EST. 2021 EMC
EDITORIAL MAR CARIBE

MÉTODOS ESTADÍSTICOS ASISTIDOS CON SOFTWARE:

Aplicaciones en Python, Octave y R

EDWIN JOHNY ASNATE SALAZAR - RAMÓN COSME CORREA BECERRA - HENRY OSWALDO VILLAREAL TORRES - MARCO ANTONIO CORCUERA DE LOS SANTOS - RAÚL YAJAHUANCA HUANCAS - EDWIN ALBERTO VALENCIA CASTILLO - SANDRA CECILIA RODRÍGUEZ AVILA

Métodos estadísticos asistidos con software: Aplicaciones en Python, Octave y R

Asnate Salazar, Edwin Johny; Correa Becerra, Ramón Cosme; Villareal Torres, Henry Oswaldo; Corcuera De los Santos, Marco Antonio; Valencia Castillo, Edwin Alberto; Rodríguez Ávila, Sandra Cecilia; Yajahuanca Huancas, Raúl

Primera edición (1ra. ed.): Octubre, 2025

Editado por:

Editorial Mar Caribe ®

www.editorialmarcaribe.es

Av. Gral. Flores 547, 70000 Col. del Sacramento, Departamento de Colonia, Uruguay.

Diseño de caratula e ilustraciones: *Isbelia Salazar Morote*

Libro electrónico disponible en:

https://editorialmarcaribe.es/ark:/10951/isbn.9789915698403

Formato: Electrónico

ISBN: 978-9915-698-40-3

ARK: ark:/10951/isbn.9789915698403

Editorial Mar Caribe (OASPA): Como miembro de la Open Access Scholarly Publishing Association, apoyamos el acceso abierto de acuerdo con el código de conducta, la transparencia y las mejores prácticas de OASPA para la publicación de libros académicos y de investigación. Estamos comprometidos con los más altos estándares editoriales en ética y deontología, bajo la premisa de «Ciencia Abierta en América Latina y el Caribe»

Editorial Mar Caribe, firmante Nº 795 de 12.08.2024 de la Declaración de Berlín

"… Nos sentimos obligados a abordar los retos de Internet como medio funcional emergente para la distribución del conocimiento. Obviamente, estos avances pueden modificar significativamente la naturaleza de la publicación científica, así como el sistema actual de garantía de calidad..." (Max Planck Society, ed. 2003, pp. 152-153).

Editorial Mar Caribe

Métodos estadísticos asistidos con software: Aplicaciones en Python, Octave y R

Colonia, 2025

Índice

Introducción

La estadística ha sido históricamente el pilar metodológico de la investigación en ciencias sociales, proporcionando las herramientas esenciales para analizar la variabilidad, construir modelos predictivos y sustentar el pensamiento racional. Sin embargo, en la era del *Big Data* y la Ciencia de Datos, la enseñanza y la práctica tradicionales de la estadística se enfrentan a un desfase crítico. Los currículos centrados en pruebas de hipótesis dicotómicas y el uso exclusivo de *software* con interfaz gráfica (GUI) ya no son suficientes para modelar la complejidad inherente a los fenómenos sociales contemporáneos.

Este libro nace de la imperiosa necesidad de tender un puente entre el rigor de la teoría estadística y las demandas computacionales del análisis de datos moderno. Su propósito es guiar al lector en la adopción de los tres ecosistemas de código abierto más relevantes para el cómputo científico y el análisis cuantitativo: Python, R y GNU Octave. Al dominar estas plataformas, el investigador social no solo se emancipa del *software* propietario, sino que adquiere las capacidades para realizar análisis que, por diseño, son más potentes, escalables y, fundamentalmente, reproducibles.

A lo largo de sus cuatro capítulos, exploraremos las fortalezas complementarias de cada lenguaje. R, diseñado desde su concepción para el cálculo estadístico y la visualización, se presenta como el entorno ideal para la inferencia metodológica profunda y la creación de gráficos de alta calidad. Por el contrario, Python es un lenguaje de programación de propósito general, destacando por su versatilidad para la integración de sistemas, la gestión de *Big Data* y el desarrollo de soluciones de *Machine Learning* a escala de producción. En teoría, se abordará GNU Octave como una alternativa robusta para el cómputo numérico y la simulación.

La ambición de este texto va más allá de la mera sintaxis. La investigación social avanzada exige ir más allá de la correlación y el valor *p* simplificado. Por ello, dedicaremos secciones cruciales a las metodologías

que definen la vanguardia: la Inferencia Causal (incluyendo técnicas como el *Double Machine Learning*), la Estadística Bayesiana para un manejo riguroso de la incertidumbre y el Modelado de Ecuaciones Estructurales (SEM) para capturar la complejidad de las variables latentes.

En última instancia, un análisis riguroso es inútil si no es transparente y verificable. Este libro enfatiza la reproducibilidad como un estándar de oro y enseña a los lectores a construir flujos de trabajo rastreables. Los lectores aprenderán a utilizar herramientas de documentación dinámica, como Quarto, para integrar código, análisis y texto narrativo en informes listos para la publicación, garantizando que el proceso analítico pueda ser verificado por cualquier colega.

Al concluir este recorrido, el lector habrá transformado su enfoque: de ser un usuario de *software* con capacidades limitadas, pasará a ser un científico de datos sociales capaz de plantear preguntas críticas, seleccionar el algoritmo más adecuado y construir soluciones que resistan el escrutinio de la metodología más avanzada. Lo invitamos a sumergirse en la práctica de la estadística del siglo XXI.

Capítulo I

Métodos Estadísticos Asistidos por Software: Análisis y Comparación

En el mundo actual, la toma de decisiones basada en datos se ha convertido en un aspecto esencial en diversas disciplinas, desde la investigación científica hasta el ámbito empresarial. Los métodos estadísticos son herramientas fundamentales que permiten analizar, interpretar y extraer conclusiones a partir de conjuntos de datos. Sin embargo, la complejidad y la cantidad de datos que se generan hoy en día han hecho que el uso de software especializado en estadística sea una necesidad más que una opción.

Los métodos estadísticos asistidos por software permiten a los analistas e investigadores realizar cálculos complejos y visualizaciones de datos de manera más eficiente y precisa. El uso de software no solo simplifica el proceso de análisis, sino que también ayuda a minimizar los errores humanos y a maximizar la reproducibilidad de los resultados. Además, muchas de estas herramientas ofrecen interfaces intuitivas y bibliotecas potentes que facilitan la implementación de técnicas estadísticas avanzadas (Perdigón y Pérez, 2022).

En este contexto, lenguajes de programación como Python, Octave y R han ganado popularidad entre los profesionales de la estadística. Cada uno de estos entornos ofrece características y ventajas propias, lo que permite a los usuarios elegir la herramienta más adecuada según sus necesidades y preferencias. Python cuenta con varias bibliotecas que facilitan la realización de análisis estadísticos. Entre las más destacadas se encuentran:

- NumPy: fundamental para realizar operaciones numéricas en Python. Proporciona estructuras de datos eficientes, como matrices

y arreglos multidimensionales, así como funciones matemáticas y estadísticas esenciales para el análisis de datos.

- Pandas: indispensables para la manipulación y el análisis de datos. Permite trabajar con estructuras de datos como DataFrames, facilitando la limpieza, transformación y análisis de conjuntos de datos, lo que simplifica las operaciones estadísticas.
- SciPy: Basada en NumPy, ofrece herramientas adicionales para cálculos científicos y técnicos. Incluye módulos de optimización, integración, interpolación, estadística y álgebra lineal, lo que la convierte en una opción ideal para análisis estadísticos avanzados.
- StatsModels: Se centra en la estimación de modelos estadísticos y en pruebas estadísticas. Proporciona una interfaz para realizar regresiones lineales y no lineales, análisis de series temporales y pruebas de hipótesis, entre otras funcionalidades.
- Matplotlib y Seaborn: Para la visualización de datos, Matplotlib es la biblioteca más utilizada en Python, mientras que Seaborn, construida sobre Matplotlib, ofrece una interfaz más sencilla y atractiva para crear gráficos estadísticos. Juntas, permiten representar datos de manera efectiva.

1.1 Implementación de Regresiones y Análisis de Varianza

La implementación de regresiones y de análisis de varianza (ANOVA) en Python es directa gracias a las bibliotecas mencionadas. Por ejemplo, para realizar una regresión lineal, se puede utilizar StatsModels de la siguiente manera:

python

```
import pandas as pd

import statsmodels.api as sm
```

Cargar datos

```
data = pd.read_csv('datos.csv')
```

Definir variables dependientes e independientes

```
X = data['variable_independiente']
y = data['variable_dependiente']
```

Agregar constante para el modelo

```
X = sm.add_constant(X)
```

Ajustar el modelo

```
modelo = sm.OLS(y, X).fit()
```

Obtener resumen de resultados

```
print(modelo.summary())
```

En cuanto al análisis de varianza, StatsModels también permite realizar ANOVA de forma sencilla, facilitando la comparación de medias entre diferentes grupos:

python

```
from statsmodels.formula.api import ols
import statsmodels.api as sm
```

Ajustar un modelo ANOVA

```
modelo_anova = ols('variable_dependiente ~ C(grupo)', data=data).fit()
tabla_anova = sm.stats.anova_lm(modelo_anova, typ=2)
print(tabla_anova)
```

La visualización de datos es crucial en el análisis estadístico, ya que permite interpretar los resultados con mayor claridad. Con Matplotlib y Seaborn, los usuarios pueden crear una variedad de gráficos, desde diagramas de dispersión hasta histogramas y gráficos de cajas (Toalombo et al., 2024). Por ejemplo, para visualizar la relación entre dos variables en un diagrama de dispersión, se puede usar el siguiente código:

python

```
import matplotlib.pyplot as plt

import seaborn as sns
```

Crear un diagrama de dispersión

```
plt.figure(figsize=(10, 6))

sns.scatterplot(x='variable_independiente', y='variable_dependiente', data=data)

plt.title('Diagrama de Dispersión')

plt.xlabel('Variable Independiente')

plt.ylabel('Variable Dependiente')

plt.show()
```

También Seaborn permite crear gráficos más complejos y estilizados, lo que ayuda a resaltar patrones y tendencias en los datos. Un gráfico de cajas puede ser útil para mostrar la distribución de una variable en diferentes grupos:

python

```
plt.figure(figsize=(10, 6))

sns.boxplot(x='grupo', y='variable_dependiente', data=data)

plt.title('Gráfico de Cajas por Grupo')

plt.xlabel('Grupo')

plt.ylabel('Variable Dependiente')

plt.show()
```

Python se ha consolidado como una herramienta poderosa y accesible para el análisis estadístico. Con sus diversas bibliotecas y capacidades de visualización, los analistas pueden emplear una amplia gama de técnicas estadísticas, lo que facilita la interpretación y la comunicación de los resultados en sus investigaciones. Octave es un software de código abierto diseñado principalmente para cálculos

numéricos y análisis matemático. Su sintaxis es similar a la de MATLAB, lo que facilita su uso para quienes ya están familiarizados con MATLAB. En el contexto de los métodos estadísticos, Octave ofrece una variedad de herramientas y funciones que permiten realizar análisis complejos de manera eficiente.

Una de las principales ventajas de Octave es su accesibilidad. Al ser un software de código abierto, cualquier usuario puede descargarlo y utilizarlo sin costo alguno. Esto lo convierte en una alternativa atractiva para estudiantes, investigadores y profesionales que desean realizar análisis estadísticos sin adquirir licencias de software comercial. Además, Octave cuenta con una comunidad activa que contribuye al desarrollo de paquetes y funciones adicionales, lo que amplía sus capacidades.

Otra característica destacable de Octave es su compatibilidad con scripts de MATLAB, lo que significa que muchos de los códigos escritos para MATLAB pueden ejecutarse en Octave con pocas o ninguna modificación. Esta compatibilidad facilita la migración de proyectos entre ambos entornos, lo que puede resultar ventajoso para los usuarios de instituciones que utilizan MATLAB. Por último, Octave ofrece una amplia gama de funciones integradas para realizar análisis estadísticos, incluyendo pruebas de hipótesis, análisis de regresión y de varianza (ANOVA). También permite manipular matrices y realizar cálculos matemáticos complejos, esenciales para muchos métodos estadísticos.

Para ilustrar el uso de Octave en el análisis estadístico, consideremos un ejemplo sencillo de regresión lineal. Supongamos que tenemos un conjunto de datos que relaciona el ingreso de un grupo de individuos con su nivel educativo. Podemos utilizar Octave para estimar la relación entre estas dos variables.

```
octave

% Cargar datos

data = load('datos.csv'); % Suponiendo que los datos están en un archivo CSV

ingresos = data(:, 1); % Primer columna
```

```
educacion = data(:, 2); % Segunda columna
% Ajustar modelo de regresión lineal
modelo = regress(ingresos, [ones(length(educacion), 1), educacion]);
% Mostrar resultados
disp('Coeficientes de regresión:');
disp(modelo);
```

Este script carga los datos de un archivo CSV, ajusta un modelo de regresión lineal y muestra los coeficientes resultantes. Octave también permite realizar diagnósticos del modelo, como analizar los residuos y verificar supuestos de normalidad. Otro ejemplo podría ser el análisis de varianza (ANOVA) para comparar las medias de tres grupos; se presenta un código básico que ilustra cómo se podría implementar esto en Octave:

```
octave
% Cargar datos
grupos = load('grupos.csv'); % Datos de los grupos
% Realizar ANOVA
[p, tbl, stats] = anova1(grupos);
% Mostrar resultados de ANOVA
disp('Resultados de ANOVA:');
disp(tbl);
```

Este código permite realizar un ANOVA de un solo factor y muestra la tabla de resultados, que incluye valores p y estadísticas F, esenciales para interpretar la significancia de las diferencias entre grupos. Al comparar Octave con otros software estadísticos como Python y R, se pueden identificar varias similitudes y diferencias. Mientras que Python y R han ganado popularidad en la comunidad de análisis de datos, Octave se destaca por su simplicidad y su enfoque en cálculos numéricos, lo que

puede resultar beneficioso para usuarios que buscan una herramienta directa y menos compleja.

Python, por ejemplo, ofrece bibliotecas como Pandas y StatsModels, muy potentes para el análisis de datos, pero que pueden requerir un aprendizaje más profundo. R, por su parte, es conocido por su robustez estadística y la gran cantidad de paquetes dedicados, lo que lo hace ideal para análisis estadísticos avanzados (Raschka et al., 2020).

Sin embargo, Octave puede ser una opción preferible para quienes ya están acostumbrados a la sintaxis de MATLAB y buscan un entorno accesible y gratuito. Su capacidad para realizar cálculos numéricos de forma eficiente lo convierte en una herramienta valiosa en el ámbito del análisis estadístico, especialmente para educadores y estudiantes.

Octave es una herramienta poderosa y accesible para realizar análisis estadísticos, con características que lo hacen competitivo frente a otros software. Su facilidad de uso, su compatibilidad con MATLAB y su amplia gama de funciones estadísticas lo convierten en una opción a considerar para quienes deseen realizar análisis complejos sin los costos asociados al software comercial.

1.2 Análisis estadístico con R

R es un lenguaje de programación y un entorno de software libre que se ha consolidado como una de las herramientas más populares para el análisis estadístico y la visualización de datos. Su capacidad para manejar grandes conjuntos de datos y realizar análisis complejos lo convierte en una opción preferida entre estadísticos, investigadores y científicos de datos.

Una de las principales ventajas de R es su amplia gama de paquetes que facilitan el análisis estadístico avanzado. La comunidad de usuarios de R contribuye constantemente al desarrollo de nuevos paquetes, lo que permite que la plataforma se mantenga actualizada con las últimas metodologías y técnicas estadísticas. Además, R es altamente extensible, lo que permite a los usuarios crear sus propias funciones y paquetes personalizados según sus necesidades específicas.

Otra ventaja significativa de R es su capacidad para manejar datos de diferentes formatos y provenientes de diversas fuentes, como bases de datos SQL, archivos CSV y APIs. Esto facilita la integración de datos en el flujo de trabajo analítico. R también cuenta con potentes capacidades de manipulación de datos mediante bibliotecas como dplyr y tidyr, que simplifican la limpieza y la transformación de datos.

El manejo de datos en R se realiza principalmente mediante data frames, que son estructuras de datos bidimensionales similares a las tablas de las bases de datos. Esto permite a los usuarios organizar y manipular datos de manera eficiente. R ofrece una variedad de funciones para explorar y resumir datos, como summary(), str() y head(), que ayudan a obtener una comprensión inicial de los datos antes de realizar análisis más profundos.

En cuanto a los modelos estadísticos, R es muy conocido por su capacidad para realizar regresiones lineales y no lineales, análisis de varianza (ANOVA), modelos mixtos y muchos otros enfoques estadísticos. La función lm() permite implementar modelos de regresión lineal de manera sencilla, mientras que aov() se utiliza para realizar análisis de varianza. Además, R permite evaluar modelos mediante técnicas como la validación cruzada y el análisis de residuos, lo cual es crucial para garantizar la robustez y la validez de los resultados.

Uno de los aspectos más destacados de R es su capacidad para crear gráficos de alta calidad y personalizados. La biblioteca ggplot2 es especialmente reconocida por su enfoque en la creación de visualizaciones elegantes y complejas a partir de datos. Con ggplot2, los usuarios pueden construir gráficos de dispersión, histogramas, boxplots y mucho más, utilizando una sintaxis intuitiva que ofrece gran flexibilidad en el diseño (Wickham, 2016).

Además, R facilita la creación de reportes dinámicos mediante R Markdown, que permite combinar código, texto y gráficos en un único documento. Esto se traduce en reportes reproducibles que pueden compartirse fácilmente. Con R Markdown, los usuarios pueden generar documentos en diferentes formatos, incluidos HTML, PDF y Word, lo que

lo convierte en una herramienta invaluable para la comunicación de resultados estadísticos.

R se presenta como una herramienta potente y versátil para el análisis estadístico. Su vasta comunidad, la riqueza de paquetes disponibles y sus capacidades de visualización lo convierten en una opción destacada para quienes buscan realizar análisis estadísticos de manera eficaz y profesional.

En la actualidad, el uso de software estadístico se ha convertido en una herramienta esencial para investigadores, analistas y profesionales de diversas disciplinas. Herramientas como Python, Octave y R han democratizado el acceso a métodos estadísticos avanzados, permitiendo a personas de diferentes niveles de experiencia realizar análisis que antes habrían requerido un conocimiento profundo y especializado.

Python, con su rica variedad de bibliotecas y su enfoque versátil, se ha consolidado como una opción preferida en la comunidad de ciencia de datos. Su facilidad para integrarse con otras tecnologías y su amplia documentación lo convierten en una opción ideal tanto para principiantes como para expertos. Las bibliotecas como Pandas y SciPy han simplificado el manejo de datos y la realización de análisis estadísticos, mientras que Matplotlib y Seaborn facilitan la visualización de resultados, transformando la forma en que los analistas presentan sus descubrimientos.

Por otro lado, Octave, aunque menos conocido, ofrece un entorno similar a MATLAB, accesible y gratuito. Sus características lo hacen atractivo para quienes buscan una solución de código abierto para realizar análisis numéricos y estadísticos. Gracias a su simplicidad y a su capacidad para ejecutar scripts de MATLAB y Octave, Octave permite a los usuarios realizar análisis complejos sin incurrir en altos costos de software.

R, reconocido por su potencia en el análisis estadístico y por su amplia gama de paquetes, sigue siendo una herramienta fundamental en los ámbitos académico y profesional. Su enfoque en el análisis de datos y la creación de gráficos de alta calidad lo ha convertido en un favorito entre estadísticos y científicos de datos. R no solo permite un manejo eficiente de

datos, sino que también facilita la implementación de modelos estadísticos avanzados, lo que lo convierte en una opción insustituible para quienes trabajan en investigación y análisis (Mejía, 2022).

El uso de software estadístico, ya sea mediante Python, Octave o R, ha transformado la forma en que se realizan los análisis de datos. Estas herramientas no solo han permitido a los usuarios realizar análisis más complejos y precisos, sino que también han hecho que la estadística sea más accesible para todos. La elección del software dependerá de las necesidades específicas del proyecto y de la familiaridad del usuario con cada herramienta, pero lo innegable es que en el futuro, la estadística asistida por software continuará desempeñando un papel crucial en la toma de decisiones informadas y en la generación de conocimientos valiosos a partir de los datos (Martínez y del Águila, 2025).

1.3 Comparativa de los Entornos

Cada uno de estos software aborda la estadística desde una filosofía distinta. Entender esto es clave para elegir la herramienta adecuada (Ver Tabla 1).

Tabla 1. Características de los entornos R, Python y GNU Octave

Característica	**R**	**Python**	**GNU Octave**
Filosofía	Creado *por* estadísticos *para* estadísticos.	Lenguaje de propósito general con librerías de ciencia de datos.	Clon de MATLAB orientado al cálculo numérico y matricial.
Puntos Fuertes	Inferencia estadística, análisis exploratorio, bioestadística, gráficos académicos (ggplot2).	Machine Learning, Deep Learning, integración con apps web, limpieza de datos (pandas).	Álgebra lineal, ecuaciones diferenciales, ingeniería, prototipado rápido de algoritmos matemáticos.

Característica	R	Python	GNU Octave
Librerías Clave	stats (nativo), tidyverse, caret.	scipy.stats, pandas, statsmodels, scikit-learn.	Paquetes nativos (limitados en estadística avanzada, en comparación con R).
Curva de Aprendizaje	Pendiente inicial pronunciada, sintaxis idiosincrásica.	Muy amigable, con sintaxis legible y consistente.	Fácil si conoces álgebra matricial o si vienes de MATLAB.

Para entender las "aplicaciones", veamos cómo se ejecuta un método estadístico clásico: una regresión lineal simple (ajustar una línea y = \beta_0 + \beta_1x + \epsilon) en los tres lenguajes.

- *R (El enfoque clásico)*

R trata los modelos estadísticos como objetos de primera clase. La salida incluye, por defecto, ricos detalles estadísticos (valores p, R^2 y residuos).

R

Cargar datos

datos <- data.frame(x = c(1, 2, 3, 4, 5), y = c(2.1, 4.0, 6.1, 8.0, 10.2))

Ajustar modelo (fórmula y ~ x)

modelo <- lm(y ~ x, data = datos)

Ver resultados detallados (ANOVA, coeficientes, p-values)

summary(modelo)

- *En Python (El enfoque programático)*

Python requiere importar librerías específicas. statsmodels es lo más cercano a R, mientras que scikit-learn se enfoca más en la predicción que en la inferencia estadística.

Python

```
import statsmodels.api as sm

import pandas as pd
```

Datos

```
df = pd.DataFrame({'x': [1, 2, 3, 4, 5], 'y': [2.1, 4.0, 6.1, 8.0, 10.2]})
```

Definir variables (añadiendo constante para el intercepto)

```
X = sm.add_constant(df['x'])

y = df['y']
```

Ajustar modelo

```
modelo = sm.OLS(y, X).fit()
```

Ver resumen estadístico

```
print(modelo.summary())
```

- *En GNU Octave (El enfoque matemático)*

Octave ve la estadística como una operación de álgebra lineal. A menudo resuelves el sistema A x = B. Para estadística descriptiva básica es directo, pero para modelos complejos requiere más pasos manuales o paquetes adicionales.

Matlab

```
% Datos

x = [1; 2; 3; 4; 5];

y = [2.1; 4.0; 6.1; 8.0; 10.2];

% Ajuste polinómico de grado 1 (Lineal)

p = polyfit(x, y, 1);
```

% Evaluar el modelo

y_pred = polyval(p, x);

% Calcular R^2 manualmente (es menos automático que R)

y_mean = mean(y);

SStot = sum((y - y_mean).^2);

SSres = sum((y - y_pred).^2);

R2 = 1 - SSres/SStot;

disp(["Coeficientes: ", num2str(p)]);

disp(["R^2: ", num2str(R2)]);

- *ANOVA de una vía en R*

La función aov es estándar y el summary te da exactamente lo que un estadístico espera ver (grados de libertad, suma de cuadrados, valor F, valor p).

R

Preparar datos (Factor y Respuesta)

fertilizante <- factor(c(rep("A", 5), rep("B", 5), rep("C", 5)))

crecimiento <- c(20, 21, 23, 19, 20, Grupo A

24, 25, 27, 23, 26, Grupo B

15, 17, 16, 18, 15) Grupo C

datos <- data.frame(fertilizante, crecimiento)

Ejecutar ANOVA

modelo <- aov(crecimiento ~ fertilizante, data = datos)

Ver tabla de resultados

summary(modelo)

- *ANOVA de una vía en Python*

En Python, tenemos dos caminos principales. scipy es rápido pero básico, statsmodels es lo que debes usar si quieres una salida profesional similar a la de R.

Python

```
import pandas as pd

import statsmodels.api as sm

from statsmodels. Formula.api import ols
```

Preparar datos

```
df = pd.DataFrame({

   'fertilizante': ['A']5 + ['B']5 + ['C']5,

   'crecimiento': [20, 21, 23, 19, 20,

                   24, 25, 27, 23, 26,

                   15, 17, 16, 18, 15]

})
```

Ajustar el modelo (usando sintaxis de fórmula estilo R)

```
modelo = ols('crecimiento ~ C(fertilizante)', data=df).fit()
```

Generar tabla ANOVA

```
tabla_anova = sm.stats.anova_lm(modelo, typ=2)

print(tabla_anova)
```

- *ANOVA de una vía en GNU Octave*

Aunque tiene funciones estadísticas, a menudo es más directo trabajar con matrices o usar la función anova1 si tienes el paquete statistics cargado. Aquí vemos la estructura matricial pura.

Matlab

```
% Preparar datos (Matriz: columnas = grupos)
```

% Cada columna es un fertilizante diferente

X = [20, 24, 15;

21, 25, 17;

23, 27, 16;

19, 23, 18;

20, 26, 15];

% Ejecutar ANOVA (requiere paquete statistics o compatible con MATLAB)

% p = valor p, tbl = tabla de resultados, stats = estructura para pruebas post-hoc

[p, tbl, stats] = anova1(X);

% Si no tienes interfaz gráfica, 'tbl' contiene la matriz de celdas con los datos del ANOVA

disp(tbl)

Octave prefiere que los datos estén estructurados geométricamente (una columna por grupo) en lugar de usar "etiquetas" o factores, como en R y Python. (ver Tabla 2)

Tabla 2. Resumen de Diferencias Críticas

Aspecto	**R**	**Python (statsmodels)**	**Octave**
Entrada de Datos	Dataframes con Factores	Dataframes de Pandas	Matrices Numéricas
Sintaxis de Modelo	Fórmulas (y ~ x)	Fórmulas (strings)	Argumentos de función
Salida por defecto	Tabla ANOVA completa	Objeto modelo (requiere print)	Gráfica interactiva + Tabla

Aspecto	**R**	**Python (statsmodels)**	**Octave**
Post-Hoc (Tukey)	TukeyHSD(modelo)	statsmodels. Stats. multicomp	multcompare(stats)

Capítulo II

El uso de lenguajes de programación R y Python en ingeniería

En la actualidad, la ingeniería está fuertemente influida por los avances tecnológicos y la disponibilidad de datos. En este contexto, los lenguajes de programación son fundamentales para los ingenieros que desean optimizar procesos, realizar análisis complejos y desarrollar soluciones innovadoras. La habilidad de programar permite automatizar tareas repetitivas, modelar y simular sistemas, analizar grandes cantidades de datos y crear algoritmos que aumenten la eficiencia y la precisión en la toma de decisiones.

Entre los diversos lenguajes de programación disponibles, R y Python se destacan como dos de los más utilizados en el ámbito de la ingeniería. Cada uno de estos lenguajes tiene características particulares que lo hacen adecuado para distintos tipos de aplicaciones y desafíos. R, por ejemplo, es ampliamente reconocido por sus capacidades en análisis estadístico y visualización de datos, lo que lo convierte en una opción preferida en áreas como la ingeniería estadística y la investigación de operaciones. Por otro lado, Python ha ganado popularidad debido a su versatilidad y facilidad de uso, lo que lo convierte en una herramienta ideal para el desarrollo de software, la automatización de tareas y aplicaciones de inteligencia artificial y de aprendizaje automático.

La elección entre R y Python se ha convertido en una cuestión fundamental para ingenieros, analistas de datos y científicos. Ambos lenguajes tienen características únicas que los hacen adecuados para distintas aplicaciones en el ámbito de la ingeniería. Para comprender mejor estas diferencias, es esencial explorar la historia y la evolución de cada lenguaje, así como sus características distintivas y sus aplicaciones en el análisis de datos (Sharma et al., 2021).

R es un lenguaje de programación y un entorno de software diseñados principalmente para el análisis estadístico y la visualización de datos. Su origen se remonta a 1993, cuando Robert Gentleman y Ross Ihaka, en la Universidad de Auckland, Nueva Zelanda, comenzaron a desarrollarlo como una implementación del lenguaje S, creado en los años 70. R ha evolucionado gracias a su comunidad, que aporta paquetes y extensiones para ampliar sus funciones. Hoy en día, R es ampliamente reconocido en el ámbito académico y entre los profesionales de la estadística y constituye una herramienta esencial para el análisis complejo y el modelado de datos.

Por otro lado, Python fue creado a finales de los años 80 por Guido van Rossum y lanzado por primera vez en 1991. Desde entonces, Python ha crecido en popularidad debido a su simplicidad y legibilidad, lo que lo convierte en un lenguaje accesible para programadores de todos los niveles. Con el tiempo, Python ha evolucionado para incluir potentes bibliotecas para la ciencia de datos, como NumPy, pandas y Matplotlib, lo que facilita su uso en diversas aplicaciones de ingeniería, desde el análisis de datos hasta el desarrollo de software.

Una de las principales diferencias entre R y Python radica en su enfoque y diseño. R está diseñado específicamente para el análisis estadístico y la visualización de datos, lo que lo convierte en la elección preferida para estadísticos y analistas de datos que requieren herramientas avanzadas para este propósito. En contraste, Python es un lenguaje de propósito general que se utiliza en una variedad de dominios, incluyendo el desarrollo web, la automatización y, por supuesto, la ciencia de datos. Esta versatilidad lo hace atractivo para ingenieros que buscan una herramienta que les permita abordar una amplia gama de problemas.

Otra diferencia notable es la forma en que cada lenguaje gestiona la manipulación de datos. R ofrece un enfoque más orientado a los datos, con funciones que permiten realizar análisis estadísticos de forma directa y eficiente. Python, con bibliotecas como pandas, también permite una manipulación de datos eficaz, pero su estilo es más programático y

orientado a objetos, lo que puede resultar más familiar para los ingenieros con experiencia en programación.

R es ampliamente utilizado para análisis estadístico, gracias a su amplio conjunto de funciones y paquetes dedicados a métodos estadísticos avanzados. Los ingenieros que trabajan en campos como la investigación y el desarrollo pueden aprovechar R para realizar análisis de regresión, pruebas de hipótesis y modelos de series temporales, lo que facilita la interpretación de datos complejos.

Python, con su biblioteca pandas, se ha convertido en un estándar para la manipulación de datos. Su capacidad para manejar grandes conjuntos de datos de manera eficiente y su sintaxis intuitiva permiten a los ingenieros realizar tareas de limpieza, transformación y análisis de datos con eficacia. Esto es especialmente útil en proyectos de ingeniería donde los datos pueden provenir de múltiples fuentes y requieren un procesamiento significativo antes de ser analizados.

La visualización de datos es otro aspecto crítico del análisis. R sobresale en este campo gracias a librerías como ggplot2, que proporciona herramientas poderosas para crear gráficos estadísticos de alta calidad. Python, por su parte, también ofrece bibliotecas robustas como Matplotlib y Seaborn, que permiten crear visualizaciones atractivas y personalizables. La elección entre R y Python para la visualización a menudo depende de la preferencia del usuario y del contexto del proyecto (Oviedo et al., 2024).

Tanto R como Python tienen ventajas y desventajas en el ámbito de la ingeniería. La elección del lenguaje más adecuado dependerá de las necesidades específicas del proyecto, así como de la experiencia y las preferencias del ingeniero. El lenguaje de programación R ha ganado una considerable popularidad entre los ingenieros debido a su robustez en el análisis estadístico y su capacidad para manejar grandes conjuntos de datos.

R fue diseñado desde sus inicios con un enfoque fuerte en la estadística, lo que lo convierte en una herramienta ideal para ingenieros que necesitan realizar análisis complejos y modelado estadístico. Su extensa

colección de paquetes, como `lm()` para regresión lineal y `glm()` para modelos lineales generalizados, permite a los ingenieros desarrollar modelos predictivos y realizar inferencias estadísticas de manera eficaz. Además, R ofrece potentes funciones de análisis multivariante, esenciales para evaluar múltiples variables y sus interacciones en proyectos de ingeniería.

El análisis de riesgo es una parte crucial del proceso de toma de decisiones en ingeniería y R proporciona herramientas valiosas para realizar simulaciones. Con paquetes como `MonteCarlo`, los ingenieros pueden ejecutar simulaciones de Monte Carlo para modelar la incertidumbre y evaluar el impacto de distintas variables en sus proyectos. Esto es especialmente útil en campos como la ingeniería civil y la financiera, donde las decisiones deben basarse en la evaluación precisa de riesgos potenciales. R cuenta con una amplia gama de paquetes específicos, particularmente útiles en el ámbito de la ingeniería. Algunos de los más destacados incluyen:

- *ggplot2:* Fundamental para la visualización de datos, permite a los ingenieros crear gráficos complejos de manera sencilla y efectiva. La capacidad de personalizar las visualizaciones es crucial para presentar los resultados de manera clara y comprensible.

- *dplyr:* Este paquete facilita la manipulación de datos, permitiendo a los ingenieros de datos limpiar y transformar conjuntos de datos de manera eficiente. Las operaciones como seleccionar, filtrar y agregar datos se vuelven intuitivas y rápidas con `dplyr`.

- *caret:* En el ámbito del machine learning, el paquete `caret` ofrece una serie de funciones que permiten construir modelos predictivos y realizar validaciones cruzadas de manera sistemática, lo cual resulta invaluable para ingenieros que buscan implementar técnicas de inteligencia artificial en sus proyectos.

R se ha consolidado como una herramienta esencial en el arsenal de los ingenieros, ofreciendo capacidades avanzadas en análisis estadístico, simulación y manipulación de datos. Su comunidad activa y la evolución

continua de sus paquetes aseguran que R seguirá siendo relevante y útil en el ámbito de la ingeniería.

2.1 Uso de Python en ingeniería

Python ha emergido como uno de los lenguajes de programación más populares y versátiles en el ámbito de la ingeniería. Su sintaxis clara y sencilla, junto con una amplia gama de bibliotecas y herramientas, lo ha convertido en una elección preferida para diversas aplicaciones en este campo.

Una de las principales ventajas de Python es su capacidad para facilitar el desarrollo de software. Los ingenieros pueden utilizar Python para crear aplicaciones que automatizan tareas repetitivas, lo que no solo ahorra tiempo, sino que también minimiza los errores humanos. Por ejemplo, en la ingeniería civil, Python puede utilizarse para automatizar cálculos complejos y generar informes de manera rápida y eficiente. Además, su integración con herramientas de control de versiones, como Git, permite a los equipos de ingeniería colaborar de manera efectiva en proyectos de software, lo que mejora la productividad y la calidad del producto final.

Python se ha consolidado como el lenguaje de referencia para el desarrollo de aplicaciones de machine learning e inteligencia artificial. Con bibliotecas como TensorFlow, Keras y scikit-learn, los ingenieros pueden construir modelos predictivos y sistemas inteligentes capaces de aprender a partir de datos y tomar decisiones de forma autónoma. En la ingeniería, esto se traduce en aplicaciones que van desde la optimización de procesos industriales hasta el análisis predictivo en el mantenimiento, donde las máquinas pueden predecir fallas antes de que ocurran, lo que ahorra tiempo y costos significativos (Mehta, 2019). La riqueza del ecosistema de Python se refleja en la gran cantidad de bibliotecas disponibles para diversas disciplinas de la ingeniería. Algunas de las más destacadas incluyen:

- NumPy: fundamental para el cálculo numérico, permite realizar operaciones sobre arrays y matrices de manera eficiente y es esencial para la manipulación de datos en ingeniería.
- Pandas: Ideal para la manipulación y el análisis de datos; facilita la limpieza y la transformación de conjuntos de datos, lo cual es crucial para el análisis de resultados experimentales.
- Matplotlib y Seaborn: Estas bibliotecas son esenciales para la visualización de datos, permitiendo a los ingenieros crear gráficos informativos y atractivos que facilitan la interpretación de los resultados.
- SciPy: Proporciona herramientas para la optimización, la integración y la resolución de ecuaciones diferenciales, siendo particularmente útil en campos como la ingeniería mecánica y la eléctrica.
- Flask y Django: son frameworks de desarrollo web que permiten a los ingenieros crear aplicaciones web robustas para la visualización de datos y la interacción con los usuarios.

Python no solo permite a los ingenieros desarrollar software y automatizar procesos, sino que también les ofrece un conjunto de herramientas potentes para abordar problemas complejos mediante machine learning y análisis de datos. Su versatilidad y facilidad de uso lo han consolidado como un pilar fundamental de la ingeniería moderna.

La elección entre R y Python en el ámbito de la ingeniería no es sencilla, ya que ambos lenguajes ofrecen ventajas significativas y se adaptan a diferentes necesidades y contextos. R, con su enfoque en el análisis estadístico y la visualización de datos, se ha consolidado como una herramienta esencial para ingenieros que requieren modelar fenómenos complejos y realizar análisis profundos. Su vasta colección de paquetes especializados permite a los usuarios realizar estudios estadísticos avanzados y generar gráficos de alta calidad, lo que la convierte en una opción preferida para quienes trabajan en campos como la investigación y el análisis de datos.

Por otro lado, Python se destaca por su versatilidad y facilidad de uso, lo que lo convierte en una opción atractiva para ingenieros que buscan

desarrollar software, automatizar procesos y aplicar técnicas de machine learning. Su sintaxis clara y su potente ecosistema de bibliotecas, como NumPy, Pandas y Scikit-learn, permiten la manipulación eficiente de datos y la implementación de algoritmos de inteligencia artificial. Esto lo hace especialmente valioso en sectores en los que la integración entre el análisis de datos y el desarrollo de software resulta crucial.

Al final, la elección entre R y Python debe basarse en las necesidades específicas del proyecto y en las habilidades del equipo. Para quienes se centran en la estadística y la visualización, R puede ser la mejor opción. En cambio, quienes buscan una solución más general y versátil, que incluya desarrollo de software y machine learning, pueden encontrar en Python la herramienta ideal. En muchos casos, la combinación de ambos lenguajes puede ofrecer una solución más completa y efectiva, aprovechando las fortalezas de cada uno para abordar los desafíos del ámbito de la ingeniería.

Por lo tanto, la decisión debe considerar tanto el contexto del trabajo como los objetivos a alcanzar, lo que permitirá una elección informada que maximice el potencial de los ingenieros en su campo. El uso de R y Python en ingeniería representa un cambio de paradigma. Tradicionalmente, la ingeniería estaba dominada por herramientas como MATLAB, Fortran o Excel. Hoy, la ingeniería moderna se basa en datos y aquí es donde entran estos dos gigantes. Mientras que en la ciencia de datos general compiten, en la ingeniería suelen desempeñar roles complementarios muy definidos.

a. Python: El "Ingeniero de Planta" (Versatilidad y Producción)

En ingeniería, Python se ha convertido en el estándar de facto para reemplazar MATLAB y automatizar tareas. Su fortaleza radica en su capacidad para interactuar con hardware, simuladores y sistemas web.

- Simulación y Elementos Finitos (FEA): Librerías como FEniCS o SfePy permiten resolver ecuaciones diferenciales complejas para análisis estructurales o de dinámica de fluidos sin pagar licencias costosas.

- Automatización de CAD: software como Rhino (con Grasshopper), Blender y FreeCAD utiliza Python para generar geometría paramétrica automáticamente.
- Mantenimiento Predictivo (IoT): Es el rey de leer sensores (Arduino/Raspberry Pi), procesar señales y predecir fallas en maquinaria mediante Machine Learning (scikit-learn).
- Librerías Clave:
    - SciPy: para optimización lineal, integración y procesamiento de señales.
    - NumPy: álgebra matricial de alto rendimiento.
    - Control: Alternativa a la "Control System Toolbox" de MATLAB.

b. R: El "Auditor de Calidad" (Estadística Rigurosa)

R no se usa típicamente para controlar un robot o diseñar un puente, sino para analizar la variabilidad de los procesos que construyen dicho puente. Es fundamental en ramas donde el riesgo y la incertidumbre deben modelarse con precisión matemática.

- Control de calidad (Six Sigma): R domina en la industria manufacturera en el diseño de gráficos de control, el análisis de la capacidad de proceso (C_p, C_{pk}) y el diseño de experimentos (DOE).
- Ingeniería Ambiental e Hidrología: Se utiliza ampliamente para modelar series temporales de lluvias, caudales y contaminantes (paquete hydroGOF).
- Confiabilidad (Reliability Engineering): Para calcular la vida útil de un producto mediante el análisis de Weibull, R ofrece paquetes específicos que superan las implementaciones básicas de Python.
- Librerías Clave:
    - qcc: Para gráficos de control de calidad.

- DoE.base: Para el diseño de experimentos.
- survival: Para el análisis de falla y de supervivencia de materiales.

c. **Comparativa por Disciplina de Ingeniería** (Ver Tabla 3).

Tabla 3. Comparativa de uso de python y R entre disciplinas

Disciplina	**Uso Principal de Python**	**Uso Principal de R**
Ing. Industrial	Optimización de rutas logísticas, simulación de eventos discretos (SimPy).	Control estadístico de procesos (SPC), Six Sigma, diseño de experimentos.
Ing. Mecánica / Civil	Cálculo numérico, elementos finitos, automatización de planos CAD.	Análisis de la fatiga de materiales y de la hidrología estadística.
Ing. Eléctrica / Electrónica	Procesamiento de señales, visión por computador, control de sistemas.	Análisis de ruido en señales, modelado estocástico de redes.
Bioingeniería	Procesamiento de imágenes médicas.	Bioinformática, ensayos clínicos, genética.

Estudio de caso # 1:

- *El escenario: Eres ingeniero de calidad en una fábrica de pernos. Necesitas monitorear si el diámetro de los pernos se mantiene estable o si la máquina se está descalibrando.*

Por qué R: La librería qcc es el estándar de oro. Genera gráficos de control con límites (LCL/UCL) y detecta automáticamente violaciones de reglas estadísticas (como "racha de 5 puntos subiendo").

R

Instalar librería si no la tienes: *install.packages("qcc")*

library(qcc)

1. Simulación de datos: 20 muestras de 5 pernos cada una

Los primeros 15 grupos son normales (media 10 mm)

Los últimos 5 grupos simulan una falla (la máquina se descalibra a 10.8 mm)

*diametros <- c(rnorm(15*5, mean=10, sd=0.5),*

*rnorm(5*5, mean=10.8, sd=0.5))*

Organizar en una matriz (20 filas x 5 columnas)

matriz_datos <- matrix(diametros, nrow=20, ncol=5)

2. Generar Gráfico de Control X-bar (Media)

R calcula automáticamente los límites de control superior e inferior

grafico <- qcc(matriz_datos, type="xbar", title="Control de Calidad - Diámetro de Pernos")

R marca en ROJO los puntos que violan las reglas estadísticas

Resultado: R te mostrará inmediatamente los puntos rojos al final, avisándote de que el proceso está "fuera de control".

Estudio de caso # 2:

- *El escenario: Diseñas la suspensión de un auto. Necesitas simular cómo oscila y se detiene el amortiguamiento del sistema después de atravesar un bache.*

¿Por qué Python? Resolver ecuaciones diferenciales (ODE) requiere métodos numéricos potentes. SciPy es una herramienta de ingeniería robusta para ello.Shutterstock

Python

import numpy as np

```
from scipy.integrate import odeint
import matplotlib.pyplot as plt
# 1. Definir la Ecuación Física (Masa-Resorte-Amortiguador)
def modelo(condiciones, t, k, m, c):
    x, v = condiciones  # x = posición, v = velocidad
    dxdt = v
    dvdt = -(c/m)*v - (k/m)*x  # Segunda ley de Newton: F = ma
    return [dxdt, dvdt]
# Parámetros físicos
k = 2.0  # Constante del resorte
m = 1.0  # Masa
c = 0.5  # Coeficiente de amortiguamiento (fricción)
# Condiciones iniciales: El resorte empieza estirado 1 metro, velocidad 0
condiciones_iniciales = [1.0, 0.0]
# Tiempo de simulación (de 0 a 20 segundos)
t = np.linspace(0, 20, 200)
# 2. Resolver la Ecuación Diferencial (El "Solver")
solución = odeint(modelo, condiciones_iniciales, t, args=(k, m, c))
# 3. Visualizar la Ingeniería
plt.figure(figsize=(10,5))
plt.plot(t, solucion[:, 0], 'b-', label='Posición (m)')
plt.title('Simulación de Amortiguamiento (Suspensión)')
plt.xlabel('Tiempo (s)')
plt.ylabel('Desplazamiento')
```

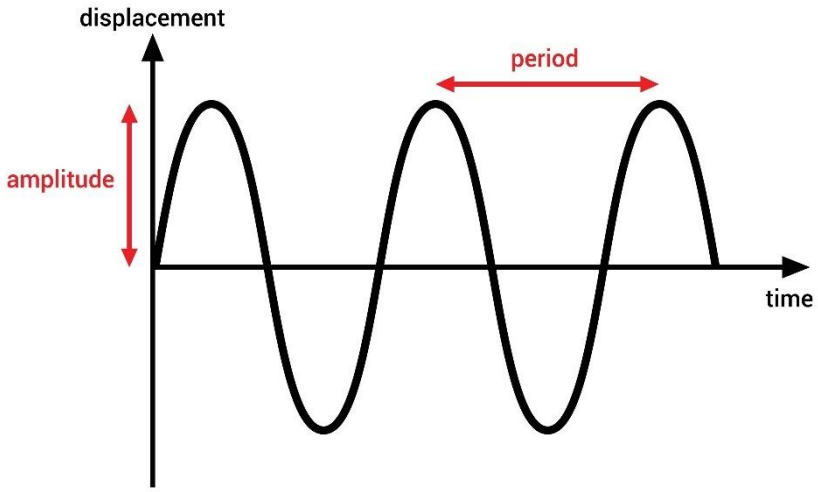
plt.grid(True)

plt.legend()

plt.show()

Resultado: Python genera una curva oscilatoria que se va aplanando, lo que muestra exactamente cómo la suspensión absorbe el impacto a lo largo del tiempo (Ver Figura 1).

Figura 1. Curva oscilatoria de sistema de amortiguación



El análisis estadístico moderno ha sido transformado por la accesibilidad a plataformas de software de código abierto. Python, R y GNU Octave representan tres pilares fundamentales, aunque con propósitos y nichos de aplicación marcadamente distintos. La elección de la herramienta más adecuada para un proyecto de ciencia de datos o de investigación rigurosa no es arbitraria; depende intrínsecamente del objetivo final, ya sea la inferencia científica, la visualización de alta calidad o la implementación a escala de producción (Nuñez et al., 2024).

2.2 Definición Conceptual y Roles Fundamentales de los Lenguajes

R es un entorno de software y un lenguaje de programación desarrollado específicamente para el cálculo estadístico y la visualización

de datos.[2] Su diseño inicial priorizó la facilidad para expresar conceptos estadísticos complejos y para manejar objetos de datos heterogéneos. Las capacidades de R se clasifican generalmente en tres grandes categorías: manipulación de datos, análisis estadístico y visualización de datos. Python se ha establecido como un lenguaje de programación de propósito general. Su atractivo principal radica en su versatilidad, lo que permite su uso en una amplia gama de soluciones más allá del análisis estadístico, como el desarrollo web, la automatización de sistemas y la ingeniería de *software*. Si bien Python ha adoptado el ecosistema de ciencia de datos (Pandas, SciPy, Scikit-learn), su diseño fundamental prioriza la funcionalidad general y la robustez en la construcción de sistemas de *software* escalables.

GNU Octave fue concebido como un lenguaje de alto nivel orientado principalmente a cálculos numéricos. Su objetivo principal es servir como alternativa de código abierto a MATLAB. Por esta razón, su nicho principal se ha centrado en la simulación de sistemas dinámicos, la resolución de problemas de álgebra lineal compleja y la generación de material didáctico en contextos de ingeniería. A diferencia de R y Python, Octave no fue diseñado con la inferencia estadística o la gestión de datos empresariales como foco central, lo que limita su utilidad en la ciencia de datos moderna (Companioni et al., 2012).

La elección entre Python y R a menudo se enmarca en una dicotomía: R para la investigación estadística y Python para la producción de *software*. Sin embargo, esta distinción requiere una comprensión matizada de dónde radica la eficiencia de cada lenguaje.

Es común escuchar la afirmación de que Python es inherentemente un lenguaje de propósito general superior, mientras que R se limita estrictamente a la estadística. Sin embargo, esta posición ha sido refutada activamente por académicos y científicos de datos avanzados. Profesionales que priorizan R destacan que han completado tareas de programación sumamente complejas con una eficiencia extraordinaria en R. Estas tareas incluyen el análisis de *Big Data* (gestionando decenas o incluso cientos de GB de datos sin procesar), tareas avanzadas de aprendizaje automático

(Machine Learning), bioinformática y la elaboración de informes profesionales dinámicos y rigurosos.

La ventaja innegable de Python en la operacionalización. La superioridad de Python no reside en su capacidad intrínseca para ejecutar un algoritmo estadístico (donde R a menudo destaca), sino en su rendimiento al escribir código de producción. Hay una diferencia fundamental entre la velocidad del ecosistema (la optimización de librerías específicas escritas en C o Fortran, común en R) y la velocidad del lenguaje base. Python aventaja a R en la velocidad del código base, con tipos fundamentales como listas y diccionarios, y en la iteración de cadenas letra por letra, siendo considerablemente más rápido que sus contrapartes en R.

Además, Python posee un sistema de programación orientada a objetos (OOP) que se considera superior y más robusto que los conjuntos de sistemas OOP de R (S3, S4, R6). Esta solidez arquitectónica es crítica cuando un análisis estadístico debe encapsularse, probarse e implementarse dentro de un sistema de *software* empresarial más amplio, lo que impulsa a Python como el lenguaje de elección para la fase de operacionalización. Por lo tanto, aunque R puede ser más eficiente para la exploración y el análisis estadístico, Python es más eficiente para la construcción de sistemas de software que incorporan ese análisis.

En síntesis, la posición marginal de GNU Octave en este debate se debe a su desempeño. Octave es notablemente más lento que MATLAB y no compite en velocidad ni en ecosistema con R o Python para tareas de computación general. Esta limitación, combinada con el riesgo de obsolescencia de sus paquetes estadísticos, lo relega a entornos muy específicos de simulación o de enseñanza.

La etapa inicial de cualquier proyecto estadístico es la limpieza y transformación de los datos, un proceso que revela las diferencias filosóficas y arquitectónicas clave entre R y Python.

El ecosistema R, en particular a través del conjunto de paquetes Tidyverse, promueve una sintaxis funcional declarativa que se ha convertido en el estándar *de facto* para la manipulación de datos. Utilizando

el operador *pipe* (%>% o |>), esta filosofía sintáctica permite encadenar operaciones de limpieza de manera altamente legible, simplificando tareas como la filtración (dplyr::filter), la creación de nuevas columnas (dplyr::mutate) o la selección de variables específicas (dplyr::select). El manejo de datos textuales también se beneficia de este enfoque, con paquetes como stringr que ofrecen funciones explícitas (p. ej., str_to_upper para convertir a mayúsculas o str_split_fixed para dividir cadenas).

Python, por su parte, basa la manipulación de datos en el *DataFrame* de Pandas, una estructura de datos orientada a objetos. Las transformaciones se realizan mediante la invocación de métodos sobre los objetos DataFrame o Series. Por ejemplo, las operaciones sobre cadenas se realizan con métodos explícitos sobre la propiedad .str de la columna (p. ej., cities['ctg1'].str.upper()) o para inspeccionar el contenido (.str.contains("f")). La elección entre el estilo idiomático y funcional de Tidyverse y el estilo de programación general y orientado a objetos de Pandas se reduce a la preferencia sintáctica y la familiaridad del analista.

La gestión eficiente de grandes volúmenes de datos es crucial para el análisis moderno. La superioridad de un lenguaje sobre el otro en este dominio depende de la escala y de la arquitectura de los datos. R está diseñado y optimizado para manejar eficientemente grandes cantidades de datos que caben en la memoria RAM de una sola máquina (*in-memory*). Esto se logra mediante paquetes altamente optimizados, como data.table, y el uso de técnicas específicas que aprovechan el código compilado. Para un analista que trabaja con decenas o cientos de gigabytes en un entorno de servidor potente, R ofrece soluciones muy veloces (Plasencia y Anías, 2017).

Sin embargo, cuando se considera el verdadero *Big Data*—conjuntos de datos que exceden el terabyte (TB) o se almacenan de manera distribuida—, Python, junto con su ecosistema, domina. Spark, una tecnología de procesamiento distribuido, se recomienda específicamente para *datasets* superiores a 1 TB, datos distribuidos en sistemas como HDFS, S3 o Azure, y flujos de trabajo que exigen clústeres distribuidos a escala empresarial.

Aunque Spark ofrece soporte para R, el núcleo de Spark está escrito en Scala, y su interoperabilidad es más fuerte con Python a través de PySpark. Además, Dask, una librería escrita completamente en Python, actúa como un componente clave dentro del ecosistema, permitiendo la mejora y paralelización de librerías nativas como NumPy y Pandas para la computación paralela y *out-of-core*. Por lo tanto, si el cuello de botella de un proyecto es la capacidad de la memoria RAM o la necesidad de acceder a datos distribuidos, Python y sus herramientas asociadas son la opción preferente.

La capacidad de conectarse y gestionar datos en bases de datos relacionales es una necesidad técnica en el entorno empresarial. El lenguaje SQL es fundamental para la gestión, actualización y eliminación de información en grandes bases de datos. Python demuestra una superioridad significativa en la capa de ingestión y conexión de datos. Permite la integración eficiente de sistemas y aplicaciones para obtener una vista integrada de los datos. Python facilita la conexión a tablas, la implementación de lógica de JOIN y la conformidad con las reglas de normalización de bases de datos, lo cual es vital cuando se manejan grandes volúmenes de datos (ESIC University, 2023).

En general, Python es más versátil en la recopilación de datos, ofreciendo un soporte más amplio para diversos formatos y una mayor facilidad para la extracción de datos web (*web scraping*) en comparación con R. Esta facilidad para la Extracción, Transformación y Carga (ETL) a menudo convierte a Python en la opción predeterminada para proyectos donde la ingesta y preparación de datos son etapas complejas. Si un proyecto requiere interactuar constantemente con APIs externas, recolectar datos web o gestionar grandes bases de datos SQL, la infraestructura más generalista de Python impulsa su adopción, incluso si el modelado posterior pudiera beneficiarse de R.

Para la estadística descriptiva y la inferencia basada (pruebas T, ANOVA, regresión lineal simple), ambos lenguajes, R y Python, son perfectamente competentes. El ecosistema de R, incluyendo su base y paquetes como stats, ofrece implementaciones históricamente las más

rigurosas, validadas y aceptadas en la literatura estadística. Al ser un lenguaje diseñado para la estadística, su sintaxis para estos modelos suele ser más limpia y directa (fórmulas de estilo *modelo~predictora*).

Python, por su parte, utiliza librerías robustas como statsmodels y SciPy para replicar la funcionalidad estadística tradicional de R. La principal ventaja de Python en este nivel es que estas herramientas están integradas en un entorno de programación de propósito general, lo que permite una transición fluida entre el análisis estadístico y otras tareas de *software*.

GNU Octave proporciona funciones estadísticas mediante su paquete statistics. Este paquete ofrece capacidades básicas de inferencia, como funciones de distribución acumulada (CDF) y de densidad de probabilidad (PDF) para distribuciones como Beta, F y Logística (betacdf, fcdf, logipdf, logiinv). También incluye funciones de regresión más avanzadas, como las clases de Modelos Aditivos Generalizados (GAM) mediante RegressionGAM.

A pesar de su capacidad de cómputo numérico, la utilidad de GNU Octave para el análisis estadístico profesional y riguroso es extremadamente limitada. Un factor crítico es el mantenimiento de su ecosistema. El repositorio Octave Forge, donde se alojan paquetes clave como statistics, ya no está activo. Para cualquier proyecto estadístico serio que requiera el uso de librerías actualizadas, seguras y validadas, depender de un entorno con desarrollo estancado introduce un riesgo significativo de obsolescencia y de errores metodológicos.

Octave se reporta como más lento que MATLAB y los usuarios que buscan una alternativa económica a MATLAB tienden a migrar directamente a Python. El coste de oportunidad de utilizar un entorno con un soporte tan limitado y una velocidad inferior a la de las alternativas resulta demasiado alto tanto para la investigación académica rigurosa como para el análisis de datos empresarial (Companioni et al., 2012). Su rol, por lo tanto, se mantiene firmemente anclado en la simulación de ingeniería o en el uso didáctico.

En los dominios de la estadística avanzada—particularmente la inferencia causal, el modelado jerárquico y la estadística bayesiana—los ecosistemas de R y Python muestran diferencias estilísticas y metodológicas cruciales. La inferencia causal busca determinar relaciones de causa y efecto, lo cual constituye una necesidad central en las ciencias sociales, la economía y la medicina. El ecosistema R tiende a liderar la implementación de metodologías de inferencia causal frecuentista y de econométrica. El paquete estimatr ofrece estimadores lineales rápidos y fáciles de usar, con énfasis en la robustez. Es esencial para diseños experimentales, ya que permite recuperar estimaciones que reflejan errores estándar robustos al clúster para diseños aleatorizados y de bloques.

Además, R alberga implementaciones clave de metodologías de vanguardia, como el *Double Machine Learning* (DML). El paquete DoubleML implementa el marco DML desarrollado por Chernozhukov. Este marco permite realizar inferencia válida en modelos causales (como modelos de regresión parcialmente lineal e interactiva) mediante métodos de Machine Learning para la estimación de componentes de molestia, así como mediante la ortogonalidad de Neyman y la división de muestras.

Python, impulsado por su comunidad de *Machine Learning* e Inteligencia Artificial, se enfoca en modelos causales basados en grafos y redes. Las redes bayesianas, accesibles en Python, constituyen una herramienta poderosa para modelar y razonar sobre la incertidumbre en diversos dominios. Estas redes son capaces de capturar relaciones complejas entre variables y facilitan la inferencia y la toma de decisiones (Vo, 2017). Los modelos causales basados en grafos, a través de librerías como CausalML, son esenciales para representar y analizar relaciones causales, lo que permite una comprensión más profunda de la interacción entre variables para la evaluación de intervenciones.

La presencia de los paquetes más rigurosos y consolidados para DML y estimación robusta, primero en R, se debe a que estos métodos fueron desarrollados inicialmente en la econometría y la estadística, campos que tradicionalmente han adoptado R. Los modelos multinivel (MLM) son

necesarios para el análisis de datos con estructura anidada o jerárquica (p. ej., estudiantes dentro de aulas, pacientes dentro de hospitales).

R es el entorno tradicional para el modelado de MLM, principalmente mediante el paquete lme4. Este ecosistema permite la especificación detallada de estructuras de varianza, como la adición de interceptos y pendientes aleatorios, incluyendo la correlación entre ellos, como se ilustra en un modelo de tres niveles (p. ej., hospital). R también facilita la interpretación de estos modelos, permitiendo graficar las interacciones de nivel cruzado (como la interacción entre la condición experimental y el tamaño del hospital) mediante librerías auxiliares.

Python y la aproximación jerárquica bayesiana. Python aborda los modelos multinivel con un fuerte sesgo hacia el marco bayesiano, utilizando librerías como PyMC. La construcción de modelos jerárquicos en PyMC ofrece una ventaja significativa al mejorar las estimaciones, especialmente en casos de muestreo desequilibrado.

La divergencia en la adopción de MLM muestra una elección estilística: los investigadores que necesitan la velocidad de estimación, la validación frecuentista tradicional y la adherencia a las pruebas de hipótesis clásicas eligen R. Los investigadores que requieren flexibilidad en la especificación del modelo, desean incorporar conocimiento previo o necesitan una cuantificación completa de la incertidumbre de los parámetros eligen Python/PyMC.

Un ejemplo típico de su uso incluye la realización de muestreos (p. ej., pm.sample) y la generación de información resumida (p. ej., pm.summary), que puede indicar rangos de probabilidad creíble para un parámetro (p. ej., un 95% de probabilidad de que el sesgo se encuentre entre 0.706 y 0.864). Aunque R cuenta con paquetes bayesianos (como Stan), la integración de PyMC con el amplio ecosistema de Machine Learning de Python ha posicionado a Python como el estándar preferido para el modelado probabilístico avanzado.

2.3 Visualización, Reportes y Reproducibilidad

La comunicación efectiva de los resultados estadísticos es tan importante como la solidez del análisis. Este proceso se apoya en la visualización y en herramientas para generar informes dinámicos. R sobresale en la visualización, ofreciendo una estética y rigurosidad mediante ggplot2, que implementa la *Grammar of Graphics*. Este marco permite crear gráficos de alta calidad para su publicación, lo que contribuye a la reputación de R por ofrecer una mayor capacidad para representar resultados.

La librería Plotly representa un punto de convergencia crucial, ya que es compatible con múltiples lenguajes, incluidos Python, R y MATLAB. Esto permite a los equipos estandarizar la interactividad de los reportes para la comunicación empresarial, independientemente del lenguaje de modelado utilizado (Mailund, 2017). Plotly se caracteriza por su sintaxis sencilla y su amplia oferta de tipos de gráficos. La reproducibilidad, en la que el código utilizado para generar un resultado forma parte del documento final, es un estándar científico y empresarial.

Tradicionalmente, R dominaba este campo mediante R Markdown. Sin embargo, la introducción de Quarto ha nivelado significativamente el campo. Quarto permite a los autores crear documentos y reportes completamente reproducibles al integrar código ejecutable en Markdown de múltiples lenguajes, incluidos Python, R, Julia y Observable.

Un documento Quarto puede contener bloques de código ejecutable en Python, asegurando que el código que produce la salida (como un gráfico generado con matplotlib) sea parte integral del documento y se vuelva a ejecutar automáticamente al renderizar. Esta capacidad de generar artículos, presentaciones y sitios web de calidad en HTML, PDF o MS Word permite a las organizaciones aprovechar la fuerza estadística de R y la capacidad de producción de Python dentro del mismo flujo de trabajo reproducible.

Octave, como alternativa a MATLAB, también posee capacidades de visualización. Sin embargo, estas se orientan a la simulación técnica de

sistemas dinámicos y al desarrollo de interfaces gráficas para material didáctico. Su enfoque difiere del análisis exploratorio de datos y de la visualización empresarial avanzada que caracterizan a R y Python. La Tabla 4 resume las herramientas clave de visualización y comunicación en los dos ecosistemas dominantes:

Tabla 4. Herramientas de Visualización y Comunicación

Aspecto	**R (Ecosistema)**	**Python (Ecosistema)**	**Ventaja Competitiva**
Gráficos Estáticos	ggplot2	Seaborn, Matplotlib	R: Rigurosidad y estética superiores mediante la gramática gráfica.
Gráficos Interactivos	Plotly, shiny	Plotly, Bokeh	Plotly: Estándar para interactividad multilingüe.
Informes Reproducibles	R Markdown, Quarto	Jupyter, Quarto	Convergencia: Quarto ha neutralizado la ventaja histórica de R en este dominio.

La capacidad de un lenguaje para escalar el análisis más allá de una única máquina e integrarse en la infraestructura de software de una organización es el factor decisivo en muchos contextos empresariales. Cuando los datos alcanzan la escala de *Big Data* (Petabytes) y requieren un procesamiento distribuido, el ecosistema de Python se vuelve dominante.

Apache Spark es la tecnología preeminente para el manejo de datos distribuidos en grandes *data lakes*. Como Spark está escrito en Scala y tiene una fuerte interoperabilidad con el ecosistema Apache, la vía más eficiente para su uso es a través de Python (PySpark). Spark es la mejor opción para manejar petabytes de datos, realizar consultas SQL y ejecutar cargas de trabajo que requieren *clústeres* distribuidos a escala empresarial.

Dask, escrito en Python, está diseñado para extender la funcionalidad de las librerías nativas de Python (como Pandas y NumPy) a la computación paralela y distribuida. A diferencia de Spark, un proyecto *all-in-one* que ha inspirado su propio ecosistema, Dask se integra y mejora el ecosistema de Python existente. Esta integración nativa facilita la transición a la computación distribuida para usuarios familiarizados con la stack de Python.

Aunque R tiene paquetes para interactuar con Spark (sparklyr), la infraestructura y la comunidad de soporte para Python en el dominio de la computación distribuida (Dask, PySpark) son significativamente más robustas, lo que consolida a Python como la herramienta de facto para proyectos de Big Data distribuidos. La principal ventaja arquitectónica de Python frente a R radica en su diseño como lenguaje de programación de propósito general (Grogan, 2018).

POO y velocidad de despliegue. Python sobresale en la escritura de código de producción. Su sistema de Programación Orientada a Objetos (OOP) es más maduro y más fácil de utilizar que los sistemas de objetos S3/S4/R6 de R. Esta ventaja de OOP permite a los ingenieros de software crear código mantenible, modular y escalable, lo cual es crucial para desplegar modelos estadísticos en producción.

Además, la mayor velocidad de los tipos de datos de base de Python (listas, iteradores, conjuntos) contribuye a un rendimiento superior en tareas que no están cubiertas por librerías estadísticas optimizadas en C. Esta velocidad del *código base* es esencial para el *scripting* general, la automatización y la creación de la infraestructura que rodea el modelo estadístico.

R y el despliegue. Aunque R ha desarrollado herramientas para el despliegue de modelos (como Shiny para aplicaciones web y Plumber para APIs), su arquitectura funcional y sus sistemas OOP se consideran menos eficientes y más difíciles de integrar en pipelines de *software* complejos que en Python.

El proceso de operacionalización actúa como un filtro final: incluso si R fue la herramienta más eficiente en la fase de exploración y modelado estadístico, la necesidad de un despliegue rápido, mantenible y escalable basado en una arquitectura robusta a menudo obliga a la transición a Python en la fase de producción. Esta tendencia en la industria valida la elección de Python como infraestructura de *software* a largo plazo.

La elección óptima entre R, Python y GNU Octave depende de una evaluación rigurosa del ciclo de vida del proyecto: desde la rigurosidad estadística y la exploración hasta la escalabilidad y el despliegue en producción.

2.4 Resumen de Fortalezas y Debilidades Específicas

R mantiene una ventaja histórica y metodológica en la inferencia estadística pura y en la visualización de alta calidad, mientras que Python domina la versatilidad, la integración empresarial y la escalabilidad del *Big Data* distribuido (Knaflic, 2020). GNU Octave, aunque es una herramienta de código abierto funcional para el cómputo numérico, está esencialmente excluida del análisis estadístico profesional moderno debido a sus limitaciones de velocidad, a la obsolescencia de su ecosistema de paquetes y a la falta de comunidad en la ciencia de datos (ver Tabla 5).

Tabla 5. Matriz de Desempeño Estratégico

Dominio	**R (Fortaleza)**	**Python (Fortaleza)**	**Octave (Nicho)**
Rigurosidad/Inferencia	Modelos lineales/MLM avanzados, DML frecuentista	Inferencia Bayesiana (PyMC), Redes Causales	Funciones de Distribución Base
Código de Producción	Menor velocidad base, OOP compleja	Tipos base rápidos, OOP superior,	Irrelevante (Lento y no mantenido)

Dominio	**R (Fortaleza)**	**Python (Fortaleza)**	**Octave (Nicho)**
		Integración API	
Escala Empresarial	Técnicas eficientes (*In-memory*/data.table)	Infraestructura distribuida (Spark, Dask)	Nulo
Visualización	ggplot2 (Estética superior)	Interactividad (Plotly), Versatilidad	Simulación Gráfica Didáctica

Las recomendaciones para la elección del software deben basarse en el objetivo principal del proyecto:

- Para el Científico de Datos Investigador (Foco en Inferencia de Alta Calidad): Se recomienda R. El acceso directo a métodos de vanguardia en inferencia causal, como DML (DoubleML), y la facilidad para obtener estimadores robustos (estimatr) lo convierten en el entorno más idiomático para el descubrimiento científico, la publicación académica y el análisis de diseños experimentales complejos.
- Para el ingeniero de machine learning o de producción (Foco en escala y despliegue): se recomienda Python. Su superioridad en la programación orientada a objetos, la velocidad de sus tipos base y la interoperabilidad con tecnologías de Big Data distribuidas, como Spark y Dask, garantizan la transición más eficiente del prototipo a la producción y la integración en flujos de trabajo de *software* empresarial.
- Para el Modelador de Sistemas Complejos (Foco en Incertidumbre y Estructura Jerárquica): Python es la elección óptima. El ecosistema

de PyMC y las herramientas para la inferencia causal probabilística (Redes bayesianas) son líderes en modelos complejos que requieren una visión jerárquica y una cuantificación detallada de la incertidumbre.

La creciente madurez de las herramientas de código abierto apunta a una tendencia clara: el futuro del cómputo estadístico profesional es el poliglotismo metodológico. La convergencia de las herramientas de reproducibilidad, en particular Quarto, ha eliminado la necesidad de adherirse dogmáticamente a un único lenguaje. Dado que Quarto soporta la ejecución de código en Python, R, Julia y otros lenguajes, la fase de comunicación de resultados ahora está desacoplada de la de cálculo. Esta nivelación significa que las organizaciones pueden aprovechar la fuerza estadística superior de R en las fases iniciales de exploración y modelado y, posteriormente, utilizar la fuerza ingenieril de Python para la escalabilidad y el despliegue.

En esencia, la mejor metodología estadística asistida por software es aquella que prioriza la selección de la herramienta más eficiente para cada etapa específica del proyecto, en lugar de intentar forzar un análisis complejo en un lenguaje que no está diseñado para ello. La coexistencia y la interoperabilidad, impulsadas por plataformas como Quarto y librerías como Plotly, definen el panorama actual.

Capítulo III

Métodos estadísticos asistidos con software: aplicaciones en Python, Octave y R

La investigación en sociología, ciencia política y economía ha experimentado una transformación fundamental, pasando del manejo de bases de datos pequeñas (encuestas y censos limitados) a la incorporación de grandes volúmenes de datos no estructurados (textos de redes sociales, registros administrativos masivos, datos georreferenciados, etc.). Esta evolución exige que los investigadores no solo dominen la inferencia estadística clásica (regresión, ANOVA, etc.), sino que también sean competentes en técnicas avanzadas de minería de datos (Data Mining) y de aprendizaje automático (Machine Learning). A pesar de esta necesidad, la formación y los recursos disponibles presentan dos deficiencias clave:

- Dependencia de Software Propietario: Gran parte de la literatura académica aún se enfoca en software comercial (p. ej., SPSS, Stata), cuya adquisición y uso están restringidos por costosas licencias institucionales. Esto limita la reproducibilidad de la investigación y el acceso a herramientas de vanguardia fuera de entornos privilegiados.
- Fragmentación Metodológica en Software Libre: Los recursos existentes sobre software libre tienden a ser exclusivos, dedicándose únicamente a R (el estándar estadístico y econométrico) o a Python (el estándar en Big Data y NLP). Esto obliga al investigador a una selección binaria, ignorando las fortalezas únicas de cada herramienta. Por ejemplo, R destaca en la publicación de resultados académicos y en el análisis de modelos de efectos mixtos, mientras que Python es insuperable en el procesamiento de lenguaje natural y en la integración con entornos de *web scraping*. La herramienta Octave, aunque menos común, sigue siendo fundamental para la

enseñanza y la aplicación de métodos numéricos y de álgebra matricial en la econometría formal.

Esta dependencia y fragmentación tienen consecuencias directas en la calidad y el alcance de la investigación en ciencias sociales:

- Limitación del Alcance: Los investigadores no pueden escalar sus análisis para incluir fenómenos complejos, como el análisis de redes sociales o la minería de texto, sin un conocimiento integrado de Python.
- Decisiones Subóptimas: La falta de una perspectiva comparativa lleva a usar una herramienta para tareas en las que otra resulta más eficiente, lo que limita la profundidad y la velocidad del análisis.
- Baja Reproducibilidad: La investigación queda sujeta a la disponibilidad de licencias y a la caducidad o a costos prohibitivos de las versiones de los programas.

Existe una clara necesidad de una obra que sirva como guía comparativa y práctica, que articule los métodos estadísticos esenciales de las ciencias sociales (desde la regresión lineal hasta las series temporales y la clasificación) y demuestre su aplicación directa y contrastada en los tres ecosistemas de software libre más relevantes: Python, Octave y R (Sarrel, 2017).

Este enfoque comparativo permitirá al investigador social moderno seleccionar el software óptimo para la pregunta de investigación específica, promoviendo la eficiencia y la escalabilidad del análisis y garantizando la transparencia y la reproducibilidad del proceso científico.

El uso de software para el análisis estadístico ha transformado radicalmente las ciencias sociales. Ya no se trata solo de encuestas y observaciones cualitativas; hoy en día, la sociología, la ciencia política y la psicología se apoyan fuertemente en la "Ciencia de Datos Social" (*Computational Social Science*).

Cada lenguaje tiene una "filosofía" distinta, adaptada a las necesidades del investigador social.

R: El Estándar Académico

R fue creado *por* estadísticos *para* estadísticos. Es, sin duda, la herramienta más potente para el análisis inferencial clásico y la visualización de datos complejos.

- Bibliotecas clave: tidyverse (manipulación de datos), ggplot2 (visualización), lme4 (modelos mixtos), psych (psicometría).
- Mejor uso: Análisis de encuestas, diseño experimental, modelos lineales jerárquicos y creación de gráficos listos para su publicación académica.

Python: La Navaja Suiza

Python es un lenguaje de propósito general. En ciencias sociales, ha ganado terreno por su capacidad para manejar "Big Data", realizar *web scraping* (extracción de datos de redes sociales) y procesar el lenguaje natural.

- Bibliotecas clave: pandas (tablas de datos), statsmodels (regresiones), scikit-learn (Machine Learning), NLTK/Spacy (análisis de texto).
- Mejor uso: minería de opiniones en Twitter/X, análisis de redes complejas, automatización de la recolección de datos y predicción de comportamientos mediante Machine Learning.

GNU Octave: El Matemático

Octave es la alternativa de código abierto a MATLAB. Su enfoque es matricial y numérico. Aunque es menos común en la sociología en general, es vital en econometría y sociología matemática.

- Enfoque: álgebra lineal numérica.
- Mejor uso: modelos de simulación, teoría de juegos y econometría avanzada en los que se requiere manipular matrices directamente en lugar de usar "cajas negras" estadísticas.

3.1 Aplicaciones Prácticas por Método Estadístico *(A partir de la experiencia de los autores)*

- Estadística Descriptiva y Visualización

Antes de modelar, necesitamos entender los datos.

- R: Insuperable en visualización. Permite crear mapas coropléticos (mapas de calor geográficos) para mostrar, por ejemplo, la distribución de la pobreza en una región.
- Python: excelente para gráficos interactivos (con Plotly o Seaborn) que pueden integrarse en páginas web para la divulgación científica.

- *Modelos de Regresión e Inferencia*

El núcleo de la investigación cuantitativa consiste en explicar la relación entre variables. Por ejemplo, una regresión lineal simple:

Y_i = \beta_0 + \beta_1 X_i + \epsilon_i

- R y Python (statsmodels): Ambos proporcionan salidas detalladas (valores p, R^2, intervalos de confianza) esenciales para validar hipótesis.
- Diferencia: R tiene más paquetes para "casos de borde" o técnicas estadísticas de nicho (p. ej., regresión logística multinomial con efectos aleatorios) que Python tarda a veces en implementar.

- *Procesamiento de Lenguaje Natural (NLP)*

Aquí es donde las ciencias sociales modernas brillan al analizar discursos políticos, entrevistas o noticias.

- Python: Es el líder indiscutible. Permite realizar análisis de sentimiento, modelado de temas (Topic Modeling con LDA) y clasificación de textos masivos que serían imposibles de leer manualmente.

- *Análisis de Redes Sociales (SNA)*

Estudio de cómo interactúan los agentes (personas, organizaciones).

- R (igraph) y Python (NetworkX): Ambos permiten calcular métricas de centralidad (quién es el más influyente en una red) y detectar comunidades (clústeres de personas que interactúan entre sí).

En las ciencias sociales modernas, la tendencia es el poliglotismo. Muchos investigadores recolectan y limpian datos con Python y luego realizan el análisis estadístico inferencial y las gráficas finales en R. El análisis de series temporales es fundamental en las ciencias sociales para estudiar fenómenos como la inflación, las tasas de criminalidad, la intención de voto o los flujos migratorios a lo largo del tiempo (Chancusig et al., 2024).

Para este ejemplo comparativo, utilizaremos el modelo clásico ARIMA (AutoRegressive Integrated Moving Average). Matemáticamente, buscamos ajustar un modelo:

y'_t = c + \phi_1 y'_{t-1} + \dots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q} + \varepsilon_t

Usaremos el famoso dataset AirPassengers (número mensual de pasajeros de aerolíneas, 1949-1960), disponible en ambos entornos. El objetivo es predecir los próximos 12 meses.

R brilla por su simplicidad en el análisis de series temporales gracias al paquete forecast. Su función auto.arima() es legendaria porque selecciona automáticamente los mejores parámetros (p,d,q) del modelo, lo que ahorra mucho tiempo al investigador.

R

```
# 1. Cargar librerías
library(ggplot2)
# 2. Cargar datos (viene incluido en R base)
data("AirPassengers")
# 3. Ajuste del modelo (Auto ARIMA)
# R busca automáticamente la mejor combinación de parámetros
```

4. Realizar la predicción (12 meses a futuro)

pronóstico <- forecast(modelo, h=12)

5. Visualización

autoplot es una función mágica que grafica datos históricos + predicción + intervalos de confianza

autoplot(pronostico) +

ggtitle("Predicción de Pasajeros (Modelo ARIMA)") +

ylab("Pasajeros") +

theme_minimal()

Lo destacado de R:

- Extremadamente conciso (menos líneas de código).
- Manejo nativo de objetos de series de tiempo (ts).
- La visualización con intervalos de confianza (la zona sombreada) es automática.

En Python, utilizamos pandas para la manipulación temporal y statsmodels para la econometría. Python requiere ser más explícito (definir las fechas, configurar el índice) y, en la librería estándar, a menudo exige que definas los parámetros del modelo manualmente, aunque existe una librería externa (pmdarima) que imita a R (enfoque estándar con statsmodels):

Python

1. Importar librerías

Import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.arima.model import ARIMA

from statsmodels. datasets import get_rdataset

```
# 2. Cargar y preparar datos
# Obtenemos el mismo dataset que en R
dataset = get_rdataset('AirPassengers').data
# En Python debemos crear explícitamente el índice de tiempo
dataset['date'] = pd.date_range(start='1949-01', periods=len(dataset), freq='M')
serie = dataset.set_index('date')['value']
# 3. Ajuste del Modelo
# En statsmodels estándar, solemos especificar el orden (p,d,q) manualmente
# (Aquí usamos (2,1,2) como ejemplo simplificado)
modelo = ARIMA(serie, order=(2,1,2))
modelo_fit = modelo.fit()
# 4. Realizar la predicción
prediccion = modelo_fit.get_forecast(steps=12)
media_pred = prediccion.predicted_mean
conf_int = prediccion.conf_int()
# 5. Visualización (Más manual que en R)
plt.figure(figsize=(10,6))
plt.plot(serie.index, serie, label='Histórico')
plt.plot(media_pred.index, media_pred, label='Pronóstico', color='red')
plt.fill_between(media_pred.index,
                 conf_int.iloc[:, 0],
                 conf_int.iloc[:, 1], color='pink', alpha=0.3)
plt.title("Predicción de Pasajeros con Python")
plt.legend()
```

plt.show()

Lo destacado de Python:

- Control total sobre el manejo de fechas con pandas.
- Sintaxis orientada a objetos (modelo.fit(), modelo.predict()), común en Machine Learning.
- Se integra mejor si luego quieres usar esos datos en una aplicación web o en un dashboard.

Si una serie no es estacionaria (su media o varianza cambia con el tiempo), las predicciones de un modelo estándar (como una regresión lineal o un modelo ARIMA básico) pueden ser espurias o falsas. Para verificarlo científicamente, usamos el test de Dickey-Fuller aumentado (ADF).

- *La Lógica del Test (Hipótesis)*

En términos estadísticos, planteamos el test así:

- Hipótesis Nula (H_0): La serie tiene una raíz unitaria (No es estacionaria).
- Hipótesis Alternativa (H_1): La serie no tiene raíz unitaria (Es estacionaria).

Si el p-valor (p-value) es menor que 0.05, rechazamos H_0.

p < 0.05 \rightarrow \text{¡Bien! La serie es estacionaria.

p > 0.05 \rightarrow \text{Problema. La serie NO es estacionaria.

En R, el paquete estándar para esto es tseries. La salida es un resumen de texto muy limpio, típico de los reportes académicos.

R

```
# 1. Cargar librería
# install.packages("tseries") si no la tienes
library(tseries)
# 2. Cargar datos de ejemplo (Pasajeros)
```

```
data("AirPassengers")


# 3. Ejecutar el test
resultado <- adf.test(AirPassengers)
# 4. Mostrar resultado
print(resultado)
# Interpretación en R:
# R te dirá algo como "p-value = 0.01" (o el valor correspondiente).
# También te advierte si el p-value es menor que el valor que puede imprimir.
```

En Python, statsmodels nos devuelve una tupla (una lista de números) cruda. No te da un texto explicativo, por lo que tú mismo debes formatear la salida para leerla cómodamente.

Python

```
import pandas as pd
from statsmodels. tsa. stattools import adfuller
from statsmodels. datasets import get_rdataset
# 1. Obtener datos
dataset = get_rdataset('AirPassengers').data['value']
# 2. Ejecutar el test
# adfuller devuelve: (adf_stat, p_value, lags, obs, critical_values, icbest)
resultado = adfuller(dataset)
# 3. Imprimir con formato legible
print(f'Estadístico ADF: {resultado[0]}')
print(f'p-valor: {resultado[1]}')
```

Interpretación Lógica:

if resultado[1] < 0.05:

print("Conclusión: La serie es ESTACIONARIA (Rechazamos H0)")

else:

print("Conclusión: La serie NO es estacionaria (No rechazamos H0)")

El p-valor será alto (> 0.05). Esto indica que la serie de pasajeros NO es estacionaria (obvio, porque la cantidad de pasajeros crece cada año).

En R: Verás p-value = 0.99 (o similar).

En Python: Verás 0.9918…

En síntesis, no se pueden introducir estos datos directamente en un modelo de regresión. Primero necesitas transformarlos. La técnica más común es la diferenciación (restar el valor de hoy del de ayer: \Delta Y_t = Y_t - Y_{t-1}).

En R, la función diff() está optimizada para objetos de tipo ts. Es directa y maneja automáticamente los índices de tiempo.

R

1. Cargar datos

data("AirPassengers")

2. Aplicar Diferenciación (Lag = 1)

Esto resta: Y(t) - Y(t-1)

pasajeros_diff <- diff(AirPassengers)

3. Visualización comparativa

par(mfrow=c(2,1)) # Dividir pantalla en 2 filas

plot(AirPassengers, main="Original (Con Tendencia)", col="black")

plot(pasajeros_diff, main="Diferenciada (Estacionaria)", col="blue")

4. (Opcional) Verificar con ADF de nuevo

```
library(tseries)

print(adf.test(pasajeros_diff))

# Ahora el p-value debería ser mucho más bajo (probablemente < 0.05)
```

En Python, usamos el método .diff() de los DataFrames o de las Series de Pandas. Al diferenciar, el primer dato se convierte en NaN (Nulo) porque no tiene un mes anterior con el que restarse. Debes eliminarlo antes de graficar o de hacer tests.

Python

```
import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.datasets import get_rdataset

from statsmodels. tsa. stattools import adfuller

# 1. Cargar datos

dataset = get_rdataset('AirPassengers').data

# Crear índice de fechas (necesario para gráficos limpios)

dataset['date'] = pd.date_range(start='1949-01', periods=len(dataset), freq='M')

serie = dataset.set_index('date')['value']

# 2. Aplicar Diferenciación

# .dropna() es CRUCIAL para eliminar el primer valor nulo

serie_diff = serie.diff().dropna()

# 3. Visualización

plt.figure(figsize=(10, 8))

plt.subplot(2, 1, 1)

plt.plot(serie, color='black')

plt.title('Original (Con Tendencia)')
```

```
plt.subplot(2, 1, 2)
plt.plot(serie_diff, color='blue')
plt.title('Diferenciada (Estacionaria)')
plt.tight_layout()
plt.show()
# 4. Verificación rápida
p_val = adfuller(serie_diff)[1]
print(f"Nuevo p-valor: {p_val}")
```

Al ejecutar esto, notarás que la línea azul (diferenciada) ya no sube eternamente, sino que se mueve alrededor del 0. Sin embargo, en el caso específico de los pasajeros de avión, es probable que veas que la "oscilación" se hace más grande con el tiempo (heterocedasticidad) o que presenta un patrón muy claro cada 12 puntos (estacionalidad). Esta es la fase de "diagnóstico fino". Una vez que la serie es estacionaria (ya la diferenciamos), necesitamos saber cuánta "memoria" tiene el proceso para configurar los parámetros p (autorregresivo) y q (Media móvil) del ARIMA (Toomey, 2014). Para ello usamos dos gráficos complementarios:

- *ACF (Función de Autocorrelación):* Mide la relación total con periodos pasados. Ayuda a identificar el componente MA (q).
- *PACF (Autocorrelación Parcial):* Mide la relación directa con un periodo pasado, eliminando la influencia de los periodos intermedios. Ayuda a identificar el componente AR (p).

En R, la función ggtsdisplay() del paquete forecast es una de las favoritas de los científicos sociales. ¿Por qué? Porque genera un panel 3-en-1 que combina la serie de tiempo, el ACF y el PACF.

R

```
library(forecast)
# Asumimos que 'AirPassengers' ya está cargado
```

Aplicamos diff directamente dentro de la función para ver el diagnóstico de la serie estacionaria

ggtsdisplay(diff(AirPassengers),

main = "Diagnóstico de Serie Diferenciada (R)",

theme = theme_minimal())

Lo que verás:

- Un gráfico superior con la serie oscilando en torno a 0.
- Dos gráficos inferiores (ACF a la izquierda, PACF a la derecha) con bandas azules discontinuas.

En Python, importamos las funciones específicas plot_acf y plot_pacf. Aquí debemos generar los gráficos uno por uno o crear un "canvas" de matplotlib para ponerlos juntos.

Python

import matplotlib.pyplot as plt

from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

import pandas as pd

from statsmodels.datasets import get_rdataset

Recuperamos la serie y diferenciamos (como en el paso anterior)

dataset = get_rdataset('AirPassengers').data

serie = pd.Series(dataset['value'].values)

serie_diff = serie.diff().dropna()

Crear un entorno gráfico de 2 filas

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10, 8))

1. Gráfico ACF

'lags=40' nos deja ver hasta 40 meses atrás

plot_acf(serie_diff, ax=ax1, lags=40, title="Autocorrelación (ACF) - Para identificar q")

2. Gráfico PACF

plot_pacf(serie_diff, ax=ax2, lags=40, title="Autocorrelación Parcial (PACF) - Para identificar p")

plt.tight_layout()

plt.show()

Capítulo IV

La Brecha Estadística en Ciencias Sociales: Desafíos Metodológicos y Computacionales Frente al Análisis de Datos Moderno

La estadística ha desempeñado históricamente un papel primordial en el desarrollo de la sociedad moderna y en la investigación en ciencias sociales (CC.SS.), proporcionando herramientas metodológicas para el análisis de la variabilidad, la determinación de relaciones entre variables y la mejora de las predicciones en situaciones de incertidumbre. El objetivo intrínseco de la estadística, que contribuye al pensamiento racional, ha asegurado su inclusión en la mayoría de los programas de posgrado y diplomados en ciencias de la educación y en las ciencias sociales.

4.1 Contexto Histórico y Definición de la Enseñanza Estadística Tradicional (ETES)

La Enseñanza Tradicional de la Estadística Aplicada a las Ciencias Sociales (ETES) se ha caracterizado históricamente por un enfoque curricular centrado en la enseñanza de técnicas descriptivas y en los fundamentos del modelo lineal clásico. Esta formación a menudo prioriza la aplicación mecánica de fórmulas por encima del desarrollo de un razonamiento estadístico profundo. Un problema recurrente es que este enfoque plantea situaciones excesivamente simplificadas, en las que la resolución se limita a la aplicación de una fórmula y al cálculo de cuentas, sin requerir contextualización ni la interconexión de las conclusiones con un entorno de aplicación real. Como resultado, el currículo se ha mantenido, en gran medida, estático, a pesar de los avances metodológicos y tecnológicos.

La sociedad contemporánea exige una adaptación radical de la docencia en estadística. Es un imperativo que la enseñanza se adecúe a los tiempos actuales, adoptando enfoques prácticos que utilicen la tecnología para desarrollar habilidades efectivas para resolver problemas contextualizados. El mundo real en el que los investigadores de CC.SS. Operan es inherentemente más complejo y "mucho más multidimensional que un solo resultado estadístico".

El mantenimiento de un currículo estadístico simplificado tiene consecuencias directas y severas para la formación profesional. Cuando el currículo se limita a técnicas descriptivas y modelos lineales básicos, los graduados carecen de competencias computacionales y metodológicas avanzadas (como la inferencia causal o el uso de Big Data) que se exigen en la investigación y en el mercado laboral moderno. Esta deficiencia en la capacidad para manejar datos complejos y aplicar modelos avanzados expone a las ciencias sociales a perder su relevancia disciplinaria frente a campos más técnicos como la ciencia de datos, que están mejor equipados para analizar fenómenos sociales masivos y complejos. Esta situación conduce a una devaluación del título académico tradicional en el ámbito del análisis cuantitativo.

La enseñanza también debe fomentar la humildad científica, promoviendo el reconocimiento de las limitaciones inherentes a los estudios, a la estadística misma y a sus enfoques e interpretaciones, lo cual resulta motivador para enriquecer toda perspectiva con nuevos planteamientos. El análisis de datos en ciencias sociales está experimentando una profunda revisión de sus fundamentos, impulsada por las deficiencias conceptuales inherentes al enfoque tradicional.

El uso predominante del contraste de hipótesis basado en la prueba de significación de la hipótesis nula (NHST) y en el valor p ha sido el eje de la enseñanza tradicional. Este enfoque, aunque ampliamente aceptado, ha generado una intensa controversia en los últimos años debido a la baja credibilidad y la falta de reproducibilidad de diversos estudios (Cumming y Calin, 2024). La simple dicotomía de "estadísticamente significativo"

basada en el umbral de $p < 0.05$ ha sido señalada por sus múltiples falacias y malas interpretaciones.

Para subsanar esta crisis de rigor, las recomendaciones académicas modernas abogan por elevar el estándar de la evidencia. Se sugiere informar valores precisos de p acompañados de intervalos de credibilidad o de confianza (IC95%), así como de medidas contextualizadas, como el riesgo relativo, la razón de momios y el tamaño del efecto. Además, se ha propuesto, dentro del marco frecuentista, sustituir el umbral tradicional de $p=0.05$ por un estándar más estricto, como $p=0.005$, y considerar los valores intermedios ($0.005 < p < 0.05$) solo como "sugerentes".

Como se ha mencionado, los docentes no deben plantear situaciones excesivamente simplificadas que se resuelvan con la aplicación de una única fórmula. Para lograr un aprendizaje significativo, los problemas deben tener la mayor cantidad posible de "ingredientes" y pertenecer al área de aplicación del futuro ejercicio profesional del estudiante. Solo así los estudiantes transitan por las distintas etapas del proceso estadístico, lo que impulsa aprendizajes significativos y contextualizados, esenciales para el desarrollo de las competencias requeridas por la población investigadora.

Un elemento fundamental que diferencia el análisis moderno del tradicional es la integración de una perspectiva crítica. La educación estadística debe ir más allá de la mera técnica. Es crucial buscar propuestas conjuntas para integrar las dimensiones sociales, culturales y políticas en las prácticas educativas de la estadística, contribuyendo así a la formación de ciudadanos participativos.

La necesidad de esta perspectiva crítica resulta evidente al considerar el riesgo de la neutralidad ficticia en la formación. Al desvincular la enseñanza de la estadística de su aplicación y de sus consecuencias sociales, el currículo tradicional omite la necesidad de cuestionar el origen, la recolección y el uso ético de los datos. Esto es crítico en un entorno donde la información se utiliza con frecuencia con fines de lucro (publicidad) o de carácter político o partidario (elecciones), como se observa en la preocupación de algunas universidades públicas. Si la formación no integra esta dimensión ética y crítica, el resultado es la formación de investigadores

que no problematizan adecuadamente los sesgos algorítmicos, la privacidad de los datos o la equidad en el análisis, elementos cruciales en la era de la ciencia de datos. El análisis de datos moderno exige ir más allá de los modelos descriptivos y exploratorios, enfocándose en la inferencia rigurosa y en la modelización de la complejidad inherente a los fenómenos sociales.

La investigación social rigurosa, especialmente en la evaluación de políticas públicas o de intervenciones, debe centrarse en determinar el efecto causal que ejercen ciertas variables sobre los resultados de interés. Esto obliga a la enseñanza a trascender el mero estudio de la correlación.

Para manejar rigurosamente el análisis, incluso con datos observacionales, la formación debe incorporar los fundamentos de la inferencia causal. Esto incluye la comprensión de métodos avanzados para extraer conclusiones causales, como las técnicas de *Matching* y otros métodos de identificación causal como el análisis de Diferencias en Diferencias (DiD) o el uso de Variables Instrumentales (IV).

Una de las transiciones metodológicas más significativas es el creciente interés por la estadística bayesiana. Este enfoque es fundamentalmente un problema causal y es esencial para un manejo riguroso de la incertidumbre. La estadística bayesiana permite incorporar explícitamente el conocimiento previo sobre un fenómeno mediante las "distribuciones a priori", que se combinan con la evidencia de los datos para generar las "distribuciones a posteriori".

Las ventajas de la inferencia bayesiana incluyen la capacidad de calcular estimados (promedios, diferencias, riesgos) con mayor precisión, y sus intervalos de credibilidad ofrecen una mejor representación de la expectativa de las variaciones del estimado (Tovar, 2015). Ya existen implementaciones bayesianas para muchos modelos comunes en CC.SS., incluyendo ANOVA, regresión lineal y modelos loglineales.

No obstante, la adopción de la estadística bayesiana enfrenta una resistencia conceptual significativa. Aunque proporciona un paradigma metodológico superior para manejar la incertidumbre, su fundamento en la

probabilidad subjetiva suscita desconfianza entre algunos investigadores y clínicos. Se teme que, al ser subjetiva, carezca de fundamento matemático o no sea reproducible. Sin embargo, la subjetividad en este contexto no implica arbitrariedad; los investigadores pueden respaldar sus probabilidades con evidencia de estudios previos o con el consenso de expertos, siempre con argumentos teóricos bien fundamentados. Superar la falta de formación en probabilidad subjetiva es crucial para que las ciencias sociales aprovechen plenamente este marco metodológico.

Los fenómenos sociales son inherentemente complejos y rara vez pueden ser capturados por un modelo lineal simple.

- Modelos de Ecuaciones Estructurales (SEM): Son una forma efectiva de analizar fenómenos sociales, especialmente cuando se incluyen variables latentes o constructos que no son directamente observables. Estos modelos permiten estudiar relaciones causales a partir de información no experimental, utilizando variantes como el análisis factorial exploratorio y confirmatorio o el estudio de efectos mediadores.
- Métodos no paramétricos: Dada la prevalencia de datos sociales que no cumplen con los supuestos de normalidad o de escalas de intervalo (como datos ordinales o categóricos), la enseñanza debe destacar la relevancia de los modelos estadísticos no paramétricos. Estos métodos son robustos y deben priorizarse en los libros de texto universitarios para garantizar la solidez de los análisis ante supuestos débiles (Ver Tabla 6).

Tabla 6. Comparativa Metodológica: Tradicional vs. Moderna en Ciencias Sociales

Dimensión de Análisis	**Enfoque Tradicional (ETES)**	**Enfoque Moderno (Ciencia de Datos Sociales)**
Paradigma Inferencial	Frecuentista (NHST, P-valor como umbral)	Inferencia causal, bayesiana, frecuentista rigurosa

Dimensión de Análisis	Enfoque Tradicional (ETES)	Enfoque Moderno (Ciencia de Datos Sociales)
Foco de Investigación	Correlación, Descripción, Significación estadística básica	Inferencia Causal, Predicción (ML), Humildad Científica
Modelos Primarios	Regresión Lineal/Logística simple, ANOVA	SEM, Modelos Multinivel, Inferencia Causal (Matching, DiD), Machine Learning
Tipo de Datos Preferido	Estructurados, encuestas con supuestos de normalidad	Masivos (Big Data), no estructurados (Texto, Redes), no paramétricos robustos

4.2 La Obsolescencia del Software de Interfaz Gráfica (GUI)

Software propietario como SPSS y Stata siguen siendo utilizados en la enseñanza de posgrado. Si bien cumplen una función didáctica inicial, estos paquetes presentan limitaciones cruciales en términos de escalabilidad, automatización y, fundamentalmente, reproducibilidad. El análisis de datos moderno, especialmente ante la irrupción de Big Data, exige familiaridad con tecnologías de procesamiento masivo, como Hadoop y Spark, y con entornos de almacenamiento en la nube, como Google BigQuery o Amazon Redshift.

La dependencia de herramientas de interfaz gráfica de usuario (GUI) como SPSS no es solo una limitación metodológica; también implica un costo oculto para las instituciones. El alto costo de las licencias de software propietario restringe la escalabilidad del análisis y consume recursos financieros que podrían reorientarse a la formación docente o la infraestructura de Big Data (Wang, 2024). Esto perpetúa una brecha tecnológica y obstaculiza la investigación colaborativa, ya que el acceso a las herramientas avanzadas no está democratizado.

Los lenguajes de programación de código abierto R y Python se han convertido en estándares *de facto* para el análisis de datos y la ciencia de datos sociales.

- R: Es el lenguaje y entorno estadístico por excelencia, diseñado específicamente para la estadística y los gráficos. Quienes trabajan con estadística y análisis de datos usan R para administrar grandes volúmenes de datos con mayor facilidad, aplicando modelos avanzados de data mining y de aprendizaje automático.
- Python: Es un lenguaje de programación de propósito general, ideal para principiantes debido a su sintaxis lógica y accesible. Es crucial en el contexto de Big Data, el *machine learning* y la automatización. La formación en Python no solo enseña a programar, sino que también permite a los estudiantes de CC.SS. Enriquecer sus análisis, discernir la información, graficarla y, sobre todo, aprender a plantear las preguntas fundamentales que debe plantear un científico de datos.

El dominio de estos lenguajes se traduce en la capacidad de escribir código eficiente y mantenible, una habilidad clave para un analista de datos moderno. Las demandas computacionales modernas van más allá del mero análisis de encuestas. La ciencia de datos sociales requiere explorar métodos para recolectar y analizar datos sociales disponibles en Internet. Esto implica el uso de técnicas de procesamiento de lenguaje natural (PLN), análisis de redes sociales y *machine learning* (Sommerville, 2005). Los estudiantes deben estar familiarizados con herramientas de visualización específicas (como Voyant Tools o Gephi) y saber acceder a fuentes digitales de información social, así como formar una visión crítica sobre el potencial y las limitaciones de estos datos.

La falta de reproducibilidad es un problema endémico en la ciencia, exacerbado en las ciencias sociales. Por la inadecuada gestión de los flujos de trabajo. En el entorno de investigación tradicional, es común encontrar un caos de archivos con nombres vagos (datos_2_limpios.csv, manuscrito_1_revision_pepe.docx), lo que hace extremadamente difícil, si no imposible, reproducir un análisis años o incluso meses después. Para un

investigador, la necesidad de rehacer o revisar análisis estadísticos previamente realizados es constante.

La solución a la gestión ineficiente de archivos y a la trazabilidad de los cambios es la adopción del control de versiones, específicamente Git, y su uso centralizado en plataformas como GitHub. Estos sistemas permiten centralizar cuadernos de investigación, datos, programas de análisis y versiones de manuscritos, cada uno con un identificador único. Esto facilita el trabajo y la edición en paralelo entre colaboradores y rastrea todo el historial de cambios, garantizando la integridad científica de un proyecto.

4.3 Documentación Dinámica: RMarkdown y Jupyter

La documentación dinámica resuelve la brecha entre el código y el informe final. RMarkdown, por ejemplo, es un mecanismo basado en texto plano que permite la construcción de documentos con códigos y análisis de datos integrados.

A diferencia de los flujos de trabajo tradicionales que requieren exportar los resultados del software estadístico (como SPSS o Minitab) a un documento de texto (Word o LaTeX), RMarkdown permite construir el informe desde el mismo entorno de análisis (como RStudio). Los documentos generados son totalmente reproducibles, ya que integran texto narrativo y código y admiten múltiples lenguajes, como R, Python y SQL (Mangrulkar & Vijay, 2025).

La exigencia de reproducibilidad, con herramientas como RMarkdown y Git, se convierte en un filtro de calidad metodológica. Para que el código sea reproducible, el proceso analítico debe estar formalizado, lo que obliga al investigador a optimizar y clarificar su código y, a su vez, fortalece intrínsecamente el pensamiento crítico y el rigor lógico en el análisis (ver Tabla 7).

Tabla 7. Transición Computacional: Herramientas y Habilidades Clave

Categoría de Herramienta	Herramienta Tradicional Típica	Herramienta Moderna Requerida	Propósito Principal
Análisis Estadístico	SPSS, Minitab, Excel	R (RStudio, Tidyverse)	Modelado estadístico complejo, visualización avanzada
Análisis Big Data/General	N/A	Python (Pandas, Scikit-learn, Spark)	Procesamiento escalable, Machine Learning, automatización
Flujo de Trabajo/Reproducibilidad	Documentos de Word/LaTeX separados	RMarkdown, Jupyter Notebooks	Documentación dinámica e integrable de código y resultados
Colaboración y Trazabilidad	Email, almacenamiento local, pendrives	Git y GitHub	Control de versiones, colaboración rastreable

4.4 Propuestas de Modelos Curriculares Integrales y Transdisciplinares

La evidencia muestra una falta de concordancia entre los temas estadísticos tradicionalmente planteados y la metodología de la clase, con docentes que no incorporan la minería de datos ni los requerimientos estadísticos de la sociedad actual, como las competencias STEAM.

Se requiere implementar modelos contemporáneos, interdisciplinares y transdisciplinares para la enseñanza de la estadística. Estos modelos deben implicar que los conocimientos se descubran, se comprueben y se sustenten activamente mediante el uso de tecnologías, para que el estudiante pueda adaptarse a la realidad profesional. Algunas instituciones ya están liderando la transición hacia la ciencia de datos sociales.

- UNQ (Argentina): Ha lanzado un ciclo extracurricular de "Ciencia de datos para ciencias sociales" que integra cursos de Python básico y R. El objetivo pedagógico es enseñar a discernir la información y a plantear preguntas críticas, aplicando el conocimiento para construir indicadores, clasificar y realizar microsegmentación (microclusters) de datos.
- UC (Chile): Ofrece un curso intensivo enfocado en la "ciencia de datos sociales", centrado en aprender a obtener y analizar datos sociales a partir de Internet. El currículo explora métodos digitales, el procesamiento del lenguaje natural y el análisis de redes sociales. Estos programas buscan proporcionar a los estudiantes una visión crítica del potencial y de las limitaciones de la información a la que tienen acceso.

El enfoque pedagógico debe centrarse en el estudiante y en la práctica contextualizada. Se recomienda:

- Trabajo Basado en Proyectos: Utilizar la metodología de trabajo basada en proyectos vinculados a contextos reales y, en lo posible, a las áreas de aplicación del futuro ejercicio profesional del estudiante.
- Aprendizaje Contextualizado: Utilizar situaciones problemáticas basadas en datos reales provenientes de reportes de investigaciones educativas y artículos científicos.
- Tecnología y Entornos Invertidos: Generar entornos educativos invertidos (*flipped classroom*) para guiar y apoyar el uso de programas de cómputo libre, agilizando el procesamiento y el análisis de datos.

La integración de la Ciencia de Datos (que incluye Big Data, análisis de datos sociales digitales y Machine Learning) no puede ser puramente técnica; debe incorporar un fuerte componente ético. El aumento del acceso a datos masivos y a técnicas avanzadas incrementa el riesgo de sesgo algorítmico y de problemas de privacidad y seguridad. Por lo tanto, la formación debe enfatizar la necesidad de desarrollar protocolos para el uso adecuado de la información, haciendo de la ética de datos un eje transversal que asegure que los futuros analistas sociales aborden las consideraciones de equidad y transparencia.

La modernización curricular no es un proceso trivial; se enfrenta a obstáculos estructurales, tecnológicos y humanos. La adopción de la estadística moderna exige una infraestructura tecnológica adecuada. Las instituciones a menudo se enfrentan a barreras significativas en el aprendizaje y la participación, relacionadas con la infraestructura universitaria y el acceso a los servicios (Silva et al., 2024). La falta de capitalización de las oportunidades que ofrece el Internet y el espacio tecnológico (acceso al conocimiento, innovación) profundiza la brecha digital, lo que afecta a los ciudadanos, a los consumidores, a las empresas y al propio Estado.

La inercia curricular es frecuentemente sostenida por la resistencia docente. La resistencia al cambio es un comportamiento observable, complejo y multidimensional que surge como respuesta al desagrado o al desafío que sienten los docentes ante la introducción de nuevas ideas, métodos o dispositivos. Esta es una problemática de larga data e irresuelta en los sistemas educativos.

Para que los cambios en el currículo de estadística resulten exitosos, el liderazgo institucional es fundamental. La formación docente debe integrarse en las prioridades institucionales, generando un entorno apropiado con el apoyo de las autoridades de todos los niveles. Esto, junto con el entusiasmo genuino del profesorado, mejorará la satisfacción docente y consolidará el aprendizaje estudiantil.

Para superar la inercia, se requiere un estudio y la caracterización de la resistencia docente para formular estrategias efectivas que mejoren la

permeabilidad de las políticas educativas. Las estrategias deben ser lideradas por un cuerpo metodológico sólido. El estancamiento curricular se mantiene porque el cuerpo docente que diseña y enseña el currículo puede ser el mismo que se resiste a su actualización. Esto demanda una formación continua integral que aborde no solo las habilidades prácticas de programación (R y Python) sino también los fundamentos conceptuales de los nuevos paradigmas (p. ej., inferencia causal, modelado bayesiano).

Sin este liderazgo institucional que promueva la cocreación de planes de estudio inter y transdisciplinares, las reformas corren el riesgo de quedar como propuestas aisladas o cursos extracurriculares, perdiendo la oportunidad de posicionar a la institución como vanguardista en la formación de científicos de datos sociales.

La dependencia metodológica del paradigma frecuentista simplificado (NHST y el p-valor) ha conducido a una crisis de reproducibilidad, mientras que la dependencia computacional del software cerrado (GUI) restringe la capacidad de los graduados para manejar la complejidad inherente al Big Data y la inferencia causal rigurosa (Patriota, 2018; Valentine et al., 2015). La brecha entre la formación y la práctica moderna es, fundamentalmente, una brecha de rigor y reproducibilidad. Se requiere una hoja de ruta estratégica para alinear la formación estadística en CC.SS. Con las exigencias del análisis de datos del siglo XXI:

- Reingeniería Conceptual y Metodológica: El centro del currículo debe migrar del NHST hacia la Inferencia Causal y el Modelado Bayesiano. Esto implica priorizar el razonamiento estadístico crítico sobre la aplicación mecánica de pruebas, fomentando el uso de intervalos de credibilidad, del tamaño del efecto y de análisis robustos.
- Alfabetización Computacional Obligatoria: La programación debe integrarse desde los niveles introductorios. Es fundamental que los estudiantes adquieran competencias en R y Python y se familiaricen con la gestión de datos masivos (Big Data y SQL) y con la extracción de datos digitales.

- Estándares de Reproducibilidad como Eje Transversal: La investigación debe ser trazable. Se debe institucionalizar el uso de Git/GitHub para el control de versiones y de RMarkdown/Jupyter Notebooks para la documentación dinámica de los análisis.

Para garantizar la formación de profesionales capaces de abordar los problemas sociales contemporáneos con rigor y profundidad, se requiere la adquisición de las siguientes competencias mínimas (Ver Tabla 8)

Tabla 8. Requisitos Mínimos para la Competencia de Análisis de Datos en CC.SS. (Siglo XXI)

Área de Competencia	**Habilidad/Herramienta Específica**	**Propósito de Investigación**
Metodología Inferencial	Estadística Bayesiana, Modelos de Ecuaciones Estructurales (SEM)	Manejo riguroso de la incertidumbre, modelado de variables latentes.
Identificación Causal	Inferencia Causal (Matching, DiD), Pensamiento Crítico	Determinar los efectos reales de las políticas e intervenciones y evitar correlaciones espurias.
Computación/Programación	R with Python (Scripting, Tidyverse, Pandas)	Procesamiento escalable, automatización, creación de código eficiente.
Gestión de Datos	Big Data (Hadoop/Spark), SQL, APIs para datos sociales digitales	Recolección, almacenamiento y análisis de grandes volúmenes de datos no estructurados.

Área de Competencia	Habilidad/Herramienta Específica	Propósito de Investigación
Rigor y Transparencia	Git/GitHub, RMarkdown/Jupyter	Garantizar la reproducibilidad de los resultados y la colaboración trazable.

Conclusión

Este recorrido a través de los ecosistemas de software libre ha revelado una verdad fundamental: el análisis de datos moderno en ciencias sociales ya no es una disciplina de una sola herramienta, sino un arte de integración y rigor. La era de la Enseñanza Tradicional, marcada por la dependencia de la significación estadística (NHST) y el uso de *software* de interfaz gráfica (GUI) como SPSS o Stata, ha llegado a su fin, dejando paso a un imperativo de transparencia y sofisticación computacional.

La lección más importante extraída de esta obra es la necesidad de convertirse en un analista políglota. Hemos demostrado que la elección óptima del *software* es contextual y estratégica:

- R se mantiene como el bastión metodológico. Su diseño intrínseco para la computación estadística y su ecosistema de vanguardia (paquetes Tidyverse, *Double Machine Learning* y visualización de alta calidad) lo convierten en el entorno preferido para la inferencia, la exploración y la validación de modelos causales complejos.
- Python es el motor de la escalabilidad y de la producción. Su superioridad como lenguaje de propósito general, su robustez en la Programación Orientada a Objetos y su integración nativa con ecosistemas de *Big Data* (como PySpark y Dask) lo hacen indispensable para la ingesta de grandes volúmenes de datos, el *Machine Learning* y la implementación de soluciones en la infraestructura empresarial.
- GNU Octave, aunque valorado por su código abierto en el cómputo numérico, se encuentra marginado del análisis estadístico avanzado, lo que reforza la dicotomía R/Python en la ciencia de datos aplicada.

El cambio de herramienta, sin embargo, es un reflejo de un cambio metodológico más profundo. El análisis moderno abandona la simplicidad de la correlación espuria para abrazar la inferencia causal y la estadística bayesiana. Este cambio exige modelos más complejos (SEM, modelos

multinivel) y una profunda humildad científica para contextualizar los hallazgos en un mundo inherentemente multidimensional.

Finalmente, la integración de herramientas como Git/GitHub para el control de versiones y Quarto para la documentación dinámica no es meros complementos; es el estándar de rigor que garantiza la reproducibilidad de la investigación. Este estándar transforma el análisis de una caja negra en un proceso auditable, cumpliendo con la responsabilidad ética de asegurar la transparencia y la justicia de los algoritmos y resultados.

Al cerrar este libro, el lector no solo ha adquirido habilidades de *scripting* en tres lenguajes, sino que también ha internalizado una nueva filosofía de trabajo: la capacidad de seleccionar el estimador más riguroso, de integrarlo en el software más escalable y de documentar todo el proceso con una transparencia inquebrantable. Esta nueva competencia posiciona al científico social no solo como un analista de datos, sino también como un constructor de conocimiento robusto, preparado para descifrar la complejidad de la sociedad con precisión computacional y rigor metodológico. El futuro de las ciencias sociales y la toma de decisiones basada en la evidencia recaen en la adopción de este paradigma.

Bibliografía

Chancusig López, M. B., Yauli Chicaiza, G. E., López Castillo, G. de las M., Andrade Valencia, J. A., & López Velasco, J. E. (2024). Estadística Descriptiva Aplicada en Python para la Investigación Científica en Ciencias Sociales y Educativas: Descriptive Statistics Applied in Python for Scientific Research in Social and Educational Sciences. *Editorial SciELa*, *1*(1). https://doi.org/10.62131/978-9942-7221-6-4

Companioni Guerra, A., Cuesta Llanes, E., Hernández Blanco, Y., Orovio Cobo, V., & Días Ramos, S. (2012). Entorno integrado para el trabajo con GNU/Octave. *Revista Cubana de Ciencias Informáticas*, 6(4), 1-8

Cumming, G., & Calin-Jageman, R. (2024). *Introduction to the New Statistics : Estimation, Open Science, and Beyond.* (2nd ed.). Taylor & Francis Group

ESIC University (Diciembre 2023). *¿Qué es y para qué sirve el lenguaje de programación SQL?*. https://www.esic.edu/rethink/tecnologia/lenguaje-sql-que-es-c

Grogan, M. (2018). *Python vs. R for Data Science* (1st edition). O'Reilly Media, Inc

Knaflic, C. N. (2020). *Storytelling with data : let's practice!* (C. Madden, Ill.; 1st edition). Wiley

Mailund, T. (2017). *Beginning Data Science in R : Data Analysis, Visualization, and Modelling for the Data Scientist* (1st ed. 2017.). Apress. https://doi.org/10.1007/978-1-4842-2671-1

Mangrulkar, R. S., & Vijay Chavan, P. (2025). *Predictive Analytics with SAS and R : Core Concepts, Tools, and Implementation* (1st ed. 2025.). Apress. https://doi.org/10.1007/979-8-8688-0905-7

Martínez, M.J., y del Águila, I.M. (2025). *Programación en Python: Estadística a través de problemas resueltos.* Almería: Editorial Universidad de Almería

Mehta, P., Wang, CH, Day, AGR, Richardson, C., Bukov, M., Fisher, CK y Schwab, DJ (2019). Introducción al aprendizaje automático para físicos: alta varianza y alta precisión. *Physics Reports* , *810* , 1–124. https://doi.org/10.1016/j.physrep.2019.03.001

Mejía Peñafiel, E. F. (2022). R como herramienta de análisis y visualización de datos usando Inteligencia de Negocios y PowerBI. *AlfaPublicaciones*, *4*(3), 186–208. https://doi.org/10.33262/ap.v4i3.246

Nuñez Laje , E. A. ., Estrella de las Mercedes Baquero Tapia, Rengel Sandoval , H., & Noboa Ramírez , M. Y. (2024). El software de código abierto, democratizando el análisis estadístico en la ciencia de datos . *Revista Científica De Innovación Educativa Y Sociedad Actual "ALCON"*, *4*(5), 179–188. https://doi.org/10.62305/alcon.v4i5.361

Oviedo Rodríguez, K., Jiménez Oviedo, B., & Aguilar Fernández, E. (2024). Introduction to data visualization with ggplot2: Introducción a la visualización de datos con ggplot2. *Revista Digital: Matemática, Educación E Internet*, *24*(2). https://doi.org/10.18845/rdmei.v24i2.6914

Patriota, A. G. (2018). Is NHST logically flawed? Commentary on: "NHST is still logically flawed." *Scientometrics*, *116*(3), 2189–2191. https://doi.org/10.1007/s11192-018-2817-4

Perdigón Llanes, R., & Pérez Pino, M.T. (2022). Herramientas de código abierto para el análisis estadístico en investigaciones científicas. *Anales de la Academia de Ciencias de Cuba*, *12*(3). Recuperado a partir de http://scielo.sld.cu/pdf/aacc/v12n3/2304-0106-aacc-12-03-e1120.pdf

Plasencia Moreno, L., & Anías Calderón, C. (2017). Arquitectura referencial de Big Data para la gestión de las telecomunicaciones. *Ingeniare. Revista chilena de ingeniería*, *25*(4), 566-577. https://dx.doi.org/10.4067/S0718-33052017000400566

Raschka, S., Patterson, J. y Nolet, C. (2020). Aprendizaje automático en Python: principales desarrollos y tendencias tecnológicas en ciencia de datos, aprendizaje automático e inteligencia artificial. *Information*, *11* (4), 193. https://doi.org/10.3390/info11040193

Sarrel, M. D. (2017). *The state of data analytics and visualization adoption* (First edition.). O'Reilly Media

Sharma, K., Shetty, A., Jain, A. and Dhanare, R.K. (2021). "A Comparative Analysis on Various Business Intelligence (BI), Data Science and Data Analytics Tools". *International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, 1-11, https://doi.org/10.1109/ICCCI50826.2021.9402226

Silva-Cueva, J., Niño Cueva, D. C. E., Niño Cueva, M. R., Alanya-Beltran, J., & Calderón Pizango, M. R. (2024). Reformas curriculares en educación tecnológica: experiencias de adecuación desde la perspectiva de directivos y docentes . *Horizontes. Revista De Investigación En Ciencias De La Educación*, *8*(32), 119 –. https://doi.org/10.33996/revistahorizontes.v8i32.709

Sommerville, I. (2005). *Ingeniería del software.* Madrid: Pearson

Toalombo Montero, M. P., Toalombo Montero, O. W., Ballestero Torres, F. A., Hernández Dávila, C. A., & Ruiz Sarzosa, J. P. (2024). Visualización de Datos Educativos y Sociales con Python: Herramientas para Decisiones Informadas. *Editorial SciELa*, *1*(1). https://doi.org/10.62131/978-9942-7173-6-8

Toomey, D. (2014). *R for data science : learn and explore the fundamentals of data science with R* (1st ed.). Packt Publishing

Tovar Cuevas, J.R. (2015). Inferencia Bayesiana e Investigación en salud: un caso de aplicación en diagnóstico clínico. *Revista Médica de Risaralda*, *21*(1), 9-16

Valentine, J. C., Aloe, A. M., & Lau, T. S. (2015). Life After NHST: How to Describe Your Data Without "p-ing" Everywhere. *Basic and Applied Social Psychology*, *37*(5), 260–273. https://doi.org/10.1080/01973533.2015.1060240

Vo. T. H, P. (2017). *Python : data analytics and visualization : understand, evaluate, visualize data* (1st edition). Packt Publishing

Wang, Z. (2024). Law Case Teaching Combining Big Data Environment With SPSS Statistics. *International Journal of Web-Based Learning and Teaching Technologies*, *19*(1), 1–15. https://doi.org/10.4018/IJWLTT.334848

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org

De esta edición de *"Métodos estadísticos asistidos con software: Aplicaciones en Python, Octave y R"*, se terminó de editar en la ciudad de Colonia del Sacramento en la República Oriental del Uruguay el 06 de octubre de 2025