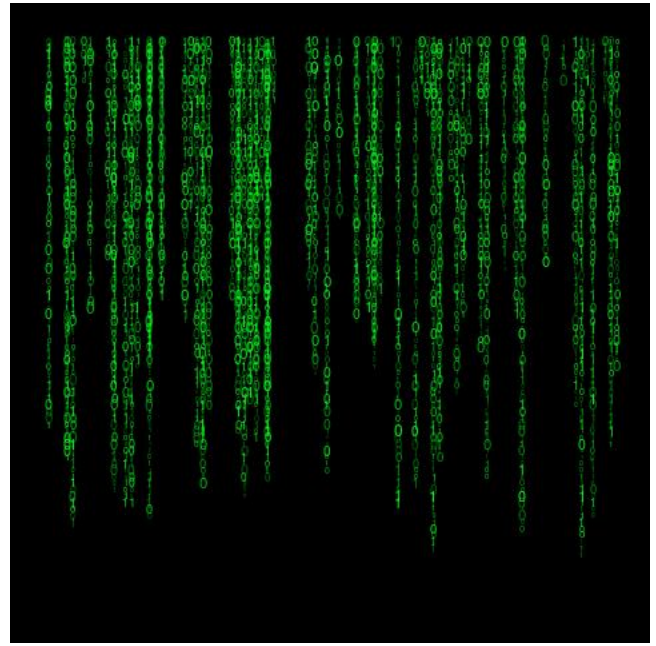
Introducción a la Programación en R

What is R?

- Un lenguaje de programación y un entorno para el análisis estadístico y la creación de gráficos.
- Basado en el lenguaje S - 1976
- Desarrollado en Bell Labs por John Chambers y colegas (el mismo laboratorio que creó los transistores, 6 Nobel de física y 1 de química, lenguaje C, tecnología de telefonía celular, satélites de comunicaciones, etc.)
- R – 1993 de Ross Ihaka y Robert Gentleman
- Software libre y de código abierto que utiliza la Licencia Pública General GNU

Lenguajes de Programación: Abstracción vs Control

Nivel	Características	Ejemplos	Usos comunes
Nivel Alto	• Fácil de aprender • Enfoque en resolución de problemas • Menos control de hardware	R, Python, MATHLAB	•Análisis de datos • Desarrollo Web • Prototipado rápido
Nivel Medio	•Complejidad moderada • Equilibrio de control • Abstracción mixta	C++, Java	• Desarrollo de Software • Programas de sistema •Videojuegos
Nivel bajo	• Sintaxis compleja • Control directo de hardware • Máximo rendimiento	Assembly, Machine Code	• Controladores de dispositivos • Sistemas operativos • Sistemas embebidos



Hecho in R

Por qué R?

- **Software libre**
- **Compatibilidad con cualquier plataforma (Windows, Mac, Linux, Cloud)**
- **Ecosistema de paquetes grandes (CRAN, Bioconductor, GitHub)**
- **Varias opciones para la manipulación de datos**

Software libre (Free Software)

- Free software Foundation's GNU General License

Libertad para:

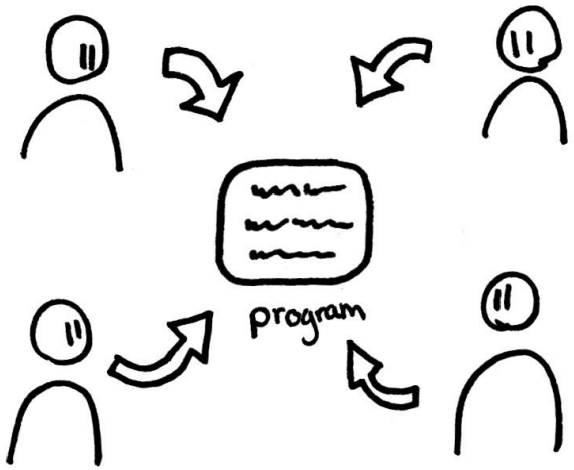
- Ejecutar el programa
- Estudiar cómo funciona
- Redistribuir copias
- Modificar y mejorar el código

https://www.r-project.org/about.html

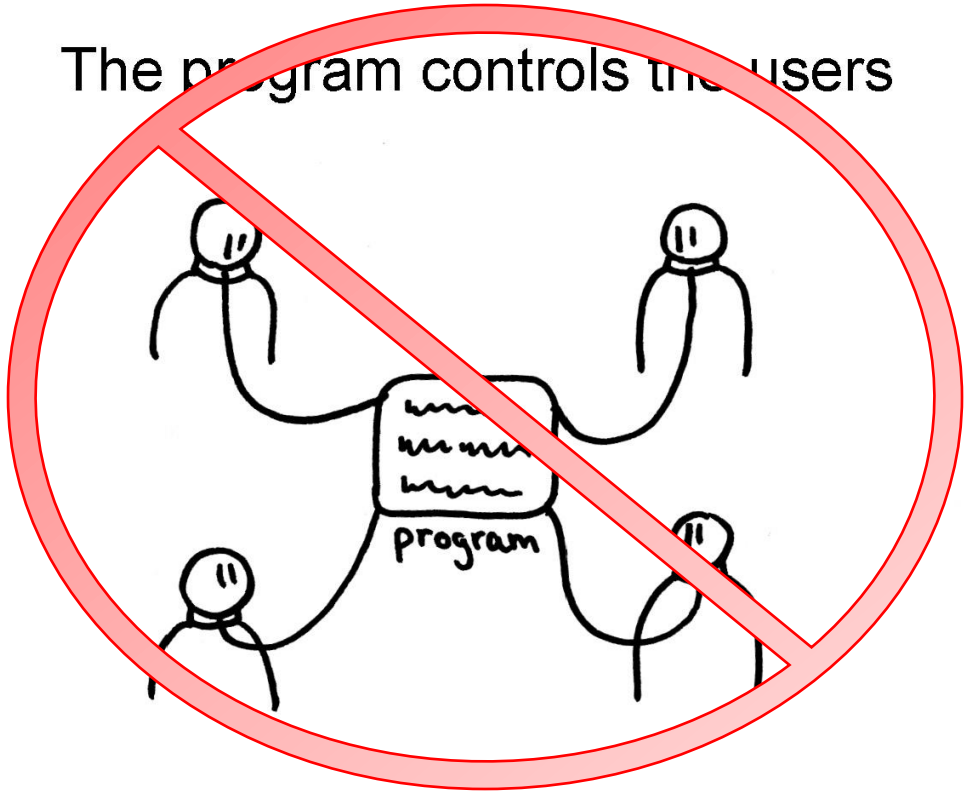
No es por el dinero!

LIBRE not GRATIS

Users control the program



The program controls the users



Expandible

- Infinite number of combinations lead to many packages available:

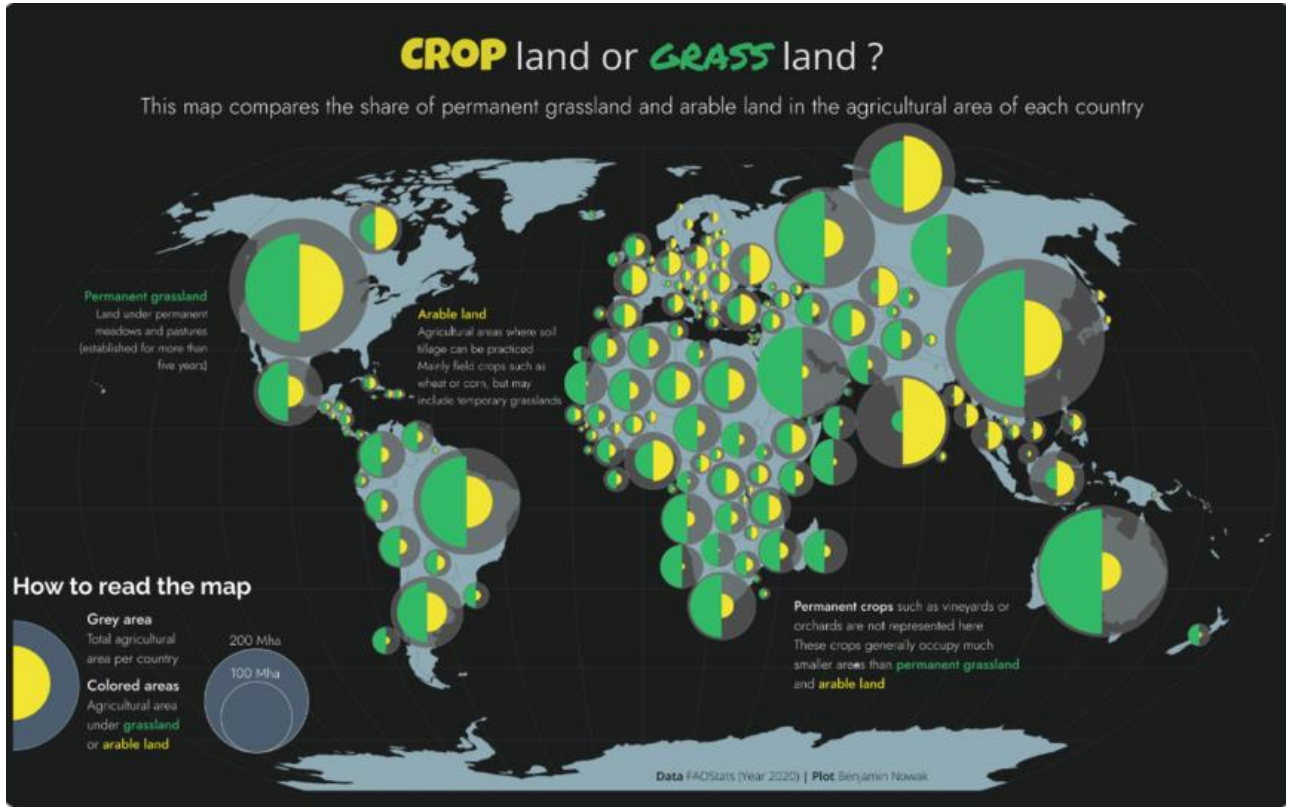
Some number taken on 03/14/2025:

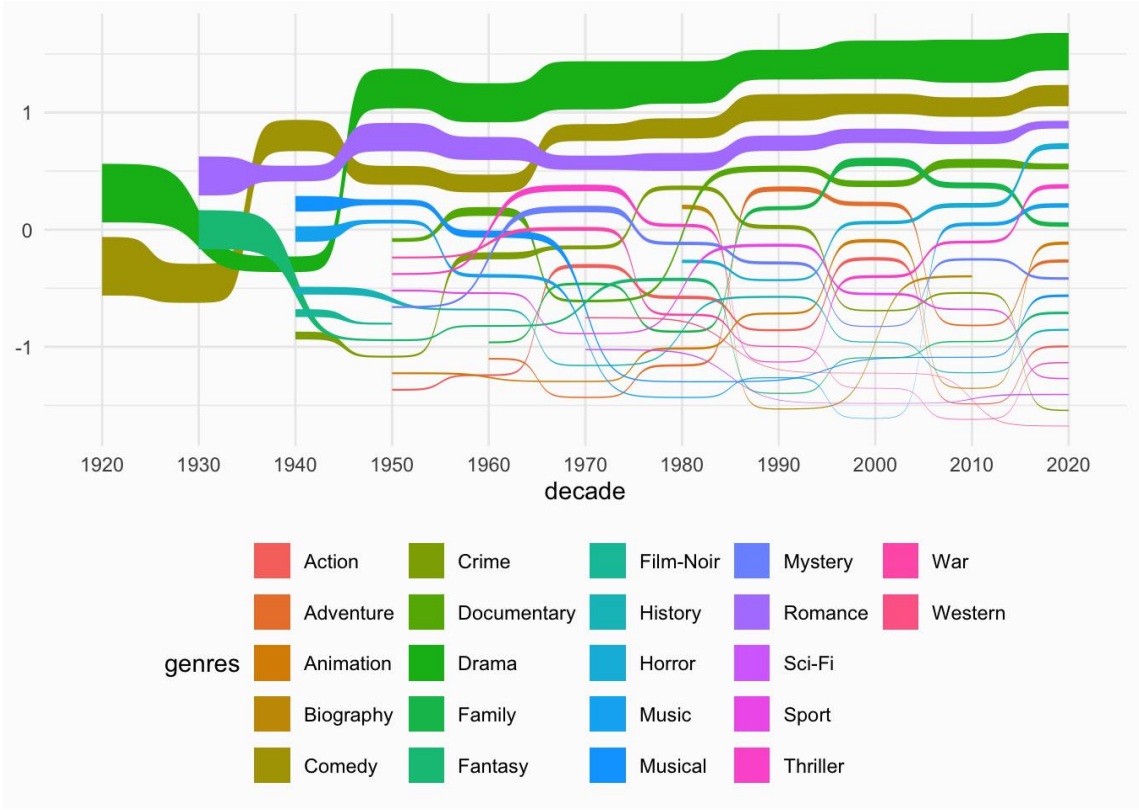
- CRAN – 22.201 packages
- Bioconductor – 2.289 packages
- R-forge – 2.144 packages
- GitHub – Likely > 100.000 R-related repositories

Por qué R?

- **Software libre**
- **Compatibilidad con cualquier plataforma (Windows, Mac, Linux, Cloud)**
- **Ecosistema de paquetes grandes (CRAN, Bioconductor, GitHub)**
- **Varias opciones para la manipulación de datos**
- **Análisis estadístico**
- **Informes y presentaciones (Rmarkdown, Shiny)**
- **Lider en la creación de gráficos (ggplot2, plotly)**

R graph gallery





Por qué R?

- **Software libre**
- **Compatibilidad con cualquier plataforma (Windows, Mac, Linux, Cloud)**
- **Ecosistema de paquetes grandes (CRAN, Bioconductor, GitHub)**
- **Varias opciones para la manipulación de datos**
- **Análisis estadístico**
- **Informes y presentaciones (Rmarkdown, Shiny)**
- **Lider en la creación de gráficos (ggplot2, plotly)**
- **Automatización y reproducibilidad**
- **Aplicaciones interactivas (Shiny)**

Shiny apps

- COVID tracker

https://shiny.posit.co/r/gallery/life-sciences/covid19-tracker/

- VIEWpoly (Tools for Polyploids)

https://cris-taniguti.shinyapps.io/viewpoly/

- BIGapp (Breeding Insight)

https://big-demo.shinyapps.io/bigapp/

Reproducibilidad

- Desafío: ejecutar el código después de 10 años

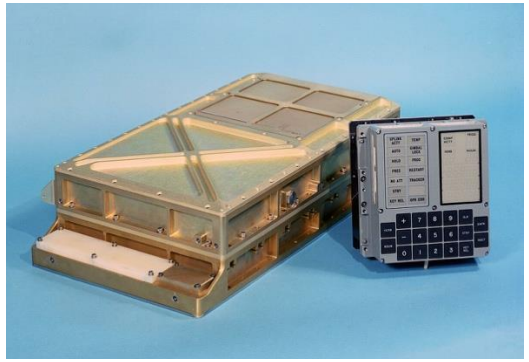
- Scripts vs capturas de pantalla
  - Mejores prácticas para la reproducibilidad:
  - Control de versiones (por ejemplo, Git)
  - Documentación con código
  - Entornos reproducibles (p. ej., Docker)
  - Flujos de trabajo automatizados
  - Pruebas automatizadas

Por qué R?

- **Compatibilidad con cualquier plataforma (Windows, Mac, Linux, Cloud)**
- **Ecosistema de paquetes grandes (CRAN, Bioconductor, GitHub)**
- **Varias opciones para la manipulación de datos**
- **Análisis estadístico**
- **Informes y presentaciones (Rmarkdown, Shiny)**
- **Lider en la creación de gráficos (ggplot2, plotly)**
- **Automatización y reproducibilidad**
- **Aplicaciones interactivas (Shiny)**
- **Procesamiento rápido (se puede conectar con C, C++ y Fortran, paralelización)**
- **Maneja bien Big Data (data.table, sparklyr)**
- **Integración con otras herramientas (Python, SQL, etc.)**
- **Integración de Machine Learning (tidymodels, caret)**

¿Por qué aprender a programar?

- Brecha entre tecnología y aplicaciones



Su teléfono inteligente es millones de veces más potente que la computadora del Apolo 11, pero a menudo lo usamos solo para tareas básicas

Diferencias

RAM: ~ 4 millones de veces más
Almacenamiento: ~ 14 millones de veces más
Velocidad de procesamiento: ~100.000 veces más rápido
Peso: ~150 veces más ligero

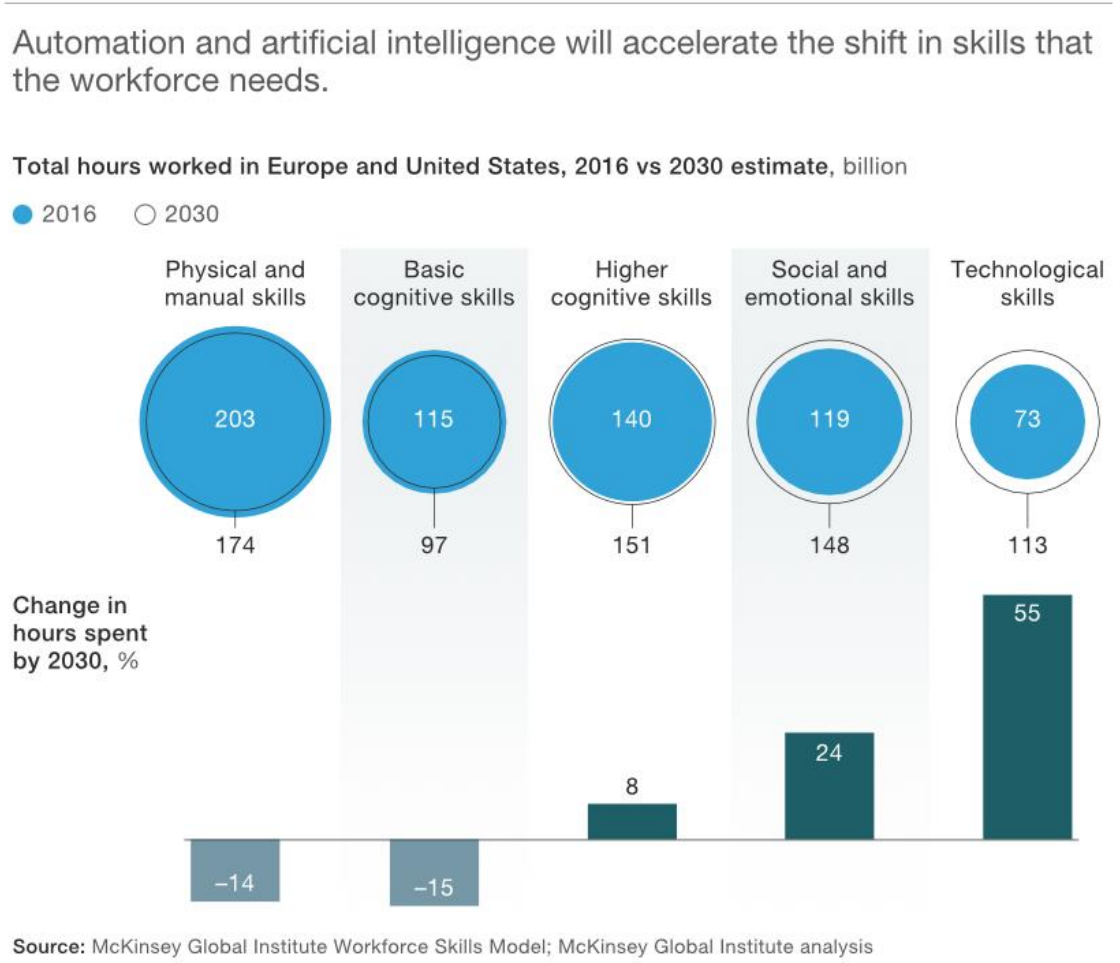
Un solo teléfono inteligente podría ejecutar miles de misiones Apolo simultáneamente

¿Por qué aprender a programar?

- Brecha entre tecnología y aplicaciones

[Su teléfono inteligente es millones de veces más potente que la computadora del Apolo 11, pero a menudo lo usamos solo para tareas básicas]

- Automatizar trabajo repetitivo
- Evolución de habilidades



How to learn coding?

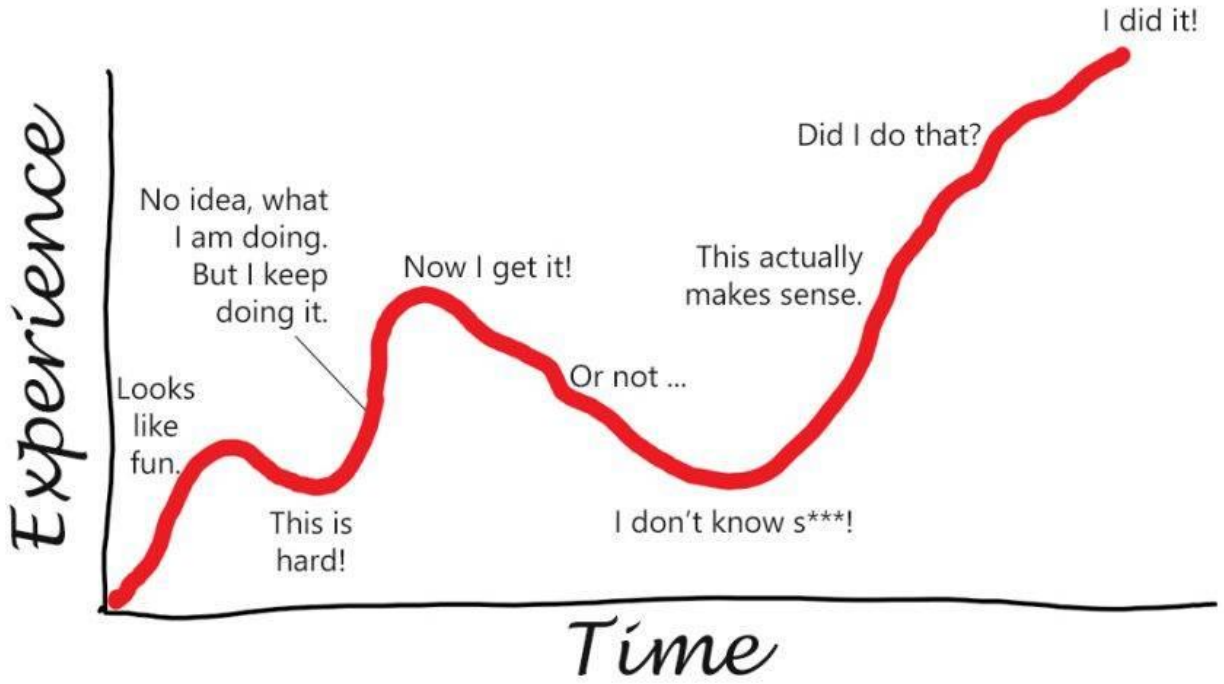


Image Reference

Evolución del código

1

```
# Script simple
data <- read.csv("data.csv")
data$new_column <- data$col1 + data$col2
mean_value <- mean(data$new_column)
plot(data$new_column)
```

2

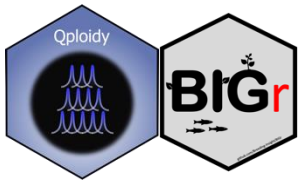
```
# Funciones
process_data <- function(file_path) {
  data <- read.csv(file_path)
  data$new_column <- data$col1 + data$col2
  return(data)
}

calculate_stats <- function(data) {
  mean_value <- mean(data$new_column)
  return(mean_value)
}

visualize_data <- function(data) {
  plot(data$new_column)
}
```

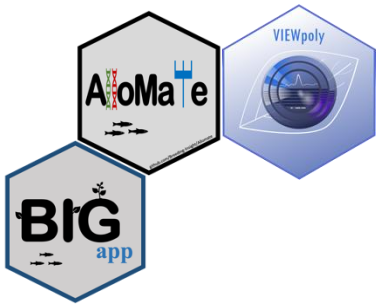
3

#Paquetes



4

#Shiny Apps



Evolución del código

Característica	Scripts	Funciones	Paquetes	Shiny Apps
Estructura	Secuencia de código lineal	código reutilizable	Directorios y componentes organizados	Arquitectura de aplicaciones basada en paquetes
Reusabilidad	Copiar y pegar	Puede ser cargado	Instalar e importar	Instalar e importar
Documentación	Comentarios básicos	Descripciones de funciones	Documentación y ejemplos	Documentación del paquete + tutoriales
Prueba	Manual	Pruebas básicas	Pruebas automatizadas	Pruebas sistemáticas + pruebas de navegador
Distribución	Archivos	Archivos	CRAN/GitHub	Repositorios/Ejecutables
Ideal para	Análisis rápido	Tareas repetidas	Herramientas compartidas	Production-grade applications
Ejemplo	Ssal_Pedigree.R	ped_check.R	BIGr	BIG.shiny

El poder de la colaboración

- Tutorial sobre git
- Tutorial sobre cómo integrar git y Rstudio
- Some examples:
  - Partículas elementales de Boson Higgs – 1200 contribuyentes
  - DeltaBreed – 9 colaboradores (equipo de desarrollo de BI)
  - BIGapp - 2 colaboradores (equipo científico de BI)
  - BIGr - 3 colaboradores (equipo científico de BI)
  - VIEWpoly – 5 colaboradores (NCState + Texas A&M + PFR-NZ + equipo científico de BI)
  - Qploidy – 2 colaboradores (Texas A&M + equipo científico de BI)
  - OneMap – 9 colaboradores (ESALQ/USP + NCState + equipo científico de BI)

Mejores prácticas para combatir ”bugs"

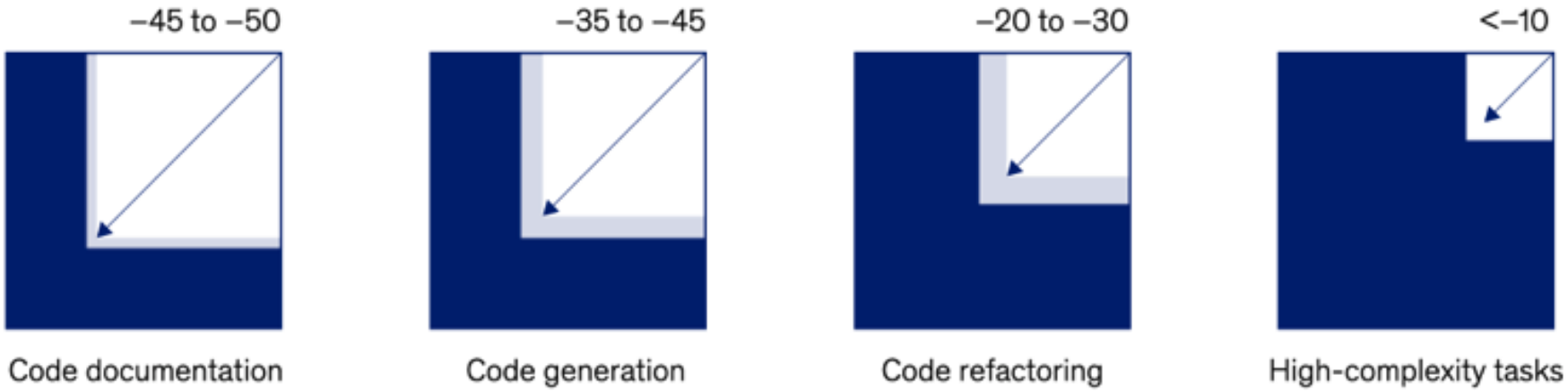
El término se originó en 1947 cuando Grace Hopper encontró una polilla atrapada en un relé de la computadora Mark II de Harvard

- **Da pequeños pasos**
  - Escribir/cambiar código de forma incremental
  - Pruebe cada pequeña modificación
  - Es más fácil aislar los problemas
- **Localice el error**
  - Usar mensajes de error
  - Agregar instrucciones de impresión/depuración
  - Comprobación de los valores de las variables
  - Uso de puntos de interrupción
- **Reproducibilidad es clave**
  - Crear un ejemplo mínimo
  - Pasos del documento para volver a crear
  - Comparta el código y los datos si es necesario
  - Controlar semillas aleatorias
- **Método científico**
  - Observar el comportamiento
  - Formular hipótesis
  - Pruebas sistemáticas
  - Documentar los hallazgos
- **Búsqueda inteligente**
  - Busca en Google el mensaje de error
  - Busca en Stackoverflow
  - Leer la documentación
  - Busca casos similares
  - Asistencia de IA (GitHub Copilot, ChatGPT, Claude, Bard, Stack Overflow AI)

Usar la IA para ayudarte

Generative AI can increase developer speed, but less so for complex tasks.

Reduction in completion time by development task, using generative AI, %



McKinsey & Company

Otros materiales de aprendizaje

- Lógica de programación:
  - Hour of Code
  - Scratch
  - Coursera
  - Khan academy
  - Code academy
  - edX
  - GitHub
  - Datacamp
- Programación R:
  - Manuales CRAN
  - Paquete interactivo Swirl
  - El canal de YouTube de The New Boston
  - TryR

Historia de este material

- 11ª versión
- Colaboradores de la Universidad de São Paulo – ESALQ-USP (Brasil)
- Equipo de Breeding-Insight

Pensamientos antes de comenzar

- Este curso tiene material **básico**. Se necesitan bases sólidas para poder escribir código avanzado de manera **eficiente** y **reproducible**.

- Este curso va a ser tan útil o tan aburrido como ustedes lo decidan. Depende de su **participación** y de sus **preguntas.**

- Zoom tiene la opción de compartir pantalla y compartir control de pantalla. Compartir los errores es la mejor forma de ayudar a los demás a aprender.