficiencia y efectividad de los activos físicos. Entre los más relevantes, Silva y Orrego [1] destacan los indicadores dependientes del tiempo, particularmente el MTBF

(Mean Time Between Failures) y el MTTR (Mean Time To Repair). Ambos se emplean de manera conjunta en los análisis RAM (Reliability, Availability and Maintainability) con el propósito de "determinar la producción perdida y la indisponibilidad de un proceso de producción, de acuerdo con la configuración de sus equipos, la confiabilidad de sus componentes, las políticas de mantenimiento y la filosofía operacional existente".

Estos dos autores definen el MTBF como el promedio del tiempo transcurrido entre dos fallas consecutivas de un componente o sistema, reflejando su confiabilidad intrínseca. Por su parte, el MTTR representa el tiempo promedio necesario para restaurar la funcionalidad del equipo tras una falla, constituyendo así un indicador de mantenibilidad, se puede observar en la Fig. 1 un esquema de estos KPI. En conjunto, ambos parámetros permiten cuantificar la capacidad de un sistema para operar de forma continua y eficiente.

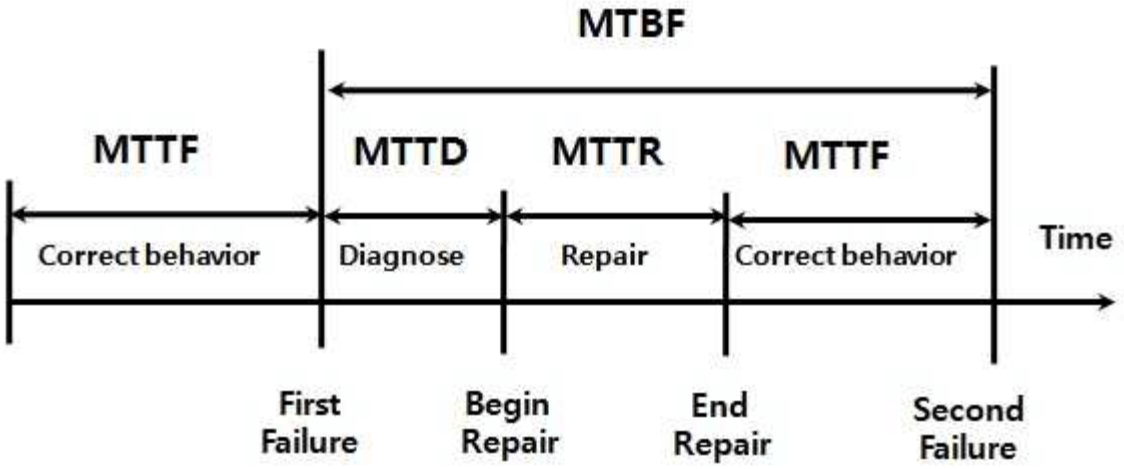

Fig. 1. Diferencias entre MTTF, MTTD, MTTR y MTBF.

Nota: Fuente [5]

Complementariamente, O'Connor y Kleyner [6] definen la confiabilidad como "la probabilidad de que un artículo cumpla una función requerida sin fallar, bajo condiciones especificadas, durante un período de tiempo determinado". Esta definición resalta el carácter probabilístico de las fallas y la necesidad de analizar estadísticamente los comportamientos de los equipos a lo largo del tiempo.

En cuanto a la disponibilidad, Silva y Orrego [1] la describen como el porcentaje del tiempo total en que un equipo se encuentra en condiciones de operar según su propósito. En la Ecuación (1) se encuentra la definición matemática de la disponibilidad, representada por la letra A.

$$A = \frac{MTBF}{MTBF + MTTR} \quad (1)$$

Esta relación evidencia el equilibrio entre confiabilidad y mantenibilidad, convirtiéndose en un indicador directo de la capacidad productiva y eficiencia global del mantenimiento. A su vez, la mantenibilidad se define como la probabilidad de restaurar un equipo a su condición operativa dentro de un tiempo determinado tras una falla no planificada, lo que la convierte en un parámetro fundamental para el diseño de estrategias de mantenimiento y la gestión de repuestos.

Si bien indicadores como el MTTF, MTBF y MTTR suelen presentarse de forma simplificada como valores promedio, en rigor estos no deben interpretarse como simples medias aritméticas. Desde el enfoque de confiabilidad, los tiempos hasta la falla y los tiempos de reparación corresponden a variables aleatorias, cuyo comportamiento debe ser analizado mediante modelos estadísticos adecuados para que los indicadores resultantes sean representativos del fenómeno físico observado.

En este sentido, el MTTF y el MTBF adquieren validez técnica únicamente cuando se derivan de un conjunto de datos homogéneos, correctamente depurados y bajo un control adecuado del proceso de ocurrencia de fallas, lo cual implica una definición clara del evento de falla, consistencia en los registros históricos y estabilidad en las condiciones operativas. Bajo estas condiciones, dichos indicadores pueden ser estimados a partir de los parámetros de la distribución de probabilidad que mejor se ajusta a los datos, y no únicamente como una media aritmética.

De manera análoga, el MTTR solo resulta representativo cuando el proceso de mantenimiento se encuentra estandarizado, es decir, cuando las actividades de diagnóstico, intervención y puesta en marcha siguen procedimientos definidos y repetibles. La estandarización de las tareas permite reducir la variabilidad artificial en los tiempos de reparación y asegura que el MTTR refleje realmente la mantenibilidad del activo y no diferencias operativas o humanas.

Bajo este enfoque, los indicadores MTTF, MTBF y MTTR dejan de ser métricas descriptivas aisladas y se convierten en parámetros técnicos que permiten evaluar y controlar tanto la confiabilidad como la disciplina operativa del proceso de mantenimiento.

C. Modelamiento estadístico del tiempo hasta la falla

Según Nowlan y Heap [2], el análisis de los datos históricos de fallas resulta indispensable para establecer patrones de comportamiento, modos dominantes de falla y oportunidades de mejora. Los autores enfatizan la importancia de contar con registros sistemáticos que integren

información proveniente de diferentes fuentes, como operación, mantenimiento y costos, de modo que sea posible evaluar la efectividad de las tareas programadas y ajustar los planes de mantenimiento con base en evidencia empírica.

En este contexto, el análisis TTF adquiere un papel central. Este análisis estudia los intervalos de tiempo transcurridos entre el inicio de la operación y la ocurrencia de una falla, permitiendo identificar el comportamiento probabilístico del deterioro de los componentes. Desde la perspectiva del RCM, dicho comportamiento es clave para seleccionar estrategias de mantenimiento acordes con el tipo de fallo [3].

Moubray también resalta que el mantenimiento moderno debe sustentarse en decisiones basadas en evidencia cuantitativa, y no en suposiciones empíricas. Así, comprender la naturaleza aleatoria de las fallas y utilizar modelos estadísticos que permitan estimar su probabilidad y severidad se convierte en un requisito esencial para una gestión técnica eficiente.

En la práctica industrial moderna, sistemas integrados como SAP ERP cumplen un papel determinante en esta tarea. Estas plataformas centralizan la información relacionada con órdenes de trabajo, tiempos de parada, costos, materiales y causas de falla, facilitando su análisis mediante técnicas estadísticas y herramientas analíticas. Gracias a ello, es posible calcular indicadores como el MTBF y el MTTR, detectar tendencias de fallas y priorizar acciones correctivas o preventivas en función de la criticidad del equipo.

En el libro Statistical Models and Methods for Lifetime Data de Lawless [7], se introduce el concepto de variable aleatoria como una magnitud numérica asociada a los fenómenos de vida o fallas que puede adoptar distintos valores de manera incierta. Estas variables describen el comportamiento estocástico del tiempo hasta la falla y constituyen la base para construir modelos de confiabilidad. En este contexto, el autor señala que una variable aleatoria T puede representarse mediante una función de distribución de probabilidad (CDF) $F(t)$, que expresa la probabilidad de que el evento ocurra antes o igual a un tiempo dado t, así como lo presenta la Ecuación (2).

$$F(t) = P(T \leq t) \quad (2)$$

Asimismo, Lawless distingue entre modelos continuos y discretos dependiendo de la naturaleza de la variable T. En los modelos continuos, típicos del análisis de confiabilidad, T puede tomar cualquier valor positivo real y su comportamiento se describe mediante una función de densidad de probabilidad (PDF) $f(t)$, la cual cuantifica la probabilidad relativa de que la falla ocurra en un intervalo infinitesimal alrededor de t. En cambio, cuando los datos se obtienen en

forma agrupada o en ciclos discretos, T se trata como una variable aleatoria discreta, definida por una función de probabilidad (PF) $f(t_i)$, donde t_i son los posibles valores de T. Esta función de probabilidad aparece definida en la Ecuación (3).

$$f(t_i) = P(T = t_i) \quad (3)$$

En el ámbito de la confiabilidad, la selección de la distribución de probabilidad adecuada para modelar el tiempo hasta la falla de un componente resulta esencial, ya que determina la precisión de los análisis y las estrategias de mantenimiento. Algunas funciones de distribución paramétricas que son comúnmente usadas para aplicaciones de confiabilidad para componentes son:

- **Weibull**: Es una de las más utilizadas debido a su flexibilidad para representar diferentes etapas del ciclo de vida de un componente, su CDF se puede observar en la Ecuación (4). Según Meeker, esta función depende del parámetro de forma (β) y del parámetro de escala (η). Su función de tasa de falla puede ser creciente, constante o decreciente dependiendo del parámetro de forma β: para $\beta < 1$ describe fallas tempranas (infantiles), $\beta = 1$ se asocia a fallas aleatorias (similar a la exponencial) y $\beta > 1$ representa procesos de desgaste o envejecimiento. Su versatilidad permite modelar tanto componentes electrónicos como mecánicos sujetos a fatiga, corrosión o desgaste progresivo [8].

$$F(t; \eta, \beta) = 1 - e^{-\left(\frac{t}{\eta}\right)^{\beta}} \quad (4)$$

- **Exponencial**: Es un caso particular de la Weibull con $\beta = 1$. Su parámetro característico es el parámetro de escala (λ), que, en aplicaciones de confiabilidad, es la tasa de fallas, la cual es constante, lo que implica independencia del tiempo. Se utiliza para describir componentes electrónicos o sistemas con fallas puramente aleatorias, donde el envejecimiento no afecta la probabilidad de falla. En la Ecuación (5) se presenta su CDF [8].

$$F(t; \lambda) = 1 - e^{-\lambda t} \quad (5)$$

- **Normal**: Se caracteriza por los parámetros media (μ) y desviación estándar (σ). Aunque es menos común en análisis de fallas, puede emplearse cuando los tiempos de vida presentan simetría y pequeñas variaciones, como en resistencias de materiales o componentes cuya variabilidad se debe a procesos de fabricación controlados. En la Ecuación (6) se ve su CDF, donde erf es la función error [8].

$$F(t;\mu,\sigma) = \frac{1}{2}\left[1 + erf\left(\frac{t-\mu}{\sigma\sqrt{2}}\right)\right] \quad (6)$$

- **Log-normal**: Resulta adecuada cuando el logaritmo del tiempo hasta la falla sigue una distribución normal. Sus parámetros son el promedio (μ) y la desviación estándar (σ) del logaritmo de T. Es útil para representar fenómenos de desgaste progresivo, donde los mecanismos de falla son multiplicativos, como la corrosión o la fatiga. Se puede ver su CDF en la Ecuación (7), donde erf es la función error [8].

$$F(t;\mu,\sigma) = \frac{1}{2}\left[1 + erf\left(\frac{\ln t-\mu}{\sigma\sqrt{2}}\right)\right] \quad (7)$$

- **Log-logística**: Similar a la log-normal, pero con colas más pesadas, lo que la hace apropiada para modelar datos con alta variabilidad o donde las fallas pueden presentarse muy temprano o muy tarde en la vida útil del componente [8]. Forma parte de la familia de distribuciones log-localización-escala junto con la Weibull y la log-normal, y así como esta primera, depende del parámetro de forma (β) y del parámetro de escala (η), como se puede ver en la Ecuación (8).

$$F(t;\eta,\beta) = \frac{1}{1 + \left(\frac{t}{\eta}\right)^{-\beta}} \quad (8)$$

- **Gamma**: Empleada para describir tiempos de falla acumulativos resultantes de múltiples procesos exponenciales. Posee dos parámetros: forma (k) y escala (θ). Su flexibilidad la hace apropiada para representar datos con tasas de falla crecientes o decrecientes, especialmente en sistemas mecánicos sometidos a desgaste. Su CDF se ve en la Ecuación (9), donde Γ es la función Gamma completa, mientras que γ es la función Gamma incompleta inferior [8].

$$F(t;k,\theta) = \frac{1}{\Gamma(k)}\gamma\left(k,\frac{t}{\theta}\right) \quad (9)$$

- **Birnbaum–Saunders**: Derivada de un modelo de crecimiento de grietas por fatiga, describe adecuadamente los tiempos hasta fractura de materiales. Se define mediante un parámetro de escala (β) y un parámetro de forma (θ). Su función de tasa de falla es unimodal, iniciando en cero e incrementándose hasta un máximo antes de disminuir, comportamiento típico de procesos de degradación mecánica. Su CDF se encuentra en la Ecuación (10), donde Φ es la CDF de la normal estándar [8].

$$F(t;\alpha,\beta) = \Phi\left(\frac{1}{\alpha}\left[\sqrt{\frac{t}{\beta}} - \sqrt{\frac{\beta}{t}}\right]\right) \quad (10)$$

- **Fréchet**: Es una de las distribuciones límite fundamentales en la teoría de valores extremos, siendo útil para casos de fatiga, ya que se adecua bien para modelar los valores máximos alcanzados por una característica crítica, como la longitud de una grieta antes de causar falla [9]. Matemáticamente, la CDF de la distribución Fréchet puede expresar en la Ecuación (11), donde γ es el parámetro de forma y θ es el parámetro de escala.

$$F(t;\gamma,\theta) = e^{-\left(\frac{t}{\theta}\right)^{-\gamma}} \quad (11)$$

- **Burr**: También conocida como distribución Burr Tipo XII o Singh-Maddala, esta distribución se caracteriza por su alta flexibilidad, lo que la convierte en una herramienta idónea para modelar datos con colas pesadas, un comportamiento frecuente en estudios de confiabilidad y análisis de vida útil. La Ecuación (12) se puede observar su CDF, la cual depende de dos parámetros, el primero (c) controla la pendiente inicial y la forma general de la curva, e influye en la velocidad de crecimiento de la CDF y la asimetría de la distribución, y el segundo parámetro (k) modifica la curvatura de la cola y determina cuán pesada o liviana es la cola de la distribución [10].

$$F(t;c,k) = 1 - \frac{1}{(1+t^c)^k} \quad (12)$$

- **Gumbel**: Esta distribución se caracteriza por tener dos parámetros principales, un parámetro de escala (α) y un parámetro de forma (λ) , tal como se observa en su CDF presentada en la Ecuación (13). Su aplicación es particularmente común en componentes sometidos a procesos de desgaste progresivo, como la corrosión, donde la probabilidad de falla aumenta con el tiempo debido a la degradación gradual del material [11].

$$F(t;\alpha,\lambda) = e^{-e^{-\frac{t-\lambda}{\alpha}}} \quad (13)$$

D. Pruebas de bondad de ajuste

En los estudios de confiabilidad, es habitual verificar si un modelo estadístico describe adecuadamente el comportamiento observado de los TTF. Para representar fielmente el patrón de los datos, es necesario comprobar si estos se ajustan a una distribución teórica específica, lo cual se realiza mediante las pruebas de bondad de ajuste.

Estas pruebas se basan en la formulación de hipótesis estadísticas, donde la hipótesis nula (H_0) plantea que la distribución empírica $F(t)$ proviene de una distribución teórica propuesta $F_0(t)$. Este proceso resulta fundamental, ya que permite validar la representatividad del modelo y, por tanto, estimar con mayor precisión la confiabilidad, la tasa de falla y la vida útil de los equipos. De esta manera, las políticas de mantenimiento preventivo, predictivo o de reemplazo pueden diseñarse con base en evidencia cuantitativa, incrementando la efectividad y la disponibilidad operativa de los activos [7].

A lo largo de los años, se han desarrollado métodos para realizar pruebas de bondad de ajuste, entre las más destacadas se encuentran:

- **Kolmogorov - Smirnov**: Es uno de los test más utilizados para evaluar la adecuación de un modelo teórico de distribución con respecto a una muestra de datos, y se define como la máxima diferencia absoluta (D_n) entre ambas funciones de distribución [7], como se puede ver en la Ecuación (14).

$$D_n = sup|F(t) - F_0(t)| \quad (14)$$

- **Chi - cuadrado:** Esta prueba se fundamenta en comparar las frecuencias observadas y esperadas de un conjunto de datos distribuidos en clases o intervalos. Su expresión general se puede ver en la Ecuación (15).

$$\chi^2_{(\ell-p)} = \sum_{i=1}^{\ell} \frac{(O_i - E_i)^2}{E_i} \quad (15)$$

Donde O_i representa el número de observaciones registradas en la clase i, E_i el número de observaciones esperadas según la distribución teórica, ℓ el número de clases y p la cantidad de parámetros estimados para dicha distribución. Para garantizar la validez estadística de la prueba, se recomienda que cada clase contenga al menos cinco observaciones esperadas [12].

- **Anderson - Darling**: Este test resulta particularmente útil para comparar distribuciones continuas, dado que considera con mayor sensibilidad las discrepancias en las colas de la distribución, a diferencia del test de Kolmogorov - Smirnov, que solo evalúa la desviación máxima entre las CDF [13]. Esta prueba calcula la diferencia entre la CDF empírica y la teórica y se obtiene el estadístico A^2, que mide la distancia acumulada entre ambas. Dicho estadístico se define en la Ecuación (16).

$$A^2 = -\frac{1}{N}\sum_{i=1}^{N}(2i-1)[\ln F(t_i) + \ln(1 - F(t_{N+1-i}))] \quad (16)$$

Donde $F(t_i)$ representa la CDF teórica evaluada en los datos ordenados t_i y N es el tamaño de la muestra. El test de Anderson - Darling ha demostrado una mayor precisión que otras pruebas en la validación de distribuciones continuas, motivo por el cual se considera más adecuado en estudios de variabilidad y desempeño donde los datos se originan de fenómenos físicos, casos comunes cuando se evalúa la confiabilidad.

- **Cramér - von Mises**: A diferencia de otras pruebas como la de Kolmogorov - Smirnov, esta se basa en la suma ponderada de las diferencias cuadráticas entre la función de distribución empírica y la función de distribución teórica acumulada, otorgando una sensibilidad más uniforme a lo largo de todo el rango de los datos [14]. Matemáticamente, el estadístico de la prueba (W^2) se expresa en la Ecuación (17):

$$W^2 = n\int_{-\infty}^{\infty}[F_n(x) - F(x)]^2 dF(x) \quad (17)$$

Donde $F_n(x)$ representa la función de distribución empírica y $F(x)$ la función de distribución teórica. Este enfoque integral permite una comparación global entre ambas distribuciones, evitando que el resultado dependa excesivamente de las diferencias máximas locales, como ocurre en el test de Kolmogorov – Smirnov [14].

E. Criterios de comparación entre modelos

Una vez se seleccionan las posibles distribuciones que podrían representar el comportamiento de la muestra de datos, se tiene que evaluar cual de estas ofrece el mejor ajuste estadístico. Para ello, es necesario aplicar criterios cuantitativos de comparación entre modelos, los cuales permiten medir la calidad del ajuste y la exactitud del modelo a la hora de representar los TTF. Entre los más en el análisis de confiabilidad se encuentran:

- **P-value**: El p-valor o p-value se define estadísticamente como la probabilidad de obtener un estadístico de prueba $T(y_{rep})$ más extremo que el observado $T(y)$, bajo la hipótesis nula H_0, así como se explica en la Ecuación (18).

$$p = \Pr\left(T\left(y_{rep}\right) > T(y)|H_0\right) \quad (18)$$

Donde y_{rep} representa una replicación hipotética de los datos bajo el modelo nulo [15]. El p-value se puede interpretar como una métrica de cuán consistente es el conjunto de datos con el modelo asumido.

En el contexto del análisis estadístico aplicado a la confiabilidad y el mantenimiento, el nivel de significancia (α) representa el umbral de decisión que permite determinar si los datos observados son consistentes con el modelo teórico propuesto. En términos formales, α se define como la probabilidad de rechazar la hipótesis nula H_0 cuando en realidad es verdadera. Este valor, comúnmente establecido en 0,05 o 5%, fija el criterio para evaluar el resultado del p-value obtenido en una prueba de bondad de ajuste [15]. Se pueden evaluar dos casos: Se rechaza la hipótesis nula H_0 y los datos no se ajustan al modelo evaluado $(p < 0{,}05)$ o que se acepte la hipótesis nula H_0 y los datos sí se ajusten al modelo $(p \geq 0{,}05)$.

- **Log-likelihood**: En términos generales, el log-likelihood mide la probabilidad (en escala logarítmica) de observar un conjunto de datos dado un modelo específico. Su variante, el test log-likelihood (TLL), también conocida como *predictive log-likelihood*, se calcula sobre un conjunto de datos de prueba y se utiliza como criterio comparativo entre diferentes modelos o algoritmos de inferencia aproximada aplicados a un mismo modelo probabilístico. Matemáticamente, el test log-likelihood se define como la media del logaritmo de la densidad predictiva del modelo sobre un conjunto de datos de prueba $D^* = \{y_n^*\}_{n=1}^{N^*}$, como se ve en la Ecuación (19).

$$TLL(D^*; \Pi) = \frac{1}{N^*} \sum_{n=1}^{N^*} \log \pi(y_n^* | D) \quad (19)$$

Donde $\pi(y_n^*|D)$ es la densidad predictiva generada por el modelo Π entrenado con los datos D. En la práctica, se selecciona el modelo con mayor TLL, bajo el supuesto de que un valor más alto implica una mejor capacidad predictiva y, por tanto, una mayor similitud entre la distribución predictiva del modelo y la distribución verdadera de los datos [16].

- **Criterio de Información Akaike (AIC)**: Este criterio surge del principio de maximización de la verosimilitud y busca equilibrar la capacidad explicativa del modelo con su complejidad. Matemáticamente, el AIC se define en la Ecuación (20).

$$AIC = -2 \log l(\hat{\theta}_{MLE} | y) + 2p \quad (20)$$

Donde $l(\hat{\theta}_{MLE}|y)$ denota la verosimilitud del modelo para el parámetro θ y para los datos y. La dimensión del parámetro θ se representa por p, y la estimación de la máxima verosimilitud

se denota por $\hat{\theta}_{MLE}$ [17]. El primer término evalúa la bondad del ajuste, mientras que el segundo penaliza el exceso de complejidad, evitando el sobreajuste. En el contexto de análisis de confiabilidad, este balance es esencial, ya que un modelo con demasiados parámetros puede describir correctamente los datos históricos de fallas, pero perder capacidad predictiva ante nuevos escenarios operativos.

En la práctica, el modelo con el menor valor de AIC se considera el más adecuado para representar los datos, siempre que las diferencias sean estadísticamente significativas.

- **Criterio de Información Bayesiano (BIC)**: También conocido como Criterio de Schwartz, es la profundización del enfoque adoptado por el AIC, pero fundamentado en principios bayesianos, es decir, considera también la probabilidad posterior de que un modelo sea verdadero dados los datos observados. Este criterio combina la verosimilitud máxima del modelo con una penalización más estricta por número de parámetros, favoreciendo aquellos modelos que logran un adecuado compromiso entre precisión y simplicidad [17]. Matemáticamente, el BIC se expresa en la Ecuación (21):

$$BIC = -2\log l\left(\hat{\theta}\middle|y\right) + p\log d \qquad (21)$$

Donde d denota el tamaño de la muestra y $\hat{\theta}$ corresponde a la estimación de la media o la moda posteriores. El BIC puede interpretarse como una aproximación al logaritmo del factor de Bayes, que mide cuánta evidencia proporcionan los datos a favor de un modelo frente a otro. Comparando metodologías, mientras el AIC prioriza la capacidad predictiva, el BIC incorpora una perspectiva más conservadora, orientada a evitar el sobreajuste y a seleccionar modelos con una estructura mínima pero suficiente para describir el comportamiento real de los datos. De esta forma, el modelo con el menor valor de BIC es considerado el más adecuado, en tanto logra representar los datos observados sin incluir parámetros innecesarios [17].

F. Confiabilidad y funciones de Confiabilidad

Una vez validado el modelo estadístico que mejor describe el comportamiento de los tiempos hasta la falla, se procede a analizar las curvas de confiabilidad, las cuales permiten caracterizar el desempeño y la evolución del riesgo de falla de un activo a lo largo del tiempo.

Anteriormente se definió la confiabilidad, denotada como $R(t)$, como la probabilidad de que un sistema o componente opere satisfactoriamente durante un intervalo de tiempo t bajo

condiciones especificadas [9]. Desde el punto de vista matemático, esta función representa el complemento de la función de distribución acumulada $F(t)$, así como se ve en la Ecuación (22):

$$R(t) = 1 - F(t) \quad (22)$$

Por su parte, la función de riesgo o hazard rate, $h(t)$, se define como la probabilidad instantánea de falla en el intervalo de tiempo siguiente, dado que el componente ha sobrevivido hasta el instante [9]. En la Ecuación (23) se puede observar su expresión matemática.

$$h(t) = \frac{f(t)}{R(t)} \quad (23)$$

Esta magnitud tiene un significado práctico esencial en confiabilidad: indica la vulnerabilidad del sistema en un momento específico de su vida útil. Por tanto, el comportamiento de $h(t)$ permite clasificar el tipo de régimen de fallas. Los tres comportamientos típicos de $h(t)$ son:

- **Tasa de fallas decreciente (DFR)**: Se caracteriza por una disminución progresiva del riesgo de falla ($h'(t) < 0$) típica de equipos con defectos de fabricación o acople, por lo que es común que se presente una alta frecuencia de fallas iniciales, seguida de una fase de estabilización a medida que los componentes defectuosos son reemplazados o ajustados [9].
- **Tasa de fallas constante (CFR)**: Asociada a sistemas con fallas aleatorias, donde la tasa de riesgo es constante ($h(t) = \lambda$). Este caso describe un proceso de fallas completamente aleatorio, lo cual es común de componentes electrónicos en su vida útil media [9].
- **Tasa de fallas decreciente (IFR)**: Aquí la tasa de riesgo aumenta con el tiempo ($h'(t) > 0$), representando procesos de envejecimiento y desgaste. Ejemplos comunes incluyen válvulas, cojinetes o engranajes sometidos a fatiga y degradación mecánica [9].

Estos tres comportamientos se integran en la conocida curva de la bañera, que combina las tres fases de vida de un componente, así como puede apreciarse en la Fig. 2, donde la forma característica de la curva ilustra la evolución del riesgo de falla a lo largo del tiempo.

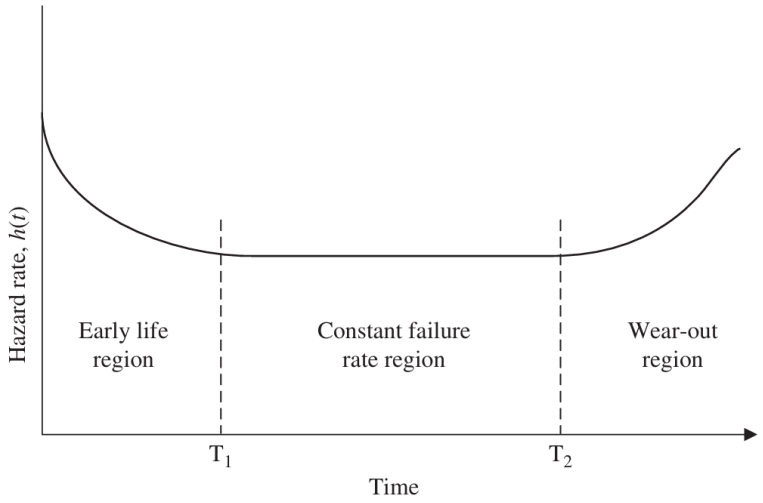

Fig. 2. Curva de la bañera.

Nota: Fuente [9].

G. Diagrama de bloques de confiabilidad

En el ámbito industrial, es frecuente la necesidad de evaluar la confiabilidad global de una máquina o sistema completo. Sin embargo, estos equipos suelen estar conformados por múltiples subsistemas y, a su vez, por miles de componentes interdependientes. Del mismo modo, puede ser necesario analizar la confiabilidad de una línea de producción completa, integrada por diversos equipos que interactúan entre sí.

Ante esta complejidad, los sistemas se descomponen en subconjuntos o componentes elementales, cuyos comportamientos de falla se modelan de manera estructurada para estimar la confiabilidad total. En este contexto, los diagramas de bloques de confiabilidad constituyen una herramienta fundamental, ya que representan gráficamente cómo la falla o supervivencia de cada elemento influye en el desempeño global del sistema. Cada bloque simboliza un componente o subsistema, mientras que las conexiones entre ellos describen la lógica funcional del conjunto, permitiendo identificar los elementos críticos que determinan la continuidad operativa [9].

En términos generales, estos diagramas pueden adoptar dos configuraciones básicas:

- **Configuración en serie:** Si dos componentes están acoplados en serie, todo el arreglo falla si al menos uno de ellos falla. Matemáticamente, la confiabilidad total se expresa como en la Ecuación (24).

$$R_s = \prod_{i=1}^{n} R_i \qquad (24)$$

Donde R_i es la confiabilidad del componente i, así como se puede ver en la Fig. 3Fig. 3. Diagrama de bloques de confiabilidad de un sistema en serie.. Este modelo es típico en cadenas de procesos sin redundancia, donde una sola interrupción provoca el paro total, por lo tanto, acoplar componentes en serie disminuye la confiabilidad global [9].

Fig. 3. Diagrama de bloques de confiabilidad de un sistema en serie.

Nota: Fuente [9].

- **Configuración en paralelo**: Si dos componentes están acoplados en paralelo, el sistema sigue operando mientras al menos un componente continúe funcionando. La confiabilidad total se obtiene como en la Ecuación (25).

$$R_p = 1 - \prod_{i=1}^{n}(1 - R_i) \qquad (25)$$

Esta configuración representa sistemas con redundancia, diseñados para mantener la operación frente a fallas parciales, como bombas gemelas o ventiladores duplicados, ya que, acoplar componentes en paralelo aumenta la confiabilidad. En la Fig. 4 se puede apreciar un sistema en paralelo.

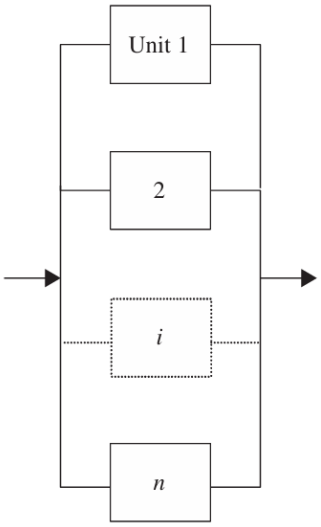

Fig. 4. Diagrama de bloques de confiabilidad de un sistema en paralelo.

Nota: Fuente [9].

H. Visualización de resultados y dashboards

En el contexto de la ingeniería de mantenimiento y confiabilidad, un dashboard de confiabilidad se entiende como una herramienta visual que permite integrar, analizar y comunicar información crítica sobre el desempeño de los activos a partir de indicadores como la confiabilidad, mantenibilidad, disponibilidad y tendencia de fallas. Hay que tener en cuenta que un dashboard no es una diapositiva estática, sino una aplicación web interactiva que sintetiza los resultados analíticos mediante estructuras jerárquicas de información, niveles visuales y componentes dinámicos que facilitan la toma de decisiones [18].

Según Knaflic [19], la eficacia de una visualización depende de su capacidad para reducir la carga cognitiva del observador mediante la eliminación del "ruido visual" y la maximización del data-ink ratio (la proporción de elementos gráficos que comunican información relevante). Este principio se traduce en dashboards de confiabilidad más claros, donde la atención del usuario se dirige a indicadores críticos como la confiabilidad por componente, sistema o equipo.

En cuanto a la integración de resultados estadísticos con Power BI, los dashboards modernos se construyen sobre un proceso robusto de preparación e integración de datos. Esto incluye la importación de resultados desde entornos de análisis como Python o R, el uso de Power Query para limpieza y modelado, y la posibilidad de establecer actualizaciones automáticas desde fuentes externas para mantener los informes sincronizados. Dichas funciones son especialmente útiles en contextos de confiabilidad, donde los resultados de pruebas de bondad de ajuste o distribuciones estadísticas pueden integrarse directamente como indicadores de tendencia o riesgo [18].

Es necesario subrayar que la visualización efectiva de datos representa una ventaja competitiva en la gestión de mantenimiento, ya que convierte grandes volúmenes de información en insights accionables. Un diseño adecuado mejora la comunicación entre equipos técnicos y directivos, fomenta decisiones basadas en evidencia y favorece la transición hacia una cultura de mantenimiento predictivo y analítico dentro de la organización [18].

El conjunto de conceptos descritos conforma el marco teórico que sustenta el desarrollo del presente proyecto, permitiendo integrar el análisis estadístico de confiabilidad con herramientas de inteligencia de negocios para fortalecer la toma de decisiones en la gestión del mantenimiento.

IV. METODOLOGÍA

La metodología desarrollada en este proyecto se fundamenta en un enfoque cuantitativo, analítico y basado en datos históricos de fallas, orientado a la evaluación y modelamiento de la confiabilidad de los equipos a partir de información real proveniente del sistema SAP. El propósito central consiste en transformar los registros de mantenimiento en conocimiento útil para la toma de decisiones, mediante la integración de herramientas estadísticas, programación y visualización de datos.

Se puede subdividir el desarrollo metodológico en las siguientes etapas:

A. Extracción y depuración de datos

En los estudios de confiabilidad, las probabilidades de supervivencia y de falla se determinan a partir de TTF, los cuales finalizan en el momento en que el activo deja de cumplir su función. No obstante, en este proyecto se identificó una limitación en la disponibilidad de información, ya que las bases de datos consultadas no registraban las fechas exactas en las que los repuestos fallaron, dado que ninguna transacción del sistema proveía dicho detalle.

Para superar esta restricción, se implementó un método indirecto de estimación de fallas, basado en el análisis de los movimientos de mercancía del almacén asociados a cada material. De esta forma, el intervalo de tiempo entre dos movimientos consecutivos se interpretó como el período operativo del repuesto, es decir, el tiempo durante el cual el componente permaneció en servicio antes de ser reemplazado. Así, el momento en que se registra una nueva salida del almacén se considera equivalente a la ocurrencia de una falla funcional del repuesto anterior.

Para la extracción de datos, se utilizaron varias transacciones de SAP. Para repuestos, se usó la IW3M y la IW39 para obtener las fechas en las que hubo movimiento de mercancía. Con la IW39 se consultaban las órdenes de trabajo en las que hubo movimiento de mercancía, filtrando por una máquina específica ya que algunos repuestos pueden ser usados en varios equipos, así de esta manera se podía mapear solo el comportamiento del repuesto bajo un mismo equipo y así evitar sesgar la información. Posterior a esto, se filtraban los movimientos de mercancía para un repuesto en específico, incluyendo en el layout de la tabla a qué tipo de orden de trabajo correspondía dicho movimiento, si se cambió debido a una falla (ZAVE o ZPRO) o por un mantenimiento preventivo (ZPRE o ZPRV).

Asimismo, se consideró el tipo de movimiento de mercancía como un campo clave dentro del proceso de análisis. Para garantizar la consistencia y representatividad de los datos, se incluyeron únicamente las salidas de almacén correspondientes al consumo de repuestos, excluyendo las devoluciones, con el fin de evitar sesgos y asegurar que los movimientos analizados reflejaran efectivamente la ocurrencia de una avería real.

En consecuencia, los campos utilizados por el script para el procesamiento de la información y el cálculo de la confiabilidad fueron: la fecha del movimiento, el tipo de movimiento, el código SAP de la orden de trabajo y el tipo de orden, permitiendo así vincular de forma precisa los eventos de mantenimiento con los componentes involucrados.

B. Automatización del proceso en Python y R

El script principal se desarrolló en Python y se estructuró como una herramienta modular orientada al procesamiento de archivos extraídos de SAP y otras fuentes. La lógica se organizó en funciones independientes para cada etapa del proceso (lectura de datos, depuración, cálculo de TTF, integración con R, exportación y manejo de errores), lo que facilita la trazabilidad y el mantenimiento del código.

En cuanto a librerías, se emplearon `pandas` y `numpy` como base para el tratamiento de datos y operaciones numéricas, `glob` y `sys` para la gestión de archivos y la interacción con la terminal, `subprocess` para la instalación dinámica de dependencias (como `rpy2`), `warnings` para suprimir mensajes deprecados y `re` para normalizar y limpiar nombres de columnas. Adicionalmente, se definió una función de impresión segura `_print_nb`, que garantiza la salida de mensajes tanto en entornos de cuaderno (Google Colab) como en consola. En el Anexo A es posible observar el script completo.

El flujo general del script se resume en los siguientes pasos:

1) Inicialización y manejo de errores: Se definen constantes globales (como el nivel de significancia `ALPHA` y el factor de conversión de días a horas) y una lista global `ERRORS`. Las funciones `add_error` y `print_error_summary` centralizan el registro y reporte de errores mediante códigos numéricos y mensajes descriptivos.

2) Lectura robusta del archivo CSV: La función `read_csv_robusto` implementa una lectura escalonada del archivo, probando diferentes combinaciones de codificación (`utf-8`, `latin-1`, `cp1252`, etc.), separadores (coma y punto y coma) y motores de lectura. Además, normaliza los encabezados (`normalize_header_text`) para eliminar caracteres especiales y

espacios, lo que hace al script tolerante frente a variaciones en el formato de exportación desde SAP.

3) Detección automática del tipo de fuente: Una vez cargado el archivo, el script identifica el tipo de layout mediante la presencia de columnas clave: Para repuestos usa las columnas CMv, Fe.contab., Clase_orden y Orden. Para equipos usa las columnas de fecha tipo Fe.entrada o Inic.prog. y, opcionalmente, una columna de identificación de equipo y para un formato general usa las columnas con TTF directo (Tiempo_dias). En función de esta detección se selecciona uno de los tres caminos de preprocesamiento: `preprocess_sap_to_ttf`, `preprocess_equipos_to_ttf` o `preprocess_general_ttf`.

4) Preprocesamiento y cálculo de TTF: Para repuestos, se realiza limpieza de clase de movimientos de mercancía CMv = 262 más su fila vecina más cercana en el tiempo, se convierten y ordenan las fechas, se filtra por clases de orden relevantes (ZPRO y ZAVE) y se calcula el tiempo entre fallos (Tiempo_dias) a partir de la diferencia entre eventos consecutivos. Para equipos, se detecta la columna de fecha, se agrupan los eventos por equipo (si la columna existe) y se calcula el TTF_dias como diferencia entre fechas consecutivas. En el modo general, se valida la columna Tiempo_dias, se eliminan valores no numéricos, no positivos o insuficientes, y se verifica que exista variabilidad en los datos. El resultado de esta etapa es un DataFrame con una columna Tiempo_dias lista para el análisis estadístico, que además se exporta como archivo intermedio de TTF para trazabilidad.

5) Flujo principal de cálculo: La función `run_full_pipeline_with_file` coordina todo el proceso: carga y validación del archivo, selección del modo de preprocesamiento, generación de la serie de TTF, integración con R para el ajuste de distribuciones, cálculo de la confiabilidad y exportación final de resultados. En paralelo, la función `run_expon_mtbf_only` permite un flujo alternativo en el cual el usuario ingresa directamente un valor de MTBF y se asume una distribución exponencial.

Finalmente, se implementó una interfaz mínima de usuario que se adapta al entorno: en Google Colab mediante widgets interactivos (`ipywidgets`) y en entornos de consola mediante preguntas tipo `input()`, manteniendo el mismo flujo lógico de decisión.

Por otro lado, Para ejecutar pruebas de ajuste de distribuciones y cálculos de confiabilidad avanzados se integró Python con R utilizando la librería `rpy2`. Esta integración permite aprovechar paquetes estadísticos especializados de R (como `EWGoF, fitdistrplus, MASS,`

`survival, actuar, evd y VGAM`) directamente desde el script de Python, manteniendo al usuario en un único flujo de trabajo.

En primer lugar, el script verifica la disponibilidad de `rpy2` y, en caso de no estar instalado, lo incorpora dinámicamente mediante `subprocess.check_call`. Posteriormente, se activan los conversores `pandas2ri` y `numpy2ri` para mapear de forma transparente estructuras de datos de Python (`numpy y pandas`) a objetos de R y viceversa. También se silencian los mensajes de consola de R para evitar ruido en la salida estándar.

Una vez establecida la conexión, se define en R la función `test_distributions(tiempos)`, la cual recibe la serie de tiempos entre fallos (TTF) como vector numérico, ajusta un conjunto amplio de distribuciones candidates (Exponencial, Normal, Weibull, Lognormal, Log-logística, Gumbel, Burr, Gamma, Frechet y Birnbaum–Saunders) utilizando `fitdistrplus, MASS, survival` y otros paquetes especializados, y además calcula para cada distribución un estadístico de bondad de ajuste (principalmente pruebas de Kolmogórov–Smirnov o Anderson–Darling vía `EWGoF::WEDF.test`) y retorna el p-value asociado junto con los parámetros estimados de la distribución.

Desde Python, se invoca esta función mediante `ro.r['test_distributions']`, y los resultados se procesan en un diccionario que asocia cada distribución con su p-value y sus parámetros. Se define un umbral de significancia `ALPHA = 0.05` y se selecciona automáticamente la distribución que presenta el mayor p-value, siempre que este sea mayor o igual que el umbral. En caso contrario, el script informa que ninguna distribución se ajusta adecuadamente a los datos y detiene el flujo, registrando el error correspondiente.

Adicionalmente, en R se define la función `calculate_reliability(dist, params, t)`, que implementa la función de confiabilidad $R(t)$ para cada distribución seleccionada, a partir de sus parámetros estimados. Esta función se llama desde Python para evaluar la confiabilidad en una malla de tiempos discretos, lo que permite generar la curva $R(t)$ de forma numérica y exportarla posteriormente.

Con este enfoque híbrido, el script combina la flexibilidad de Python en el manejo de datos con la robustez de los paquetes estadísticos de R, logrando un flujo automatizado de ajuste de distribuciones y cálculo de confiabilidad basado en pruebas de bondad de ajuste formales.

A su vez, el diseño del script contempló la generación de salidas claras y reutilizables tanto para el análisis técnico como para la construcción de dashboards en herramientas de visualización (por ejemplo, Power BI).

Una vez seleccionada la mejor distribución, el script calcula y muestra en consola estadísticas básicas de la serie de TTF: media, desviación estándar, mínimo y máximo (en días). Además, imprime los parámetros estimados de la distribución seleccionada (por ejemplo, β y η para Weibull, λ para Exponencial, μ y σ para Normal, etc.), junto con el p-value asociado a la prueba de bondad de ajuste.

Luego de esto, La función `exportar_confiabilidad_excel` encapsula la generación del archivo final de confiabilidad. A partir del vector de tiempos (en días) y de los valores de $R(t)$ calculados en R, se construye un DataFrame con las columnas: Horas, que es una conversión de los días a horas mediante un factor promedio (`HORAS_POR_DIA`), Días, que es un horizonte temporal en días, Confiabilidad (%), el cual es el valor de confiabilidad expresado en porcentaje, redondeado a tres decimales y acotado al rango [0, 100] y Confiabilidad (decimal) que es el valor de confiabilidad en formato decimal, redondeado a seis decimales y acotado al rango [0, 1].

Este DataFrame se exporta a un archivo Excel con el nombre Confiabilidad_<base>_<Distribución>.xlsx, donde <base> corresponde al nombre del archivo de entrada y <Distribución> a la distribución seleccionada (por ejemplo, Weibull, Gamma, etc.). En el flujo alternativo basado únicamente en MTBF, se genera un archivo análogo con sufijo Exponencial_MTBF, lo que permite comparar escenarios con o sin datos históricos suficientes.

C. Interacción con el usuario: códigos de errores con try/except

La interacción con el usuario se diseñó bajo el principio de "fallar de forma controlada", de manera que cualquier problema en la carga de datos, procesamiento o integración con R se traduzca en mensajes claros y accionables. Para ello, se implementó un sistema centralizado de manejo de errores basado en la combinación de bloques `try/except` y códigos de error estandarizados.

La función `add_error(code, msg, detail=None)` registra cada incidente en la lista global `ERRORS` mediante una tupla (código, mensaje, detalle) y, al mismo tiempo, muestra en pantalla un mensaje con un icono y el identificador del error. Adicionalmente, el parámetro opcional `detail` permite incluir información técnica más específica (como trazas de excepción o detalles del intento de lectura) que facilitan el diagnóstico.

Al finalizar cada bloque crítico del flujo, se invoca la función `print_error_summary()`. Esta función revisa la lista `ERRORS` y, si existen registros, imprime un resumen estructurado con todos los códigos y mensajes acumulados durante la ejecución. De esta forma, el usuario recibe una visión global de qué falló, en qué etapa y por qué.

Los códigos de error se organizaron por rangos temáticos, lo que facilita su interpretación:

- 1xx-2xx: problemas de entrada y preprocesamiento (archivo no cargado, formato incorrecto, columnas faltantes, fechas inválidas, etc.).
- 3xx: problemas de ajuste estadístico (por ejemplo, cuando ninguna de las distribuciones evaluadas supera el nivel de significancia).
- 4xx: errores relacionados con la integración con R y el cálculo de confiabilidad (`rpy2` no disponible, fallos al ejecutar funciones de R, parámetros inválidos, etc.).
- 5xx: fallos al exportar archivos Excel.
- 6xx: errores de interacción en la carga de archivos (por ejemplo, múltiples CSV detectados, archivos .zip, tamaños excesivos, cancelación por parte del usuario).

Cada sección del código está protegida con bloques `try/except`, de manera que, ante una excepción, el script captura el error, lo registra mediante `add_error` y detiene el flujo de forma controlada devolviendo un diccionario de estado. Esto permite, además, que la capa de interfaz (widgets en Colab o inputs en consola) pueda reaccionar adecuadamente, ya sea solicitando información adicional (como un valor de MTBF cuando hay menos de tres TTF) o informando al usuario que no es posible continuar con el análisis.

En conjunto, este mecanismo de manejo de errores convierte al script en una herramienta más robusta y amigable, que guía al usuario en la corrección de problemas de origen de datos y minimiza la probabilidad de resultados silenciosamente incorrectos.

D. Diseño del dashboard de confiabilidad en Power BI

Para garantizar la actualización continua de la información y facilitar la integración con Power BI, se estableció una conexión directa con las tablas de Google Sheets, donde se centraliza el registro de datos de confiabilidad. El proceso de conexión se realizó mediante el conector nativo de Power BI para Google Sheets, el cual permite importar rangos definidos o tablas completas directamente desde la nube. Esta metodología evita la manipulación manual de archivos y asegura que cualquier modificación realizada en las hojas de cálculo, ya sea la adición de nuevos equipos,

actualizaciones en la criticidad o ajustes en las confiabilidades calculadas se refleje automáticamente en los tableros de control.

Antes de la importación, las tablas fueron estructuradas siguiendo un modelo relacional estandarizado, compuesto por tablas de hechos y tablas de dimensiones. Las primeras contienen los valores numéricos y registros históricos de confiabilidad por día, sistema y componente, mientras que las segundas almacenan la información contextual de cada entidad, como líneas de producción, equipos, sistemas y repuestos.

Adicionalmente, se implementaron rutinas de validación automática desde Google Apps Script, con el fin de transformar y depurar la información antes de su carga al modelo. Estas rutinas garantizan la coherencia de los formatos, la correcta asignación de códigos SAP y la homogeneidad de las fechas, evitando errores comunes en los procesos de actualización. En el Anexo B se puede observar el código completo de App Script.

1) Tablas de hechos y dimensiones

Con el fin de estructurar adecuadamente el análisis de confiabilidad y garantizar la integridad de los cálculos realizados, se diseñó un modelo de datos basado en el enfoque estrella (star schema), ampliamente utilizado en entornos de analítica avanzada y sistemas de gestión de mantenimiento. Este modelo diferencia claramente las tablas de hechos, donde se concentran los eventos y mediciones numéricas, de las tablas de dimensiones, que proporcionan el contexto necesario para interpretar dichos valores.

La tabla de Hechos de Confiabilidad almacena la confiabilidad diaria calculada para cada equipo, sistema y componente. En ella se registran las variables clave del análisis: fecha, identificador del equipo (EquipoID), sistema o componente asociado, factor de falla y confiabilidad obtenida. Esta tabla constituye la base sobre la cual se construyen las curvas temporales, los cálculos de tendencias y los análisis comparativos entre jerarquías de mantenimiento. En la TABLA I se muestra un fragmento de la tabla de hechos de confiabilidad.

TABLA I

TABLA DE HECHOS DE CONFIABILIDAD

Fecha	EquipoID	EquipoSistemaID	ComponenteID	Factor ID	Nivel	Confiabilidad
1/06/2025	391	5001			Sistema	0,99
1/06/2025	391	5002			Sistema	0,99
1/06/2025	391	5003	90001	1	Componente	1,00
1/06/2025	391	5004	90002	2	Componente	1,00

Fecha	EquipoID	EquipoSistemaID	ComponenteID	Factor ID	Nivel	Confiabilidad
1/06/2025	391	5005			Sistema	0,99
1/06/2025	391	5006			Sistema	0,99
1/06/2025	391	5007			Sistema	0,99
1/06/2025	391	5008			Sistema	0,99
…	…	…	…	…	…	…

Nota: Elaboración propia

Complementariamente, se incluyó una segunda tabla de hechos de averías, la cual contiene la información proveniente de las órdenes de trabajo y avisos SAP. Allí se registran datos como el repuesto utilizado, sistema afectado, factor de falla reportado por el técnico, tiempos de avería y criticidad del evento. Esta tabla brinda un soporte narrativo y causal al análisis probabilístico, permitiendo correlacionar las fallas reales con los patrones de confiabilidad estimados. En la TABLA II es posible visualizar una parte de la tabla de hechos de averías.

TABLA II

TABLA DE HECHOS DE AVERÍAS

Fecha	Aviso	Orden	EquipoSistemaID	ComponenteID	FactorID	…
18/06/2025	1202840603	108117210	5003	90001	1	…
4/06/2025	1202832316	108084724	5003	90001	3	…
20/06/2025	1202841513	108121889	5003	90001	2	…
12/06/2025	1202837534	108104374	5017	90005	3	…
17/06/2025	1202839638	108114527	5042			…
20/06/2025	1202841643	108121789	5034	90010	1	…
23/06/2025	1202842543	108126134	5033		1	…
21/06/2025	1202842249	108123987	5049	90012	2	…
…	…	…	…	…	…	…

Nota: Elaboración propia

Por su parte, las tablas de dimensiones proporcionan el contexto necesario para interpretar cada registro. Las tablas de dimensiones usadas fueron:

- **Dimensión de líneas**: En esta tabla aparecen todas las líneas de producción de la planta y se puede consultar en la TABLA III.

TABLA III

TABLA DE DIMENSIÓN DE LÍNEA

LineaID	Linea
10	Línea #1
20	Línea #2

LineaID	Linea
30	Línea #3
40	Línea #4
…	…

Nota: Elaboración propia

- **Dimensión de equipos**: Esta tabla consolida los atributos técnicos relevantes, como el código SAP, la denominación, la línea de producción y la criticidad del activo. Se puede ver en la TABLA IV un fragmento de esta tabla.

TABLA IV

TABLA DE DIMENSIÓN DE EQUIPOS

EquipoID	SAP_Equipo	Denominacion_Equipo	Criticidad_Equipo	LineaID	PCC
101	20000003	EMBUTIDORA	A	10	No
102	20000013	BOMBA	A	20	No
103	20000020	BOOSTER	A	50	Sí
104	20000021	VÁLVULA	A	50	No
…	…	…	…	…	…

Nota: Elaboración propia

- **Dimensión Sistema-Categoría**: Esta tabla se encarga de recopilar los nombres de los sistemas de los equipos. En la TABLA V se puede observar algunos sistemas de los equipos.

TABLA V

TABLA DE DIMENSIÓN DE SISTEMAS

SistemaCatID	SISTEMA
1	SEGURIDAD
2	TRANSPORTE
3	SELLADO
4	ALIMENTACIÓN
…	…

Nota: Elaboración propia

- **Dimensión Equipo-Sistema**: Esta tabla sirve para relacionar los equipos con los sistemas que lo conforman, relacionando el EquipoID con el SistemaCatID. Se puede considerar como una tabla intermedia que sirve como puente para luego relacionar los componentes que hacen parte de cada sistema. En la TABLA VI se puede consultar esta información.

TABLA VI

TABLA DE DIMENSIÓN DE EQUIPO-SISTEMA

EquipoSistemaID	EquipoID	SistemaCatID
5053	125	1
5054	125	2
5055	125	10
5056	125	11
…	…	…

Nota: Elaboración propia

- **Dimensión Componente**: Se implementó para identificar los repuestos físicos sujetos a análisis, vinculándolos tanto a su sistema como a sus códigos SAP correspondientes. Se puede visualizar en la TABLA VII es posible visualizar esta tabla de dimensión.

TABLA VII

TABLA DE DIMENSIÓN DE COMPONENTE

ComponenteID	SAP_Componente	Componente	SistemaCatID
90023	6165501	Empaque	3
90024	6009814	Marco de corte	7
90025	6086083	Engranaje planetario	1
90026	6085958	Resorte	16
…	…	…	…

Nota: Elaboración propia

- **Dimensión Factor de Falla**: Esta tabla almacena los factores de falla que originaron las averías, contemplando el FactorID, el código del factor y una descripción. Estos factores de falla se referencian en la TABLA VIII.

TABLA VIII

TABLA DE DIMENSIÓN DE FACTOR DE FALLA

FactorID	CodigoFactor	Descripcion
1	EH	ERROR HUMANO - OPERACIÓN
2	DN	DETERIORO NATURAL - FIN DE VIDA ÚTIL
3	DF	DETERIORO FORZADO
4	CO	MATERIAL EMPAQUE FUERA ESPECIFICACION
…	…	…

Nota: Elaboración propia

2) Tabla Dimensión Calendario

Esta tabla facilita el análisis temporal, habilitando filtraciones jerárquicas por día, mes o año, además de permitir cálculos como acumulados, comparativos y tendencias intermensuales. En la TABLA IX se puede ver el contenido de esta dimensión.

TABLA IX

TABLA DE DIMENSIÓN CALENDARIO

Fecha	Año	Mes	Día	Semana del año	Nombre del mes	Año-Mes
1/06/2025	2025	6	1	23	Junio	2025-06
2/06/2025	2025	6	2	23	Junio	2025-06
3/06/2025	2025	6	3	23	Junio	2025-06
4/06/2025	2025	6	4	23	Junio	2025-06
…	…	…	…	…	…	…

Nota: Elaboración propia

3) Tabla Dimensión Medidas

Esta tabla agrupa un conjunto de indicadores calculados en DAX y su propósito es ofrecer un nivel adicional de control sin sobrecargar la tabla de hechos principal, manteniendo así un modelo más limpio, modular y fácilmente escalable. anterior, lo cual permite evaluar la evolución reciente del equipo o sistema analizado. Estas medidas funcionan como puntos de referencia clave para identificar mejoras o deterioros en el desempeño operacional.

Asimismo, se incluyen medidas que determinan el fin de mes del período en curso y del mes anterior, por lo que estas referencias temporales son particularmente útiles para la creación de visualizaciones dinámicas. Un componente especialmente relevante dentro de esta tabla es la medida “Hay Proyección”, la cual verifica si existen datos suficientes para extender la curva de confiabilidad hacia el mes siguiente. Esto permite habilitar visualizaciones proyectadas únicamente cuando la información disponible lo justifica, evitando interpretaciones erróneas o extrapolaciones no deseadas. En otras palabras, el modelo solo muestra la proyección futura cuando existe una base confiable para hacerlo.

Por otro lado, se incluye la medida “Mostrar Solo Mes Anterior”, que actúa como un filtro de control para visualizar únicamente la confiabilidad correspondiente al período precedente. Esta medida resulta útil en análisis retrospectivos o cuando se desea eliminar ruido visual y centrarse exclusivamente en el comportamiento consolidado del mes anterior.

La articulación entre estas tablas, mediante relaciones uno-a-muchos, permite construir un modelo robusto, escalable y completamente alineado con las mejores prácticas de análisis de confiabilidad. De esta manera, los indicadores generados representan no solo el comportamiento estadístico de los activos, sino también su correspondencia con los eventos operativos reales, así como se puede ver en la FIG, donde aparecen las relaciones del modelo completo del Power BI.

Fig. 5. Relaciones del modelo del Power BI.

Nota: Elaboración propia

Finalmente, en la Fig. 6 se presenta un diagrama de flujo que sintetiza de manera estructurada todas las etapas del proceso de actualización del dashboard de confiabilidad en Power BI, evidenciando cómo cada fase, desde la extracción y depuración de datos hasta el cálculo de indicadores y la visualización de resultados contribuye de forma integrada a la estimación de la confiabilidad final de los equipos.

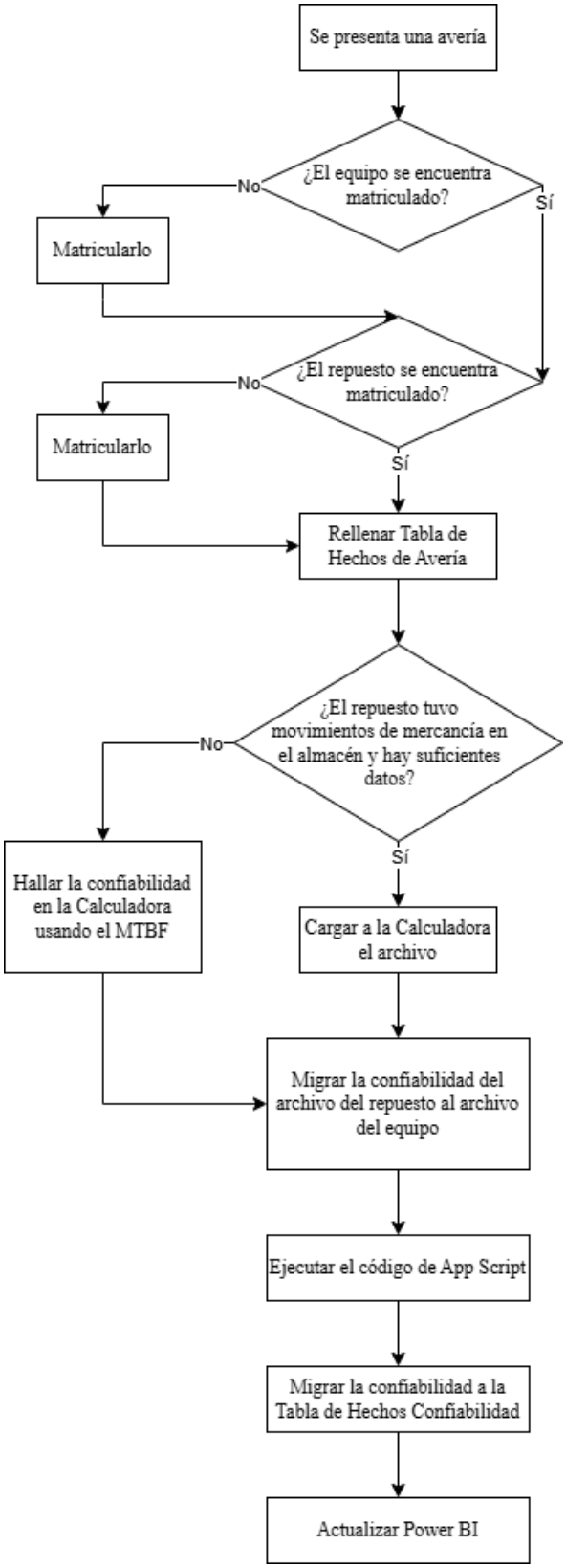

Fig. 6. Diagrama de flujo de las etapas de actualización del Power BI de confiabilidad.

Nota: Elaboración propia.

V. ANÁLISIS DE RESULTADOS

A. Resultados del procesamiento y análisis automatizado de datos

Durante el análisis de confiabilidad realizado, se evaluó un conjunto de aproximadamente 39 repuestos, a cada uno de los cuales se le asoció una distribución estadística representativa de su comportamiento de falla. La proporción de las distribuciones ajustadas permitió caracterizar el patrón de comportamiento de los repuestos en operación. En la Fig. 7 se presenta la distribución global de las funciones ajustadas, la cual resume el comportamiento estadístico observado en el conjunto de repuestos analizados.

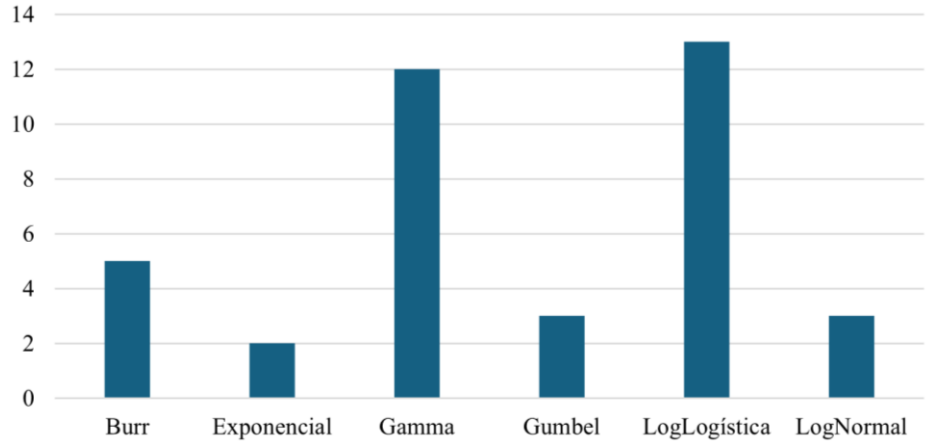

Fig. 7. Diagrama de barras que muestra la distribución global de las funciones que mejor se adaptaron a los repuestos.

Nota: Elaboración propia

El gráfico evidencia una predominancia clara de determinadas distribuciones estadísticas en el ajuste de los tiempos hasta la falla. En particular, se observa que las distribuciones Log-Logística y Gamma concentran la mayor cantidad de ajustes, con valores significativamente superiores al resto. Este resultado sugiere que una proporción importante de los repuestos presenta comportamientos de falla no aleatorios, asociados a procesos de desgaste progresivo, degradación acumulativa o mecanismos de daño que se intensifican con el tiempo, características típicas de estos modelos.

La presencia de la distribución Burr en un número moderado de casos refuerza esta interpretación, ya que su flexibilidad permite modelar colas pesadas y comportamientos intermedios entre fallas tempranas y desgaste avanzado. Esto indica que ciertos repuestos exhiben alta variabilidad en sus tiempos de falla, posiblemente influenciada por condiciones operativas, carga o entorno.

Por el contrario, la distribución exponencial aparece en un número reducido de repuestos, lo que sugiere que solo una fracción menor de los componentes presenta un comportamiento de fallas puramente aleatorio, con tasa de falla aproximadamente constante, posiblemente asociados a componentes eléctricos y electrónicos.

Finalmente, las distribuciones Gumbel y Lognormal, aunque menos frecuentes, indican la existencia de repuestos sensibles a eventos extremos o a procesos multiplicativos de degradación, respectivamente. En conjunto, la distribución observada en el gráfico evidencia que el comportamiento dominante de los repuestos en la planta está gobernado por mecanismos de desgaste y envejecimiento.

B. Resultados de la integración y visualización en Power BI

El dashboard se encuentra estructurado en tres zonas funcionales claramente diferenciadas, tal como se observa en la Fig. 8. En primer lugar, la zona de filtros, destacada en color azul, en segundo lugar, la zona de confiabilidad, identificada en color amarillo y finalmente, la zona de averías, resaltada en color rojo.

La exploración de resultados se ve significativamente fortalecida por la capacidad del dashboard para segmentar la información mediante filtros avanzados, particularmente por fecha específica de análisis y por factor de falla. Esta funcionalidad permite evaluar la confiabilidad bajo escenarios hipotéticos, como la exclusión de fallas asociadas a error humano, deterioro forzado o deterioro natural, lo que impacta de forma dinámica la estructura del árbol jerárquico de análisis y facilita la comprensión del comportamiento del sistema bajo distintas condiciones operativas.

En la sección de confiabilidad, se presenta un árbol jerárquico que organiza la taxonomía de línea, equipo, sistema y componente, mostrando la confiabilidad mediante barras horizontales acompañadas de un formato condicional tipo semáforo. Este esquema visual permite una interpretación rápida del desempeño: verde para niveles altos de confiabilidad, amarillo para valores intermedios y rojo para confiabilidades bajas, orientando de manera inmediata la priorización de acciones de mantenimiento.

Adicionalmente, el dashboard incluye una tabla de confiabilidad diaria, que despliega el valor de la confiabilidad desde el inicio hasta el cierre del mes, evitando la necesidad de aplicar filtros repetitivos para analizar días específicos. Esta tabla utiliza el mismo formato condicional del árbol jerárquico, garantizando consistencia visual y facilitando la detección de variaciones significativas a lo largo del periodo analizado. De forma complementaria, se incorpora una curva

de confiabilidad en función del tiempo, en la que se comparan la confiabilidad correspondiente al mes anterior y la confiabilidad estimada para el mes actual, esta última actuando como una proyección del comportamiento esperado en ausencia de intervenciones de mantenimiento.

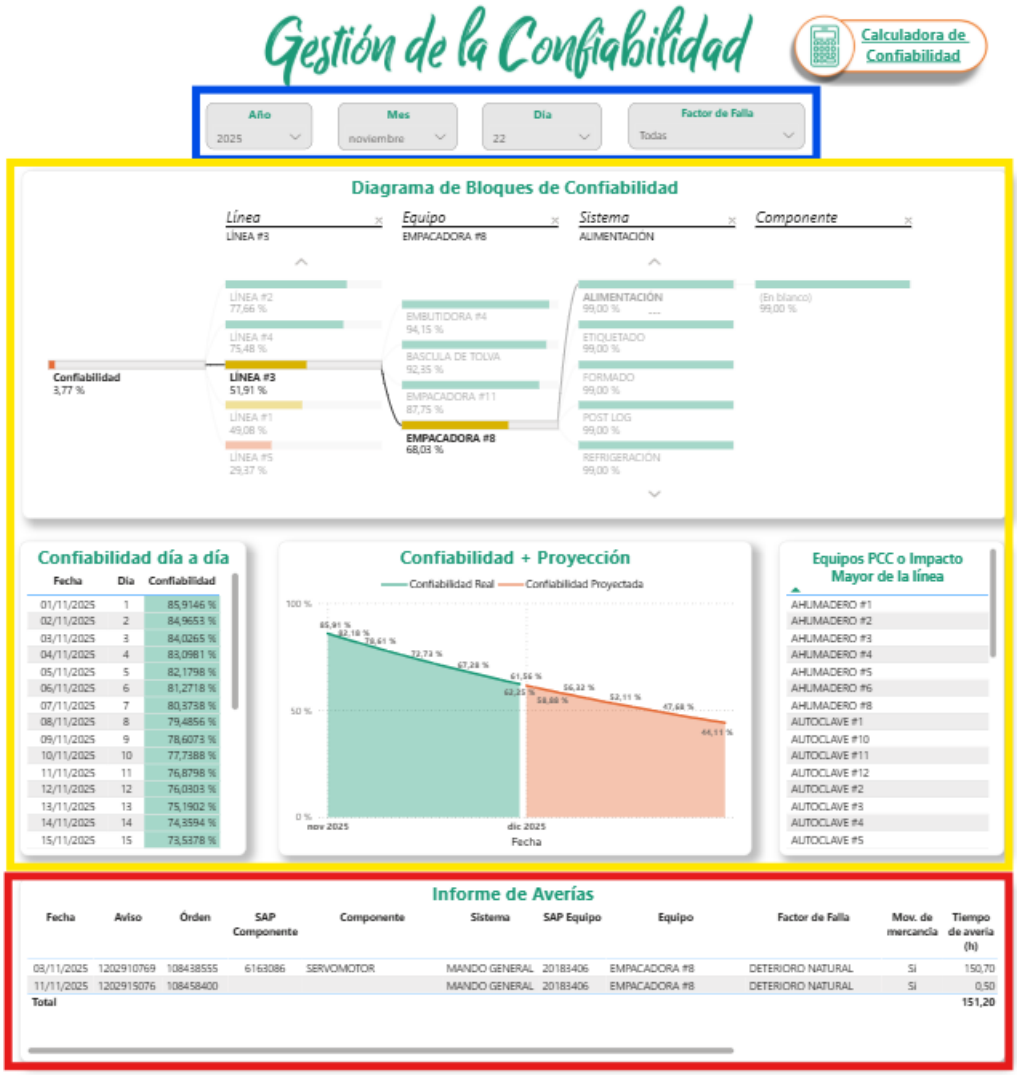

Fig. 8. Secciones del dashboard de confiabilidad.

Nota: Elaboración propia

Asimismo, se dispone de una tabla específica para los equipos PCC (Puntos Críticos de Control) y aquellos de mayor impacto dentro de cada línea, lo que garantiza una identificación permanente de los activos prioritarios, en los cuales no es aceptable operar con niveles bajos de confiabilidad debido a su criticidad para la operación. En la parte inferior del dashboard se presenta una tabla detallada de las averías del mes, que consolida información clave como la fecha del

evento, el código SAP del aviso y de la orden de trabajo, el componente y sistema afectados, el equipo involucrado y la existencia de movimientos de mercancía, entre otros campos relevantes.

Finalmente, en la parte superior del dashboard se dispone de un botón de acceso directo que permite redirigir al código desarrollado en Python y R para el cálculo de la confiabilidad. Esta funcionalidad garantiza que el usuario tenga acceso permanente a la herramienta de análisis, facilitando la trazabilidad del proceso, la validación de los resultados y la consulta directa del algoritmo que sustenta los indicadores presentados en el tablero.

En comparación con los reportes estáticos generados directamente desde SAP, el enfoque implementado mediante el dashboard de confiabilidad ofrece ventajas sustanciales en términos de análisis, flexibilidad y soporte a la toma de decisiones. Mientras que los reportes tradicionales presentan información histórica de forma fragmentada y con escasa capacidad de exploración, el dashboard permite integrar múltiples fuentes de datos, actualizar indicadores de manera dinámica y analizar escenarios específicos mediante filtros e interactividad. Esto facilita la identificación temprana de tendencias, la priorización de activos críticos y la evaluación del impacto de distintas estrategias de mantenimiento, transformando los datos operativos en información analítica accionable.

C. Validación de resultados con el área de mantenimiento

Los resultados obtenidos a partir del análisis de confiabilidad y su posterior visualización en el dashboard fueron socializados con analistas e ingenieros del área de mantenimiento, quienes aportaron su conocimiento operativo y experiencia en planta para evaluar la pertinencia de los indicadores generados. La retroalimentación recibida permitió validar la claridad de las visualizaciones, la utilidad de los filtros y la coherencia de los resultados estadísticos, destacándose el valor del tablero como herramienta de apoyo para el análisis técnico y la priorización de acciones de mantenimiento.

En términos generales, se evidenció una alta correspondencia entre los resultados obtenidos mediante el análisis estadístico y la experiencia empírica del personal de mantenimiento. Los equipos y repuestos identificados con baja confiabilidad o comportamientos de desgaste acelerado coincidieron con aquellos reconocidos en planta como activos problemáticos o críticos, lo que reforzó la credibilidad del modelo a pesar de las limitantes respecto a los datos históricos de SAP.

Como resultado del proceso de validación, se realizaron ajustes tanto en el tratamiento de los datos como en la presentación de los resultados. Entre las principales mejoras se incluyeron la

refinación de criterios de filtrado, la exclusión de ciertos eventos no representativos de fallas reales y la optimización de algunas visualizaciones para facilitar su interpretación. Estos ajustes permitieron incrementar la robustez del análisis y mejorar la usabilidad del dashboard, asegurando que la información presentada fuera técnicamente consistente y operativamente relevante.

El nivel de aceptación de la herramienta por parte del equipo de mantenimiento fue alto, destacándose su capacidad para centralizar información, reducir tiempos de análisis y facilitar la toma de decisiones basada en datos. Los usuarios valoraron especialmente la interactividad del dashboard, la posibilidad de explorar escenarios específicos y la transparencia del proceso de cálculo, al contar con acceso directo al código que sustenta los indicadores. En conjunto, la herramienta fue percibida como un aporte significativo a la gestión del mantenimiento, con potencial para ser integrada de manera regular en los procesos de análisis y planificación.

VI. CONCLUSIONES Y RECOMENDACIONES

A. Conclusiones

El desarrollo del presente proyecto permitió cumplir de manera integral el objetivo general, al consolidar una aplicación computacional capaz de calcular la confiabilidad de repuestos y equipos a partir de los TTF, mediante el uso de análisis estadístico y pruebas de bondad de ajuste, integrando los resultados en un entorno de visualización interactivo en Power BI. La metodología implementada demostró que es posible transformar los datos históricos de mantenimiento provenientes de SAP en información analítica estructurada, útil para la toma de decisiones en la gestión del mantenimiento.

En relación con los objetivos específicos, se logró diseñar e implementar una herramienta en Python con integración en R que automatiza el procesamiento de datos, la validación de TTF y la selección objetiva del modelo estadístico más representativo para cada repuesto, a partir de criterios cuantitativos como pruebas de bondad de ajuste y comparación de modelos. El análisis de aproximadamente 39 repuestos evidenció que la mayoría presenta comportamientos de falla asociados a mecanismos de desgaste y envejecimiento, reflejados en la predominancia de distribuciones como Log-Logística, Gamma y Burr, mientras que los comportamientos puramente aleatorios, modelados mediante la distribución exponencial, resultaron ser minoritarios. Este hallazgo confirma que, en el contexto analizado, las estrategias de mantenimiento deben orientarse principalmente hacia enfoques preventivos y predictivos, más que reactivos.

Asimismo, la generación de tablas de confiabilidad y curvas de supervivencia a partir de las distribuciones seleccionadas permitió caracterizar de forma probabilística el desempeño de los repuestos, facilitando la identificación de tendencias de riesgo a lo largo del tiempo. La exportación de estos resultados en un archivo estructurado de Excel demostró ser un mecanismo efectivo para garantizar la interoperabilidad entre el análisis estadístico y la capa de visualización, asegurando la trazabilidad del proceso y la reproducibilidad de los cálculos.

Por otra parte, el dashboard interactivo desarrollado en Power BI cumplió satisfactoriamente su propósito como herramienta de visualización y análisis, al permitir explorar dinámicamente los resultados, segmentar la información por fecha y factor de falla, y analizar la confiabilidad a diferentes niveles jerárquicos (línea, equipo, sistema y componente). La inclusión de visualizaciones como árboles jerárquicos, curvas de confiabilidad, tablas diarias y listados de averías, junto con indicadores visuales tipo semáforo, facilitó la interpretación rápida del estado de los activos y la priorización de equipos críticos, superando las limitaciones propias de los reportes estáticos de SAP.

Finalmente, la validación de los resultados con el área de mantenimiento evidenció una alta coherencia entre los hallazgos del análisis estadístico y la experiencia operativa en planta, lo que reforzó la confiabilidad del modelo y la pertinencia de la herramienta desarrollada. El alto nivel de aceptación por parte de los usuarios, sumado a los ajustes realizados tras la retroalimentación recibida, confirma que la solución propuesta constituye un aporte significativo a la gestión del mantenimiento, con potencial para ser integrada de forma regular en los procesos de análisis, planificación y mejora continua basados en datos.

B. Recomendaciones

A partir de los resultados obtenidos y del desarrollo del proyecto, se identificaron diversas oportunidades de mejora que podrían abordarse en trabajos futuros, con el fin de incrementar la precisión del análisis de confiabilidad, fortalecer la calidad de los datos y ampliar el alcance de la herramienta desarrollada, estas recomendaciones son:

- Mejorar la recopilación de información relacionada con los eventos de falla, mediante la implementación o habilitación de una transacción específica en SAP que permita identificar de forma directa el momento exacto en el que un componente entra en falla y genera una avería. Actualmente, el uso de los movimientos de mercancía como aproximación indirecta introduce un margen de incertidumbre en el cálculo de los TTF, por lo tanto, contar con datos explícitos de falla

permitiría reducir el error asociado al modelo y mejorar significativamente la calidad del análisis estadístico.

- Fortalecer la documentación de las averías en SAP, ya que se evidenció que, en múltiples casos, el componente registrado no correspondía al elemento realmente fallado, o bien quedaba indeterminado dentro de los catálogos. Esta situación limita la trazabilidad del análisis y afecta la confiabilidad de los resultados. Una correcta identificación del componente fallado es esencial para mejorar los estudios de confiabilidad, análisis de modos de falla y toma de decisiones de mantenimiento.
- De manera complementaria, se identificó que no todas las averías están siendo documentadas de forma consistente, lo que genera vacíos de información y reduce la representatividad de los datos históricos. Por ello, se recomienda reforzar los procedimientos operativos y la disciplina en el registro de eventos de falla, asegurando que cada intervención correctiva quede debidamente documentada en el sistema, independientemente de su severidad o duración.
- Estructurar los catálogos de factores de falla bajo el enfoque del RCM, incorporando una taxonomía que contemple de manera explícita el componente fallado, el estado o condición de falla (adjetivo) y la causa raíz. Esta estructuración permitiría elevar el nivel de análisis de fallas, facilitar la identificación de patrones recurrentes y mejorar la efectividad de las acciones correctivas y preventivas, además de potenciar el uso futuro de la herramienta para análisis más avanzados, como estudios de criticidad, análisis causa-raíz y mantenimiento predictivo.

REFERENCIAS

[1] P. E. Silva Ardila y J. C. Orrego Barrera, *Confiabilidad en la Práctica. Lo que un Gerente de Confiabilidad debe saber*, 1a ed. Barranquilla / Medellín - Colombia, 2014.

[2] F. S. Nowlan y H. F. Heap, *Reliability-Centered Maintenance. Report No. ADA066579*. San Francisco, California: U.S. Department of Defense, 1978.

[3] J. Moubray, *RCM II: Reliability Centered Maintenance*, 2a ed. Oxford, Inglaterra: Elsevier, 1999.

[4] A. M. Smith y G. R. Hinchcliffe, *RCM - GATEWAY TO WORLD CLASS MAINTENANCE*. Oxford, Inglaterra: Elsevier, 2004.

[5] S. Woo, "Modern Definitions in Reliability Engineering", 2020, pp. 53–99. doi: 10.1007/978-981-13-7236-0_3.

[6] P. D. T. O'Connor y A. Kleyner, *Practical Reliability Engineering*, 5a ed. John Wiley & Sons, Ltd, 2012.

[7] J. F. Lawless, *Statistical Models and Methods for Lifetime Data*, 2a ed. Hoboken, New Jersey: John Wiley & Sons, Ltd, 2003.

[8] W. Meeker y L. A. Escobar, *Statistical Methods for Reliability Data*, 1a ed. Nueva York, Estados Unidos: John Wiley & Sons, Ltd, 1998.

[9] E. A. Elsayed, *Reliability Engineering*, 3a ed. John Wiley & Sons, Ltd, 2021.

[10] O. J. Obulezi *et al.*, "Type-I Heavy-Tailed Burr XII Distribution with Applications to Quality Control, Skewed Reliability Engineering Systems and Lifetime Data", *CMES - Comput. Model. Eng. Sci.*, vol. 144, núm. 3, pp. 2991–3027, sep. 2025, doi: 10.32604/cmes.2025.069553.

[11] L. Lazzari, *Engineering Tools for Corrosion - Design and Diagnosis (EFC 68)*. Elsevier, 2017. [En línea]. Disponible en: https://app.knovel.com/hotlink/toc/id:kpETCDD00L/engineering-tools-corrosion/engineering-tools-corrosion

[12] S. Thijssen Pierre, Hadjiloucas, *State Estimation in Chemometrics - The Kalman Filter and Beyond (2nd Edition)*. Elsevier, 2020. [En línea]. Disponible en: https://app.knovel.com/hotlink/toc/id:kpSECTKF02/state-estimation-in-chemometrics/state-estimation-in-chemometrics

[13] G. Andrieu, *Electromagnetic Reverberation Chambers - Recent Advances and Innovative Applications*. Institution of Engineering and Technology (The IET), 2021. [En línea]. Disponible en: https://app.knovel.com/hotlink/toc/id:kpERCRAIA2/electromagnetic-reverberation/electromagnetic-reverberation

[14] E. Calixto, *Gas and Oil Reliability Engineering - Modeling and Analysis (2nd Edition)*. Elsevier, 2016. [En línea]. Disponible en: https://app.knovel.com/hotlink/toc/id:kpGOREMAE1/gas-oil-reliability-engineering/gas-oil-reliability-engineering

[15] A. Gelman, "P values and statistical practice", *Epidemiology (Cambridge, Mass.)*, vol. 24, núm. 1, pp. 69–72, enero de 2013.

[16] S. K. Deshpande, S. Ghosh, T. D. Nguyen, y T. Broderick, "Are you using test log-likelihood correctly?", *Cornell University*, vol. 4, pp. 1–23, 2024.

[17] H. Pham, *Springer Handbook of Engineering Statistics (2nd Edition)*. Springer Nature, 2023. [En línea]. Disponible en: https://app.knovel.com/hotlink/toc/id:kpSHES0001/springer-handbook-engineering/springer-handbook-engineering

[18] A. Kolokolov y M. Zelensky, *Data Visualization with Microsoft Power BI: How to Design Savvy Dashboards*, 1a ed. Sebastopol, Ucrania: O'Reilly Media, Inc., 2024.
[19] C. N. Knaflic, *Storytelling with data*, 1a ed. Canadá: Wiley, 2015.

ANEXOS

Anexo A. Código de Python y R para el cálculo de la confiabilidad

```
# Elaborado por: Adrián Cogollo Sáez

import warnings, sys, subprocess, glob, re
warnings.filterwarnings("ignore", category=DeprecationWarning)
# warnings para errores | sys para outputs de terminal | subprocess para instalar
librerías
# glob para detectar archivos | re para limpiar caracteres

# ---------- print seguro en Colab/Notebook ----------
def _print_nb(*args, sep=" ", end="\n"):
    try:
        sys.stdout.write(sep.join(str(a) for a in args) + end)
        sys.stdout.flush()
    except Exception:
        pass
print = _print_nb

import pandas as pd
import numpy as np

ALPHA = 0.05
HORAS_POR_DIA = 520.0 / 30.0  # 17.3333...
ERRORS = []  # lista para el resumen de errores

# ---------- Manejo de errores ----------
def add_error(code, msg, detail=None):
    ERRORS.append((code, msg, detail))
    print(f"❌ ERROR {code}: {msg}")
    if detail:
        print(f"   Detalle: {detail}")

def print_error_summary():
    if not ERRORS:
        print("☑ Sin errores reportados.")
        return
    print("\n====== RESUMEN DE ERRORES ======")
    for code, msg, detail in ERRORS:
        if detail:
            print(f"• {code} - {msg} | {detail}")
        else:
            print(f"• {code} - {msg}")
    print("================================\n")
```

```
# ---------- Utilidades texto/fechas ----------
def normalize_header_text(s: str) -> str:
    import unicodedata
    s = str(s).replace("\ufeff", "").replace("\ufffe", "")
    return unicodedata.normalize("NFKC", s).strip()

def canonical_key(name: str) -> str:
    return re.sub(r"[^a-zA-Z0-9]+", "", str(name)).lower()

def find_column(df, candidates):
    norm_map = {canonical_key(c): c for c in df.columns}
    for cand in candidates:
        key = canonical_key(cand)
        if key in norm_map:
            return norm_map[key]
    return None

def parse_fechas_robusto(serie):
    s = serie.astype(str).str.strip()
    dt = pd.to_datetime(s, errors="coerce", dayfirst=True)
    mask = dt.isna()
    if mask.any():
        dt2 = pd.to_datetime(s[mask], format="%d/%m/%Y", errors="coerce")
        dt.loc[mask & dt2.notna()] = dt2; mask = dt.isna()
    if mask.any():
        dt3 = pd.to_datetime(s[mask], format="%d.%m.%Y", errors="coerce")
        dt.loc[mask & dt3.notna()] = dt3; mask = dt.isna()
    if mask.any():
        dt4 = pd.to_datetime(s[mask], format="%Y-%m-%d", errors="coerce")
        dt.loc[mask & dt4.notna()] = dt4; mask = dt.isna()
    if mask.any():
        dt5 = pd.to_datetime(s[mask], errors="coerce")
        dt.loc[mask & dt5.notna()] = dt5
    return dt

# ---------- Lector robusto CSV ----------
def read_csv_robusto(path):
    intentos_err = []
    primary_attempts = [
        {"encoding": "utf-8-sig", "sep": ",", "engine": "python", "quotechar": '"',
"skipinitialspace": True},
        {"encoding": "utf-8",     "sep": ",", "engine": "python", "quotechar": '"',
"skipinitialspace": True},
    ]
    for kw in primary_attempts:
        try:
            try:
                df = pd.read_csv(path, on_bad_lines="skip", **kw)
            except TypeError:
```

```
                kw2 = kw.copy(); kw2.pop("quotechar", None);
kw2.pop("skipinitialspace", None)
                df = pd.read_csv(path, **kw2)
            df.columns = [normalize_header_text(c) for c in df.columns]
            if df.shape[1] == 1:
                try:
                    df2 = pd.read_csv(path, on_bad_lines="skip",
encoding=kw["encoding"], sep=";", engine="python")
                except TypeError:
                    df2 = pd.read_csv(path, encoding=kw["encoding"], sep=";",
engine="python")
                df2.columns = [normalize_header_text(c) for c in df2.columns]
                return df2, {"encoding": kw["encoding"], "sep": ";", "engine":
"python"}
            return df, {"encoding": kw["encoding"], "sep": ",", "engine": "python"}
        except Exception as e:
            intentos_err.append(f"[primary {kw}] {e}")

    encodings = ["utf-8", "utf-8-sig", "latin-1", "cp1252", "utf-16", "utf-16le", "utf-
16be"]
    combos = [
        {"sep": ";", "engine": None},
        {"sep": ",", "engine": None},
        {"sep": ";", "engine": "python"},
        {"sep": ",", "engine": "python"},
        {"sep": None, "engine": "python"},
    ]
    for enc in encodings:
        for cfg in combos:
            try:
                kwargs = {"encoding": enc}
                if cfg["sep"] is not None: kwargs["sep"] = cfg["sep"]
                if cfg["engine"] is not None: kwargs["engine"] = cfg["engine"]
                try:
                    df = pd.read_csv(path, on_bad_lines="skip", **kwargs)
                except TypeError:
                    df = pd.read_csv(path, **kwargs)
                df.columns = [normalize_header_text(c) for c in df.columns]
                sep_info = cfg["sep"] if cfg["sep"] is not None else "infer"
                eng_info = cfg["engine"] or "c"
                return df, {"encoding": enc, "sep": sep_info, "engine": eng_info}
            except Exception as e:
                intentos_err.append(f"[enc={enc}, cfg={cfg}] {e}")

    return None, intentos_err

# ---------- Preproceso REPUESTOS (SAP) ----------
def preprocess_sap_to_ttf(df_raw, col_mov="CMv", col_fecha="Fe.contab.",
col_clase="Clase_orden", col_orden="Orden"):
    df_raw.columns = [normalize_header_text(c) for c in df_raw.columns]
```

```
    need_map = {
        "CMv":          ["CMv", "Cl.mov.", "Clmov", "Cl_mov", "Cl mov", "Movimiento"],
        "Fe.contab.":   ["Fe.contab.", "Fe.contab", "Fecha contab.", "Fecha",
"Fechacontab"],
        "Clase_orden": ["Clase_orden", "Clase orden", "Clase.de.orden", "Tipo_orden",
"Clase Orden"],
        "Orden":        ["Orden", "N° orden", "Numero de orden", "N° orden", "Orden
ID"],
    }
    resolved = {}
    for std, cands in need_map.items():
        got = find_column(df_raw, cands)
        if got is None:
            add_error(201, f"Falta la columna requerida '{std}'.")
            return None, None
        resolved[std] = got

    df = df_raw.copy()
    try:
        df["_FECHA_DT"] = parse_fechas_robusto(df[resolved["Fe.contab."]])
        n_invalid = df["_FECHA_DT"].isna().sum()
        if n_invalid > 0:
            add_error(202, f"Se encontraron {n_invalid} filas con fecha inválida en
'{resolved['Fe.contab.']}'. Serán descartadas.")
        df = df.dropna(subset=["_FECHA_DT"])
    except Exception as e:
        add_error(202, "Error al convertir fechas (Repuestos).", str(e)); return None,
None

    df = df.sort_values("_FECHA_DT").reset_index(drop=True)

    df_removed_out = pd.DataFrame(columns=["CMv", "Orden", "Fe.contab."])
    try:
        mov = df[resolved["CMv"]].astype(str).str.strip()
        idx_262 = df.index[mov == "262"].tolist()
        drop_idx = set()
        for i in idx_262:
            drop_idx.add(i)
            prev_i = i-1 if i-1 >= 0 else None
            next_i = i+1 if i+1 < len(df) else None
            def dt(idx): return df.loc[idx, "_FECHA_DT"] if idx is not None else None
            prev_dt, next_dt = dt(prev_i), dt(next_i)
            if prev_dt is not None and next_dt is not None:
                dprev = abs((df.loc[i, "_FECHA_DT"] - prev_dt).days)
                dnext = abs((next_dt - df.loc[i, "_FECHA_DT"]).days)
                neighbor = prev_i if dprev <= dnext else next_i
            elif prev_dt is not None:
                neighbor = prev_i
            elif next_dt is not None:
                neighbor = next_i
```

```
            else:
                neighbor = None
            if neighbor is not None: drop_idx.add(neighbor)

        if drop_idx:
            drop_idx = sorted(list(drop_idx))
            removed = df.loc[drop_idx, [resolved["CMv"], resolved["Orden"],
"_FECHA_DT"]].copy()
            removed["Fe.contab."] = removed["_FECHA_DT"].dt.strftime("%d/%m/%Y")
            df_removed_out = removed.rename(columns={resolved["CMv"]: "CMv",
resolved["Orden"]: "Orden"})[["CMv", "Orden", "Fe.contab."]].reset_index(drop=True)
            print(f"ℹ Eliminando {len(drop_idx)} filas por regla CMv=262
(devoluciones + vecino más cercano).")
            df = df.drop(index=drop_idx).reset_index(drop=True)
        else:
            print("ℹ No se encontraron filas con CMv = 262.")
    except Exception as e:
        add_error(202, "Error al eliminar CMv=262 y su vecino.", str(e)); return None,
None

    try:
        df["_FECHA_SIG"] = df["_FECHA_DT"].shift(-1)
        df["Tiempo_dias"] = (df["_FECHA_SIG"] - df["_FECHA_DT"]).dt.days
        n_nan = df["Tiempo_dias"].isna().sum()
        if n_nan > 0: print(f"ℹ Se descartan {n_nan} filas sin siguiente fecha
(últimas).")
        df = df.dropna(subset=["Tiempo_dias"])
        df = df[df["Tiempo_dias"] > 0]
    except Exception as e:
        add_error(202, "Error al calcular Tiempo_dias (Repuestos).", str(e)); return
None, None

    try:
        keep_classes = {"ZPRO", "ZAVE"}
        clase = df[resolved["Clase_orden"]].astype(str).str.strip().str.upper()
        df = df[clase.isin(keep_classes)].copy()
        print(f"ℹ Filtrado por Clase_orden en {keep_classes}. Quedan {len(df)}
filas.")
    except Exception as e:
        add_error(202, "Error al filtrar por Clase_orden (Repuestos).", str(e)); return
None, None

    try:
        df_out = pd.DataFrame({
            "Orden": df[resolved["Orden"]].astype(str),
            "Fe.contab.": df["_FECHA_DT"].dt.strftime("%d/%m/%Y"),
            "Tiempo_dias": df["Tiempo_dias"].astype(int),
            "Clase_orden": df[resolved["Clase_orden"]].astype(str)
        }).reset_index(drop=True)
        return df_out, df_removed_out
```

```
    except Exception as e:
        add_error(202, "Error al construir salida TTF SAP (Repuestos).", str(e));
return None, None

# ---------- Preproceso EQUIPOS ----------
def preprocess_equipos_to_ttf(df_raw, col_fecha_candidates=None,
col_equipo_candidates=None):
    """
    Detecta la columna de fecha para equipos: prioriza 'Fe.entrada', luego
'Inic.prog.'.
    Soporta dd.mm.aaaa (y otros formatos comunes) con parseo robusto.
    """
    df_raw.columns = [normalize_header_text(c) for c in df_raw.columns]

    if col_fecha_candidates is None:
        col_fecha_candidates = [
            "Fe.entrada", "Fe.entrada.", "Fecha entrada", "Fecha Entrada",
            "Inic.prog.", "Inicio programado", "Inicio.programado", "Fecha de inicio"
        ]
    col_fecha = find_column(df_raw, col_fecha_candidates)
    if col_fecha is None:
        add_error(201, "Falta columna de fecha para equipos (p. ej., 'Fe.entrada' o
'Inic.prog.').")
        return None

    if col_equipo_candidates is None:
        col_equipo_candidates = ["Equipo", "Equipo_ID", "N° equipo", "Num_equipo",
"ID_Equipo", "ID Equipo", "Cod_Equipo", "Código equipo"]
    col_equipo = find_column(df_raw, col_equipo_candidates)

    df = df_raw.copy()
    try:
        df["_FECHA_DT"] = parse_fechas_robusto(df[col_fecha])
        n_invalid = df["_FECHA_DT"].isna().sum()
        if n_invalid > 0:
            add_error(202, f"Se encontraron {n_invalid} filas con fecha inválida en
'{col_fecha}'. Serán descartadas.")
        df = df.dropna(subset=["_FECHA_DT"])
    except Exception as e:
        add_error(202, f"Error al convertir fechas en '{col_fecha}'.", str(e)); return
None

    try:
        if col_equipo:
            df = df.sort_values([col_equipo, "_FECHA_DT"]).reset_index(drop=True)
            df["_FECHA_SIG"] = df.groupby(col_equipo)["_FECHA_DT"].shift(-1)
        else:
            df = df.sort_values("_FECHA_DT").reset_index(drop=True)
            df["_FECHA_SIG"] = df["_FECHA_DT"].shift(-1)
        df["TTF_dias"] = (df["_FECHA_SIG"] - df["_FECHA_DT"]).dt.days
```

```
        n_nan = df["TTF_dias"].isna().sum()
        if n_nan > 0: print(f"ⓘ Se descartan {n_nan} filas sin siguiente fecha
(último evento).")
        df = df.dropna(subset=["TTF_dias"]); df = df[df["TTF_dias"] > 0]
    except Exception as e:
        add_error(202, "Error al calcular TTF (Equipos).", str(e)); return None

    try:
        fecha_col_out = col_fecha
        if col_equipo:
            df_out = pd.DataFrame({
                "Equipo": df[col_equipo].astype(str),
                fecha_col_out: df["_FECHA_DT"].dt.strftime("%d/%m/%Y"),
                "TTF_dias": df["TTF_dias"].astype(int)
            }).reset_index(drop=True)
            print(f"ⓘ TTF por equipo usando '{col_equipo}' y fecha '{fecha_col_out}'.
Total: {len(df_out)}.")
        else:
            df_out = pd.DataFrame({
                fecha_col_out: df["_FECHA_DT"].dt.strftime("%d/%m/%Y"),
                "TTF_dias": df["TTF_dias"].astype(int)
            }).reset_index(drop=True)
            print(f"ⓘ TTF global usando fecha '{fecha_col_out}'. Total:
{len(df_out)}.")
        return df_out
    except Exception as e:
        add_error(202, "Error al construir salida TTF (Equipos).", str(e)); return None

# ---------- Preproceso GENERAL (TTF directo) ----------
def preprocess_general_ttf(df_raw, col_ttf="Tiempo_dias"):
    df_raw.columns = [normalize_header_text(c) for c in df_raw.columns]
    if col_ttf not in df_raw.columns:
        add_error(201, f"Falta la columna requerida '{col_ttf}'."); return None
    s = df_raw[col_ttf]
    try:
        has_comma = s.dropna().astype(str).str.contains(r",").any()
        comma_decimal_pattern =
s.dropna().astype(str).str.match(r"^\s*\d+,\d+\s*$").any()
        if has_comma or comma_decimal_pattern:
            add_error(206, "Se detectaron valores con coma decimal en 'Tiempo_dias'.
Reemplaza comas por puntos.")
    except Exception:
        pass
    serie_num = pd.to_numeric(s, errors="coerce").dropna()
    serie_num = serie_num[serie_num > 0]
    if len(serie_num) < 3:
        add_error(204, "Se requieren al menos 3 valores positivos en 'Tiempo_dias'.");
return None
    if np.ptp(serie_num.values) == 0:
        add_error(205, "Todos los valores de 'Tiempo_dias' son iguales."); return None
```

```
    df_out = pd.DataFrame({"Tiempo_dias": serie_num.reset_index(drop=True)})
    print(f"ℹ️ TTF (General) validado. Total filas: {len(df_out)}.")
    return df_out

# ---------- Exportar confiabilidad ----------
def exportar_confiabilidad_excel(base, dias, conf_array, sufijo):
    try:
        horas = (dias.astype(float) * HORAS_POR_DIA).astype(int)
        df_confiabilidad = pd.DataFrame({
            "Horas": horas,
            "Días": dias.astype(int),
            "Confiabilidad (%)": (conf_array * 100),
            "Confiabilidad (decimal)": conf_array
        })
        df_confiabilidad["Confiabilidad (%)"] = df_confiabilidad["Confiabilidad
(%)"].round(3).clip(0, 100)
        df_confiabilidad["Confiabilidad (decimal)"] = df_confiabilidad["Confiabilidad
(decimal)"].round(6).clip(0, 1)
        nombre_archivo = f"Confiabilidad_{base}_{sufijo}.xlsx"
        df_confiabilidad.to_excel(nombre_archivo, index=False)
        print(f"\n☑️ Se ha generado el archivo de Excel: {nombre_archivo}")
    except Exception as e:
        add_error(501, "No se puede exportar el xlsx.", str(e))

# ---------- Camino MTBF sin archivo ----------
def run_expon_mtbf_only(mtbf_value, base_name="SinArchivo"):
    try:
        max_tiempos = max(mtbf_value * 1.2, 2)
        dias = np.arange(1, int(max_tiempos) + 1)
        if len(dias) > 10000: dias = np.arange(1, 10001)
        lamb = 1.0 / float(mtbf_value)
        conf_array = np.exp(-lamb * dias.astype(float))
        exportar_confiabilidad_excel(base_name, dias, conf_array, "Exponencial_MTBF")
    except Exception as e:
        add_error(401, "Fallo en cálculo/exportación MTBF.", str(e))
    finally:
        print_error_summary()

# ---------- Pipeline principal con archivo ----------
def run_full_pipeline_with_file(file_name=None, IS_COLAB=False,
return_on_need_mtbf=False):
    ERRORS.clear()
    base = "SinNombre"

    # 1) Carga
    if file_name is None:
        if IS_COLAB:
            try:
                from google.colab import files
                uploaded = files.upload()
```

```
                if uploaded is None or len(uploaded) == 0:
                    add_error(604, "Usuario cancela la carga o no adjunta archivo.")
                    print_error_summary(); return {"status":"error"}
                if len(uploaded) > 1: add_error(601, "Se cargaron múltiples archivos.
Sube solo un archivo CSV.")
                names = list(uploaded.keys())
                if len(set(names)) < len(names): add_error(602, "Se detectaron archivos
con el mismo nombre.")
                file_name = names[0]
                if file_name.lower().endswith(".zip"): add_error(605, "Se subió un
.zip. Sube un CSV/TXT.")
                size_bytes = len(uploaded[file_name])
                if size_bytes > 50 * 1024 * 1024: add_error(603, "Archivo demasiado
grande (>50 MB).")
                if not (file_name.lower().endswith(".csv") or
file_name.lower().endswith(".txt")):
                    add_error(102, "Archivo no parece CSV (.csv/.txt).")
                else:
                    with open(file_name, "wb") as f: f.write(uploaded[file_name])
            except Exception as e:
                add_error(101, "No se cargó ningún archivo.", str(e))
        else:
            try:
                csvs = glob.glob("*.csv") + glob.glob("*.CSV") + glob.glob("*.txt") +
glob.glob("*.TXT")
                if len(csvs) == 0:
                    add_error(101, "No se cargó ningún archivo. Deja 1 CSV/TXT en la
carpeta.")
                elif len(csvs) > 1:
                    add_error(601, "Hay múltiples CSV/TXT en la carpeta. Deja solo
uno.")
                else:
                    file_name = csvs[0]
            except Exception as e:
                add_error(101, "No se pudo localizar un CSV en la carpeta.", str(e))

    if any(code in {101,102,603,604,605,601,602} for code,_,_ in ERRORS):
        print_error_summary(); return {"status":"error"}

    # 2) Lectura
    df = None; meta_csv = None
    if file_name is None:
        add_error(101, "No se obtuvo un nombre de archivo válido tras la carga.")
        print_error_summary(); return {"status":"error"}
    else:
        df, meta_or_errs = read_csv_robusto(file_name)
        if df is None:
            detalle = " | ".join(meta_or_errs[:8])
            if len(meta_or_errs) > 8: detalle += f" | ... ({len(meta_or_errs)-8}
intentos más)"
```

```
            add_error(102, "Archivo no se pudo leer con encodings/formatos comunes.",
detalle)
            print_error_summary(); return {"status":"error"}
        else:
            meta_csv = meta_or_errs
            print(f"ℹ️ CSV leído con encoding={meta_csv['encoding']},
sep='{meta_csv['sep']}', engine={meta_csv['engine']}.")
            base = file_name.rsplit(".", 1)[0]

    df.columns = [normalize_header_text(c) for c in df.columns]

    # 3) Detección de fuente (estricta)
    SOURCE_CHOICE = "auto"
    SAP_REP_COLS = ["CMv", "Fe.contab.", "Clase_orden", "Orden"]

    def has_equipos_cols(df_):
        return ("Fe.entrada" in df_.columns) or ("Inic.prog." in df_.columns)

    if has_equipos_cols(df):
        SOURCE_CHOICE = "equipos"
    elif all(c in df.columns for c in SAP_REP_COLS):
        SOURCE_CHOICE = "repuestos"
    elif "Tiempo_dias" in df.columns:
        SOURCE_CHOICE = "general"
    else:
        SOURCE_CHOICE = "auto"  # no match

    # 4) Preprocesos según fuente (con salidas tempranas si faltan columnas)
    df_ttf = None; df_removed = None

    if SOURCE_CHOICE == "repuestos":
        if not all(c in df.columns for c in SAP_REP_COLS):
            for col in [c for c in SAP_REP_COLS if c not in df.columns]:
                add_error(201, f"Falta la columna requerida '{col}' del layout SAP
(Repuestos).")
            print_error_summary(); return {"status":"error"}  # <- detener
        print("🔧 [Repuestos] Depuración SAP y cálculo TTF ...")
        df_ttf, df_removed = preprocess_sap_to_ttf(df, "CMv", "Fe.contab.",
"Clase_orden", "Orden")
        if df_ttf is None:
            print_error_summary(); return {"status":"error"}
        try:
            ttf_name = f"TTF_SAP_{base}.xlsx"
            with pd.ExcelWriter(ttf_name, engine="openpyxl") as writer:
                df_ttf.to_excel(writer, index=False, sheet_name="TTF")
                startrow = len(df_ttf) + 2
                if df_removed is not None and not df_removed.empty:
                    pd.DataFrame({"": ["Filas eliminadas por CMv=262 (+ vecino más
cercano)"]}) \
```

```
                        .to_excel(writer, index=False, header=False, startrow=startrow,
sheet_name="TTF")
                    startrow += 1
                    df_removed_out = df_removed[["CMv", "Orden", "Fe.contab."]].copy()
                    df_removed_out.to_excel(writer, index=False, startrow=startrow,
sheet_name="TTF")
            print(f"☑ Exportado TTF (Repuestos) con eliminadas: {ttf_name}")
        except Exception as e:
            add_error(501, "No se pudo exportar el TTF depurado (.xlsx).", str(e))

    elif SOURCE_CHOICE == "equipos":
        fecha_col = "Fe.entrada" if "Fe.entrada" in df.columns else ("Inic.prog." if
"Inic.prog." in df.columns else None)
        if fecha_col is None:
            add_error(201, "Falta columna de fecha para equipos (p. ej., 'Fe.entrada' o
'Inic.prog.').")
            print_error_summary(); return {"status":"error"}  # <- detener
        print(f"🔧 [Equipos] Construyendo TTF usando fecha '{fecha_col}' ...")
        df_ttf_equip = preprocess_equipos_to_ttf(df, col_fecha_candidates=[fecha_col])
        if df_ttf_equip is None:
            print_error_summary(); return {"status":"error"}
        df_ttf = df_ttf_equip.rename(columns={"TTF_dias": "Tiempo_dias"}) if "TTF_dias"
in df_ttf_equip.columns else df_ttf_equip
        try:
            ttf_name = f"TTF_EQUIPOS_{base}.xlsx"
            with pd.ExcelWriter(ttf_name, engine="openpyxl") as writer:
                df_ttf.to_excel(writer, index=False, sheet_name="TTF")
            print(f"☑ Exportado TTF (Equipos): {ttf_name}")
        except Exception as e:
            add_error(501, "No se pudo exportar el TTF (Equipos) (.xlsx).", str(e))

    elif SOURCE_CHOICE == "general":
        if "Tiempo_dias" not in df.columns:
            add_error(201, "Falta la columna requerida 'Tiempo_dias' para el modo
General (TTF directo).")
            print_error_summary(); return {"status":"error"}  # <- detener
        print("🔧 [General] Usando 'Tiempo_dias' ...")
        df_ttf_gen = preprocess_general_ttf(df, "Tiempo_dias")
        if df_ttf_gen is None:
            print_error_summary(); return {"status":"error"}
        df_ttf = df_ttf_gen.copy()
        try:
            ttf_name = f"TTF_GENERAL_{base}.xlsx"
            with pd.ExcelWriter(ttf_name, engine="openpyxl") as writer:
                df_ttf.to_excel(writer, index=False, sheet_name="TTF")
            print(f"☑ Exportado TTF (General): {ttf_name}")
        except Exception as e:
            add_error(501, "No se pudo exportar el TTF (General) (.xlsx).", str(e))

    else:
```

```
        # AUTO: si no calza con ninguno, reportar error 201 y detener (NO hay fallback
a columnas numéricas)
        add_error(
            201,
            "El archivo no coincide con ningún layout esperado.",
            "Repuestos: CMv, Fe.contab., Clase_orden, Orden | "
            "Equipos: Fe.entrada o Inic.prog. | General: Tiempo_dias"
        )
        print_error_summary(); return {"status":"error"}

    # 5) Selección de tiempos (sin 'auto-legacy')
    tiempos_serie = None
    insufficient_ttf_any = False

    if df_ttf is not None and "Tiempo_dias" in df_ttf.columns:
        try:
            serie_num = pd.to_numeric(df_ttf["Tiempo_dias"], errors="coerce").dropna()
            if len(serie_num) < 3:
                print("ⓘ TTF insuficientes (<3). Se solicitará MTBF y se asumirá
distribución exponencial.")
                insufficient_ttf_any = True
                tiempos_serie = serie_num[serie_num > 0]
            else:
                tiempos_serie = serie_num[serie_num > 0]
                if tiempos_serie.empty:
                    add_error(204, "TTF resultantes no contienen valores positivos.")
        except Exception as e:
            add_error(202, "Error al procesar 'Tiempo_dias' en TTF.", str(e))

    # Si se necesita MTBF y caller desea manejar UI => devolver señal
    if (tiempos_serie is None or len(tiempos_serie) < 3 or insufficient_ttf_any):
        if return_on_need_mtbf:
            print_error_summary()
            return {"status":"need_mtbf", "base_name": base}
        add_error(204, "Se requieren <3 TTF y no hay UI para MTBF en este modo.")
        print_error_summary(); return {"status":"error"}

    # 6) Pruebas con R
    r_ready = True
    try:
        try:
            import rpy2.robjects as ro
            from rpy2.robjects import pandas2ri, numpy2ri
        except ImportError:
            subprocess.check_call([sys.executable, "-m", "pip", "install", "rpy2"])
            import rpy2.robjects as ro
            from rpy2.robjects import pandas2ri, numpy2ri
        pandas2ri.activate(); numpy2ri.activate()
        import rpy2.rinterface_lib.callbacks
        rpy2.rinterface_lib.callbacks.consolewrite_print = lambda x: None
```

```
        def instalar_y_cargar(pkg):
            ro.r(f"""
            if (!require('{pkg}', quietly=TRUE)) {{
                install.packages('{pkg}', repos='https://cloud.r-project.org')
                library({pkg})
            }} else {{
                suppressPackageStartupMessages(library({pkg}))
            }}
            """)
        for pkg in ["EWGoF", "MASS", "fitdistrplus", "survival", "actuar", "evd",
"VGAM"]:
            instalar_y_cargar(pkg)

        ro.r("""
        test_distributions <- function(tiempos) {
            tiempos <- as.numeric(tiempos)
            tiempos <- tiempos[is.finite(tiempos) & tiempos > 0]
            resultados <- list()

            # Exponencial
            tryCatch({
                lambda <- 1/mean(tiempos)
                ks_exp <- suppressWarnings(stats::ks.test(tiempos, "pexp",
rate=lambda))
                resultados$Exponencial <- list(pvalue=as.numeric(ks_exp$p.value),
lambda=as.numeric(lambda))
            }, error=function(e){ resultados$Exponencial <- list(pvalue=NA_real_) })

            # Normal
            tryCatch({
                if (length(tiempos) <= 5000) {
                    sw <- stats::shapiro.test(tiempos)
                    mu <- mean(tiempos); sigma <- stats::sd(tiempos)
                    resultados$Normal <- list(pvalue=as.numeric(sw$p.value),
mu=as.numeric(mu), sigma=as.numeric(sigma))
                } else {
                    mu <- mean(tiempos); sigma <- stats::sd(tiempos)
                    ks_norm <- suppressWarnings(stats::ks.test(tiempos, "pnorm",
mean=mu, sd=sigma))
                    resultados$Normal <- list(pvalue=as.numeric(ks_norm$p.value),
mu=as.numeric(mu), sigma=as.numeric(sigma))
                }
            }, error=function(e){ resultados$Normal <- list(pvalue=NA_real_) })

            # Weibull
            tryCatch({
                shape <- NA_real_; scale <- NA_real_; ok <- FALSE
                try({ fd <- fitdistrplus::fitdist(tiempos, "weibull")
                      shape <- as.numeric(fd$estimate[["shape"]])
```

```
                    scale <- as.numeric(fd$estimate[["scale"]]); ok <- TRUE },
silent=TRUE)
                if (!ok) try({ md <- MASS::fitdistr(tiempos, densfun="weibull")
                               shape <- as.numeric(md$estimate[["shape"]])
                               scale <- as.numeric(md$estimate[{"scale"}]); ok <- TRUE
}, silent=TRUE)
                if (!ok) {
                    sr <- survival::survreg(survival::Surv(tiempos) ~ 1,
dist="weibull")
                    shape <- 1/as.numeric(sr$scale)
                    scale <- exp(as.numeric(stats::coef(sr))); ok <- TRUE
                }
                pval <- NA_real_
                try({ ad <- EWGoF::WEDF.test(tiempos, type="AD", funEstimate="MLE",
paramKL=2, nsim=200)
                      pval <- as.numeric(ad$p.value) }, silent=TRUE)
                if (!is.finite(pval)) {
                    ks_w <- suppressWarnings(stats::ks.test(tiempos, "pweibull",
shape=shape, scale=scale))
                    pval <- as.numeric(ks_w$p.value)
                }
                resultados$Weibull <- list(pvalue=pval, beta=shape, eta=scale)
            }, error=function(e){ resultados$Weibull <- list(pvalue=NA_real_) })

            # Lognormal
            tryCatch({
                fd <- fitdistrplus::fitdist(tiempos, "lnorm")
                meanlog <- as.numeric(fd$estimate[["meanlog"]])
                sdlog   <- as.numeric(fd$estimate[["sdlog"]])
                ks <- suppressWarnings(stats::ks.test(tiempos, "plnorm",
meanlog=meanlog, sdlog=sdlog))
                resultados$Lognormal <- list(pvalue=as.numeric(ks$p.value),
meanlog=meanlog, sdlog=sdlog)
            }, error=function(e){ resultados$Lognormal <- list(pvalue=NA_real_) })

            # Log-logística
            tryCatch({
                fd <- fitdistrplus::fitdist(tiempos, "llogis")
                shape <- as.numeric(fd$estimate[["shape"]])
                scale <- as.numeric(fd$estimate[["scale"]])
                ks <- suppressWarnings(stats::ks.test(tiempos, actuar::pllogis,
shape=shape, scale=scale))
                resultados$LogLogistica <- list(pvalue=as.numeric(ks$p.value),
shape=shape, scale=scale)
            }, error=function(e){ resultados$LogLogistica <- list(pvalue=NA_real_) })

            # Gumbel
            tryCatch({
                fd <- fitdistrplus::fitdist(tiempos, "gumbel",
start=list(loc=mean(tiempos), scale=stats::sd(tiempos)))
```

```
                loc   <- as.numeric(fd$estimate[["loc"]])
                scale <- as.numeric(fd$estimate[["scale"]])
                ks <- suppressWarnings(stats::ks.test(tiempos, evd::pgumbel, loc=loc,
scale=scale))
                resultados$Gumbel <- list(pvalue=as.numeric(ks$p.value), loc=loc,
scale=scale)
            }, error=function(e){ resultados$Gumbel <- list(pvalue=NA_real_) })

            # Burr
            tryCatch({
                fd <- fitdistrplus::fitdist(tiempos, "burr")
                shape1 <- as.numeric(fd$estimate[["shape1"]])
                shape2 <- as.numeric(fd$estimate[["shape2"]])
                scale  <- as.numeric(fd$estimate[["scale"]])
                ks <- suppressWarnings(stats::ks.test(tiempos, actuar::pburr,
shape1=shape1, shape2=shape2, scale=scale))
                resultados$Burr <- list(pvalue=as.numeric(ks$p.value), shape1=shape1,
shape2=shape2, scale=scale)
            }, error=function(e){ resultados$Burr <- list(pvalue=NA_real_) })

            # Gamma
            tryCatch({
                fd <- fitdistrplus::fitdist(tiempos, "gamma")
                shape <- as.numeric(fd$estimate[["shape"]])
                rate  <- as.numeric(fd$estimate[["rate"]])
                ks <- suppressWarnings(stats::ks.test(tiempos, "pgamma", shape=shape,
rate=rate))
                resultados$Gamma <- list(pvalue=as.numeric(ks$p.value), shape=shape,
rate=rate)
            }, error=function(e){ resultados$Gamma <- list(pvalue=NA_real_) })

            # Frechet
            tryCatch({
                fd <- fitdistrplus::fitdist(tiempos, "frechet")
                shape <- as.numeric(fd$estimate[["shape"]])
                scale <- as.numeric(fd$estimate[["scale"]])
                ks <- suppressWarnings(stats::ks.test(tiempos, VGAM::pfrechet,
shape=shape, scale=scale))
                resultados$Frechet <- list(pvalue=as.numeric(ks$p.value), shape=shape,
scale=scale)
            }, error=function(e){ resultados$Frechet <- list(pvalue=NA_real_) })

            # Birnbaum-Saunders
            tryCatch({
                fd <- fitdistrplus::fitdist(tiempos, "bisa")
                shape <- as.numeric(fd$estimate[["shape"]])
                scale <- as.numeric(fd$estimate[["scale"]])
                ks <- suppressWarnings(stats::ks.test(tiempos, VGAM::pbisa,
shape=shape, scale=scale))
```

```
                resultados$BirnbaumSaunders <- list(pvalue=as.numeric(ks$p.value),
shape=shape, scale=scale)
            }, error=function(e){ resultados$BirnbaumSaunders <- list(pvalue=NA_real_)
})

            return(resultados)
        }

        calculate_reliability <- function(dist, params, t) {
            if (dist == "Exponencial") {
                lambda <- params$lambda; return(exp(-lambda * t))
            } else if (dist == "Normal") {
                mu <- params$mu; sigma <- params$sigma; return(1 - pnorm(t, mu,
sd=sigma))
            } else if (dist == "Weibull") {
                beta <- params$beta; eta <- params$eta; return(exp(-(t/eta)^beta))
            } else if (dist == "Lognormal") {
                meanlog <- params$meanlog; sdlog <- params$sdlog; return(1 - plnorm(t,
meanlog=meanlog, sdlog=sdlog))
            } else if (dist == "LogLogistica") {
                shape <- params$shape; scale <- params$scale; return(1 -
actuar::pllogis(t, shape=shape, scale=scale))
            } else if (dist == "Gumbel") {
                loc <- params$loc; scale <- params$scale; return(1 - evd::pgumbel(t,
loc=loc, scale=scale))
            } else if (dist == "Burr") {
                shape1 <- params$shape1; shape2 <- params$shape2; scale <-
params$scale; return(1 - actuar::pburr(t, shape1=shape1, shape2=shape2, scale=scale))
            } else if (dist == "Gamma") {
                shape <- params$shape; rate <- params$rate; return(1 - pgamma(t,
shape=shape, rate=rate))
            } else if (dist == "Frechet") {
                shape <- params$shape; scale <- params$scale; return(1 -
VGAM::pfrechet(t, shape=shape, scale=scale))
            } else if (dist == "BirnbaumSaunders") {
                shape <- params$shape; scale <- params$scale; return(1 - VGAM::pbisa(t,
shape=shape, scale=scale))
            }
            return(NA_real_)
        }
        """)
    except Exception as e:
        r_ready = False
        add_error(401, "No se pudieron preparar las pruebas de ajuste (R/rpy2).",
str(e))

    if not r_ready:
        print_error_summary(); return {"status":"error"}

    # 7) Ejecutar pruebas R
```

```
    try:
        import rpy2.robjects as ro
        res = ro.r['test_distributions'](tiempos_serie.values)
    except Exception as e:
        add_error(401, "Fallo al ejecutar test_distributions en R.", str(e))
        print_error_summary(); return {"status":"error"}

    def r_get(d, key):
        try:
            val = d.get(key)
            try: return float(val[0])
            except Exception:
                try: return float(val)
                except Exception: return None
        except Exception:
            return None

    dist = None; params = None; pbest = None; resultados = {}; pvals = {}
    try:
        for nombre, vals in zip(res.names, list(res)):
            sub = {str(k): v for k, v in vals.items()}
            resultados[str(nombre)] = sub
        for k, sub in resultados.items():
            p = r_get(sub, "pvalue"); p = (-np.inf) if (p is None or not
np.isfinite(p)) else p
            pvals[k] = p
        all_fail = all((p < ALPHA) for p in pvals.values()) if len(pvals) > 0 else True
        if all_fail:
            add_error(301, f"Los datos no cumplen ninguna de las {len(pvals)}
distribuciones (p < {ALPHA}).")
            print_error_summary(); return {"status":"error"}
        dist, pbest = max(pvals.items(), key=lambda kv: kv[1])
        params = resultados.get(dist, {})
    except Exception as e:
        add_error(401, "Error procesando resultados de ajuste.", str(e));
print_error_summary(); return {"status":"error"}

    # 8) Estadísticos básicos
    try:
        media = tiempos_serie.mean(); sd = tiempos_serie.std(); mn =
tiempos_serie.min(); mx = tiempos_serie.max()
        print("📊 Estadísticas descriptivas:")
        print(f"   Media: {media:.4f} días")
        print(f"   Desviación Estándar: {sd:.4f} días")
        print(f"   Mínimo: {mn:.4f} días")
        print(f"   Máximo: {mx:.4f} días")
    except Exception:
        pass

    print("\n---"); print(f"☑ Mejor ajuste: {dist} (p = {pbest:.4f})")
```

```
    try:
        if dist == "Exponencial":
            lamb = r_get(params, "lambda")
            if lamb is not None: print(f"   lambda = {lamb:.6f}")
        elif dist == "Normal":
            mu = r_get(params, "mu"); sigma = r_get(params, "sigma")
            if (mu is not None) and (sigma is not None):
                print(f"   mu = {mu:.4f}"); print(f"   sigma = {sigma:.4f}")
        elif dist == "Weibull":
            beta = r_get(params, "beta"); eta = r_get(params, "eta")
            if (beta is not None) and (eta is not None):
                print(f"   beta = {beta:.4f}"); print(f"   eta = {eta:.4f}")
        elif dist == "Lognormal":
            meanlog = r_get(params, "meanlog"); sdlog = r_get(params, "sdlog")
            if (meanlog is not None) and (sdlog is not None):
                print(f"   meanlog = {meanlog:.4f}"); print(f"   sdlog = {sdlog:.4f}")
        elif dist == "LogLogistica":
            shape = r_get(params, "shape"); scale = r_get(params, "scale")
            if (shape is not None) and (scale is not None):
                print(f"   shape = {shape:.4f}"); print(f"   scale = {scale:.4f}")
        elif dist == "Gumbel":
            loc = r_get(params, "loc"); scale = r_get(params, "scale")
            if (loc is not None) and (scale is not None):
                print(f"   loc = {loc:.4f}"); print(f"   scale = {scale:.4f}")
        elif dist == "Burr":
            s1 = r_get(params, "shape1"); s2 = r_get(params, "shape2"); scale =
r_get(params, "scale")
            if (s1 is not None) and (s2 is not None) and (scale is not None):
                print(f"   shape1 = {s1:.4f}"); print(f"   shape2 = {s2:.4f}");
print(f"   scale = {scale:.4f}")
        elif dist == "Gamma":
            shape = r_get(params, "shape"); rate = r_get(params, "rate")
            if (shape is not None) and (rate is not None):
                print(f"   shape = {shape:.4f}"); print(f"   rate = {rate:.4f}")
        elif dist in ("Frechet", "BirnbaumSaunders"):
            shape = r_get(params, "shape"); scale = r_get(params, "scale")
            if (shape is not None) and (scale is not None):
                print(f"   shape = {shape:.4f}"); print(f"   scale = {scale:.4f}")
    except Exception:
        pass

    # 9) Confiabilidad y exportación
    try:
        r_calc_rel = ro.r['calculate_reliability']
    except Exception as e:
        add_error(401, "No se encontró calculate_reliability en R.", str(e))
        print_error_summary(); return {"status":"error"}

    try:
        max_tiempos = tiempos_serie.max()
```

```
        dias = np.arange(1, int(max_tiempos * 1.2) + 1)
        if len(dias) > 10000: dias = np.arange(1, 10001)
    except Exception as e:
        add_error(401, "No se puede calcular malla de tiempo.", str(e));
print_error_summary(); return {"status":"error"}

    confiabilidades = []; calc_fail_count = 0
    try:
        params_r = res.rx2(dist)
    except Exception as e:
        add_error(401, "No se pudieron obtener parámetros de la distribución.",
str(e)); print_error_summary(); return {"status":"error"}

    if params_r is None or dias.size == 0:
        add_error(401, "Parámetros o malla inválidos."); print_error_summary(); return
{"status":"error"}

    for d in dias:
        try:
            val = float(r_calc_rel(dist, params_r, float(d))[0])
            if not np.isfinite(val): raise ValueError("no finito")
            confiabilidades.append(val)
        except Exception:
            calc_fail_count += 1; confiabilidades.append(np.nan)
    if calc_fail_count > 0:
        add_error(401, f"No se pudo calcular la confiabilidad en {calc_fail_count}
instantes.")

    conf_array = np.asarray(confiabilidades, dtype=float)
    exportar_confiabilidad_excel(base, dias, conf_array, dist)
    print_error_summary()
    return {"status":"ok"}

# ==========================
# UI INICIAL: ¿Tienes archivo?
# ==========================
IS_COLAB = False
try:
    from google.colab import files  # noqa
    IS_COLAB = True
except Exception:
    IS_COLAB = False

if IS_COLAB:
    try:
        import ipywidgets as widgets
        from IPython.display import display, clear_output

        ERRORS.clear()
        out = widgets.Output()
```

```
        question = widgets.HTML("<b>¿Tienes un archivo .CSV con datos?</b>")
        choice = widgets.ToggleButtons(options=[("Sí", "si"), ("No", "no")],
value="si")

        # NO: MTBF directo
        mtbf_input_no = widgets.FloatText(description="MTBF (días):", value=0.0)
        btn_mtbf_no = widgets.Button(description="Calcular con Exponencial",
button_style="success")
        box_no = widgets.VBox([mtbf_input_no, btn_mtbf_no])

        # SÍ: subir archivo
        btn_file = widgets.Button(description="Cargar .CSV y analizar",
button_style="primary")
        box_si = widgets.VBox([widgets.HTML("Selecciona tu archivo <b>.CSV</b> para
procesar."), btn_file])

        # MTBF por <3 TTF tras archivo
        mtbf_input_from_file = widgets.FloatText(description="MTBF (días):", value=0.0)
        btn_mtbf_from_file = widgets.Button(description="Continuar con Exponencial",
button_style="success")
        box_mtbf_from_file = widgets.VBox([
            widgets.HTML("<b>TTF insuficientes.</b> Ingresa el MTBF (en días) para
asumir distribución exponencial:"),
            mtbf_input_from_file, btn_mtbf_from_file
        ])
        box_mtbf_from_file.layout.display = "none"
        base_for_mtbf = {"name": "SinArchivo"}

        container = widgets.VBox([question, choice, box_si, box_mtbf_from_file, out])

        def show_si():
            box_si.layout.display = None
            box_no.layout.display = "none"
            box_mtbf_from_file.layout.display = "none"

        def show_no():
            box_si.layout.display = "none"
            box_no.layout.display = None
            box_mtbf_from_file.layout.display = "none"

        def show_mtbf_from_file(base_name):
            base_for_mtbf["name"] = base_name or "SinArchivo"
            box_si.layout.display = "none"
            box_no.layout.display = "none"
            box_mtbf_from_file.layout.display = None

        def on_choice_change(change):
            if change["name"] == "value":
                with out:
```

```
                    clear_output(wait=True)
                if change["new"] == "si":
                    if container.children[2] is not box_si:
                        container.children = [question, choice, box_si,
box_mtbf_from_file, out]
                    show_si()
                else:
                    if container.children[2] is not box_no:
                        container.children = [question, choice, box_no,
box_mtbf_from_file, out]
                    show_no()

        def on_btn_file_clicked(b):
            with out:
                clear_output(wait=True)
                print("⏳ Abriendo selector de archivos...")
            res = run_full_pipeline_with_file(file_name=None, IS_COLAB=True,
return_on_need_mtbf=True)
            if isinstance(res, dict) and res.get("status") == "need_mtbf":
                show_mtbf_from_file(res.get("base_name", "SinArchivo"))

        def on_btn_mtbf_no_clicked(b):
            ERRORS.clear()
            with out:
                clear_output(wait=True)
                try:
                    val = float(mtbf_input_no.value)
                    if not np.isfinite(val) or val <= 0:
                        raise ValueError("MTBF debe ser > 0")
                    print(f"⏳ Calculando confiabilidad Exponencial con MTBF={val:.4f}
días ...")
                    run_expon_mtbf_only(val, base_name="SinArchivo")
                except Exception as e:
                    add_error(204, "MTBF inválido.", str(e))
                    print_error_summary()

        def on_btn_mtbf_from_file_clicked(b):
            ERRORS.clear()
            with out:
                clear_output(wait=True)
                try:
                    val = float(mtbf_input_from_file.value)
                    if not np.isfinite(val) or val <= 0:
                        raise ValueError("MTBF debe ser > 0")
                    base_name = base_for_mtbf.get("name", "SinArchivo")
                    print(f"⏳ Calculando confiabilidad Exponencial con MTBF={val:.4f}
días ...")
                    run_expon_mtbf_only(val, base_name=base_name)
                except Exception as e:
                    add_error(204, "MTBF inválido.", str(e))
```

```
                    print_error_summary()

        choice.observe(on_choice_change, names="value")
        btn_file.on_click(on_btn_file_clicked)
        btn_mtbf_no.on_click(on_btn_mtbf_no_clicked)
        btn_mtbf_from_file.on_click(on_btn_mtbf_from_file_clicked)

        show_si()
        display(container)

    except Exception:
        # Fallback consola
        ans = input("¿Tienes un archivo .CSV con datos? [s/n]: ").strip().lower()
        if ans.startswith("s"):
            run_full_pipeline_with_file(file_name=None, IS_COLAB=False,
return_on_need_mtbf=False)
        else:
            try:
                mtbf_str = input("Ingresa el MTBF (en días) para asumir distribución
exponencial: ").strip()
                mtbf_value = float(mtbf_str.replace(",", "."))
                run_expon_mtbf_only(mtbf_value, base_name="SinArchivo")
            except Exception as e:
                add_error(204, "No se obtuvo un MTBF válido.", str(e))
                print_error_summary()
else:
    # Entorno no Colab: consola
    ans = input("¿Tienes un archivo .CSV con datos? [s/n]: ").strip().lower()
    if ans.startswith("s"):
        run_full_pipeline_with_file(file_name=None, IS_COLAB=False,
return_on_need_mtbf=False)
    else:
        try:
            mtbf_str = input("Ingresa el MTBF (en días) para asumir distribución
exponencial: ").strip()
            mtbf_value = float(mtbf_str.replace(",", "."))
            run_expon_mtbf_only(mtbf_value, base_name="SinArchivo")
        except Exception as e:
            add_error(204, "No se obtuvo un MTBF válido.", str(e))
            print_error_summary()
```

Anexo B. Código de Google App Script para automatizar la transformación de formato para exportar a Power BI.

```
const INPUT_SHEET  = 'DICIEMBRE';            // hoja origen
const OUTPUT_SHEET = 'DICIEMBRE PROCESADO';  // hoja destino

function transponerReliability() {
  const ss = SpreadsheetApp.getActive();
```

```
  const inSh = ss.getSheetByName(INPUT_SHEET);
  if (!inSh) throw new Error(`No existe la hoja "${INPUT_SHEET}"`);

  const outSh = ss.getSheetByName(OUTPUT_SHEET) || ss.insertSheet(OUTPUT_SHEET);
  outSh.clear();

  const lastRow = inSh.getLastRow();
  const lastCol = inSh.getLastColumn();
  if (lastRow < 5 || lastCol < 2) {
    throw new Error('La hoja parece incompleta: se esperan al menos 5 filas y 2
columnas.');
  }

  // ---------- 1) Leer EquipoID (A1 etiqueta, B1 valor) ----------
  const etiquetaEquipo = String(inSh.getRange(1,1).getValue() ||
'').trim().toUpperCase();
  const equipoIdStr    = String(inSh.getRange(1,2).getValue() || '').toString().trim();
  if (etiquetaEquipo !== 'EQUIPOID' || !equipoIdStr) {
    throw new Error('No se encontró EquipoID en A1/B1 (A1="EquipoID", B1=<valor>).');
  }
  // EquipoID puede venir como número o texto; lo dejamos como número si aplica
  const EquipoID = isNaN(Number(equipoIdStr)) ? equipoIdStr : Number(equipoIdStr);

  // ---------- 2) Detectar fila de encabezado con "Fecha" en col A ----------
  let headerRow = -1;
  for (let r = 1; r <= Math.min(lastRow, 50); r++) {
    const v = String(inSh.getRange(r,1).getValue() || '').trim().toUpperCase();
    if (v === 'FECHA') { headerRow = r; break; }
  }
  if (headerRow === -1) {
    throw new Error('No se encontró la fila de encabezado con "Fecha" en la columna
A.');
  }


  const rowESID   = headerRow;
  const rowCompID = headerRow + 1;
  const rowFactID = headerRow + 2;
  const rowData   = headerRow + 3;

  if (lastRow < rowData) {
    throw new Error('No hay filas de datos debajo de los encabezados detectados.');
  }

  // ---------- 3) Leer encabezados por columna ----------
  const hdrESID = inSh.getRange(rowESID, 1, 1, lastCol).getValues()[0];   // A=Fecha,
B..=EquipoSistemaID
```

```
  const hdrComp = inSh.getRange(rowCompID, 1, 1, lastCol).getValues()[0];  //
B..=ComponenteID (o vacío)
  const hdrFact = inSh.getRange(rowFactID, 1, 1, lastCol).getValues()[0];  //
B..=FactorID (o vacío)

  // ---------- 4) Leer datos ----------
  const data = inSh.getRange(rowData, 1, lastRow - rowData + 1, lastCol).getValues();

  // ---------- 5) Preparar salida ----------
  const outHeader =
['Fecha','EquipoID','EquipoSistemaID','ComponenteID','FactorID','Nivel','Origen','Confi
abilidad'];
  const outRows = [outHeader];

  // Recorremos filas de datos
  for (let r = 0; r < data.length; r++) {
    const row = data[r];

    // Col A: Fecha
    const fecha = row[0]; // A
    if (fecha === '' || fecha === null || typeof fecha === 'undefined') continue;

    // Recorremos columnas desde B (col index 1) en adelante
    for (let c = 1; c < lastCol; c++) {
      const valorRaw = row[c];
      if (valorRaw === '' || valorRaw === null || typeof valorRaw === 'undefined')
continue;

      // EquipoSistemaID desde cabecera (fila headerRow)
      const esidRaw = hdrESID[c];
      if (esidRaw === '' || esidRaw === null || typeof esidRaw === 'undefined')
continue;

      // Normalizar IDs posibles (pueden venir texto/número)
      const EquipoSistemaID = isNaN(Number(esidRaw)) ? String(esidRaw).trim() :
Number(esidRaw);

      // ComponenteID & FactorID desde filas headerRow+1 y headerRow+2
      const compRaw = hdrComp[c];
      const factRaw = hdrFact[c];

      const tieneComponente = !(compRaw === '' || compRaw === null || typeof compRaw
=== 'undefined');
      const ComponenteID = tieneComponente
        ? (isNaN(Number(compRaw)) ? String(compRaw).trim() : Number(compRaw))
        : '';
```

```
      const FactorID = (tieneComponente && !(factRaw === '' || factRaw === null ||
typeof factRaw === 'undefined'))
        ? (isNaN(Number(factRaw)) ? String(factRaw).trim() : Number(factRaw))
        : '';

      // Confiabilidad (decimal; admite coma)
      let confiab = Number(valorRaw);
      if (isNaN(confiab)) {
        confiab = Number(String(valorRaw).replace(',', '.'));
      }
      if (isNaN(confiab)) continue; // si no se puede convertir, saltamos

      const Nivel  = tieneComponente ? 'Componente' : 'Sistema';
      const Origen = tieneComponente ? 'Calculado'  : 'Asumido';

      outRows.push([
        fecha,
        EquipoID,
        EquipoSistemaID,
        tieneComponente ? ComponenteID : '',
        tieneComponente ? FactorID    : '',
        Nivel,
        Origen,
        confiab
      ]);
    }
  }

  // ---------- 6) Escribir salida ----------
  outSh.getRange(1, 1, outRows.length, outRows[0].length).setValues(outRows);
  outSh.setFrozenRows(1);

  // Formato numérico (Confiabilidad = col 8)
  const lastOutRow = outSh.getLastRow();
  if (lastOutRow > 1) {
    outSh.getRange(2, 8, lastOutRow - 1, 1).setNumberFormat('0.000000');
  }
  SpreadsheetApp.flush();
}
```