



Estadística aplicada potenciada por las nuevas tecnologías: *integrando la IA generativa*

Mgtr. Carolina Del Pilar Díaz

*IT Patagonia S.A.
AI Model Accelerator
B.A. Argentina.*

Dr. Pablo M. Demetrio

*Profesor Geoestadística
Investigador Adjunto
UNLP | CONICET*

Corrientes, Argentina.

En este link de NotebookLM pueden consultar sobre el contenido del curso

<https://notebooklm.google.com/notebook/04a3e267-3977-4787-a4c8-aaf3801bc100>



MIÉRCOLES 1/10

**¿Cómo pensamos
este minicurso?**

Planificación

Miércoles

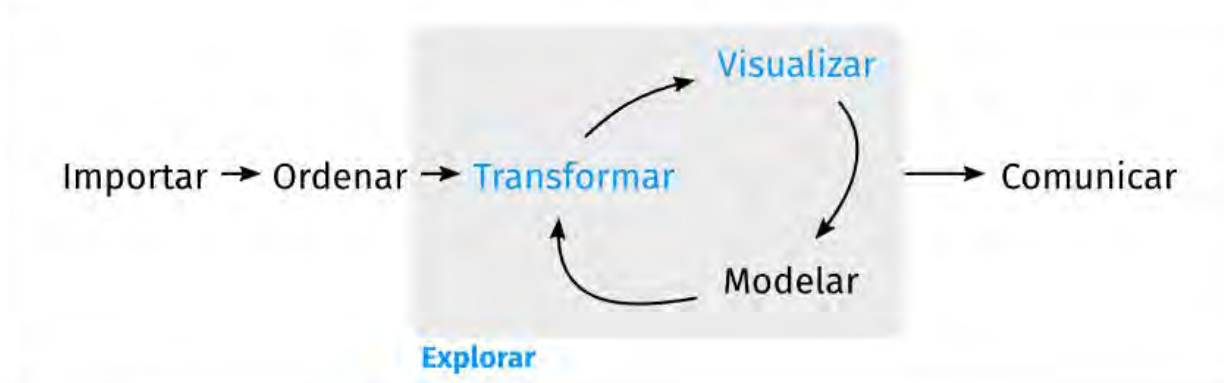
- Introducción
- Flujo de trabajo
- gen IA: asistentes
- Herramientas: R+

Jueves

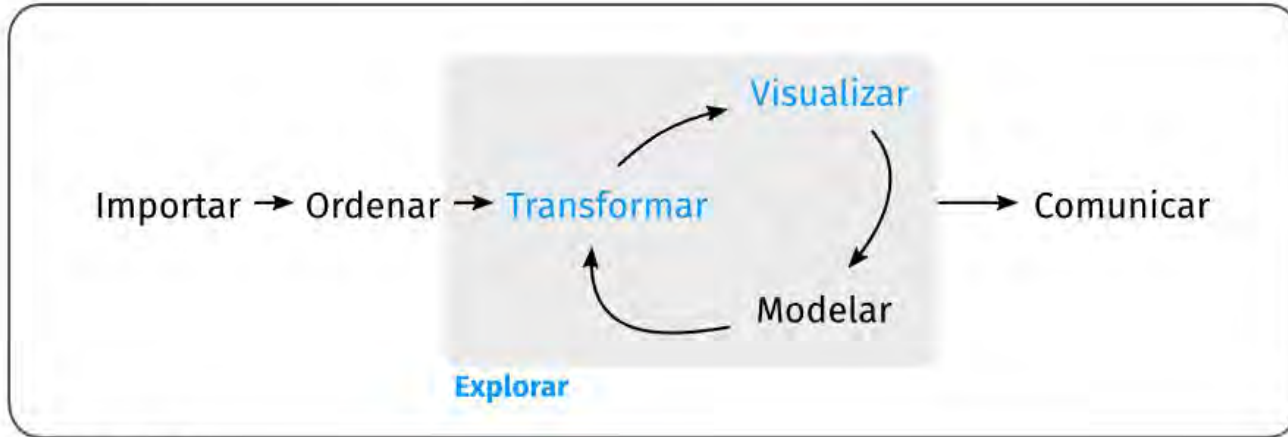
- retomando
- Herramientas: Py+
- Agentes y un poco más
- (re)pensando nuestros roles

Flujo de trabajo

Flujo de trabajo

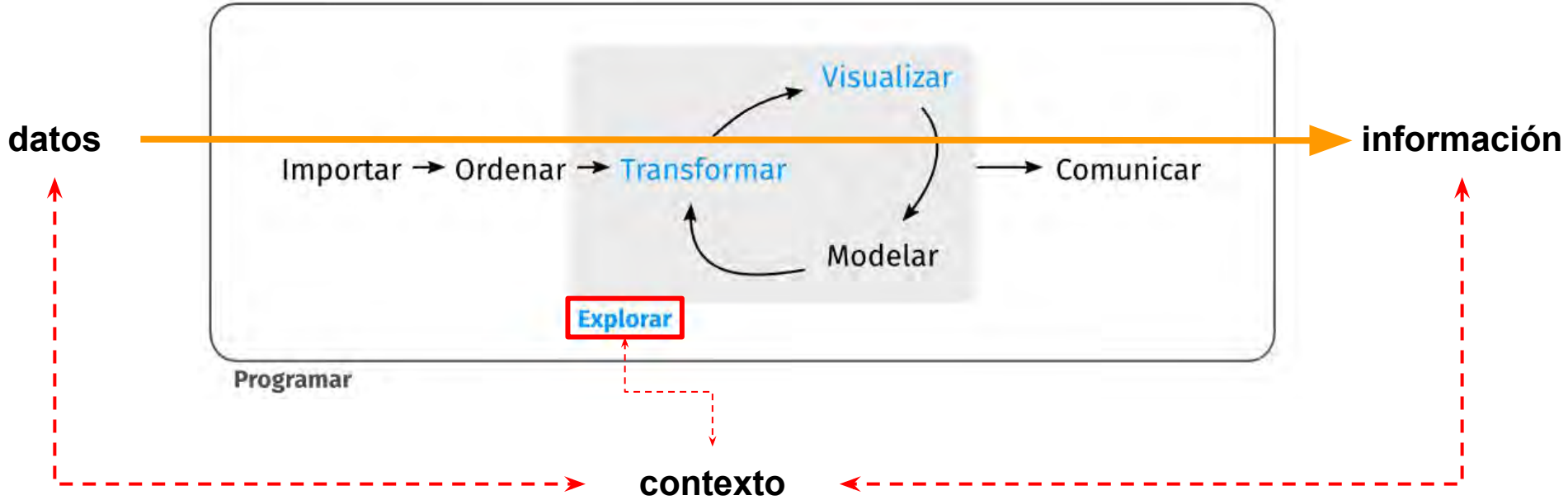


Flujo de trabajo



Programar

Flujo de trabajo



Flujo de trabajo



Algo de IA Generativa

...explicada por la IA Generativa

Algunos modelos de IA Generativa

GPT (*Generative Pre-trained Transformer*): Desarrollado por **OpenAI**, uno de los modelos de lenguaje más influyentes. Sus versiones (GPT-2, ..., GPT-5) han marcado hitos en generación de texto coherente y versátil.

LLaMA (*Large Language Model Meta AI*): modelos de lenguaje desarrollados por **Meta** (Facebook). Son de código abierto y se han convertido en base para muchos proyectos derivados.

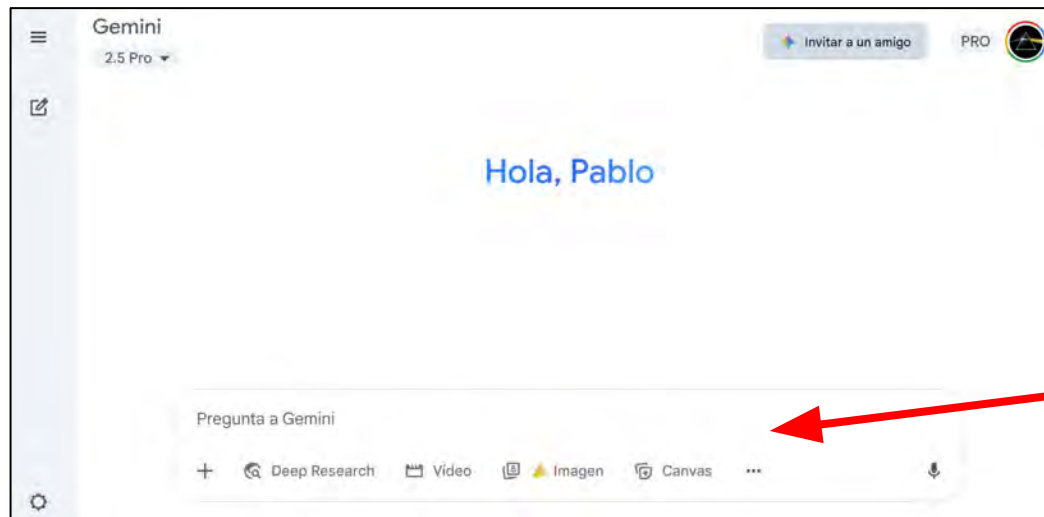
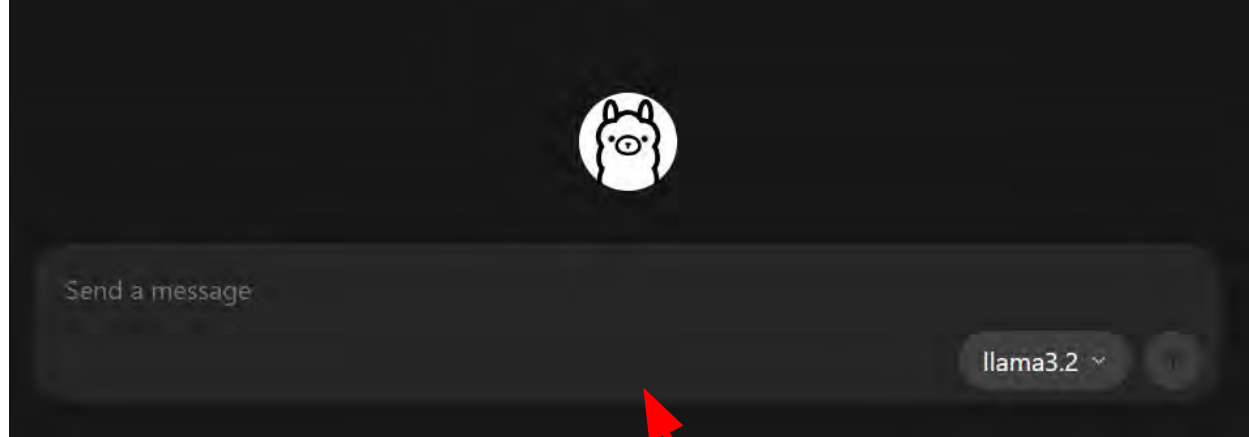
Gemini: Familia de modelos de **Google** DeepMind. Están diseñados para ser multimodales (procesar texto, imágenes, audio y código).

Claude: Modelo desarrollado por **Anthropic**, orientado a seguridad, confiabilidad y alineación ética.

Mistral: Modelos de lenguaje de código abierto muy eficientes, conocidos por su rendimiento en hardware reducido y por modelos de tipo *mixture of experts* (MoE).

Interactuando con la genAI

chateando con los LLMs



Indicaciones



Interacción con la IA

“...trata a la IA como a un nuevo compañero de trabajo **infinitamente paciente** que **olvida todo lo que le cuentas en cada nueva conversación**, alguien que **viene muy recomendado**, pero **cuyas habilidades reales no están tan claras**.”

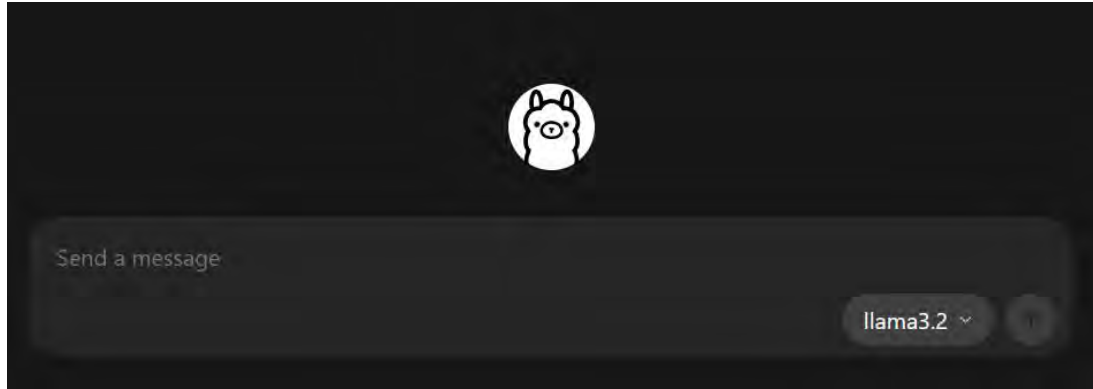


Dos aspectos de esto son análogos a trabajar con humanos (ser nuevo en el trabajo y ser compañero de trabajo)...

....y dos de ellos son muy ajenos (olvidarlo todo y ser infinitamente paciente).

Deberíamos empezar por dónde las IA se acercan más a los humanos, porque esa es la ***clave para una estimulación suficientemente buena...***”

prompts / indicaciones



Utilizar lenguaje natural

Ser claro y conciso

Proporcionar contexto

Utilizar palabras clave específicas y relevantes

Dividir las tareas complejas en indicaciones separadas

ROCKET

R – Rol -> Definir el rol o contexto que querés que adopte la IA. *“En rol de experto en estadística aplicada en temas ambientales”*

O – Objetivo → Aclarar el propósito de tu consulta: ¿qué querés lograr? *“Quiero aplicar los métodos estadísticos disponibles para ver diferencias entre tratamientos experimentales.”*

C – Contexto → Proporcionar información adicional y el escenario donde se aplicará la respuesta. *“Es en el contexto de una consultoría estadística, pero debo generar un informe de resultados para un público sin conocimiento técnico”*

K – Key details → Sumá datos específicos que guíen la respuesta. Cosas que sí y que no. *“Limitar la aplicación a métodos de estadística frecuentista.”*

E – Expectativa → Indicar el formato o tipo de respuesta a esperar. *“Responder en orden desde metodologías más estándar a más sofisticadas; incluyendo fortalezas y debilidades.”*

T – Tono → Definir el estilo y tono de la respuesta. *“Explicarlo con un tono pedagógico y claro.”*

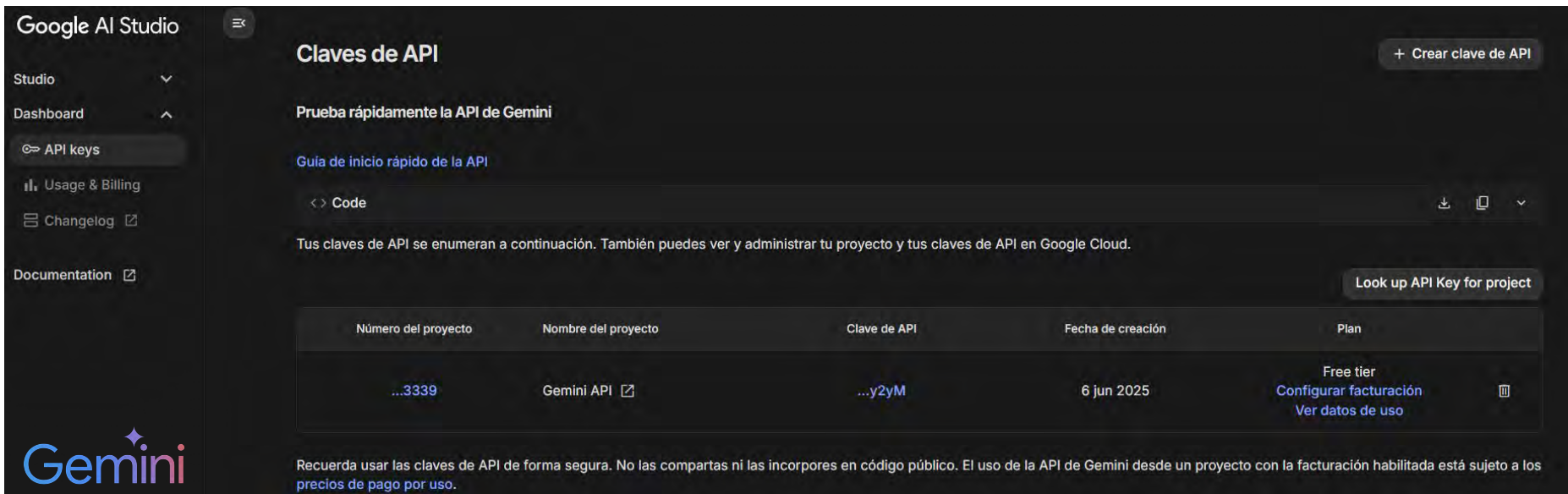
ejemplo



**...y dónde está el
modelo?**

API

application programming interface



The screenshot shows the Google AI Studio interface for managing API keys. On the left is a navigation sidebar with options like Studio, Dashboard, API keys, Usage & Billing, Changelog, and Documentation. The main content area is titled 'Claves de API' and includes a '+ Crear clave de API' button, a link to the Gemini API quick start guide, and a 'Code' editor. Below this is a table of existing API keys with columns for project number, project name, API key, creation date, and plan. A 'Look up API Key for project' button is also present. At the bottom, a security warning is displayed.

Google AI Studio

Studio

Dashboard

API keys

Usage & Billing

Changelog

Documentation

Claves de API

+ Crear clave de API

Prueba rápidamente la API de Gemini

[Guía de inicio rápido de la API](#)

<> Code

Tus claves de API se enumeran a continuación. También puedes ver y administrar tu proyecto y tus claves de API en Google Cloud.

Look up API Key for project

Número del proyecto	Nombre del proyecto	Clave de API	Fecha de creación	Plan
...3339	Gemini API	...y2yM	6 jun 2025	Free tier Configurar facturación Ver datos de uso

Recuerda usar las claves de API de forma segura. No las compartas ni las incorpores en código público. El uso de la API de Gemini desde un proyecto con la facturación habilitada está sujeto a los precios de pago por uso.



Cloud models are now available in Ollama

Chat & build with open models

[Download](#)

Available for macOS,
Windows, and Linux

Ollama



Es una **herramienta de software de código abierto** diseñada para **simplificar la descarga, configuración y ejecución** de *grandes modelos de lenguaje (LLMs)* *directamente en tu propia computadora.*

Ollama actúa como un gestor y un servidor local. Se encarga de:

- **Administrar los modelos** que has descargado.
- **Asignar los recursos** de tu hardware para que el modelo funcione.
- **Crear una interfaz** (generalmente de línea de comandos o a través de aplicaciones de terceros) para que puedas interactuar con el modelo de forma sencilla.

Esta arquitectura local te ofrece ventajas clave:

- **Privacidad Total:** Tus datos son tuyos y nunca son vistos por ninguna empresa. Ideal para trabajar con información sensible o personal.
- **Funcionamiento Offline:** Una vez descargado el modelo, puedes usarlo sin conexión a internet.
- **Control Absoluto:** Tienes control total sobre qué modelos usas y cómo los configuras.
- **Sin Costos por Uso:** Al utilizar tus propios recursos, no pagas por cada pregunta que haces, a diferencia de las APIs de servicios en la nube.

NotebookLM

Es una herramienta experimental de Google diseñada para funcionar como un *asistente de investigación virtual*.

A diferencia de los chatbots de IA tradicionales, se basa exclusivamente en los documentos que le proporcionas.

"Source-Grounded" (Basado en tus fuentes). *No busca respuestas en todo internet, sino que razona y genera contenido a partir de tu propia colección de documentos (PDFs, texto, Google Docs, etc.). Esto reduce drásticamente el riesgo de "alucinaciones" o información inventada.*

Imagina tener un becario experto que ha leído y memorizado perfectamente todos tus papers, libros y notas. Puedes "conversar" con tus documentos: hacerles preguntas, pedirles resúmenes, que identifiquen conexiones o te expliquen conceptos complejos.

Fuentes

+ Añadir Descubrir

Base de Conocimiento (Sources)



Saved sources will appear here

Click Add source above to add PDFs, websites, text, videos, or audio files. Or import a file directly from Google Drive.

Chat

Asistente de IA (Chat)



Añade una fuente para comenzar

Subir una fuente

NotebookLM

Sube una fuente para empezar 0 fuentes 

Studio

Generación de contenidos

Resultados

Mapa mental Informes

Tarjetas didácticas Cuestionario

El Bloc de Notas (Noteboard)



Los resultados de Studio se guardarán aquí.

Después de añadir las fuentes, haz clic para añadir un resumen de audio, una guía de estudio o un mapa conceptual, entre otros.

Añadir nota



NotebookLM

Generar contenido de fuentes específicas seleccionadas por el usuario



NotebookLM

Desde  y 

IDE + LLM





...un buen prompt, mis datos y...

Ctrl C + Ctrl V





setear_API_key



IDE + LLM



ellmer



facilita el uso de
LLM

- Anthropic's Claude: `chat_anthropic()`.
- AWS Bedrock: `chat_aws_bedrock()`.
- Azure OpenAI: `chat_azure_openai()`.
- Cloudflare: `chat_cloudflare()`.
- Databricks: `chat_databricks()`.
- DeepSeek: `chat_deepseek()`.
- GitHub model marketplace: `chat_github()`.
- Google Gemini/Vertex AI: `chat_google_gemini()`, `chat_google_vertex()`.
- Groq: `chat_groq()`.
- Hugging Face: `chat_huggingface()`.
- Mistral: `chat_mistral()`.
- Ollama: `chat_ollama()`.
- OpenAI: `chat_openai()`.
- OpenRouter: `chat_openrouter()`.
- perplexity.ai: `chat_perplexity()`.
- Snowflake Cortex: `chat_snowflake()` and `chat_cortex_analyst()`.
- VLLM: `chat_vllm()`.



ellmer



IDE + LLM integrado



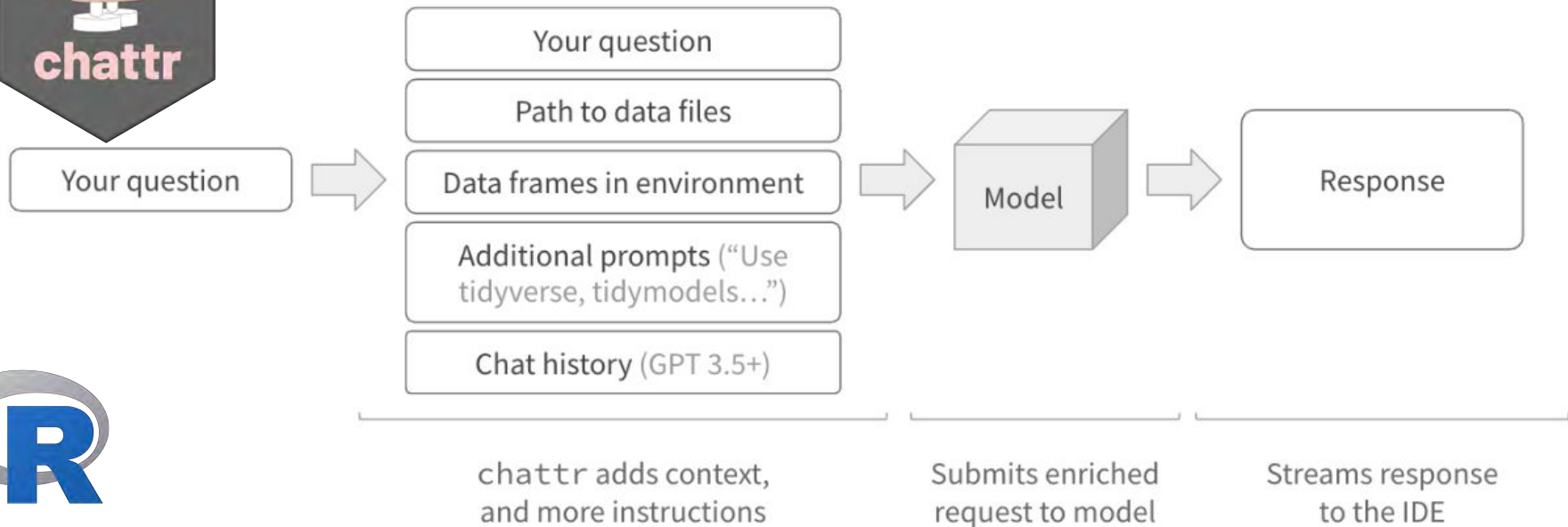
chattr



Permite enviar una solicitud al LLM desde el script o mediante el gadget Shiny proporcionado.

Enfocada en el **análisis exploratorio de datos**.

La información adicional adjunta a su solicitud proporciona una especie de "barrera de seguridad"



```

1 ## chattr
2
3 library(chattr)
4
5 ## ejemplo con LLM con Api key
6
7
8 chattr_use(my_chat_API)|
9
10 chattr_app()
11
12 chattr_defaults(include_history = TRUE,
13                include_doc_contents = TRUE,
14                max_data_frames = 20,
15                max_data_files = 20
16                )
17
18
19 ## ejemplo con LLM Local
20
21 chattr_use(my_chat_LOCAL)
22 chattr_app()
23
24 (Top Level)



```

indicaciones

como incorporar la
respuesta

Environment History Connections Tutorial

Files Plots Packages Help Viewer Presentation

Submit  

gemin-2.5-flash
(Google/Gemini)

```

library(tidyverse)

# Cargar el conjunto de datos de ejemplo (iris es un conjunto de datos incorpora
data <- iris

# 1. Estadísticas descriptivas para variables numéricas (todos los datos)
data %>%
  summarise(
    across(where(is.numeric),
            list(
              media = mean,
              mediana = median,
              desviacion_estandar = sd,
              minimo = min,
              maximo = max,
              Q1 = ~quantile(., 0.25),
              Q3 = ~quantile(., 0.75)
            )
          ),
    .names = "{.col}_{.fn}"
  )

# 2. Estadísticas descriptivas para variables numéricas (agrupadas por una varia
data %>%
  group_by(Species) %>%
  summarise(
    across(where(is.numeric),
            list(
              media = mean,
              mediana = median,
              desviacion_estandar = sd,
              minimo = min,
              maximo = max,
              Q1 = ~quantile(., 0.25),
              Q3 = ~quantile(., 0.75)
            )
          )
  )

```

respuesta
LLM

Console Terminal Background Jobs

R 4.5.1 · C:/Users/ASUS/OneDrive - quimica.unlp.edu.ar/2025/GAB/curso/data/ejemplos_curso/

```

> Sys.setenv(GEMINI_API_KEY = My_APIkey)
> library(e11mer)
> chat_google_gemini(model = "gemini-2.5-flash")
<Chat Google/Gemini/gemini-2.5-flash turns=0 tokens=0/0 $0.00>
> my_chat_API <- chat_google_gemini(model = "gemini-2.5-flash")
> library(chattr)
> chattr:::chattr_app()

```

- Provider: Google/Gemini
- Model: gemini-2.5-flash
- Label: gemini-2.5-flash (Google/Gemini)

Listening on http://127.0.0.1:3168

chattr



IDE + LLM integrado + addin



gander



Mayor rendimiento

el modelo tiene todo el contexto que necesita para proporcionar una respuesta lo más precisa posible (código ya escrito!)

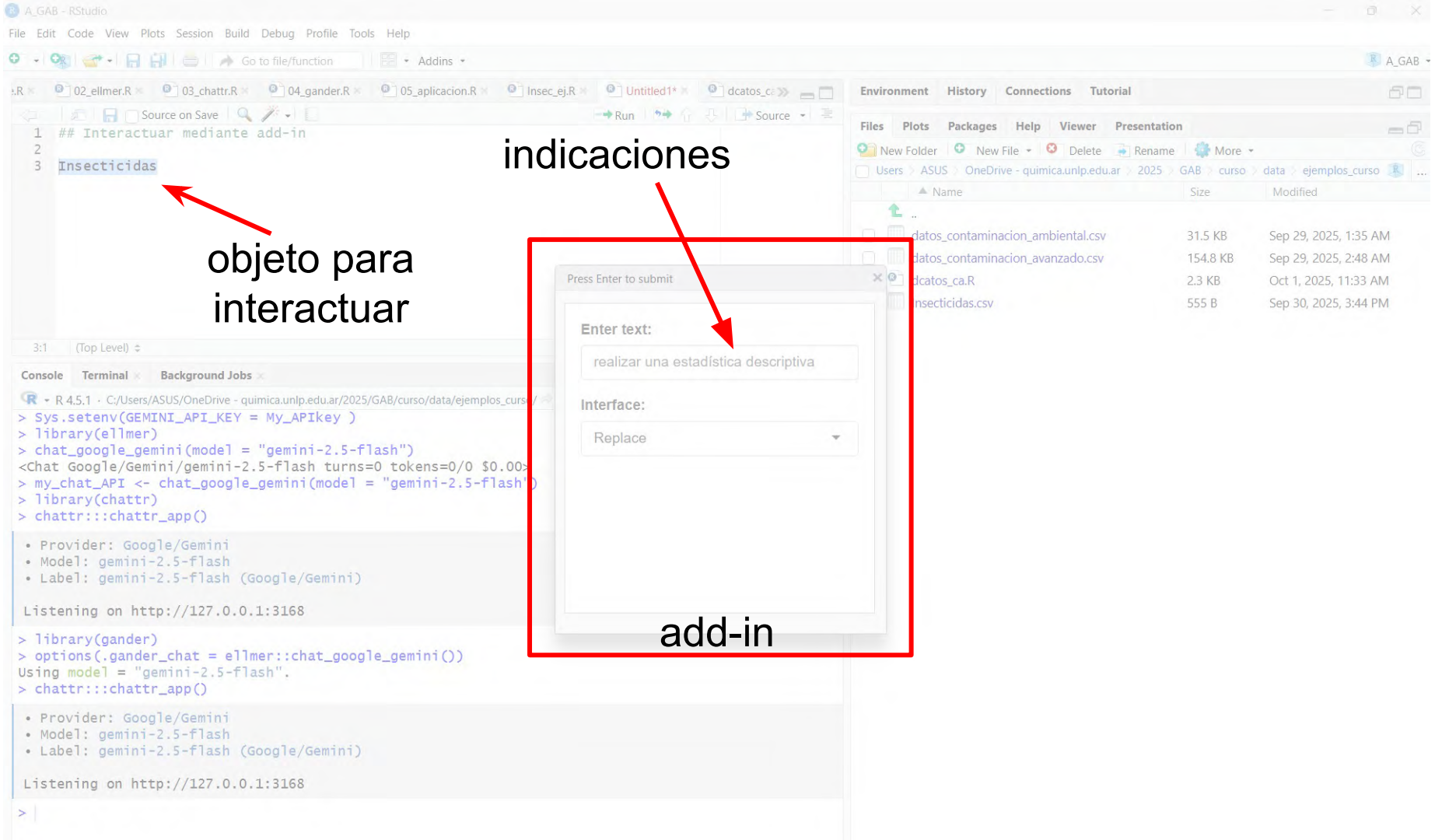
Menor fricción

elimina la necesidad de copiar y pegar código relevante en la ventana de chat y devolver su resultado al documento.

no es necesario escribir nada más que la solicitud

iterar sobre fragmentos de contenido es más sencillo, simplemente activar el complemento de nuevo.





gander



chattr

gander

Interfaz Principal	Ventana de chat tipo shiny app dentro de RStudio. Se siente como tener un chatbot en el IDE.	Inserta el código directamente en tu script.
Flujo de Trabajo	Orientado a la conversación y exploración.	Orientado a la generación de código en el acto.
Manejo del Contexto	Enriquece el prompt enviando nombres y estructura de data frames, archivos en el directorio y el historial del chat.	Es altamente consciente del contexto para generar respuestas precisas.
Fortalezas	<ul style="list-style-type: none">✓ Muy intuitivo para quienes están acostumbrados a ChatGPT.✓ Flexible: Permite un diálogo de ida y vuelta.✓ Excelente para tareas de EDA y para obtener ayuda conceptual.✓ Incluye botones para copiar el código al portapapeles o insertarlo en el script.	<ul style="list-style-type: none">✓ Baja fricción: Elimina la necesidad de copiar y pegar, integrándose de forma fluida en la escritura de código.✓ Respuestas muy precisas gracias a su profundo conocimiento del entorno de R.✓ Promueve la generación de código mínimo y viable, sin añadidos innecesarios.
Debilidades	✗ El flujo conversacional puede no ser tan directo para la simple generación de una línea de código.	✗ Menos conversacional. No es ideal para pedir explicaciones largas o debatir un enfoque.
¿Cuándo usarlo?	“Necesito ayuda para pensar cómo empezar este análisis. ¿Qué visualizaciones me sugieres para estas variables?”	“Estoy en medio de este script y necesito una función ggplot para visualizar columna_A vs columna_B.”

Muchas Gracias!!!!

JUEVES 2/10



Estadística aplicada potenciada por las nuevas tecnologías: *integrando la IA generativa*

Mgtr. Carolina Del Pilar Díaz

*IT Patagonia S.A.
AI Model Accelerator
B.A. Argentina.*

Dr. Pablo M. Demetrio

*Profesor Geoestadística
Investigador Adjunto
UNLP | CONICET*

Corrientes, Argentina.

Retomando....



Flujo de trabajo



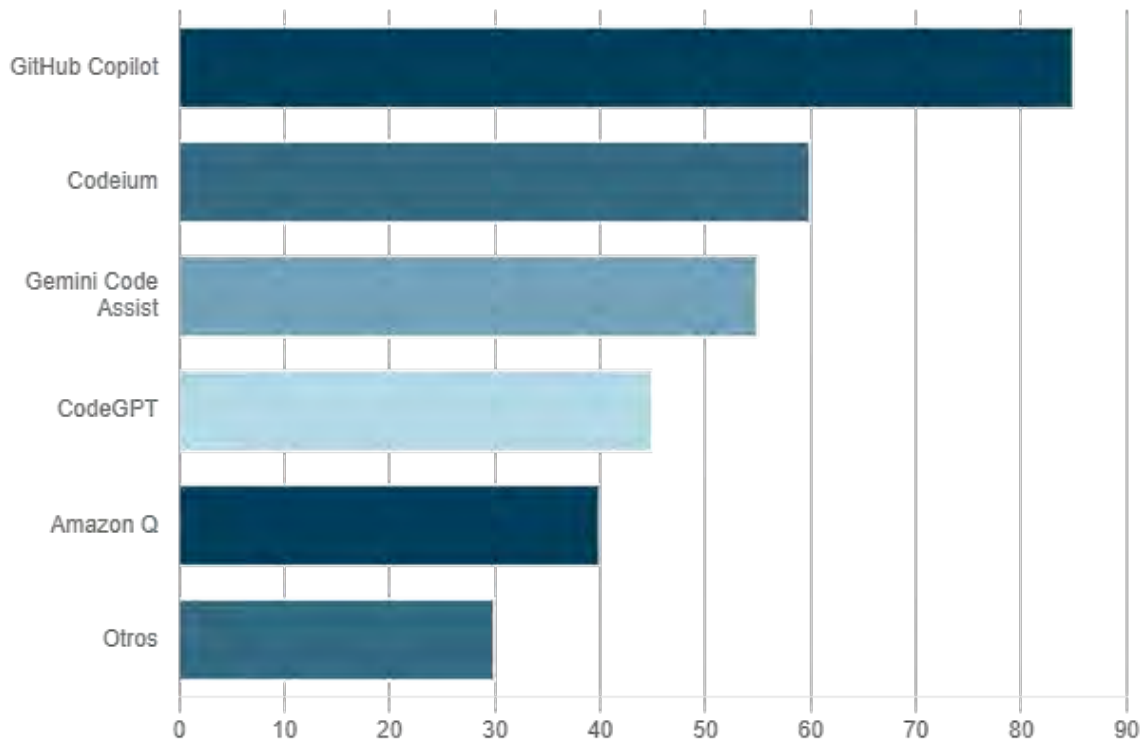
I.Asistentes de IA: Tu Copiloto de Codificación

Integrados a través de extensiones, los asistentes actúan como copilotos inteligentes.



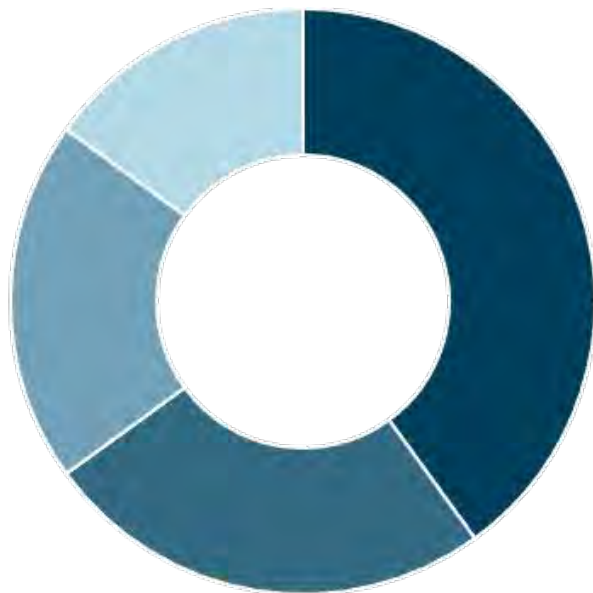
Popularidad Relativa de Asistentes de Código

Aunque la popularidad cambia, herramientas como GitHub Copilot lideran el mercado, seguidas por una variedad de soluciones competitivas que ganan terreno rápidamente.



Distribución de Funciones

- Finalización de Código
- Generación de Código
- Chat Interactivo
- Explicación y Debugging

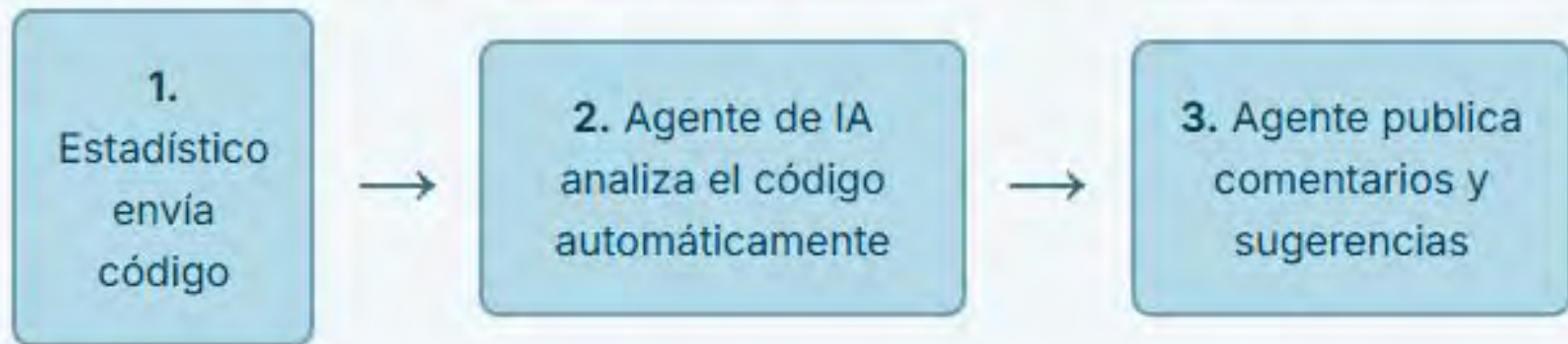


II. Agentes de IA: Tu Equipo de Especialistas Autónomos

A diferencia de los asistentes, los agentes de IA son roles más **autónomos y especializados** diseñados para **ejecutar tareas estructuradas y complejas**



Flujo de Trabajo de un Agente: Revisor de Código Autónomo



Este proceso automatizado ahorra tiempo y asegura la consistencia en la calidad del código, una tarea perfecta para un agente especializado.

Diferencias Claves: Asistente vs. Agente

Asistente (Copiloto)

Rol Principal

Ayuda interactiva, finalización de código y chat.

Naturaleza

Soporte general reactivo.

Ejemplos

GitHub Copilot, Gemini Code Assist.

Agente (Especialista)

Rol Principal

Realiza tareas autónomas y especializadas.

Naturaleza

Proactivo y enfocado en un ámbito específico.

Ejemplos

Agentes de gestión de proyectos de GitHub, Agentes de automatización de Gemini.

I. Protocolo de Contexto de Modelo (MCP)

Es un **estándar abierto** diseñado para que las aplicaciones basadas en **Modelos de Lenguaje Grandes (LLM)**, como los que impulsan a los chatbots avanzados y agentes de IA, puedan comunicarse de forma estandarizada y segura con **datos específicos, estructurados o fijos, aplicaciones y servicios externos directamente en el contexto del modelo.**



II. Ajuste Fino (Fine-Tuning)

Adaptar el modelo a un **dominio, estilo, tono o conjunto de reglas específico** que no estaba bien representado en sus datos de entrenamiento iniciales (ej. jerga legal, formato de informes internos o personalidad de marca).

El proceso **MODIFICA directamente los parámetros** (o pesos) de la red neuronal. Esto **cambia permanentemente la memoria interna y estática** del modelo.

Es un proceso **costoso y lento**, pero el resultado es un modelo que genera respuestas que son nativas del nuevo dominio.



III. Generación aumentada de recuperación (RAG)

La generación aumentada de recuperación combina el poder de grandes modelos de lenguaje con la capacidad de **recuperar y referenciar información externa en tiempo real**

RAG, que es una solución de conocimiento externo y temporal, el Ajuste Fino es una solución de **conocimiento interno y permanente**.



¿De qué manera
potenciamos nuestro
conocimiento?

¡R y Python! La dupla perfecta en VS Code

Extensiones a full



Agentes de VS Code

R + Python: Un Ecosistema para la Investigación Rigurosa

1. Manipulación de Datos y Preprocesamiento Avanzado

Python, con librerías como **pandas** y **numpy**, es esencial para la limpieza y transformación de datasets masivos. Es una herramienta clave para preparar datos para modelos en R.

Ingeniería de Features: Texto

Extrae features de texto para analizar notas de campo, reportes de estudios o artículos científicos.

Ingeniería de Features: Imagen/Señal

Procesa imágenes satelitales y señales biológicas para crear características relevantes.

Ingeniería de Features: Temporal

Crea features a partir de series de tiempo para predecir el rendimiento de un cultivo o patrones de migración.

Gestión de Bases de Datos

Facilita la interacción con bases de datos como **MongoDB** para datos semi-estructurados.

2. Generación de Datos Sintéticos y Simulación

Python es una herramienta poderosa para crear datasets sintéticos, lo cual es crucial para la validación de metodologías y la protección de datos sensibles en la investigación.

Privacidad y Escalabilidad

Crea datos genómicos o de cultivos para compartir con fines de investigación sin exponer información sensible.

Exploración de Escenarios

Genera datos bajo diferentes supuestos para evaluar la robustez de los métodos estadísticos y simular el impacto del cambio climático.

Creación de "Mundos" de Datos

Usa modelos generativos para crear entornos de datos controlados y probar nuevas hipótesis cuando los datos reales son escasos.

3. Modelado Predictivo Avanzado y Machine Learning

Python ofrece acceso a algoritmos de ML de última generación para construir modelos predictivos complejos que identifican patrones sutiles en datos biológicos, complementando los modelos inferenciales de R.

Descubrimiento de Patrones

Explora grandes conjuntos de datos (genómicos, de imágenes) en busca de biomarcadores de enfermedades o patrones de resistencia a plagas.

Optimización Rigurosa

Aplica validación cruzada y tuning de hiperparámetros para optimizar y validar el rendimiento de modelos complejos.

4. Desarrollo de Herramientas y Prototipos Rápidos

Python acelera el ciclo de investigación, permitiendo el desarrollo ágil de nuevos algoritmos y herramientas que pueden ser integrados fácilmente con R a través de **reticulate**.

Creación de Prototipos

Desarrolla rápidamente nuevos algoritmos o funciones estadísticas que luego pueden ser adaptados para usar en R.

APIs para la Investigación


Construye pequeñas APIs para que otros investigadores accedan y usen modelos o análisis específicos, democratizando el acceso a las herramientas.

Python es una herramienta esencial para la **investigación fundamental**: desde la preparación de datos complejos y la simulación, hasta la exploración de patrones predictivos avanzados, lo que enriquece enormemente la rigurosidad y el alcance del trabajo del científico de datos en ciencias de la vida.



Visual Studio
Code

The open source AI code editor

 [Download for Windows](#)

[Web](#), [Insiders edition](#), or [other platforms](#)

By using VS Code, you agree to its [license](#) and [privacy statement](#).

Solo tu cuenta de GitHub.

1. Descarga e Instalación de Software Base



[R-Project \(CRAN\)](#) Añadir R al PATH de Windows



[Descargar Python](#) MARCAR "Add python.exe to PATH"



[Descargar VSC](#) MARCAR "Add python.exe to PATH"



[Descargar Git para Windows](#) conexión github-vscode



[Suscripción Gratuita Github](#) Repositorio de código. Para obtener GITHUB COPILOT

2. Configuración de Jupyter (Python y R)



Instalar el paquete principal de Jupyter en Python

1. Elegir Kernel Python

2. En Powershell ejecutar `pip install jupyter`



Instalar y Registrar el Kernel de R (IRkernel)

ABRIR Consola de R (o la consola de RStudio) y ejecutar los siguientes comandos:

1. Instala el paquete IRkernel `install.packages('IRkernel')`

2. Registra el kernel de R en Jupyter `IRkernel::installspec()`

3. Visual Studio Code. Instalar Extensiones en VSC



Soporte para sintaxis y snippets de R



Soporte principal para el lenguaje



Habilita la creación el uso de R y Python en el mismo ambiente, proyecto



GITHUB COPILOT, agente AI, orquestador MCP



GEMINI agente AI. Google AI Studio, puedes obtener una clave API desde "Get API key"

4. Activación de GitHub Copilot

- **Iniciar Sesión en GitHub en VSC:** VSC pedirá iniciar sesión y **autorizar VSC** a acceder a la [cuenta de GitHub](#)
- **Verificar Suscripción:** **GitHub Copilot** requiere una **suscripción activa** asociada a la cuenta de GitHub que se realiza teniendo una cuenta en gmail.
- **Verificación Final:** El icono de Copilot se iluminará en la barra de estado inferior de VSC, indicando que está listo.

5. Orquestación MCP

Orquestador de Agentes en VSC (Protocolo MCP)

El "**Orquestador**" es el "**Agent Mode**" de VSC (parte de Copilot Chat), que utiliza el protocolo **MCP (Model Context Protocol)** para coordinar agentes especializados llamados **VSCo de MCP Servers**.

Demo

Análisis de Predicción de Rendimiento de Cultivos.

Con el crecimiento continuo de la población mundial y los efectos del cambio climático, comprender y predecir el rendimiento agrícola.

Objetivos del Análisis

- **Modelado Predictivo: Desarrollar modelos para predicción de rendimiento**
- **Análisis de Correlaciones: Identificar factores críticos que afectan el rendimiento**
- **Segmentación Geográfica: Entender patrones regionales**
- **Análisis Temporal: Detectar tendencias y estacionalidad**
- **Optimización Agrícola: Proporcionar insights para mejores prácticas**



“Vivimos en una sociedad profundamente dependiente de la ciencia y la tecnología, y en la que nadie sabe nada de estos temas. Ello constituye una fórmula segura para el desastre.”

Carl Sagan

El mundo y sus demonios

!MUCHAS GRACIAS!



