



02. Uso de LLMs para generar código

Ciencia de datos para la toma de decisiones II

Jorge
Juvenal
Campos Ferreira

juvenal.campos@tec.mx

Programa de la clase

- Ejercicios para realizar un prompt
- Introducción a **Claude Code**
- Introducción super básica a la terminal
- Ejercicio guiado de generación de código
- Introducción a RStudio
- Proyectos en R

Fundamentos de ingeniería de prompts



Prompt: Es la instrucción o pregunta que le damos al modelo para que genere una respuesta.

Ingredientes mínimos de un buen prompt

Enfoque: La actividad que queremos que realice el modelo.

Contexto: La información previa que queremos que el modelo tome en consideración al momento de procesar el prompt

Límites: Lo que el modelo debe o no debe incluir en su respuesta

Rol: El rol que debe asumir el modelo al momento de dar la respuesta.

Ejemplos de Prompts

Prompt sencillo

Realiza una investigación completa sobre el INFONAVIT de México. Incluye su historia, funcionamiento actual, noticias recientes y cualquier dataset disponible público. Proporciona información detallada y actualizada.

¿Cuál dará el mejor resultado?

Prompt más elaborado

```
<rol>
Actúa como un investigador especializado en instituciones de vivienda de México con experiencia en análisis de políticas públicas y manejo de datos gubernamentales.
</rol>

<enfoque>
Realizar una investigación exhaustiva y estructurada sobre el Instituto del Fondo Nacional de la Vivienda para los Trabajadores (INFONAVIT), abarcando múltiples dimensiones: histórica, institucional, operativa y de disponibilidad de datos.
</enfoque>

<contexto>
El INFONAVIT es una institución tripartita mexicana creada en 1972 que administra el fondo de vivienda de los trabajadores formales. Es crucial para el sector habitacional mexicano y maneja grandes volúmenes de datos sobre créditos, viviendas y trabajadores cotizantes. La investigación debe considerar su evolución, reformas recientes, impacto socioeconómico y transparencia en datos.
</contexto>

<tareas_específicas>
1. Investigar la historia institucional del INFONAVIT desde 1972 hasta la actualidad
2. Analizar noticias y desarrollos recientes (últimos 2 años)
3. Identificar y catalogar todos los datasets públicos disponibles
4. Examinar programas actuales y productos crediticios
5. Evaluar el marco regulatorio y reformas recientes
</tareas_específicas>

<limites>
- NO incluir información de otras instituciones de vivienda sin relación directa con INFONAVIT
- NO especular sobre datos no públicos o confidenciales
- NO incluir opiniones políticas partidistas
- Sí incluir únicamente fuentes verificables y oficiales
- Sí proporcionar enlaces directos a datasets cuando sea posible
- Sí mantener un enfoque analítico y objetivo
</limites>

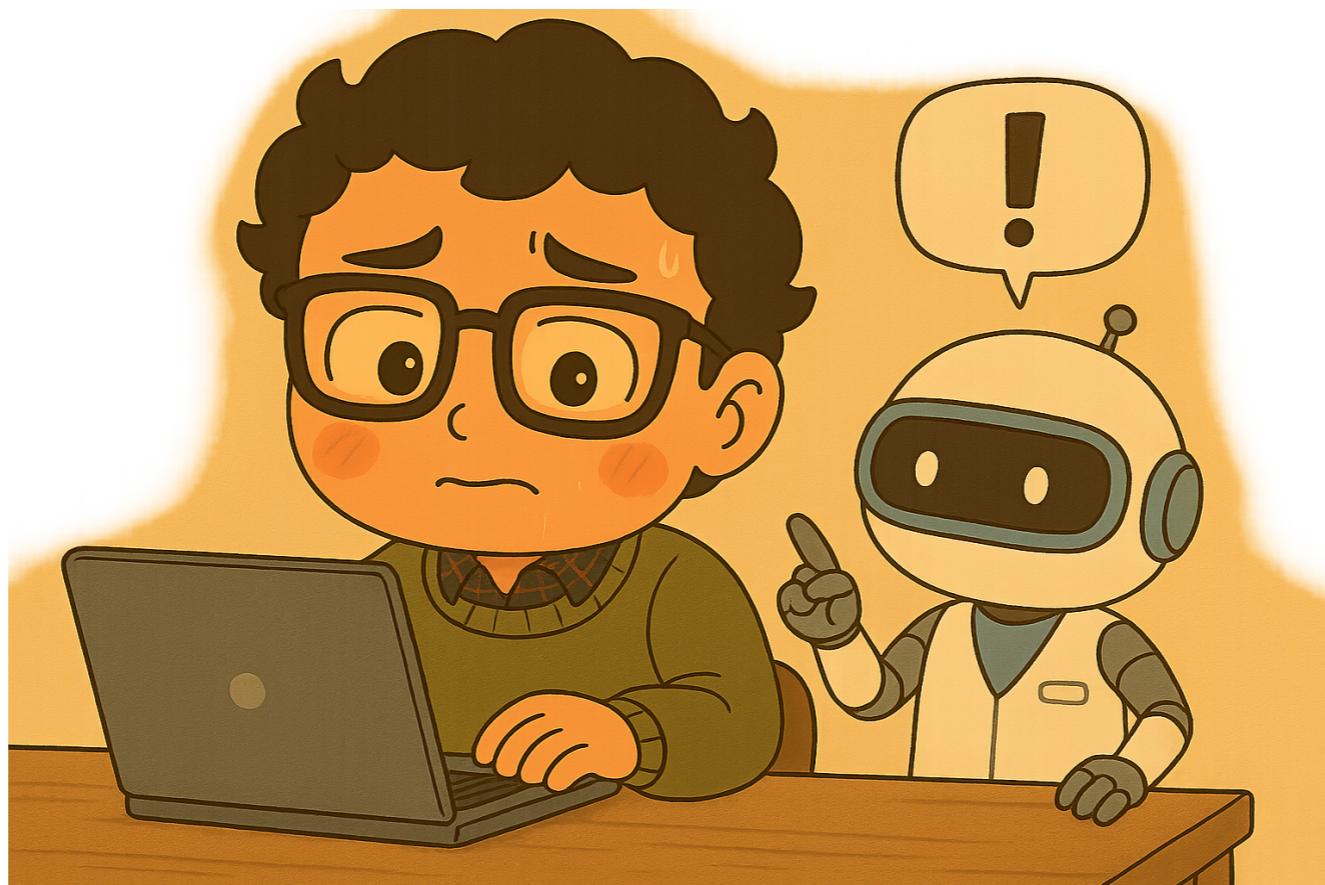
<formato_salida>
Estructura la investigación en las siguientes secciones:
1. Resumen ejecutivo
2. Historia institucional
3. Marco legal y regulatorio actual
4. Programas y productos actuales
5. Noticias y desarrollos recientes (2023-2025)
6. Datasets disponibles (con URLs directas)
7. Análisis de transparencia y acceso a información
8. Conclusiones y recomendaciones para investigadores
</formato_salida>

<fuentes_prioritarias>
- Sitio oficial de INFONAVIT
- Portal de datos abiertos del gobierno mexicano
- INAI (Instituto Nacional de Transparencia)
- Informes anuales oficiales
- Boletines de prensa institucionales
- Bases de datos del IMSS relacionadas
</fuentes_prioritarias>
```

Meta prompting

El **metaprompting** es una técnica avanzada de ingeniería de prompts que utiliza un modelo de lenguaje grande (LLM) para generar, optimizar o refinar prompts para otros LLMs.

En esencia, se trata de **usar la IA para ayudar a crear prompts más efectivos**, lo que permite obtener mejores resultados del modelo.



Ejercicio

Redacte un *prompt* para investigar un tema que le sea de interés o para generar un texto que le sea de interés.

Redáctelo tomando en cuenta **Enfoque, Contexto, Límites y Rol**.

Utilice la sintaxis XML del ejemplo anterior (*envolver cada elemento en los símbolos “>” y “<”*). ¿Considera que redactar este prompt le representa alguna ventaja?

Una vez que lo termines, pídele al LLM que tome el rol de un “ingeniero de prompt” y que te mejore el prompt que acabas de escribir.

LLMs para generar código

Modelos de generación de código

Los LLMs pueden escribir, corregir y optimizar código porque fueron entrenados no solo con lenguaje natural, sino también con **repositorios de código** y documentación técnica.

Aplicaciones:

- **Autocompletar** y escribir funciones enteras.
- Detectar y corregir errores (*debugging*).
- Explicar código existente.
- Convertir descripciones en código (prompt → script).
- Traducir entre lenguajes de programación.

Modelos de generación de código

- Hay muchas herramientas para generar código con IA. ChatGPT, Claude, Gemini o Grok funcionan bien
- También hay herramientas especializadas para mezclar IA y código, como Cursor, o CLIs (*Command Line Interface*), como Claude Code o Gemini CLI.
- Las herramientas especializadas en código facilitan la programación, y en algunos casos ejecutan y prueban el código.



Modelos de generación de código

- Mi herramienta favorita (ahorita a agosto del 2025) es Claude Code.
- Es un CLI que permite acceder a carpetas de trabajo, leer y escribir archivos y también generar código.
- Claude Code ha tenido un buen desempeño al momento de trabajar con código de R, y particularmente en generar aplicaciones shiny básicas.

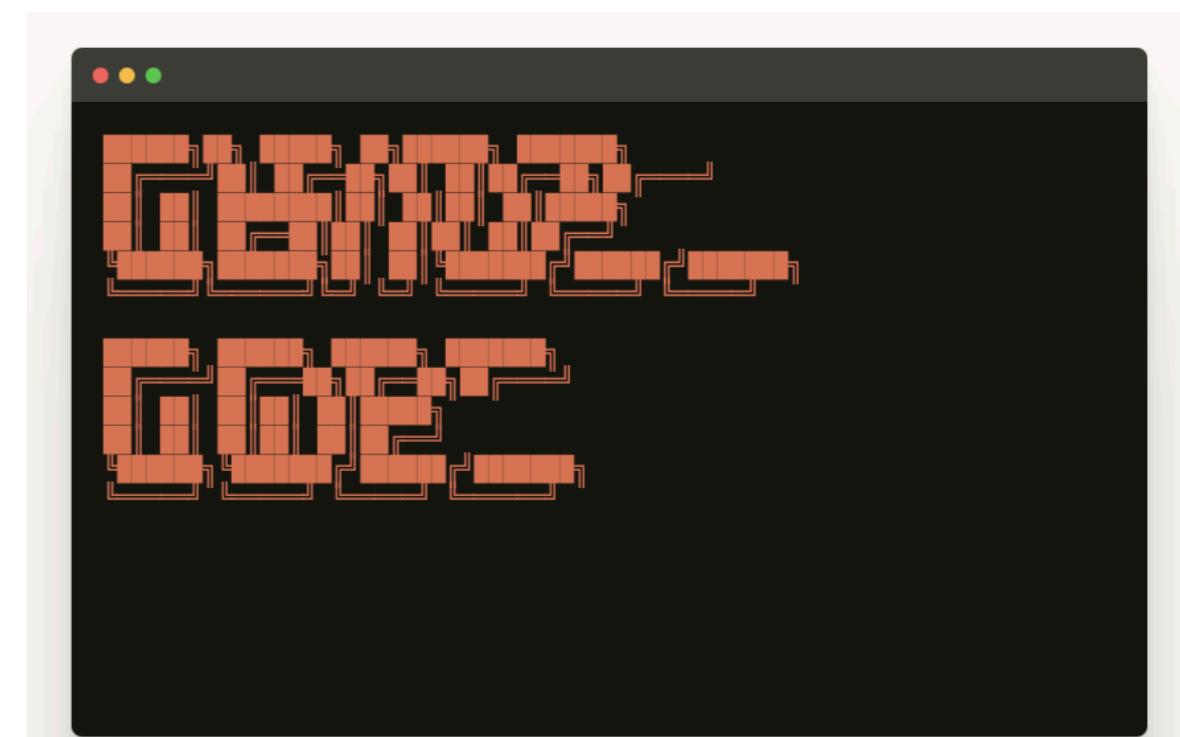


Claude Code

- Instrucciones de instalación: [https://www-anthropic-com.translate.goog/claude-code?
_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc](https://www-anthropic-com.translate.goog/claude-code?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc)

Se necesita usar terminal.

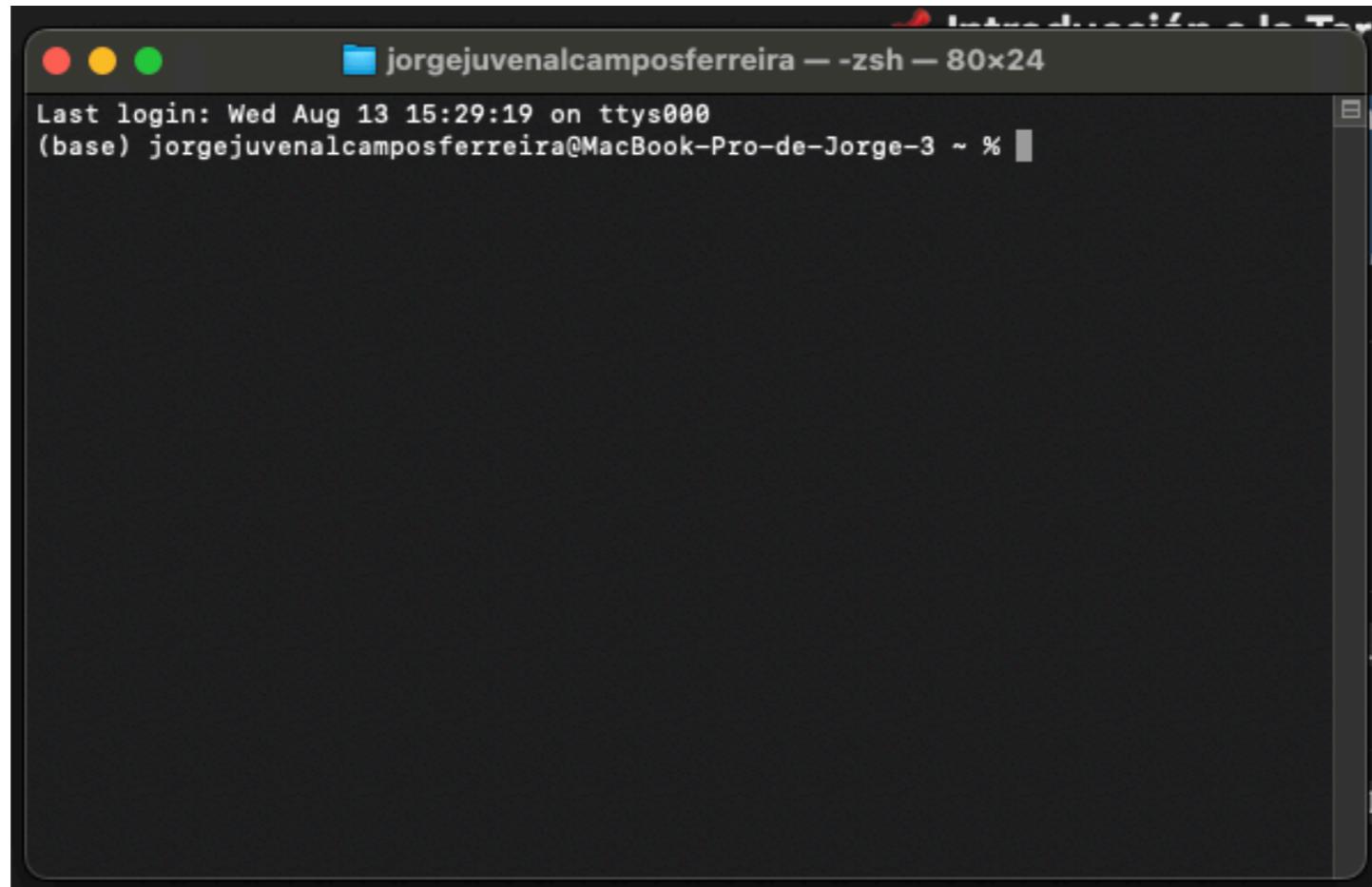
Antes de la instalación, hay que instalar Node.js 18+



Terminal

La **terminal** (o línea de comandos) es una herramienta para **interactuar directamente con el sistema operativo** mediante texto, sin usar ventanas o íconos.

Permite navegar por carpetas, ejecutar programas y administrar archivos con comandos.



Comandos básicos

Comando	Mac / Linux	Windows (PowerShell / CMD)	Descripción
cd	cd nombre_carpeta	cd nombre_carpeta	Cambia de directorio (moverse entre carpetas).
ls	ls o ls -l	dir	Lista los archivos y carpetas en el directorio actual.
pwd	pwd	cd (sin parámetros)	Muestra la ruta completa de la carpeta actual.
clear	clear	cls	Limpia la pantalla de la terminal.
mkdir	mkdir nombre_carpeta	mkdir nombre_carpeta	Crea una nueva carpeta.
rm	rm archivo.txt	del archivo.txt	Elimina un archivo.

Ejercicio guiado

Instale Claude Code o Gemini en su computadora
Ejecutelo y corra un prompt que genere un archivo para generar una
gráfica básica en ggplot.
Ejecute el código y verifique que funcione.

Vibe Coding



Es una **técnica de programación** dependiente de IA en la que una persona describe un problema en unas pocas oraciones como instrucciones para un LLM.

El LLM genera software, cambiando el rol del programador de la codificación manual a la guía, prueba y refinamiento del código fuente generado por IA

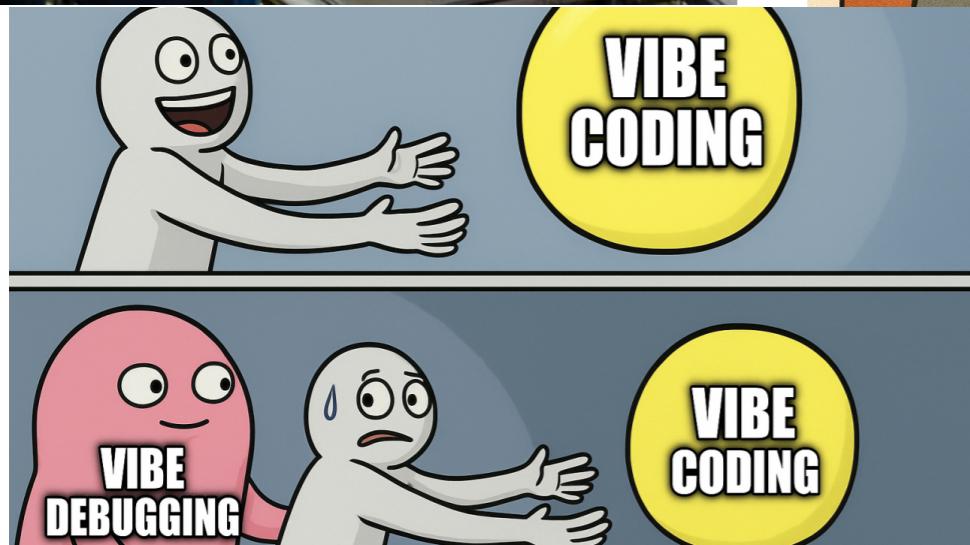
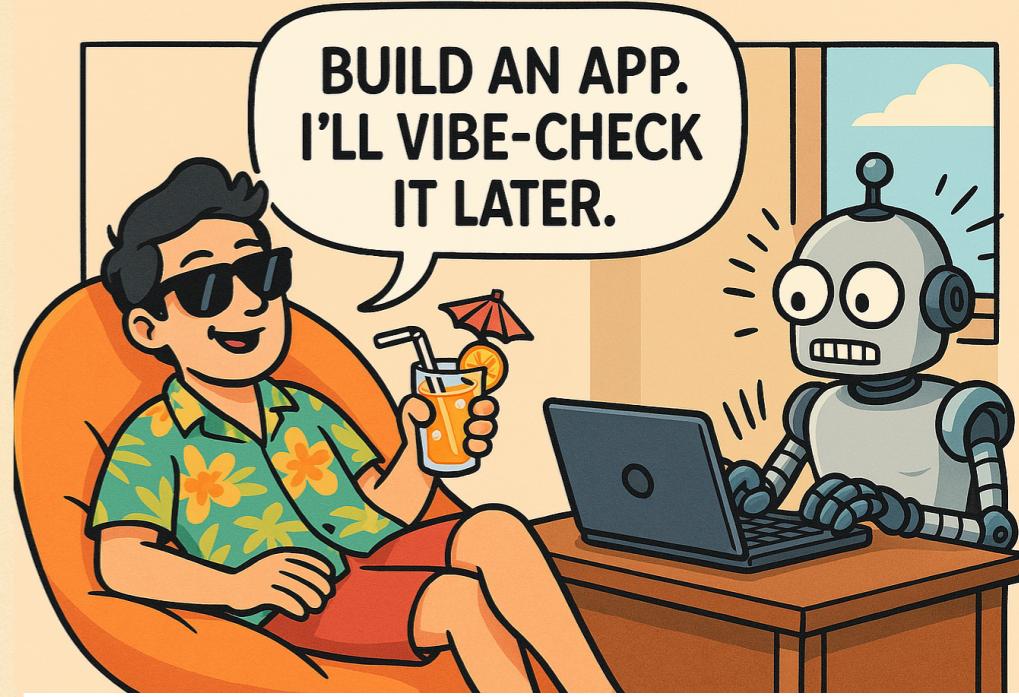
Vibe Coding



El **vibe coding** es un término acuñado por Andrej Karpathy (cofundador de OpenAI y ex-líder de IA en Tesla) en febrero de 2025. Como él mismo lo describió: "**There's a new kind of coding I call 'vibe coding', where you fully give in to the vibes, embrace exponentials, and forget that the code even exists**"



Vibe Coding



Instalar Github Desktop



Y hacer una cuenta de Github



Tecnológico
de Monterrey

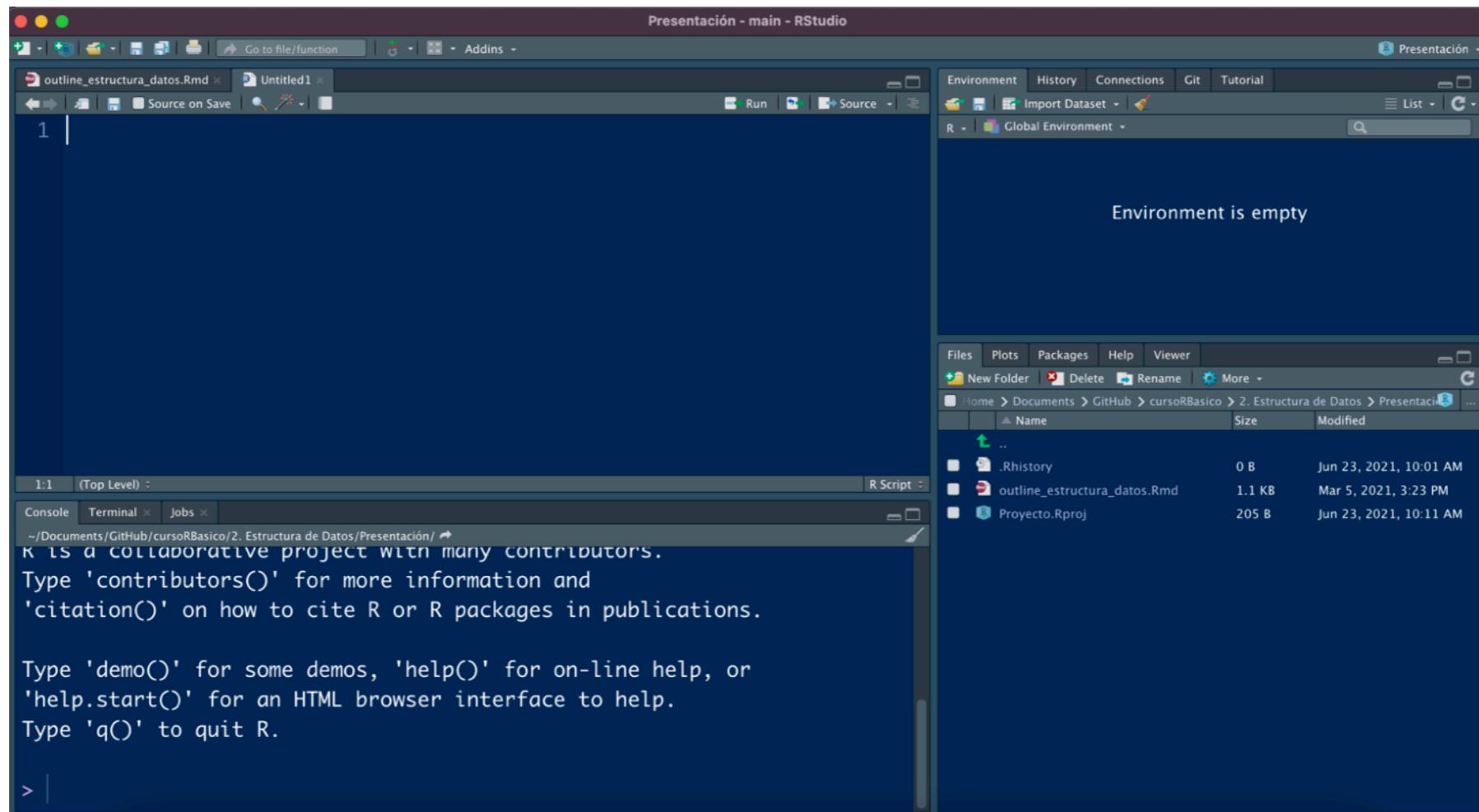
02. Introducción a RStudio

Ciencia de datos para la toma de decisiones II

Jorge
Juvenal
Campos Ferreira

 juvenal.campos@tec.mx

RStudio es un programa que provee un entorno de desarrollo (IDE) que nos da las herramientas necesarias para poder programar en R.



Ventanas



The screenshot illustrates the RStudio interface with several windows highlighted:

- Editor de texto**: The top-left window shows R code for data manipulation, with the first 10 lines highlighted by a yellow box.
- Consola**: The bottom-left window shows the output of the R code, with the last few lines highlighted by a green box.
- Visualizador**: The right side of the interface contains two panes:
 - Ambiente**: Shows the global environment with variables like `años`, `pers...`, and `sabe...`.
 - Visualizador**: Shows the file structure of the current project, with files like `.Rhistory`, `outline_estructura_datos.Rmd`, and `Proyecto.Rproj`.

Ventanas



Editor de texto

Sección del programa en la cual registramos las instrucciones que se van a correr en R. Estas instrucciones se guardan en scripts para volver a ellos más adelante.

Acá se pueden escribir códigos de R, HTML, Python, CSS, Markdown, etc.

The screenshot illustrates the RStudio interface with several colored boxes highlighting different components:

- Editor de texto**: The code editor pane, highlighted in yellow, containing R code for data manipulation and analysis.
- Consola**: The console pane, highlighted in green, showing the output of the R code run in the editor.
- Ambiente**: The environment pane, highlighted in pink, displaying the current global variables and their values.
- Visualizador**: The file browser pane, highlighted in red, showing the project structure and files.

The code in the Editor pane:

```
1 # Librerias ----  
2 library(tidyverse)  
3  
4 # Bases de datos ----  
5 personas <- c("Juvenal", "María")  
6 años <- c(29, 30)  
7 escuelas <- c("Colmex", "UNAM")  
8 sexo <- factor(c("M", "F"),  
9                   levels = c("M", "F"))  
10 sabe_r <- c(TRUE, FALSE)
```

The output in the Consola pane:

```
+ library(tidyverse)  
+ personas  
[1] Juvenal  María  
+ años  
[1] 29 30  
+ escuelas  
[1] Colmex  UNAM  
+ sexo  
[1] M F  
Levels: M F  
+ sabe_r <- c(TRUE, FALSE)
```

Ventanas



Consola

Sección en la cual se ejecuta el código que vamos a escribir en el editor de texto.

Igualmente, podemos correr acá código de R que no requerimos guardar para más adelante.

The screenshot shows the RStudio interface with several windows open:

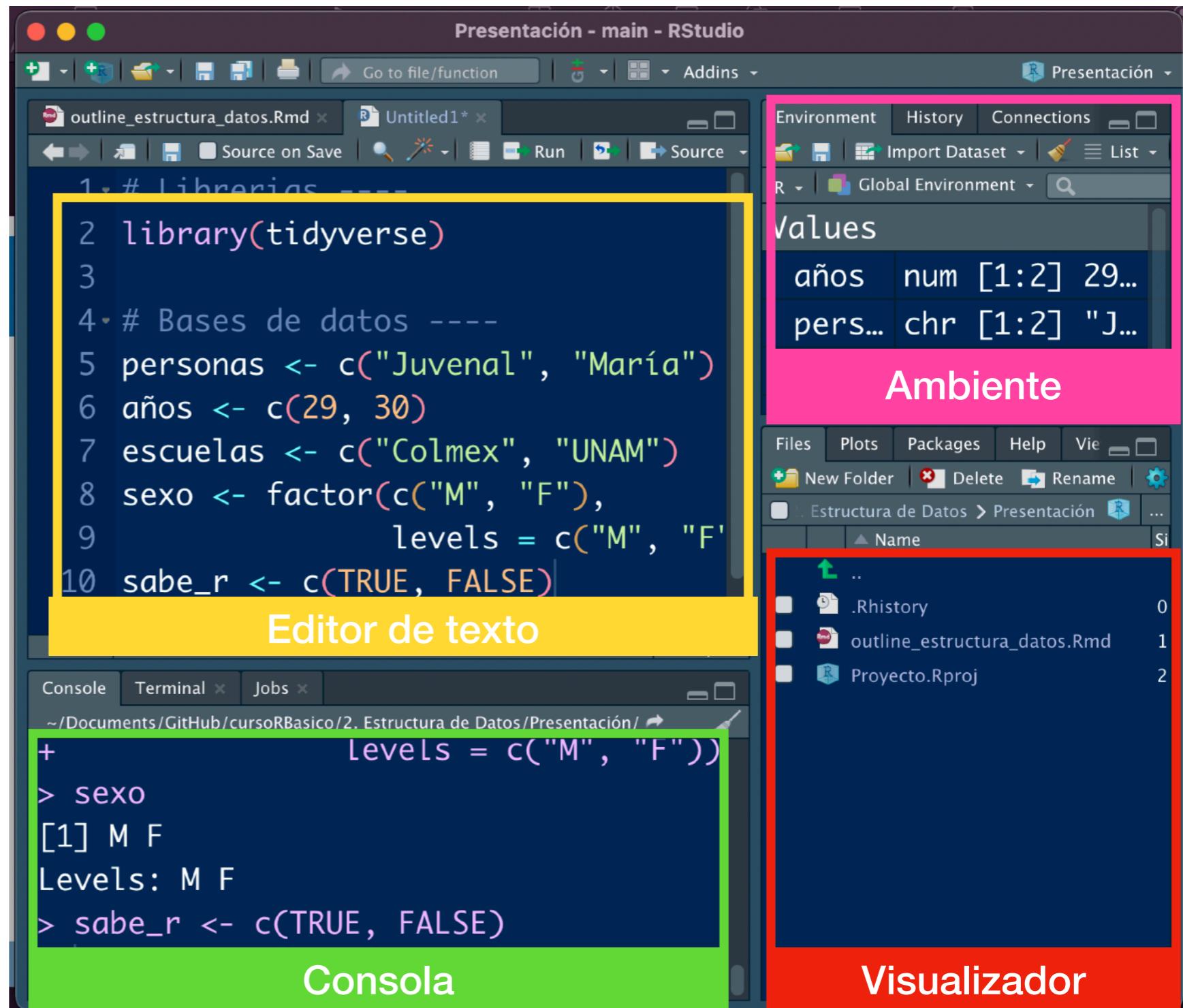
- Editor de texto**: A yellow box highlights the code editor window containing R code for creating variables (personas, años, escuelas, sexo, sabe_r) and setting levels for the sexo factor.
- Consola**: A green box highlights the console window showing the execution of the R code and its output.
- Environment**: A pink box highlights the environment pane showing the global variables: años, pers..., sabe... (with their respective types and values).
- History**: A pink box highlights the history pane showing the commandLevels = c("M", "F") and its output [1] M F.
- Connections**: A pink box highlights the connections pane.
- Global Environment**: A pink box highlights the global environment pane.
- Ambiente**: A pink box highlights the ambiente pane.
- Visualizador**: A red box highlights the visualizer pane showing the project structure: .., .Rhistory, outline_estructura_datos.Rmd, and Proyecto.Rproj.

Ventanas

Visualizador

Sección para visualizar cosas:

- 1) Archivos ubicados en nuestro directorio de trabajo
- 2) Gráficas estáticas generadas con ggplot2 o RBase.
- 3) Las librerías instaladas en nuestro RStudio.
- 4) Las visualizaciones web generadas con R.



Personalización



Para personalizar RStudio
vamos a
Tools > Global Options

Podemos configurar:

- como se visualiza el código,
- los colores de las ventanas,
- el tamaño y fuente de las letras,
- el espacio a ocupar del código,
- las cuentas para publicar resultados, etc.

The screenshot shows the 'Global Options' dialog box in R Studio. The left sidebar lists various categories: General (selected), Code, Console, Appearance, Pane Layout, Packages, R Markdown, Sweave, Spelling, Git/SVN, Publishing, Terminal, Accessibility, and Python. The main area has tabs for Basic, Graphics, and Advanced, with Basic selected. Under the R Sessions section, the default working directory is set to ~/Library/Mobile Documents/com~apple~CloudDocs, with a 'Browse...' button. Under Workspace, there are checkboxes for restoring .RData files and saving workspaces. The History section includes options for saving history and removing duplicates. The Other section contains checkboxes for wrap-around navigation, update notifications, and crash reporting. At the bottom are OK, Cancel, and Apply buttons.



Archivos nativos de R

Al trabajar con R y RStudio, generamos cuatro tipos de archivos nativos de R, estos son los siguientes:

- *.Rproj,
- *.RData,
- *.R y
- *.rds.

Los scripts son aquellos archivos que terminan en *.r