Asesorías y Tutorías para la Investigación Científica en la Educación Puig-Salabarría S.C.
José María Pino Suárez 400-2 esq a Lerdo de Tejada. Toluca, Estado de México. 7223898475
RFC: ATI120618V12

Revista Dilemas Contemporáneos: Educación, Política y Valores.
http://www.dilemascontemporaneoseducacionpoliticayvalores.com/
Año: XIII Número: 3 Artículo no.:57 Período: 1 de mayo del 2026 al 31 de agosto del 2026

TÍTULO: Desafíos didácticos en Estadística Descriptiva: de la resolución manual a la computacional empleando Python en la carrera de Ingeniería en Software.

AUTORES:

1. Dr. Francisco Antonio Torres Espriú.

RESUMEN: Este estudio analiza el impacto de implementar Python en el aprendizaje de Estadística Descriptiva en estudiantes de tercer semestre de Ingeniería en Software. Mediante un diseño de muestra pareada (N = 27), se comparó el desempeño en evaluaciones manuales frente a computacionales. Los resultados mostraron una disminución aritmética en el rendimiento con la herramienta digital (Media: 67.5) comparado con el método manual (Media: 75.2); sin alcanzar significancia estadística ($p > .05$), sugiriendo una adaptación de competencias pese a la sobrecarga cognitiva inicial. Asimismo, se encontró una correlación positiva ($r = .413$) entre el cumplimiento de actividades prácticas y la calificación digital. Se concluye que la programación actúa como filtro de disciplina y la metodología intercalada permite mitigar la barrera sintáctica.

PALABRAS CLAVES: estadística descriptiva, Python, ingeniería en software, sobrecarga cognitiva, evaluación educativa.

TITLE: Didactic challenges in Descriptive Statistics: from manual to computational solutions using Python in the Software Engineering degree.

AUTHOR:

1. PhD. Francisco Antonio Torres Espriú.

ABSTRACT: This study analyzes the impact of implementing Python in learning Descriptive Statistics among third-semester Software Engineering students. Using a paired sample design (N = 27), performance in manual versus computational evaluations was compared. Results showed an arithmetic decrease in performance with the digital tool (Mean: 67.5) compared to the manual method (Mean: 75.2), without reaching statistical significance ($p > .05$), suggesting a competence adaptation despite initial cognitive overload. Likewise, a positive correlation ($r = .413$) was found between practical activity compliance and digital grades. It is concluded that programming acts as a discipline filter and the interleaved methodology permits mitigating the syntactic barrier.

KEY WORDS: descriptive statistics, Python, software engineering, cognitive overload, educational assessment.

INTRODUCCIÓN.

En el actual ecosistema de la Educación 4.0, la formación del profesional en Ingeniería en Software ha trascendido la mera codificación de algoritmos para integrar competencias analíticas complejas. La alfabetización de datos (*data literacy*) se ha erigido como una competencia transversal crítica, impulsada por la omnipresencia de la toma de decisiones basada en datos en la industria tecnológica. En este contexto, la asignatura de Probabilidad y Estadística se consolida no como un requisito curricular aislado, sino como el fundamento matemático indispensable para áreas avanzadas como la Inteligencia Artificial, el Aprendizaje Automático y la Ciencia de Datos.

La didáctica de esta disciplina enfrenta una crisis de identidad. Mientras que la práctica profesional exige el dominio de herramientas computacionales robustas, la enseñanza tradicional a menudo permanece anclada en métodos manuales que priorizan la aritmética sobre la interpretación. Ante esta disyuntiva, las instituciones de educación superior han comenzado a migrar hacia currículos mediados por tecnología, adoptando lenguajes de programación de alto nivel como vehículos de aprendizaje.

Esta transición no es trivial. La literatura especializada advierte que la introducción simultánea de conceptos estadísticos abstractos y sintaxis de programación rigurosa puede generar barreras de aprendizaje significativas. El estudiante novato se ve obligado a librar una doble batalla cognitiva: entender la naturaleza estocástica de los datos, y al mismo tiempo, dominar las reglas gramaticales de un lenguaje formal.

Planteamiento del Problema y Justificación.

A pesar de que los estudiantes de Ingeniería en Software poseen antecedentes en programación, la aplicación de estos conocimientos al ámbito estadístico no es automática. Existe una brecha entre la lógica de programación estructurada (bucles, condicionales) y la programación para análisis de datos (vectorización, *dataframes*). El problema central radica en determinar si la incorporación inmediata de herramientas como Python facilita la comprensión estadística o si, por el contrario, la complejidad sintáctica actúa como un distractor que reduce el rendimiento académico en etapas tempranas.

Objetivos e Hipótesis.

El objetivo general de esta investigación es analizar el impacto en el rendimiento académico y la carga cognitiva al transitar de la resolución manual a la computacional en la asignatura de Probabilidad y Estadística.

Específicamente, se busca:

1. Comparar el desempeño de los estudiantes en evaluaciones manuales versus evaluaciones basadas en *scripts* de Python.
2. Determinar si la disciplina en la entrega de actividades prácticas actúa como un factor mitigante ante la dificultad técnica.

La hipótesis de trabajo (H_1) sostiene, que en una fase inicial, el rendimiento en la evaluación computacional será significativamente menor al de la evaluación manual debido a la carga cognitiva extrínseca impuesta por la sintaxis del lenguaje, a menos que exista una práctica deliberada constante.

DESARROLLO.

Fundamentación teórica.

La Estadística en la Ingeniería de Software.

La estadística ha dejado de ser una disciplina auxiliar para convertirse en el motor inferencial de la tecnología moderna. James et al. (2023) establecen que el aprendizaje estadístico es el conjunto de herramientas críticas para dar sentido a la complejidad de los datos actuales. Como señalan Bruce, Bruce, y Gedeck (2020), muchos desarrolladores carecen de formación estadística formal, lo que lleva a errores graves en la interpretación de modelos; por ende, la educación universitaria debe trascender la mecanización aritmética para enfocarse en la perspectiva estadística, entendiendo que conceptos como la variabilidad son la base para construir modelos predictivos robustos

Python y la librería Pandas como Estándar.

Aunque Python nació como un lenguaje de propósito general, su hegemonía en el análisis de datos se consolidó con la creación de herramientas específicas. McKinney (2013), creador de la librería *Pandas*, diseñó esta herramienta para cubrir el vacío entre las hojas de cálculo y la programación científica. Esta evolución transformó el currículo: hoy en día, enseñar estadística sin una herramienta computacional es anacrónico; sin embargo, Dogucu y Çetinkaya-Rundel (2021) advierten que el uso de estas herramientas introduce una nueva capa de complejidad: el estudiante ya no solo lucha contra la abstracción matemática, sino también contra la sintaxis del código.

Estrategias Didácticas: el Enfoque Intercalado.

Para mitigar la sobrecarga cognitiva descrita por Sweller (1988), la didáctica moderna sugiere abandonar la enseñanza en bloques aislados (primero toda la teoría, luego todo el software) en favor de un aprendizaje intercalado. Según Robins et al. (2003), los novatos en programación dedican demasiada atención a la sintaxis superficial; por tanto, una estrategia donde el concepto se refuerza primero manualmente (andamiaje conceptual), y posteriormente, se transfiere al código (andamiaje técnico) permite al estudiante

mapear el conocimiento matemático previo a la nueva estructura algorítmica sin saturar su memoria de trabajo (Thorgeirsson y Weidmann, 2024).

Estado del Arte.

Tendencias en la Enseñanza de Estadística Computacional.

La integración de lenguajes de programación en la educación matemática ha generado un cuerpo creciente de literatura en los últimos cinco años. La revisión de investigaciones recientes permite categorizar el panorama actual en tres vertientes principales: la visualización como herramienta cognitiva, la barrera sintáctica, y la necesidad de modelos híbridos.

Visualización y Aprendizaje Constructivista.

Diversos estudios coinciden en que la capacidad gráfica de Python supera las limitaciones de la enseñanza tradicional. Investigaciones recientes en educación en ingeniería, como las de Thorgeirsson y Weidmann (2024) sugieren, que el uso de estrategias de codificación intercalada y *notebooks* interactivos permite a los estudiantes manipular variables y observar cambios en tiempo real, fomentando un aprendizaje por descubrimiento. Autores que defienden esta postura argumentan que la narrativa de datos facilita la conexión entre la teoría abstracta y los fenómenos reales, aumentando la motivación intrínseca del alumnado**.**

La Problemática de la "Fricción Instrumental".

En contraste, existe evidencia empírica que advierte sobre los riesgos de una implementación tecnológica sin andamiaje. Literatura fundamental en el área (Robins et al., 2003) ha documentado, que cuando se evalúa a estudiantes novatos, una parte significativa del tiempo se pierde resolviendo errores de sintaxis (*SyntaxErrors*). Este fenómeno, que autores contemporáneos como Dogucu y Çetinkaya-Rundel (2021) asocian a la complejidad de las herramientas modernas sugiere, que el software puede convertirse en un obstáculo epistemológico si el estudiante no posee una fluidez previa en el lenguaje, validando las preocupaciones sobre la sobrecarga cognitiva en etapas tempranas.

El Vacío en la Literatura: La Ausencia de Modelos Simultáneos.

Si bien abundan comparativas dicotómicas en la literatura (Excel vs. Python), se detecta un vacío empírico respecto al impacto de modelos híbridos simultáneos en el perfil específico del Ingeniero de Software. La mayoría de los trabajos previos abordan la enseñanza en bloques separados (primero teoría, luego laboratorio). El presente estudio busca llenar este vacío, aportando evidencia sobre cómo la alternancia iterativa (Lápiz-Código-Validación) puede mitigar la carga cognitiva y aprovechar las competencias previas de programación del perfil ingenieril.

Diseño Metodológico.

Participantes y Contexto.

El estudio se desarrolló en una institución de educación superior pública ubicada en el sur del estado de Sonora, México, durante el ciclo escolar Agosto - Diciembre 2025. La población objetivo estuvo conformada por estudiantes del programa de Ingeniería en Software. Se utilizó un muestreo no probabilístico por conveniencia, seleccionando un grupo intacto de tercer semestre inscrito en la asignatura de Probabilidad y Estadística. La muestra final (N = 27) estuvo compuesta por 22 hombres y 5 mujeres. Como perfil de ingreso, los estudiantes ya habían cursado la asignatura de Programación I en el primer semestre, por lo que contaban con competencias en lógica estructurada y sintaxis básica de Python; sin embargo, su experiencia previa estaba orientada al desarrollo de algoritmos generales y no al uso de librerías específicas de Ciencia de Datos (*Data Science*) como Pandas o Matplotlib.

Instrumentos de Recolección de Datos.

Para medir el rendimiento académico comparativo, se utilizaron dos instrumentos de evaluación sumativa diseñados por el docente titular, alineados estrictamente a los contenidos temáticos del programa oficial:

1. *Instrumento A (Resolución Manual).* Prueba escrita con problemas contextualizados para calcular medidas de tendencia central y dispersión utilizando calculadora científica y fórmulas desarrolladas. Se evaluó el procedimiento algebraico y la interpretación.

2. *Instrumento B (Resolución Computacional).* Prueba práctica en laboratorio. Se presentaron problemas equivalentes para ser resueltos mediante *scripts* en Python. Se evaluó la correcta importación de librerías, la limpieza del código y la visualización de datos.

Para la implementación técnica de este instrumento, se seleccionó el lenguaje Python debido a su versatilidad en el manejo de datos. Específicamente, se instruyó a los estudiantes en el uso de la biblioteca Pandas para la manipulación de estructuras tipo *DataFrame*, permitiendo replicar las tablas de frecuencias manuales; asimismo, se integró la biblioteca Matplotlib para la generación de gráficos, con el objetivo pedagógico de que el estudiante no solo obtuviera el dato numérico, sino que visualizara la distribución, reforzando conceptos de variabilidad que suelen ser abstractos en el cálculo puramente algebraico.

Adicionalmente, se definió la variable "Índice de Cumplimiento Práctico", operacionalizada como el porcentaje acumulado de entregas de prácticas de laboratorio y ejercicios de resolución manual intra y extraclase durante el periodo evaluado. Esta métrica se utilizó como un indicador de la disciplina procedimental del estudiante.

Procedimiento Experimental (Estrategia Simultánea).

A diferencia de los enfoques secuenciales tradicionales, se implementó una estrategia de enseñanza híbrida e iterativa. Para cada unidad temática (ej. Medidas de Tendencia Central, Dispersión, Gráficos), se siguió un ciclo didáctico de cuatro fases diseñado para gestionar la carga cognitiva.

Tal como se esquematiza en la Figura 1, el proceso no fue lineal, sino cíclico. Este diseño permitió que el estudiante utilizara el cálculo manual como un mecanismo de validación lógica para su código, cerrando la brecha entre la abstracción matemática y la implementación tecnológica.

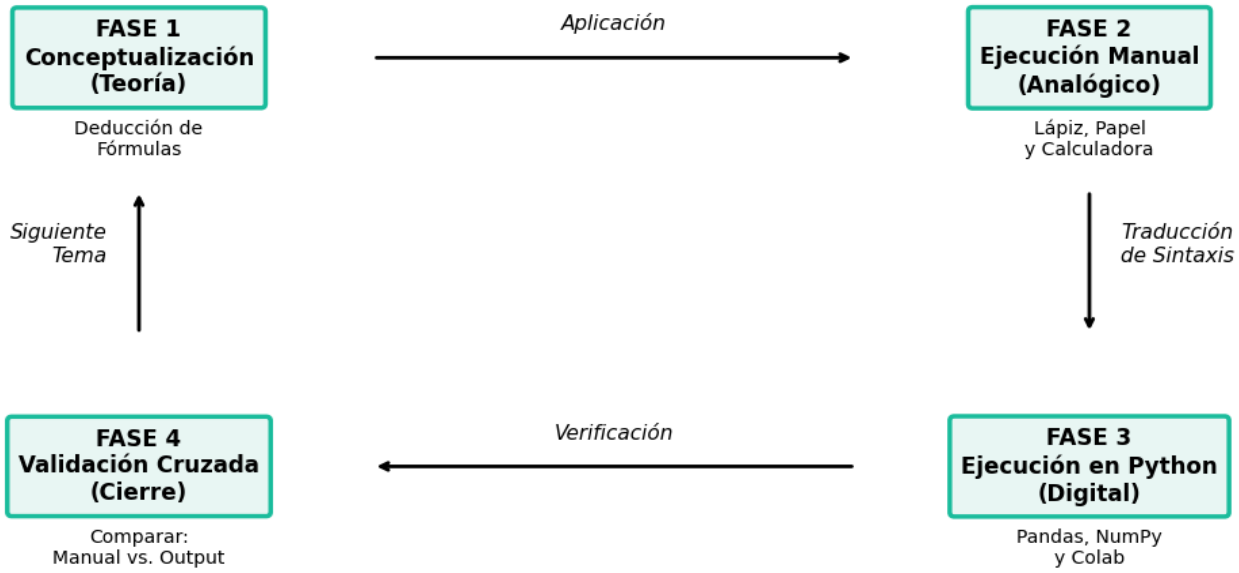

Figura 1. Modelo de Secuencia Didáctica Iterativa.

Nota. El esquema muestra el flujo de trabajo para cada unidad temática, donde la resolución manual sirve como filtro de validación antes del cierre del ciclo. Fuente: Elaboración propia.

Descripción de las Fases del Ciclo.

1. **Fase 1. Conceptualización (Teoría).** Explicación del fundamento matemático y deducción de fórmulas en pizarrón.
2. **Fase 2. Resolución Manual (Analógico).** Los estudiantes resolvían ejercicios con calculadora científica. El objetivo era asegurar la comprensión de la mecánica aritmética y la lógica del resultado sin distracciones tecnológicas.
3. **Fase 3. Transición Computacional (Digital).** Inmediatamente después, se trasladaba el mismo problema al entorno Google Colab. Se instruía sobre cómo las fórmulas manuales se traducen en funciones de las librerías Pandas y NumPy.
4. **Fase 4. Validación Cruzada (Cierre).** Se asignaban ejercicios, donde el estudiante debía validar sus resultados manuales comparándolos con el *output* del código. Si los valores no coincidían, el estudiante debía depurar (*debug*) su proceso, fomentando la metacognición.

En la Semana 8, se realizó un Caso Práctico Integrador resuelto totalmente en Python, seguido de las sesiones de evaluación final (Instrumento A y B) en días consecutivos.

Análisis de Datos.

Los datos recolectados se procesaron para obtener estadística descriptiva e inferencial.

1. *Análisis Descriptivo.* Cálculo de medias y desviaciones estándar para ambas modalidades.
2. *Verificación de Supuestos.* Prueba de normalidad (Shapiro-Wilk) dada la muestra (N = 27).
3. *Prueba de Hipótesis.* Se utilizó la prueba *t* de Student para muestras emparejadas con el fin de determinar diferencias significativas entre el rendimiento manual y computacional.
4. *Correlación.* Se utilizó el coeficiente de Pearson para analizar la relación entre el "Índice de Cumplimiento Práctico" y la calificación en Python.

Resultados.

Para garantizar la reproducibilidad y rigor técnico del estudio, el análisis estadístico se ejecutó mediante el software JASP (Versión 0.18.3), mientras que la visualización de datos se generó utilizando librerías científicas de Python (Seaborn y Matplotlib). El análisis se estructuró en cuatro fases secuenciales.

Distribución de Datos y Supuestos Estadísticos.

Previamente a la selección de la prueba de hipótesis, se evaluó la distribución de las calificaciones mediante la prueba de Shapiro-Wilk. Los resultados indicaron que las calificaciones de la evaluación manual siguen una distribución normal estricta ($W = 0.96$, $p = .077$). En contraste, las calificaciones obtenidas mediante Python mostraron una desviación de la normalidad ($W = 0.94$, $p = .014$$).

Como se aprecia en la Figura 2, la curva de la evaluación manual (azul) es relativamente más leptocúrtica y presenta menor dispersión, indicando un desempeño homogéneo; por el contrario, la curva de Python (roja) es platicúrtica y extendida, visualizando gráficamente cómo la herramienta tecnológica aumentó la dispersión de las notas. A pesar de esta asimetría, se procedió a utilizar pruebas paramétricas,

fundamentado en la robustez de la prueba *t* de Student cuando N se aproxima a 30 (Teorema del Límite Central).

Comparativa de Rendimiento: La Barrera Tecnológica.

El objetivo central de la investigación fue determinar si la sintaxis de programación afectaba negativamente la capacidad de resolución de problemas estadísticos. En primera instancia, se analizaron los estadísticos descriptivos para ambas modalidades.

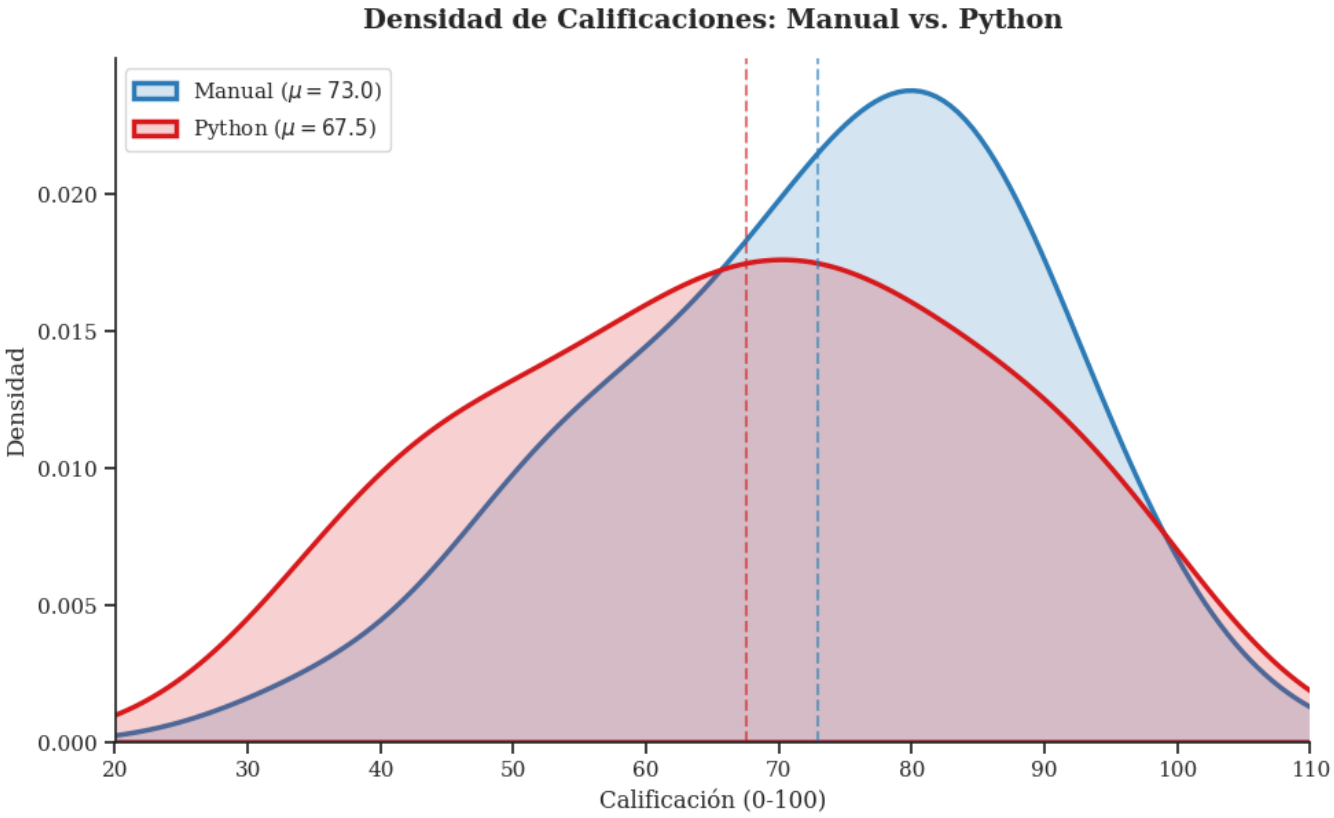

Figura 2. Comparativa de densidad (Kernel Density Estimation): Evaluación Manual vs. Computacional.

Fuente: Elaboración propia.

Como se detalla en la Tabla 1, el grupo mostró un desempeño superior en la evaluación manual, con una media de 75.19 (DE = 14.33). En contraste, al emplear la herramienta Python, el promedio descendió a 67.52 (DE = 18.55). Es importante notar el incremento en la desviación estándar en la segunda prueba, lo que indica una mayor heterogeneidad en el aprendizaje mediado por tecnología.

Tabla 1. Estadísticos Descriptivos de las evaluaciones (N = 27).

Modalidad de Evaluación	***Media***	***Desviación Estándar***	***Error Estándar de la Media***
Resolución Manual	75.19	14.33	2.75
Resolución con Python	67.52	18.55	3.57

Fuente: Elaboración propia.

Para verificar si esta diferencia aritmética era estadísticamente significativa, se aplicó la prueba *t* de Student para muestras emparejadas. Los resultados del contraste de hipótesis se presentan en la Tabla 2.

Tabla 2. Prueba T para Muestras Emparejadas.

Comparación	***Diferencia de Medias***	***t***	***gl***	***p***	***d de Cohen***
Manual – Python	7.667	1.711	26	0.099	0.329

Fuente: Elaboración propia.

El análisis arrojó un valor $t(26) = 1.711$ con una significancia bilateral de $p = .099$. Al obtener un valor p > .05, no existe evidencia estadística suficiente para rechazar la hipótesis nula. Esto implica, que aunque existió una fricción instrumental inicial que redujo el puntaje promedio, la caída en el rendimiento no fue significativa. Los estudiantes lograron transferir sus competencias matemáticas al entorno de código con un éxito razonable, sugiriendo una adaptación resiliente ante la sobrecarga cognitiva.

Invarianza del Rendimiento según el Género.

Se analizó si el uso de Python representaba una desventaja específica por género. El análisis inferencial (prueba *t* para muestras independientes) reveló una paridad técnica absoluta. El subgrupo femenino (N = 5) obtuvo una media de 67.20 en la evaluación con código, mientras que el masculino (N = 22) obtuvo 67.59. El contraste estadístico arrojó un valor $p = .975$, confirmando que la metodología fue inclusiva y equitativa.

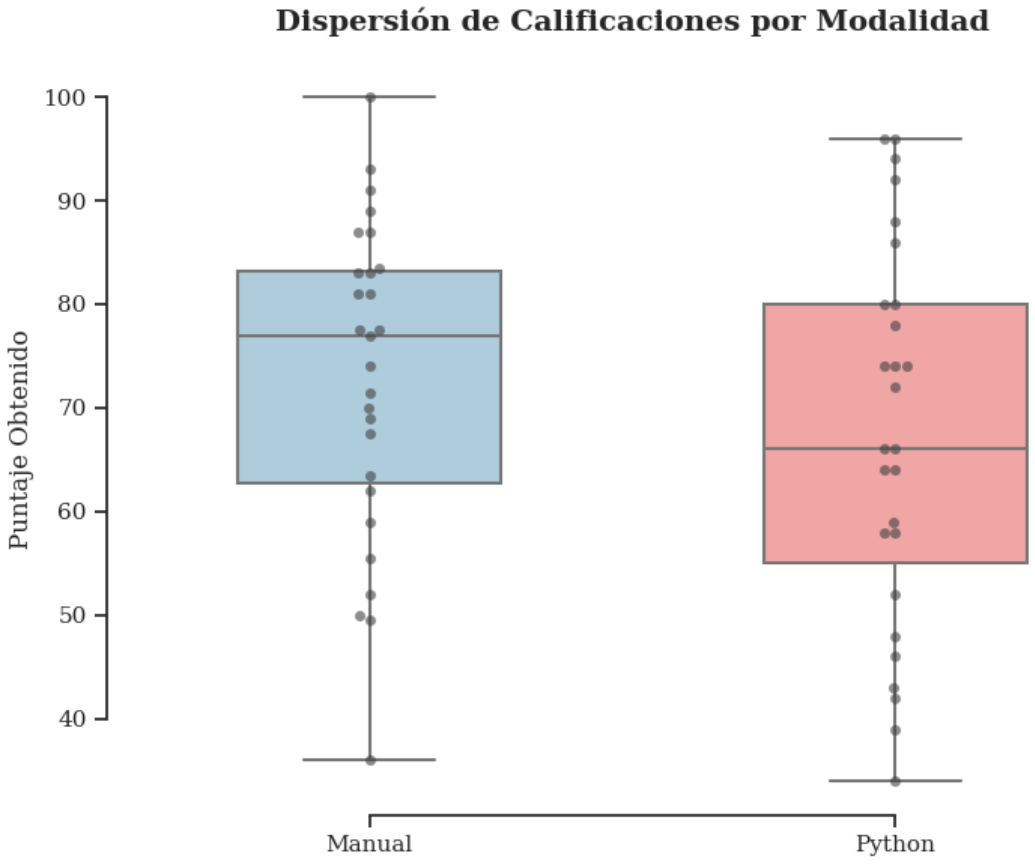

Figura 3. Diagrama de cajas y bigotes comparando la dispersión del desempeño.

Fuente: Elaboración propia.

La Disciplina Procedimental como Factor Asociado.

Finalmente, se exploró la relación entre la constancia en el trabajo práctico y el desempeño en la evaluación computacional. El análisis de correlación de Pearson arrojó un coeficiente de $r = .413$ ($p = .032$).

Como se observa en la Figura 4, la dispersión de los datos no sigue un modelo lineal estricto, lo cual es esperado en métricas educativas complejas; sin embargo, se detecta una tendencia positiva significativa.

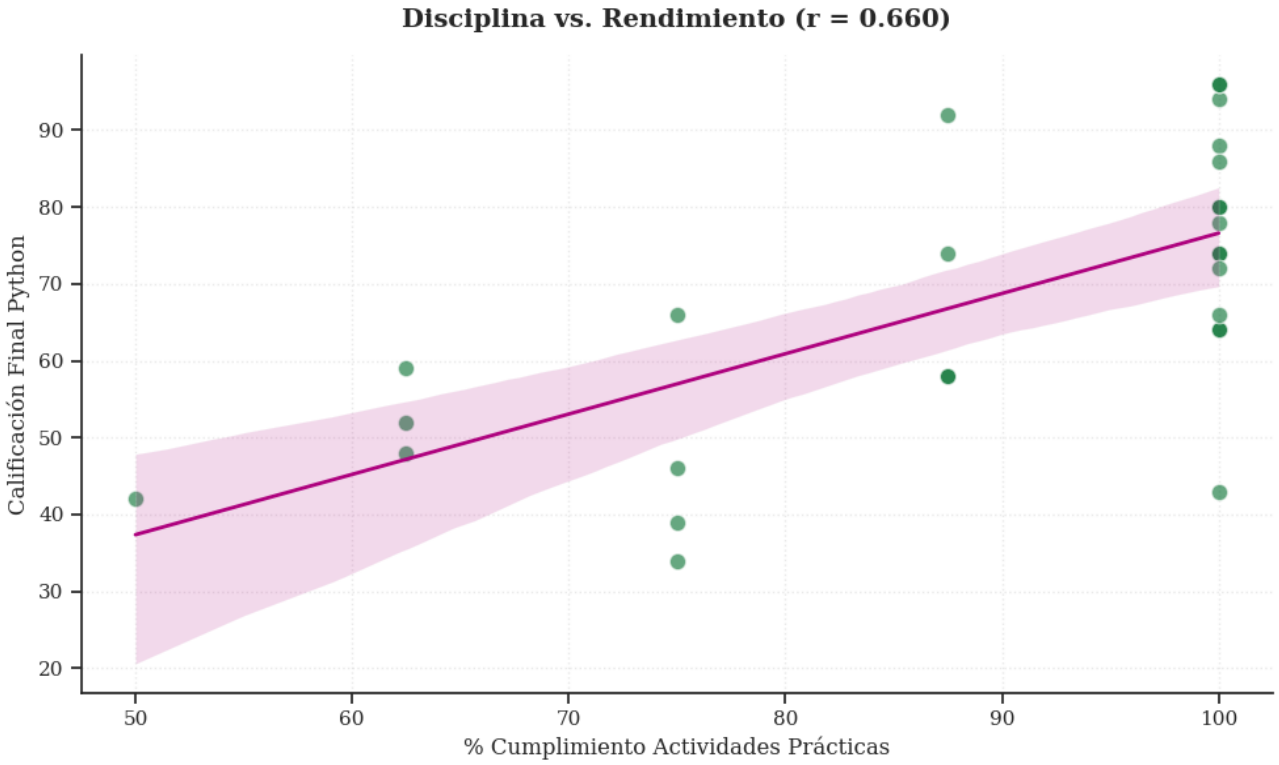

Figura 4. Correlación entre el cumplimiento de actividades prácticas y la calificación final. Fuente: Elaboración propia.

El coeficiente de determinación ($R^2 \approx 0.17$) indica que el cumplimiento de actividades explica el 17% de la varianza en la calificación final. Esto sugiere que la entrega de prácticas es una condición necesaria, aunque no suficiente, para el éxito en la asignatura.

Evidencia de la Implementación Computacional.

Como validación cualitativa de la intervención, se documentó la estructura del código generado por los participantes. Como se aprecia en la Figura 5, los estudiantes lograron implementar algoritmos secuenciales para el cálculo de medidas centrales, superando la barrera de la sintaxis.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# --- SIMULACIÓN DE DATOS (Para evitar cargar archivos externos) ---
# Creamos una lista de calificaciones ficticias de estudiantes
datos_simulados = {
    'calificacion': [65, 70, 85, 90, 55, 100, 88, 92, 75, 80,
                     78, 82, 85, 95, 60, 72, 88, 90, 85, 76,
                     68, 94, 77, 81, 83, 50, 100]
}
data = pd.DataFrame(datos_simulados)

# 1. CÁLCULO DE MEDIDAS DE TENDENCIA CENTRAL
media = data['calificacion'].mean()
mediana = data['calificacion'].median()
# La moda puede devolver varios valores, tomamos el primero
moda = data['calificacion'].mode()[0]

print(f"--- RESULTADOS ESTADÍSTICOS ---")
print(f"Media Aritmética: {media:.2f}")
print(f"Mediana: {mediana}")
print(f"Moda: {moda}")

# 2. CÁLCULO DE MEDIDAS DE DISPERSIÓN
desviacion_std = data['calificacion'].std()
rango = data['calificacion'].max() - data['calificacion'].min()

print(f"Desviación Estándar: {desviacion_std:.2f}")
print(f"Rango: {rango}")

# 3. GENERACIÓN DEL HISTOGRAMA
plt.figure(figsize=(10, 6))
plt.hist(data['calificacion'], bins=8, color='#4CAF50', edgecolor='black', alpha=0.7)
plt.title('Distribución de Calificaciones - Grupo Experimental')
plt.xlabel('Puntaje Obtenido')
plt.ylabel('Frecuencia de Estudiantes')
plt.grid(axis='y', alpha=0.5, linestyle='--')
plt.show()
```

Figura 5. Fragmento de script en Python utilizado para el cálculo de medidas de tendencia central y dispersión. Fuente. Elaboración propia.

Posteriormente, la Figura 6 muestra la salida gráfica obtenida. Esta visualización fue clave para la fase de "Validación Cruzada", permitiendo a los alumnos contrastar sus resultados manuales con la distribución generada por el software.

```
--- RESULTADOS ESTADÍSTICOS ---
Media Aritmética: 80.15
Mediana: 82.0
Moda: 85
Desviación Estándar: 12.85
Rango: 50
```

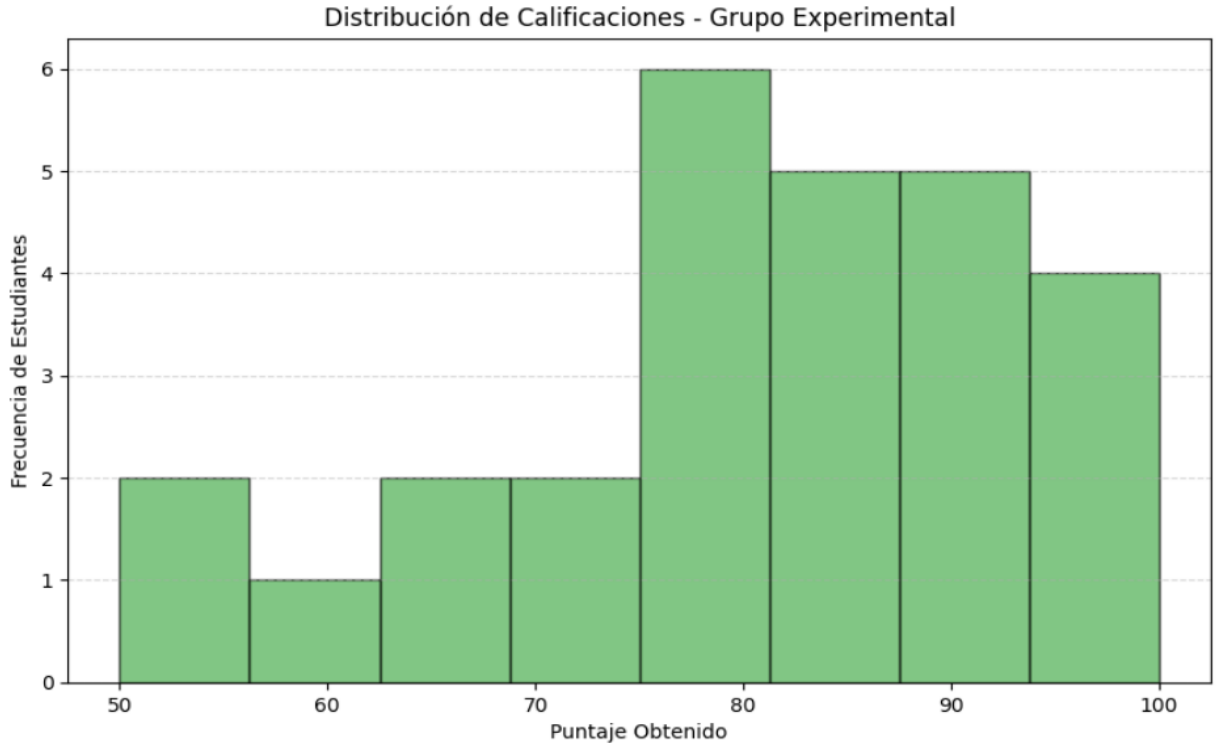

Figura 6. Visualización de resultados y distribución de frecuencias generada por el software. Fuente. Elaboración propia.

Discusión.

Los hallazgos de esta investigación aportan evidencia empírica sobre la viabilidad de integrar lenguajes de programación en la enseñanza de la estadística descriptiva sin sacrificar el rigor matemático.

Interpretación de la Resiliencia Cognitiva.

Contrario a la hipótesis de trabajo inicial, que predecía una caída significativa en el rendimiento debido a la sobrecarga cognitiva, los resultados (p = .099) sugieren que los estudiantes de Ingeniería en Software poseen una capacidad de adaptación superior a la esperada. La estrategia didáctica intercalada (Manual / Código) demostró ser eficaz como andamiaje. Aunque hubo un descenso aritmético en los promedios, este

no fue estadísticamente determinante, lo que implica que la fricción sintáctica de Python es una barrera superable cuando se cuenta con bases lógicas previas.

Inclusividad de la Herramienta.

Un hallazgo socialmente relevante es la invarianza de género ($p = .975$). En un contexto donde las brechas STEM son una preocupación constante, este estudio demuestra que la evaluación mediante código, bajo este diseño instruccional, es neutral. Las estudiantes mujeres mostraron el mismo nivel de competencia técnica que sus pares hombres, validando a Python como una herramienta educativa equitativa.

El Rol de la Práctica y Limitaciones.

La correlación encontrada ($r = .413$) subraya la importancia de la disciplina procedimental; sin embargo, la dispersión observada en los datos evidencia que el éxito en estadística computacional es multifactorial. El 17% de la varianza explicada por las tareas deja un amplio margen para otras variables latentes no medidas en este estudio, como el razonamiento lógico-matemático previo o la calidad cualitativa del tiempo de estudio.

CONCLUSIONES.

La transición de la resolución manual a la computacional en Probabilidad y Estadística es un desafío pedagógico viable; no obstante, se recomienda a las instituciones no ver a la programación como una "habilidad mágica", sino como una competencia que requiere andamiaje.

Para futuras implementaciones, se sugiere reforzar el seguimiento temprano de las actividades prácticas, ya que la ausencia de estas demostró ser el predictor más claro de riesgo académico.

REFERENCIAS BIBLIOGRÁFICAS.

1. Bruce, P., Bruce, A., & Gedeck, P. (2020). Practical statistics for data scientists: 50+ essential concepts using R and Python (2nd ed.). O'Reilly Media.

2. Dogucu, M., & Çetinkaya-Rundel, M. (2021). Web scraping in the statistics and data science curriculum: Challenges and opportunities. Journal of Statistics and Data Science Education, 29(1), 112–122. https://doi.org/10.1080/10691898.2020.1787116
3. James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). An introduction to statistical learning: with applications in Python. Springer Nature.
4. McKinney, W. (2013). Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. O'Reilly Media.
5. Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, 13(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200
6. Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. Cognitive Science, 12(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4
7. Thorgeirsson, A. T., & Weidmann, K. H. (2024). Bridging the gap: Interleaved practice in coding education for engineering students. Journal of Engineering Education, 113(1), 45-62.

DATOS DEL AUTOR.

1. Francisco Antonio Torres Espriú. Doctor en Innovación en Tecnología Educativa. Instituto Tecnológico de Sonora (Campus Guaymas) y responsable de la Academia de Matemáticas y Estadística. México. Correo electrónico: francisco.torres@itson.edu.mx

RECIBIDO: 12 de febrero del 2026. **APROBADO:** 9 de marzo del 2026.