

Ingreso a Google Colaboratory

Introducción al entorno de trabajo en análisis de datos

Manuel Fernández San Martín
Juan Ignacio da Torre

Laboratorio de Métodos Cuantitativos
Facultad de Ciencias Económicas
Universidad de Buenos Aires

Primer Cuatrimestre de 2026

Objetivo de la clase:

Conocer el entorno Colab y aprender a explorar datasets de forma eficiente

Agenda

- 1 Motivación
- 2 Google Colaboratory
- 3 Primeros pasos
- 4 La Notebook
- 5 Hardware y recursos
- 6 Trabajando con datos
- 7 Explorador de Datos
- 8 Buenas prácticas
- 9 Cierre

¿Por qué Google Colaboratory?

El análisis de datos requiere:

- Entornos reproducibles
- Bajo costo
- Facilidad de colaboración

Google Colab resuelve:

- **Instalación:** No necesitas configurar nada
- **Recursos de cómputo:** Acceso gratuito a CPU/GPU/TPU
- **Compartición:** Colaboración en tiempo real
- **Lenguaje:** R o Python

★ **Nos vamos a enfocar en pensar y analizar datos, no en configurar la máquina**

¿Qué es Google Colaboratory?

Definición

Entorno **gratuito** de Jupyter Notebook que permite programar y ejecutar código de **Python** o **R** en el navegador, desde la nube de Google.

Usos principales:

- Análisis de datos
- Machine Learning
- Educación
- Prototipado rápido

Ventajas principales de Google Colab

Sin configuración

No requiere instalación. Todo funciona desde el navegador.

Recursos y almacenamiento:

- Acceso gratuito a CPU, GPU y TPU
- Integrado con Google Drive
- Guarda automáticamente

Colaboración y contenido:

- Compartir notebooks fácilmente
- Colaboración en tiempo real
- Combina código, texto y resultados
- Integrado con Gemini / IA

Más información: <https://colab.research.google.com/>

Primeros pasos en Colab

Paso a paso:

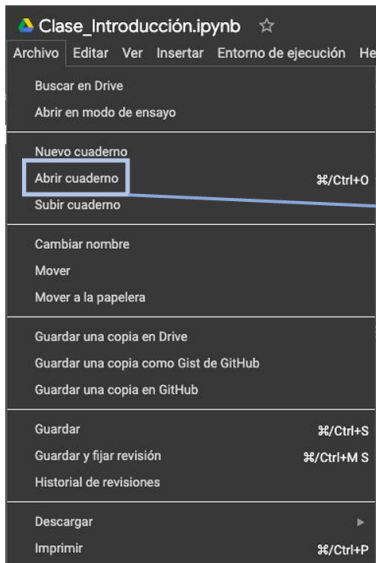
- 1 Acceder desde Google Drive
- 2 Crear o abrir una notebook
- 3 Entender la interfaz
- 4 Comenzar a programar

Activar Google Colaboratory

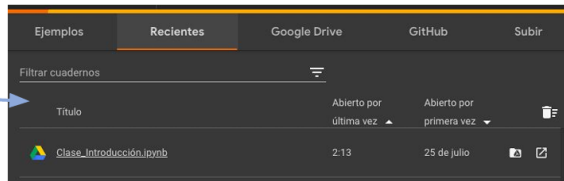
Desde tu Drive personal:

- Clic derecho → “Más” → “Conectar más aplicaciones”
- Buscar “Colaboratory” → Instalar

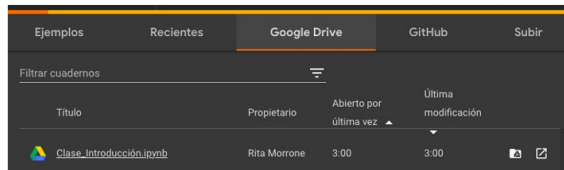
Cómo abrir una notebook existente



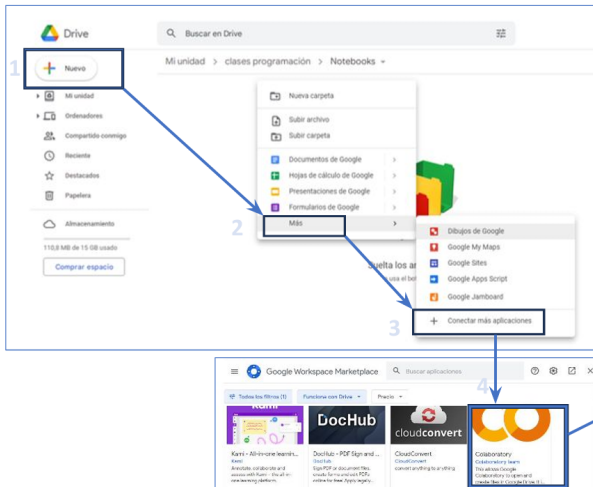
Si tenemos una notebook creada, tener en cuenta que son archivos de extensión .ipynb, se puede abrir de esta manera.



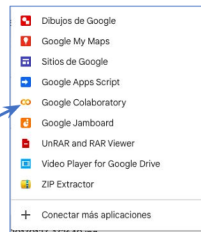
También podemos buscar en el drive personal



Paso 1 - Activar Google Colaboratory



Ingresando al drive personal, utilizamos el menu y elegimos la opción “Conectar más aplicaciones” Luego buscamos Colaboratory y elegimos la opción Instalar



Anatomía de una Notebook

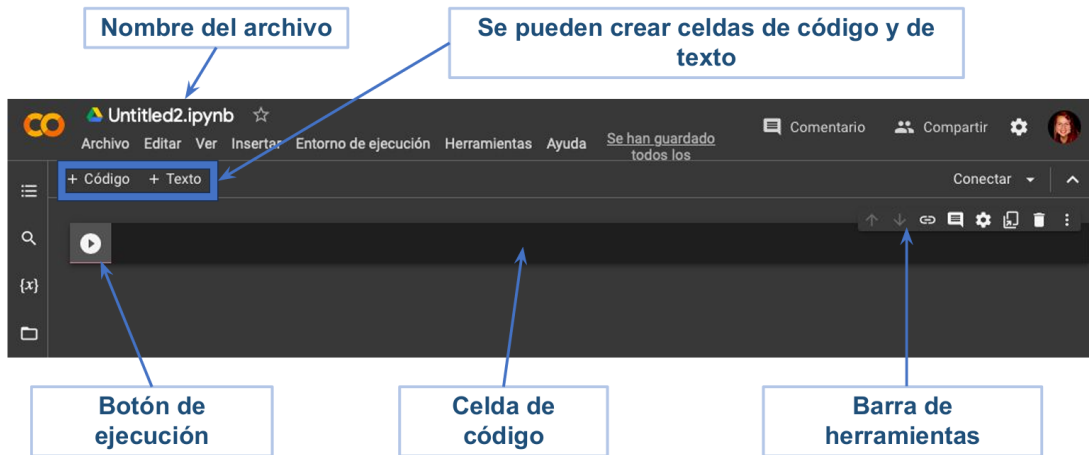
Elementos principales:

- **Nombre del archivo** (extensión `.ipynb`)
- **Barra de herramientas** (arriba)
- **Celdas de código** (donde escribís Python)
- **Celdas de texto** (explicaciones en Markdown)
- **Botón de ejecución** (▷ Ejecutar)
- **Salida** (debajo de cada celda)

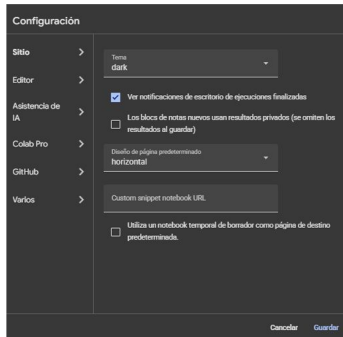
★ Concepto:

Una notebook es un *documento vivo*: código + explicación + resultados

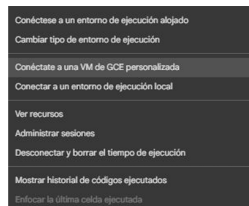
Anatomía de una Notebook



Configuración del entorno



Configuración a gusto de cada persona.



Ajuste de recursos según el proyecto, y status de uso

Configuración personalizable según el proyecto y estado de uso

Tipos de celdas

Celda de código

- Python o R
- Variables persistentes
- Ejecutar: Shift + Enter

Celda de texto (Markdown)

- Explicaciones
- Fórmulas (LaTeX)
- Imágenes y docs

Importante

- El código puede estar en una o varias celdas
- Las variables persisten en toda la notebook
- La salida aparece debajo de cada celda

Ejemplo de fórmula:


$$EC = \int_0^{q^*} [D(q) - p^*] dq$$

⇒ **Documentación reproducible**

Variables y ejecución de código



```
[8] x=3
x
3
y=x+5
y
8
```



```
[15] saludo="Hola Mundo!!!, estamos programando"
saludo
'Hola Mundo!!!, estamos programando'
```

- * Se pueden escribir celdas de textos o celdas de código
- * El código se puede escribir en una o varias celdas.
- * Luego de ejecutar la misma las variables y funciones que creamos quedan disponibles en la notebook.
- * La salida de cada ejecución se muestra a continuación de la celda de código.

Importante:

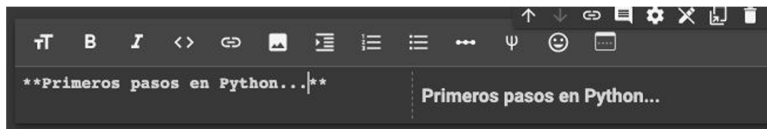
- El código puede estar en varias celdas
- Las variables persisten en la notebook
- La salida aparece debajo de cada celda

Celdas de texto con Markdown y LaTeX

En las celdas de texto se puede agregar explicaciones, imágenes que complementan el desarrollo de la notebook.

También se pueden escribir fórmulas utilizando LaTeX.

A medida que escribimos, en el panel derecho se visualiza como será mostrado.



The screenshot shows a Jupyter Notebook text cell. The top toolbar includes icons for text formatting (bold, italic, code), linking, inserting images, lists, and other controls. The input area on the left contains the text `**Primeros pasos en Python...**code>. The output area on the right displays the rendered text: Primeros pasos en Python...`



The screenshot shows a Jupyter Notebook text cell. The top toolbar is identical to the one above. The input area on the left contains the LaTeX code: `$\forall (x,y) \in \mathbb{R}^2 / \sqrt{x^2+y^2} < k$code. The output area on the right displays the rendered mathematical formula: $\forall (x, y) \in \mathbb{R}^2 / \sqrt{x^2 + y^2} < k$`

Recursos de hardware en Colab

Configuración personalizable:

- **RAM:** Memoria momentánea (para procesos en ejecución)
- **Disco:** Almacenamiento total temporal
- **Hardware:** Cantidad de recursos utilizados

Tipos de procesamiento

CPU (Default) Microprocesador estándar. Tareas secuenciales y complejas.

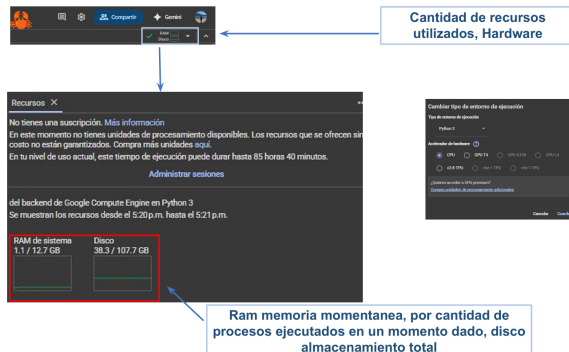
GPU Ideal para IA, imágenes y cálculos intensivos. Procesamiento en paralelo.

TPU Tensor Processing Unit. Deep learning avanzado.

Concepto clave

Los recursos son **temporales** e **intercambiables**. Lo importante es la **lógica del análisis**.

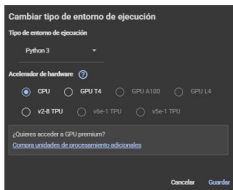
Monitoreo de recursos



RAM: Memoria momentánea para procesos en ejecución

Disco: Almacenamiento total temporal

Selección de tipo de procesamiento



En cuanto al procesamiento podemos utilizar procesamiento de CPU (Default), es decir de un microprocesador. Tareas secuenciales y complejas

Tenemos opciones como GPU ideal para entrenamiento de IA, imágenes y cálculos matemáticos. Ideal para mucho procesamiento en paralelo. Y por ultimo TPU

Introducción a Cloud TPU

[Enviar comentarios](#)

Las unidades de procesamiento tensorial (TPU) son circuitos integrados personalizados específicos de aplicaciones (ASIC) de Google que se utilizan para acelerar las cargas de trabajo de aprendizaje automático. Para obtener más información sobre el hardware de las TPU, consulta [Arquitectura de las TPU](#). [Cloud TPU es un servicio web que pone las TPU a disposición como recursos de procesamiento escalables en Google Cloud.](#)

Las TPU entrenan tus modelos de manera más eficiente con hardware diseñado para realizar operaciones de matrices grandes que suelen encontrarse en los algoritmos de aprendizaje automático. Las TPU tienen memoria de gran ancho de banda (HBM) en el chip, lo que te permite usar modelos y tamaños de lotes más grandes. Las TPU se pueden conectar en grupos llamados slices que aumentan la escala de tus cargas de trabajo con pocos o sin cambios de código.

Carga de datos en Colab

Formas comunes de cargar datos

- **Subir archivos** (CSV, Excel, etc.)
Temporales; útil para pruebas rápidas
- **Conectar Google Drive**
Persistente; acceso a todos tus archivos
- **APIs y URLs**
Uso avanzado

Panel izquierdo

- Carpetas** Archivos subidos o del Drive
- Variables** Objetos en memoria

Ojo

El panel de variables puede afectar el rendimiento.

DataFrame: la unidad central del análisis

Concepto

Un **DataFrame** es una tabla de datos donde:

- **Filas** = observaciones
- **Columnas** = variables

Ejemplo básico:

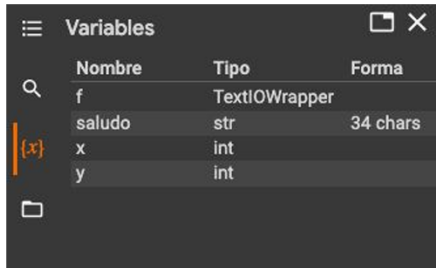
```
import pandas as pd

# Cargar datos
df = pd.read_csv('datos_supermercado.csv')

# Ver primeras filas
df.head()
```

Los DataFrames son la estructura central en análisis de datos

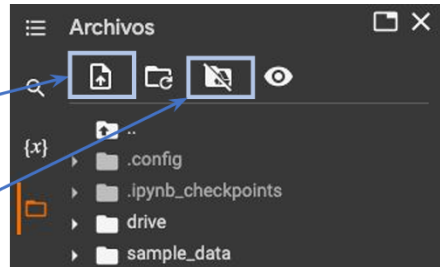
Panel lateral: Archivos y Variables



Aquí podemos ver el enviroment con las variables que vamos creando, ojo si lo dejamos abierto puede disminuir el rendimiento.

Aquí vemos los archivos, se pueden subir cada vez desde la computadora,

o también se puede optar por conectar al Drive personal y disponer de todos los archivos.



Nueva Feature: Explorador de Datos (Data Inspector)

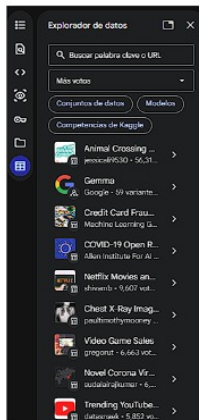
¿Qué es?

Una herramienta **visual integrada** en Colab que permite explorar un DataFrame sin escribir código adicional.

¿Dónde aparece?

- Se abre al visualizar un DataFrame
- Panel lateral derecho
- Muestra columnas y tipos automáticamente

acceso a data frames de prueba



Bases de datos ofrecidas por google para análisis

Descripción sobre los datatypes y columnas del Data frame (DF)

df	Filas
gender	OBJECT
race/ethnicity	OBJECT
parental level of educati...	OBJECT
lunch	OBJECT
test preparation course	OBJECT
math score	INT64
reading score	INT64

¿Qué información nos da el Explorador?

Ejemplo con dataset de estudiantes:
Columnas categóricas (object):

- gender
- race/ethnicity
- lunch
- parental level of education

Columnas numéricas (int64):

- math score
- reading score
- writing score

★ Concepto clave:

Antes de analizar, hay que entender qué tipo de datos tenemos

¿Por qué explorar datos antes de analizar?

Detectar problemas

- Tipos de datos incorrectos
- Variables mal clasificadas
- Valores nulos o faltantes
- Inconsistencias

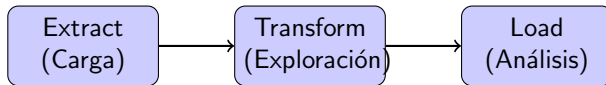
Beneficios

- ✓ Evitar errores
- ✓ Ahorrar tiempo
- ✓ Mejores decisiones
- ✓ Entender estructura

Idea clave

Datos mal entendidos \Rightarrow conclusiones erróneas

Relación con el flujo de datos (ETL)



- **Extract:** Cargamos el dataset
- **Transform:**
 - Verificar tipos de datos
 - Limpiar valores nulos
 - ← **Acá ayuda el Data Inspector**
- **Load:** Análisis, visualización, modelado

Comparación: Antes vs Ahora

Antes del Data Inspector

Teníamos que escribir:

```
1 df.info()  
2 df.dtypes  
3 df.describe()  
4 df.head()  
5 df.isnull().sum()
```

Problemas:

- Más fricción
- Menos intuitivo
- Más código

Ahora con Data Inspector

Ventajas:

- Vista rápida
- Exploración visual
- Sin código adicional
- Ideal para principiantes

✓ Mejor experiencia
✓ Más eficiente

Buenas prácticas usando Colab

1 Documentar cada paso

- Usar celdas de texto (Markdown)
- Explicar la lógica del análisis
- Agregar comentarios en el código

2 Nombrar bien los objetos

- DataFrames: `df_ventas`, `df_clientes`
- Variables descriptivas

3 Explorar antes de modelar

- Usar el Data Inspector
- Verificar tipos de datos
- Analizar valores nulos

4 No confiar ciegamente en los datos

- Siempre validar
- Buscar inconsistencias
- Cuestionar outliers

Ideas clave para llevarse

- ❶ **Colab es una herramienta, no el objetivo**
 - Lo importante es el análisis, no el entorno
- ❷ **El análisis empieza entendiendo los datos**
 - Exploración antes de modelado
- ❸ **El Data Inspector:**
 - Reduce errores
 - Mejora comprensión
 - Facilita el aprendizaje
- ❹ **Documentación reproducible**
 - Notebooks como documentos vivos
 - Código + explicaciones + resultados

Próximos pasos en el laboratorio

Temas por venir:

- 1 Limpieza de datos
- 2 Estadística descriptiva
- 3 Visualización avanzada
- 4 Modelos de regresión
- 5 Machine Learning básico

*El camino del análisis
de datos arranca acá!*

¡Gracias!

¿Preguntas?

Referencias y recursos

Documentación oficial:

- <https://colab.research.google.com/>
- <https://pandas.pydata.org/docs/>
- <https://numpy.org/doc/>

Bibliografía recomendada:

- *Viajar al futuro (y volver para contarlo)* - Walter Sosa Escudero
- *Big data* - Walter Sosa Escudero
- *Tecnofeudalismo, Crítica de la economía digital* - Cédric Durand

Consejo final:

La mejor forma de aprender es practicando.
Abrí Colab y empecé a explorar datos.