Modelos matemáticos

Una introducción con Python

Emmanuel Alcalá

ITESO

Presentación

- Emmanuel Alcalá:
    - Dr. en Ciencias del Comportamiento en UDG.
    - Profesor asociado en ITESO desde el 2021: Teoría de Juegos, Econometría, Análisis estadístico multivariado (Maestría en Ciencia de Datos).
    - Desde 2024 Data Analyst en HP Inc.
    - 12 publicaciones científicas en revistas indizadas.

Slides, datos y código en Github: Modelos matemáticos.

Programa general

Hora	**Duración**	**Tema y Actividad Principal**
08:00 - 08:15	15 min	Bienvenida e Introducción
08:15 - 09:15	90 min	Sección 1: ¿Qué es la Modelación Matemática?
09:45 - 10:30	45 min	Sección 2: Tipos de Modelos Matemáticos
10:30 - 10:35	5 min	Receso
10:35 - 11:00	25 min	Sección 3, Parte I: Instalación de Herramientas y Ejercicios
11:00 - 12:00	60 min	Sección 3, Parte II: El Proceso de Modelación
12:00 - 12:20	20 min	Receso Principal
12:20 - 13:20	60 min	Sección 4, Parte I: Fundamentos de Modelos probabilísticos
13:20 - 13:25	5 min	Receso
13:25 - 14:25	60 min	Sección 4, Parte II: Ajuste e Interpretación
14:25 - 15:00	35 min	Cierre, Preguntas y Discusión Final

Sección 1: ¿Qué es la Modelación Matemática?

Objetivo: Establecer una base conceptual y filosófica sobre qué es un modelo, por qué los usamos y cuáles son sus limitaciones.

Qué es un modelo

- Nociones convencionales sobre el concepto de “modelo”.

Qué es un modelo

- El modelo como abstracción simplificada de la realidad.

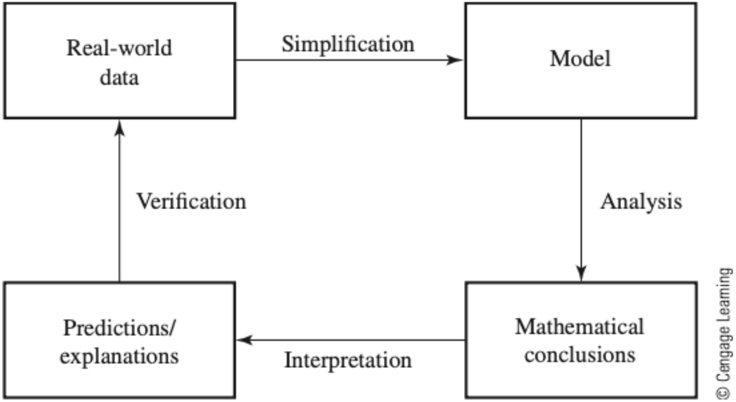


Figure 1: Relaciones entre realidad, modelo, interpretación.

- Analogía: los modelos son como *mapas*.
    - Los mapas no son el territorio.
    - Preservan ciertas relaciones.
    - Son más “simples”.
    - Su utilidad no está en que sean detallados.

Relación entre complejidad, utilidad y precisión

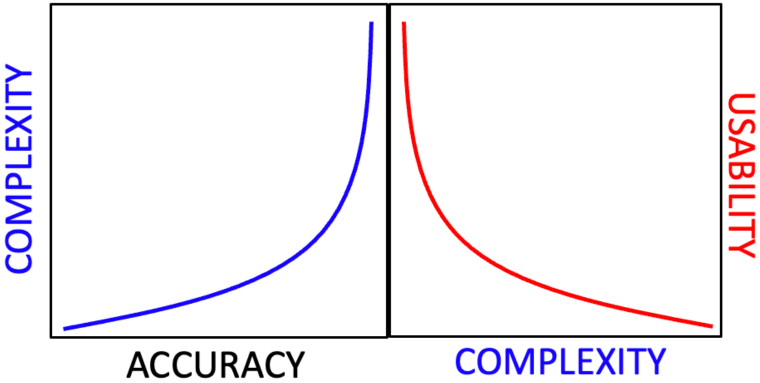


Figure 2: Cuanto más preciso, más complejo; cuanto más complejo, menos útil.

GPS como modelo: un mapa digital del mundo

- Fenómeno real:
  - Red de calles.
  - Topografía del terreno.
  - Estado del pavimento.
  - La ubicación de árboles, edificios, casas.
  - Aspectos cambiantes (tráfico, construcciones, etc).

¿Qué aspectos retirar y abstraer para convertirlo en un (buen) modelo?

- Una aplicación GPS no intenta replicar esa realidad.
- Crea una abstracción, un modelo matemático simplificado.
- La clave está en la simplificación deliberada: se omiten los detalles irrelevantes para el propósito de navegar.

Abstracción: grafo ponderado

El modelo del GPS es, en esencia, un grafo ponderado: Las calles son aristas y las intersecciones los nodos. Los "pesos" de las aristas son variables (distancia, el límite de velocidad, el tiempo de viaje estimado) que se actualiza con datos en tiempo real.

Realidad (Compleja)	**Modelo GPS (Abstracción)**
Calles con ancho, baches, múltiples carriles.	Líneas o arcos (vectores) en un grafo.
Intersecciones complejas con semáforos, señales.	Nodos que conectan las líneas.
Flujo de tráfico caótico y variable.	Atributos numéricos en las líneas (ej. velocidad promedio, tiempo de recorrido estimado).
Edificios, parques, topografía.	Ignorados/presentados como polígonos sin impacto en los cálculos de la ruta.

Todos los modelos son erróneos, pero algunos son útiles - *George Box*

El GPS como modelo fue construido con un propósito específico:

- **Encontrar una ruta óptima** de acuerdo a ciertas restricciones o parámetros entre un punto A y un punto B.
- Para lograrlo, se ejecutan algoritmos como el de Dijkstra sobre el grafo simplificado (computacionalmente intratable si se ejecuta sobre el mundo real).

¿Por qué es un modelo erróneo*? ¿Por qué es útil?*

- Es erróneo por diseño: no contiene al detalle el terreno, tráfico, etc., por lo que las predicciones no son exactas sino **aproximadas**.
- Pero es útil porque sus **aproximaciones** son mejores que nada, y además aunque no prediga con exactitud el punto y tiempo de llegada, nos permite llegar siempre (aproximadamente cerca, aproximadamente a tiempo). Es más, podríamos predecir incluso el margen de error.

Si ya tenemos la realidad ¿para qué sirve un modelo?

Si los modelos *no se supone que sean realistas y exactos*, ¿para qué sirven?

- Primero, para entender mejor algún aspecto del mundo (e.g., modelo de ADN).
- Segundo, para realizar predicciones sobre el mundo.
- Tercero, para probar una idea (que esa idea tiene una correspondencia con el mundo).
- ¿El modelo de GPS puede responder a esas preguntas?

Actividad (15 min)

Formar equipos de k tal que $n \bmod k = 0$. Elegir una app, objeto o representación que usemos de forma cotidiana (que no sea un mapa/GPS) que pueda entenderse como un modelo.

Discutir:

- ¿Qué sistema complejo de la realidad está representando o abstrayendo este modelo?
- ¿Cuál es su propósito principal? Es decir, ¿para qué fue diseñado y qué problema nos ayuda a resolver?
- ¿Qué simplifica o ignora deliberadamente de la realidad para poder ser útil?
- ¿En qué sentido es "incorrecto" o "impreciso"? ¿Cuáles son sus limitaciones y en qué situaciones podría fallar?

Contestando a la pregunta: ¿qué es un modelo matemático?

Una representación **abstracta** y **simplificada** de un sistema o fenómeno expresada en términos de un lenguaje formal (matemático). Su propósito es **entender, predecir** y/o **probar ideas de cómo funciona el mundo**.

- Que sea simplificado no significa que sea *trivial*. Las matemáticas pueden ser demandantes.
- No pretende ser una réplica de la realidad, sino una representación que captura ciertas relaciones e ignora otras.
- Qué ignorar y qué incluir requiere decisiones basadas en asunciones.
- Pueden mejorarse.

Sección 2: Tipos de modelos matemáticos

Objetivo: Presentar un panorama de las distintas clases de modelos según diferentes aspectos.

- El universo de los “modelos” es muy amplio.
- Incluye modelos físicos (ej. maqueta de un edificio, doble hélice del ADN) y modelos computacionales (ej. simulación del clima).
- Nos centraremos en un tipo: los modelos matemáticos.
  - Estos usan el lenguaje de las matemáticas (ecuaciones, funciones, lógica) para describir un sistema.
  - Hay sistemas tan complejos (ej. clima, dinámicas sociales) que es inviable describirlos con una sola ecuación. En su lugar, se usan modelos de simulación (como los basados en agentes) que aplican reglas matemáticas a miles de componentes individuales.
  - Los clasificamos en ejes o dicotomías para entender su naturaleza y elegir las herramientas correctas.

Determinístico vs. Estocástico

- **Determinístico**: No incluye aleatoriedad. Con las mismas condiciones iniciales, el resultado es siempre el mismo.
  - *Ejemplo:* La ley de enfriamiento de Newton,

$$T(t) = T_a + (T_0 - T_a)e^{-kt}$$

.

- **Estocástico (Probabilístico)**: Incorpora incertidumbre. El resultado puede variar en cada ejecución.
  - *Ejemplo:* Un modelo de caminata aleatoria para el precio de una acción.

Discreto vs. Continuo

- **Discreto**: El estado del sistema se evalúa en puntos o intervalos de tiempo específicos (días, años).
  - *Ejemplo:* Un modelo de crecimiento poblacional anual, $P_{n+1} = r \cdot P_n$.
- **Continuo**: El estado del sistema cambia constantemente en el tiempo, descrito por ecuaciones diferenciales.
  - *Ejemplo:* El decaimiento radioactivo,

$$\frac{dN}{dt} = -\lambda N$$

.

Mecanicista vs. Empírico

- **Mecanicista (Teórico)**: Se basa en los mecanismos o primeros principios del fenómeno. Intenta explicar el **porqué**.
  - *Ejemplo:* Modelo de órbitas planetarias basado en la Ley de Gravitación Universal.
- **Empírico (Fenomenológico)**: Se basa en ajustar una función a los datos observados, sin explicar el mecanismo. Describe el **qué**.
  - *Ejemplo:* Regresión que relaciona ventas de helados con la temperatura.

Lineal vs. No Lineal

- **Lineal**: Todas las relaciones entre variables son proporcionales (líneas rectas).
  - *Ejemplo:* La Ley de Ohm en un circuito simple ($V = IR$).
- **No Lineal**: Al menos una relación importante no es lineal. Son más realistas para sistemas complejos.
  - *Ejemplo:* El modelo depredador-presa de Lotka-Volterra.

Caso de estudio: crecimiento de una población de conejos

Modelo 1: Discreto y Determinístico

Idea: La población del próximo año es simplemente un 20% mayor que la de este año.

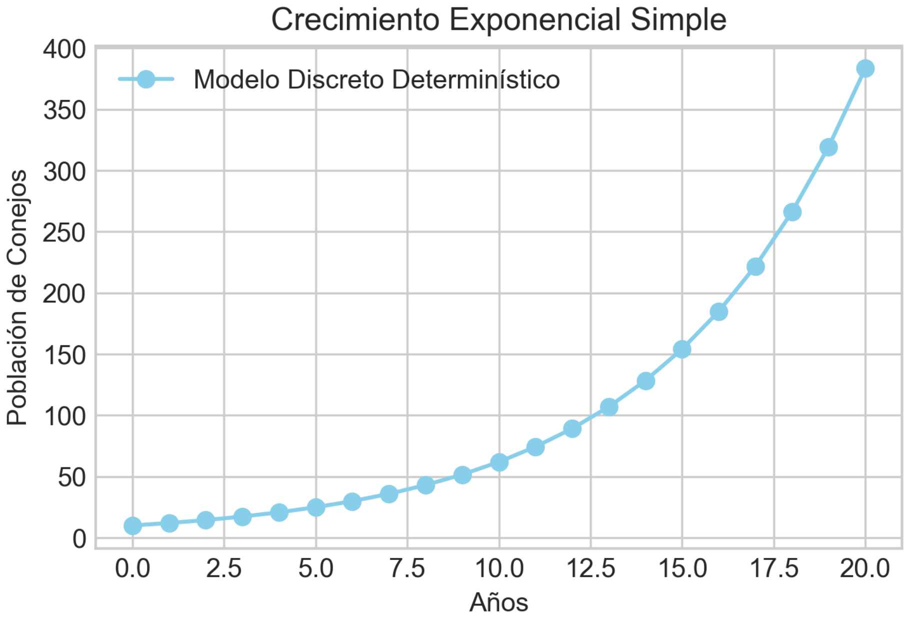
Modelo: Una relación de recurrencia simple.

$$P_{n+1} = 1.2 \cdot P_n$$

- P: Población
- n: Año.
- r: Tasa de Crecimiento

Clasificación:

- Discreto: Calculamos la población en pasos anuales.
- Determinístico: No hay aleatoriedad. El resultado es predecible.
- Mecanicista (simple): Se basa en una regla de reproducción.


Crecimiento Exponencial Simple
Modelo Discreto Determinístico
Población de Conejos
Años
0
50
100
150
200
250
300
350
400
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

- Predicción sencilla: Si en el año 0 comenzamos con 20 conejos, ¿cuántos tendremos en 20 años?

```python
P0 = 20
r = 0.2
n_years = 20
poblacion = P0 * (1 + r)**n_years
print(f"Población después de {n_years} años: {poblacion:.0f} conejos")
```

```
Población después de 20 años: 767 conejos
```

Modelo 2: Continuo y Determinístico

Idea: La población crece, pero está limitada por los recursos del entorno (una "capacidad de carga"). **Modelo**: La Ecuación Diferencial Logística.

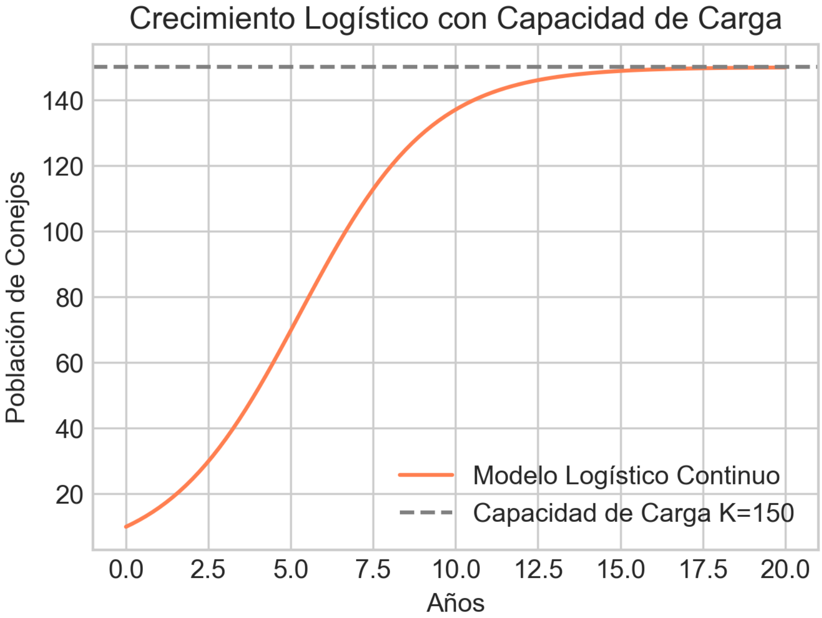
$$\frac{dP}{dt} = rP(1 - \frac{P}{K})$$

Donde:

- P: Población
- K : Capacidad de Carga
- r: Tasa de Crecimiento

Clasificación:

- Continuo: El cambio se modela en cada instante del tiempo.
- Determinístico: El comportamiento está definido por la ecuación.
- Mecanicista: Incluye un mecanismo (competencia por recursos, K).


Crecimiento Logístico con Capacidad de Carga
Población de Conejos
Años
140
120
100
80
60
40
20
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Modelo Logístico Continuo
Capacidad de Carga K=150

Modelo 3: Discreto y Estocástico

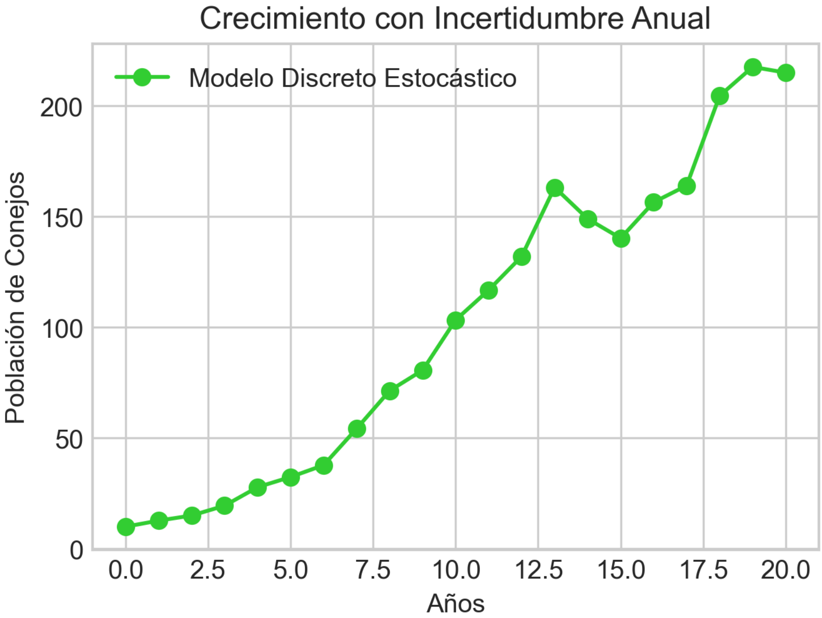
Idea: El crecimiento anual no es fijo; hay años buenos y malos debido a factores impredecibles (clima, enfermedades).

Modelo: El modelo discreto, pero con un factor de crecimiento aleatorio.

$$P_{n+1} = (1.2 + \epsilon) \cdot P_n, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Clasificación:

- Discreto: Pasos anuales.
- Estocástico: Incorpora incertidumbre explícitamente.
- Mecanicista (con ruido): La regla base es la misma, pero con variabilidad.


Crecimiento con Incertidumbre Anual
Modelo Discreto Estocástico
Población de Conejos
Años
0
50
100
150
200
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

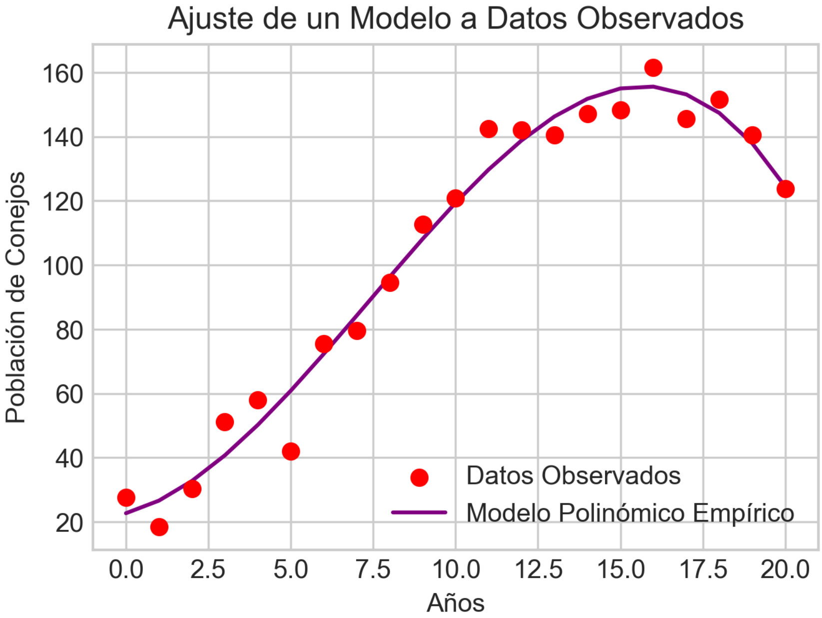
Modelo 4: Empírico (Basado en Datos)

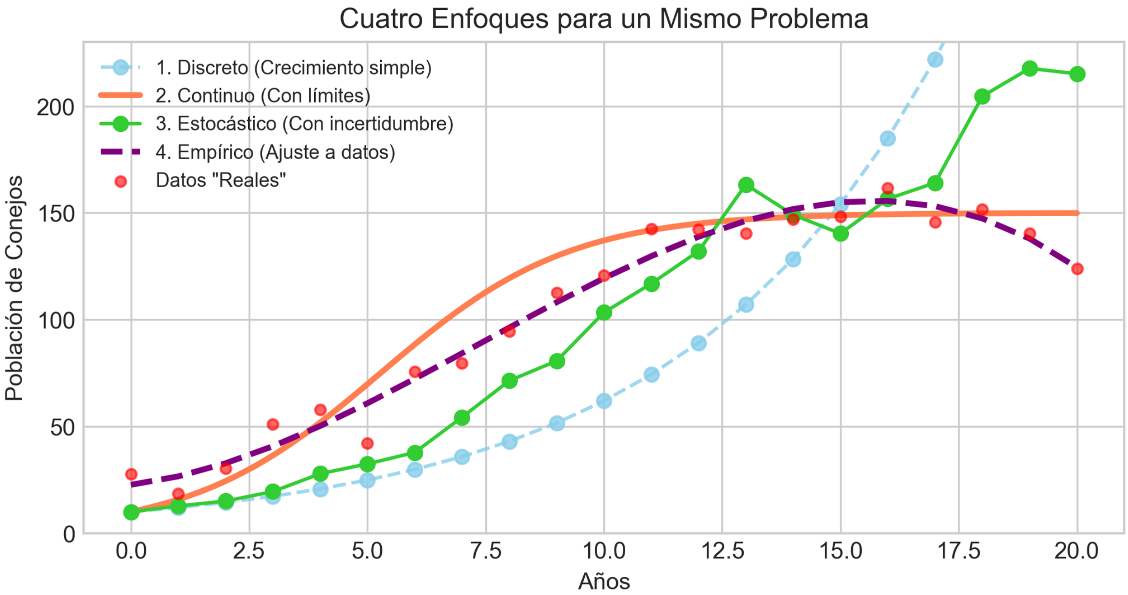
Idea: No conocemos la biología de los conejos, pero tenemos datos históricos. Ajustamos una función a esos datos.

Modelo: Una regresión (en este caso, polinómica de grado 3) que minimiza el error con respecto a los datos observados.

Clasificación:

- Empírico: Describe el "qué" (la tendencia en los datos), no el "porqué".
- Puede ser discreto o continuo en su formulación.
- Puede ser determinístico (la curva) o estocástico (si consideramos los residuos).


Ajuste de un Modelo a Datos Observados
160
140
120
100
80
60
40
20
Población de Conejos
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Años
Datos Observados
Modelo Polinómico Empírico


Cuatro Enfoques para un Mismo Problema
1. Discreto (Crecimiento simple)
2. Continuo (Con límites)
3. Estocástico (Con incertidumbre)
4. Empírico (Ajuste a datos)
Datos "Reales"
Población de Conejos
0
50
100
150
200
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Años

Asumiendo una tasa de crecimiento base del **20%** y una población inicial (P_0) de **20 conejos**, la predicción a 20 años varía drásticamente según las suposiciones del modelo:

```
- Modelo Discreto-Determinístico: 767 conejos
- Modelo Logístico-Determinístico: 134 conejos
- Modelo Discreto-Estocástico: 430 conejos (un resultado posible)
- Modelo Empírico (por ajuste): 116 conejos
```

Receso breve (5min)

Sección 3, Parte I: Instalación de Herramientas y Ejercicios

Objetivo: Instalar y verificar instalación de Python y Jupyter notebooks para realizar modelamiento, simulaciones y ajustes de modelos.

- Si no se tiene instalado Python, la forma más sencilla es hacerlo desde la distribución Anaconda.
- Las librerías necesarias son:
    - `numpy`
    - `pandas`
    - `matplotlib`
    - `scipy`

Ejercicios en Python

Notebook `code/sec3-tools.ipynb`.

Objetivo:

El Proceso de Modelación es un Ciclo Iterativo

Antes de ver los pasos, es crucial entender que la modelación no es un proceso lineal, sino un **ciclo iterativo**.

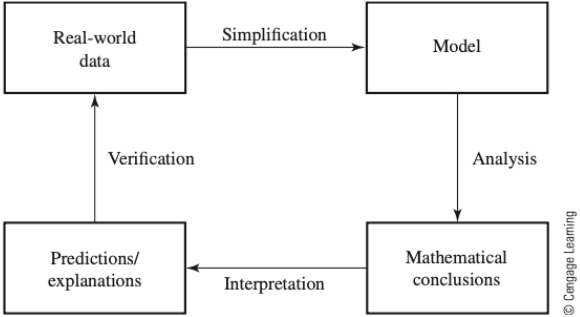


Figure 3: Relaciones entre realidad, modelo, interpretación.

- Rara vez se construye un modelo perfecto al primer intento.
- El objetivo es empezar con algo simple, validarlo y **refinarlo** progresivamente.
- Cada paso informa al siguiente y, a menudo, nos obliga a regresar a un paso anterior.

Paso 1: Identificación del Problema

Todo buen modelo comienza con una **pregunta clara y bien definida**. Este es el paso más importante. Si la pregunta es ambigua, el modelo será inútil.

- **El Objetivo:** ¿Qué queremos lograr?
  - **Explicar:** Entender por qué las filas se hacen largas a mediodía.
  - **Predecir:** Estimar el tiempo de espera promedio mañana a las 10 am.
  - **Optimizar:** Determinar si necesitamos abrir una segunda caja para que nadie espere más de 5 minutos.
- **Variables y Alcance:** ¿Qué mediremos y cuáles son los límites?
  - **Variables:** Tiempo de espera, número de clientes, tasa de servicio.
  - **Alcance:** Modelaremos una sola caja en un día laboral normal.

Paso 2: Establecimiento de Suposiciones

Los modelos son simplificaciones. Las **suposiciones** son las reglas explícitas de esa simplificación y definen la base (y las limitaciones) de nuestro modelo.

- Deben ser **explícitas** y **justificables**.
- Nos permiten transformar un problema complejo y "sucio" en uno matemático y "limpio".
- **Ejemplo: La Fila de la Cafetería**
  - **Llegada de clientes:** Asumimos que los clientes llegan a un ritmo constante y predecible (ej. 1 cliente cada 60 segundos). Esto nos da una tasa de llegada, λ.
  - **Tiempo de servicio:** Asumimos que el barista tarda el mismo tiempo con cada cliente (ej. 45 segundos). Esto nos da una tasa de servicio, μ.
  - **Comportamiento de la fila:** Asumimos que es una sola fila, nadie se rinde y se van, y el primer cliente en llegar es el primero en ser atendido (FIFO).

Paso 3: Formulación del Modelo

Traducimos nuestras suposiciones a un lenguaje lógico o matemático. En este caso, no usaremos una sola ecuación, sino un **algoritmo de simulación** que opera paso a paso en el tiempo.

Variables Clave del Modelo:

- `tiempo_actual`: El reloj de nuestra simulación (en segundos).
- `clientes_en_cola`: Número de personas esperando.
- `tiempo_para_llegada`: Contador para saber cuándo llega el siguiente cliente.
- `tiempo_servicio_restante`: Contador para saber cuándo el barista se desocupa.

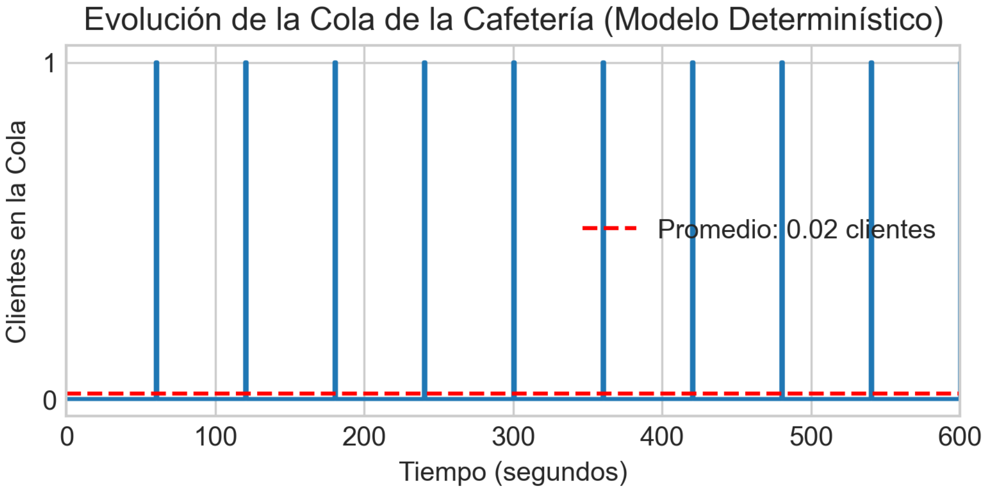
Lógica de la Simulación (Pseudocódigo):

1. Inicializar todas las variables a cero.
2. Para cada segundo de 1 hora (3600 segundos):
   a. Reducir en 1 los contadores de tiempo.
   b. **¿Llegó un cliente?** Si `tiempo_para_llegada` llega a 0, añadir 1 a la cola y reiniciar el contador.
   c. **¿El barista está libre?** Si `tiempo_servicio_restante` es 0 y hay gente en la cola, atender a 1 cliente (restarlo de la cola) y reiniciar el contador de servicio.
   d. Registrar el número de clientes en cola en este segundo.

Análisis y Solución

Ahora, implementamos el algoritmo en Python para “resolver” el modelo, es decir, para ver qué resultados produce.

Notebook: `code/sec3-modeling.ipynb`.




```
Simulación completada.
Tamaño máximo de la cola: 1 clientes.
```

El resultado del modelo (la simulación) debe ser interpretado en el contexto del problema original.

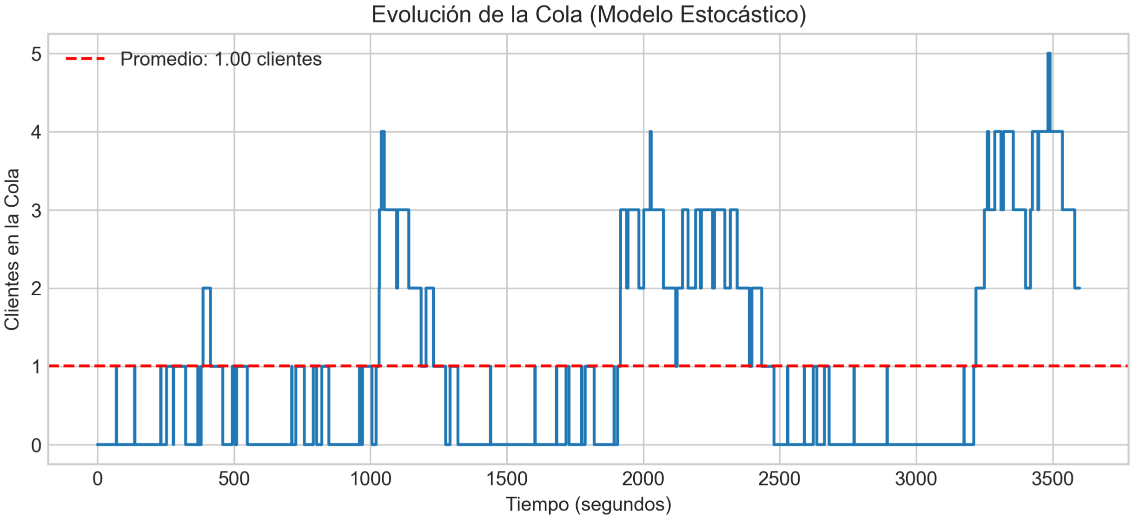
- ¿Qué nos dice el gráfico? La simulación muestra que la cola nunca crece. Permanece en 0 o salta brevemente a 1.
- ¿Por qué ocurre esto? Nuestras suposiciones iniciales lo garantizan. La tasa de servicio ($=1/45$ clientes/seg) es mayor que la tasa de llegada ($=1/60$ clientes/seg). El barista es más rápido que la llegada de clientes, por lo que el sistema es estable y no se congestiona.
- Validación: ¿Es esto realista? No del todo. La llegada de clientes y los tiempos de servicio no son constantes. Este resultado nos lleva al siguiente paso del ciclo.

Refinamiento del Modelo

Nuestro modelo simple nos dio una primera respuesta, pero su irrealismo nos invita a mejorarlo. El ciclo iterativo de modelación consiste en cuestionar las suposiciones y refinarlas.

- Pregunta de Refinamiento: ¿Qué pasaría si la llegada de clientes fuera aleatoria, siguiendo una distribución de Poisson, que es más común para este tipo de fenómenos?
- Nueva Suposición: En lugar de un cliente cada 60 segundos, *la probabilidad de que un cliente llegue en un segundo dado es de 1/60*.
- Modificación: Cambiamos la lógica de llegada para incorporar aleatoriedad (`np.random.rand()`).

Este proceso de **crítica** -> **refinamiento** -> **nueva simulación** es el núcleo del proceso de modelación.




```
Simulación estocástica completada.
Tamaño máximo de la cola: 5 clientes
```

Ahora que tenemos un modelo estocástico funcional, podemos usarlo para lo que realmente sirve: experimentar con el futuro y tomar decisiones informadas.

Análisis de Escenario: ¿Invertir en una máquina más rápida?

Planteamos una pregunta de negocio concreta para guiar nuestro análisis.

- **Situación Actual:** Nuestro barista tarda **45 segundos** por cliente.
- **Propuesta:** Invertir en una máquina que reduce el tiempo de servicio a **30 segundos**.
- **Pregunta Clave:** ¿Cómo se benefician los clientes? ¿Cuál es el impacto cuantificable en la longitud de la cola y el tiempo de espera?

Metodología: Simulación de Monte Carlo

Debido a que nuestro modelo es aleatorio, una sola simulación no es suficiente. El resultado podría ser por pura suerte. Para obtener una estimación confiable, realizamos un **Análisis de Monte Carlo**:

1. **Encapsular la Lógica:** Convertimos nuestro código de simulación en una función reutilizable que acepta el `tiempo_de_servicio` como parámetro.
2. **Ejecutar Múltiples Veces:** Corremos la simulación un número grande de veces (ej., 100 o 1,000) para cada escenario (45s y 30s).
3. **Promediar Resultados:** Calculamos el promedio de las métricas clave (longitud máxima y promedio de la cola) a través de todas las ejecuciones para cada escenario.
4. **Comparar y Decidir:** Con los promedios estables, podemos comparar de manera fiable el rendimiento de ambos sistemas y tomar una decisión basada en datos.

Sección 4, Parte I: Fundamentos y Ajuste del Modelo

Objetivo: Introducir la intuición detrás de GMM y ajustar un primer modelo.

K-means

¿Qué es K-Means?

K-Means es uno de los algoritmos más fundamentales de **aprendizaje no supervisado**. Su objetivo es simple:

> *Particionar un conjunto de datos en **K** grupos (clusters) distintos y no superpuestos, donde cada punto de datos pertenece al cluster cuyo centro (centroide) es el más cercano.*

El algoritmo busca que los clusters sean lo más **compactos** y **separados** posible, minimizando la varianza dentro de cada cluster.

K-means

¿Qué es K-Means?

K-Means es uno de los algoritmos más fundamentales de **aprendizaje no supervisado**. Su objetivo es simple:

> *Particionar un conjunto de datos en **K** grupos (clusters) distintos y no superpuestos, donde cada punto de datos pertenece al cluster cuyo centro (centroide) es el más cercano.*

El algoritmo busca que los clusters sean lo más **compactos** y **separados** posible, minimizando la varianza dentro de cada cluster.

K-means

¿Qué es K-Means?

K-Means es uno de los algoritmos más fundamentales de **aprendizaje no supervisado**. Su objetivo es simple:

> *Particionar un conjunto de datos en* ***K*** *grupos (clusters) distintos y no superpuestos, donde cada punto de datos pertenece al cluster cuyo centro (centroide) es el más cercano.*

El algoritmo busca que los clusters sean lo más **compactos** y **separados** posible, minimizando la varianza dentro de cada cluster.

K-means

¿Qué es K-Means?

K-Means es uno de los algoritmos más fundamentales de **aprendizaje no supervisado**. Su objetivo es simple:

> *Particionar un conjunto de datos en **K** grupos (clusters) distintos y no superpuestos, donde cada punto de datos pertenece al cluster cuyo centro (centroide) es el más cercano.*

El algoritmo busca que los clusters sean lo más **compactos** y **separados** posible, minimizando la varianza dentro de cada cluster.

Matemáticamente, el modelo K-Means postula que una agrupación ideal es aquella que minimiza la **inercia**, también conocida como la suma de cuadrados dentro de cada cluster. Su objetivo es resolver esta ecuación:

$$J = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mu_i||^2$$

Donde:

- K es el número de clusters.
- C_i es el conjunto de puntos en el cluster i.
- μ_i es el centroide (la media) del cluster i.

El algoritmo iterativo que usamos es, en realidad, una técnica numérica para encontrar una solución a esta función de coste. Por lo tanto, K-Means es un modelo porque **abstrae la estructura de los datos** (usando centroides) y tiene un **objetivo matemático formal**.

¿Cómo Funciona? El Proceso Iterativo

K-Means encuentra los clusters a través de un proceso iterativo de dos pasos:

1. **Paso de Asignación**: A cada punto de datos se le asigna el centroide más cercano.
2. **Paso de Actualización**: Se recalcula la posición de cada centroide, moviéndolo al centro (promedio) de todos los puntos de datos que le fueron asignados.

Estos dos pasos se repiten hasta que los centroides dejan de moverse y las asignaciones de los clusters se estabilizan.

¿Por qué GMM?

K-Means es una excelente herramienta, pero tiene supuestos rígidos que no siempre se cumplen.

- **Asignación "Dura"**: Un punto pertenece a un cluster o a otro, no hay incertidumbre. ¿Qué pasa con los puntos en las fronteras?
- **Clusters Esféricos**: K-Means asume que los clusters tienen una forma circular y un tamaño similar, ya que solo optimiza la distancia al centroide.
  Necesitamos un modelo más flexible que pueda capturar clusters de diferentes formas y orientaciones, y que además nos diga qué tan "seguro" está de la asignación de cada punto.

¿Por qué GMM?

K-Means es una excelente herramienta, pero tiene supuestos rígidos que no siempre se cumplen.

- **Asignación "Dura"**: Un punto pertenece a un cluster o a otro, no hay incertidumbre. ¿Qué pasa con los puntos en las fronteras?
- **Clusters Esféricos**: K-Means asume que los clusters tienen una forma circular y un tamaño similar, ya que solo optimiza la distancia al centroide.
  Necesitamos un modelo más flexible que pueda capturar clusters de diferentes formas y orientaciones, y que además nos diga qué tan "seguro" está de la asignación de cada punto.

¿Por qué GMM?

K-Means es una excelente herramienta, pero tiene supuestos rígidos que no siempre se cumplen.

- **Asignación “Dura”**: Un punto pertenece a un cluster o a otro, no hay incertidumbre. ¿Qué pasa con los puntos en las fronteras?
- **Clusters Esféricos**: K-Means asume que los clusters tienen una forma circular y un tamaño similar, ya que solo optimiza la distancia al centroide.
  Necesitamos un modelo más flexible que pueda capturar clusters de diferentes formas y orientaciones, y que además nos diga qué tan “seguro” está de la asignación de cada punto.

Intuición de GMM:

Imaginar que los datos no provienen de una sola fuente, sino de varias fuentes superpuestas.

Gaussian Mixture Models (GMM) asume que los datos son una combinación de varias distribuciones Normales (Gaussianas), cada una con su propio:

- **Centro** (Media): ¿Dónde está el cluster?
- **Forma y Orientación** (Covarianza): ¿Es circular, elíptico, alargado?
- **Tamaño** (Peso): ¿Qué proporción de los datos pertenece a este cluster?

Intuición de GMM:

Imaginar que los datos no provienen de una sola fuente, sino de varias fuentes superpuestas.

Gaussian Mixture Models (GMM) asume que los datos son una combinación de varias distribuciones Normales (Gaussianas), cada una con su propio:

- **Centro** (Media): ¿Dónde está el cluster?
- **Forma y Orientación** (Covarianza): ¿Es circular, elíptico, alargado?
- **Tamaño** (Peso): ¿Qué proporción de los datos pertenece a este cluster?

GMM un modelo de **asignación "suave"**: en lugar de una etiqueta, cada punto de datos obtiene una **probabilidad** de pertenecer a cada uno de los clusters dada por:

$$f(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

- x es un dato de D dimensiones.
- π_k es el coeficiente de mezcla para la k-ésima componente Gaussiana, con $0 \leq \pi_k \leq 1$ y $\sum_{k=1}^{K} \pi_k = 1$.
- $\mathcal{N}(x|\mu_k, \Sigma_k)$ es la distribución Gaussiana de D dimensiones con vector de medias μ_k y matriz de covarianza Σ_k.

¿Por qué “Mezcla Gaussiana”?

Característica	K-Means (Modelo Geométrico)	GMM (Modelo Probabilístico)
Suposición Clave	Los datos forman cúmulos esféricos.	Los datos son una **mezcla** de varias distribuciones **Gaussianas**.
Parámetros del Modelo	Un conjunto de K centroides: $\{\mu_1, ..., \mu_K\}$.	Un conjunto de K tripletas de parámetros: $\{(\mu_k, \Sigma_k, \pi_k)\}_{k=1}^{K}$ (Media, Covarianza y Peso).
Resultado	Asignación “dura” a un cluster.	**Probabilidad** de pertenencia a cada cluster.

- K-Means tiene una visión **geométrica**
- GMM tiene una visión **generativa**: asume que los datos fueron *generados* por una combinación de diferentes procesos (las gaussianas).
- Por eso es un modelo probabilístico mucho más rico que no solo agrupa, sino que intenta describir la **distribución de probabilidad** de la que provienen los datos.

¿Cómo encuentra el modelo los parámetros (centro, forma, tamaño) de estas gaussianas?

El principio rector es la **Estimación por Máxima Verosimilitud** (*Maximum Likelihood Estimation* o MLE).

- **La Pregunta Clave**: De todas las combinaciones posibles de gaussianas, ¿cuál es la que con **mayor probabilidad** pudo haber generado los datos que observamos?

¿Cómo encuentra el modelo los parámetros (centro, forma, tamaño) de estas gaussianas?

El principio rector es la **Estimación por Máxima Verosimilitud** (*Maximum Likelihood Estimation* o MLE).

- **La Pregunta Clave**: De todas las combinaciones posibles de gaussianas, ¿cuál es la que con **mayor probabilidad** pudo haber generado los datos que observamos?

Proceso de MLE

El objetivo de MLE es encontrar el valor del parámetro de un modelo (θ) que hace que nuestros datos observados ($\mathbf{x}$) sean **lo más probables posible**. Para modelos simples, podemos encontrar esta solución analíticamente.

Pensemos en el caso de lanzar una moneda 10 veces y obtener 7 caras. Queremos estimar p, la probabilidad de que salga cara.

1. **Definir la Función de Verosimilitud ($\mathcal{L}(\theta|\mathrm{x})$)**
   - Es una función que nos dice qué tan probables son nuestros datos para un valor específico del parámetro. Para nuestro ejemplo, es la función de masa de probabilidad binomial.

$$\mathcal{L}(p|k=7, n=10) = \binom{10}{7} p^7 (1-p)^3$$

2. **Simplificar con Log-Verosimilitud ($\ln \mathcal{L}$)**
   - Maximizar $\mathcal{L}$ es lo mismo que maximizar $\ln \mathcal{L}$. El logaritmo convierte productos en sumas, lo que facilita enormemente la derivación.

$$\ln \mathcal{L} = \ln\left(\binom{10}{7}\right) + 7\ln(p) + 3\ln(1-p)$$

Pensemos en el caso de lanzar una moneda 10 veces y obtener 7 caras. Queremos estimar p, la probabilidad de que salga cara.

1. **Definir la Función de Verosimilitud ($\mathcal{L}(\theta|\mathbf{x})$)**
   - Es una función que nos dice qué tan probables son nuestros datos para un valor específico del parámetro. Para nuestro ejemplo, es la función de masa de probabilidad binomial.

$$\mathcal{L}(p|k=7, n=10) = \binom{10}{7} p^7 (1-p)^3$$

2. **Simplificar con Log-Verosimilitud ($\ln \mathcal{L}$)**
   - Maximizar $\mathcal{L}$ es lo mismo que maximizar $\ln \mathcal{L}$. El logaritmo convierte productos en sumas, lo que facilita enormemente la derivación.

$$\ln \mathcal{L} = \ln\left(\binom{10}{7}\right) + 7\ln(p) + 3\ln(1-p)$$

3. **Derivar e Igualar a Cero (Condición de Primer Orden)**
   - Para encontrar el máximo de la función, buscamos el punto donde su pendiente es cero.

$$\frac{d \ln \mathcal{L}}{dp} = \frac{7}{p} - \frac{3}{1-p} = 0$$

4. **Resolver para el Parámetro ($\hat{\theta}_{MLE}$)**
   - Al resolver la ecuación, encontramos el valor del parámetro que maximiza la verosimilitud.

$$7(1-p) = 3p \implies 7 - 7p = 3p \implies 10p = 7 \implies \hat{p}_{MLE} = 0.7$$

El Problema: *Para modelos complejos como GMM, la derivada de la log-verosimilitud es demasiado complicada para resolverla analíticamente. Aquí es donde entra la **optimización numérica**.*

$$\begin{aligned}
\underset{\{\pi_k,\mu_k,\Sigma_k\}_{k=1}^{K}}{\text{maximizar}} \quad & \sum_{i=1}^{N} \ln\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_i|\mu_k,\Sigma_k)\right) \\
\text{s.a.} \quad & \sum_{k=1}^{K} \pi_k = 1 \\
& \pi_k \geq 0, \quad \forall k \\
& \Sigma_k \text{ es positiva semidefinida}, \quad \forall k
\end{aligned}$$

Si $D = 1$ y $K = 2$, y si volvemos negativa $\ln \mathcal{L}$, tenemos:

$$
\begin{aligned}
\underset{\mu_1, \mu_2, \sigma_1, \sigma_2, \pi_1}{\text{minimizar}} \quad & -\sum_{i=1}^{N} \ln \left(\pi_1 \mathcal{N}(x_i | \mu_1, \sigma_1^2) + (1 - \pi_1) \mathcal{N}(x_i | \mu_2, \sigma_2^2) \right) \\
\text{s.a.} \quad & 0 \leq \pi_1 \leq 1 \\
& \sigma_1 > 0, \quad \sigma_2 > 0
\end{aligned}
$$

Notebook: `code/sec4-gmm.ipynb`.

Actividad (30min):

Imagina que somos un equipo de control de calidad y hemos probado 150 unidades de un nuevo componente electrónico hasta que fallan. Hemos registrado el tiempo de vida de cada uno en horas. Queremos modelar esta distribución para poder hacer predicciones, como estimar la vida media del componente o la probabilidad de que falle antes de cierto tiempo.

Los datos podrían ser los tiempos de falla y no siguen una distribución normal simple.

El Modelo: La Distribución de Weibull

La distribución de Weibull es extremadamente flexible y se define por dos parámetros:

- k (parámetro de **forma**): Describe el modo de falla.
  - $k < 1$: La tasa de falla disminuye con el tiempo (fallas infantiles).
  - $k = 1$: La tasa de falla es constante (fallas aleatorias, se reduce a la dist. exponencial).
  - $k > 1$: La tasa de falla aumenta con el tiempo (fallas por desgaste).
- λ (parámetro de **escala**): Representa la "vida característica" del componente.

Su función de densidad de probabilidad (PDF) es:

$$f(x;k,\lambda) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} \quad \text{para } x \geq 0$$

Nuestro objetivo es encontrar los valores de k y λ que mejor se ajustan a nuestros datos observados.

Usa los datos `data\failure-times.csv`.

El objetivo es minimizar la siguiente función $\ell\ell$ (simplificada ya de la pdf).

$$\ell\ell^*(k, \lambda|\mathbf{x}) = -\sum_{i=1}^{N} \left[\ln(k) - \ln(\lambda) + (k-1)(\ln(x_i) - \ln(\lambda)) - \left(\frac{x_i}{\lambda}\right)^k \right]$$

Evaluación del taller

Link al form