

Flujo de trabajo reproducible con R y {targets}

Laboratorio Abre Tu Ciencia



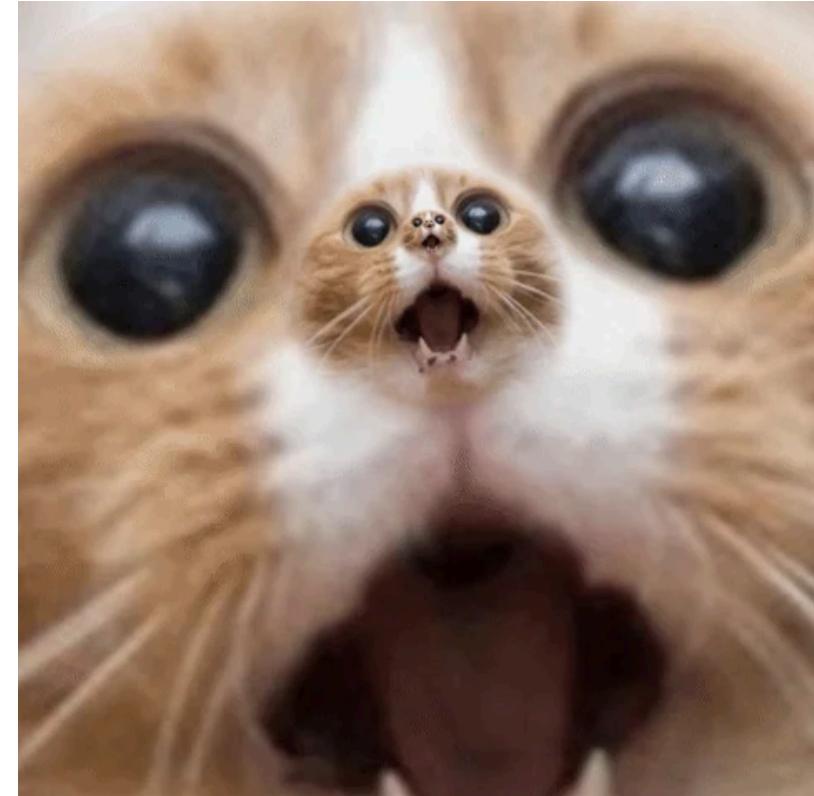
José Daniel Conejeros, Msc.

22 de enero de 2026

Hoy hablaremos sobre

- Automatización de **workflows**
- Introducción de **targets** en R

Link al material: [!\[\]\(99f58673407353e96a019fbca558fd72_img.jpg\) Código](#)



(i) Nota

En Python el proceso es análogo con [snakemake](#).

¿Por qué automatizar un flujo?

Por **workflow** o flujo de trabajo nos referimos a la serie de pasos para llevar a cabo un análisis de datos.

- Reproducibilidad: Obtener resultados utilizando los mismos datos de entrada, pasos computacionales, métodos, código y condiciones de análisis.

Pero además:

- Eficiencia.
- Escalabilidad.
- Mantenimiento.

Desafíos de la reproducibilidad

1. Directorios/rutas de trabajo.
2. Paquetes y versionamiento.
3. Pipeline o flujo poco claro.
4. Sin información de dependencias.
5. Cambios en el código.

Targets



El paquete `targets` es una herramienta de pipeline para R enfocada en análisis estadístico y ciencia de datos: coordina una serie de pasos computacionales (llamados *targets*) y, al igual que otros sistemas de flujo de trabajo, evita recalcular pasos cuyos datos o código no han cambiado, lo que ahorra tiempo y mejora la reproducibilidad de proyectos complejos.

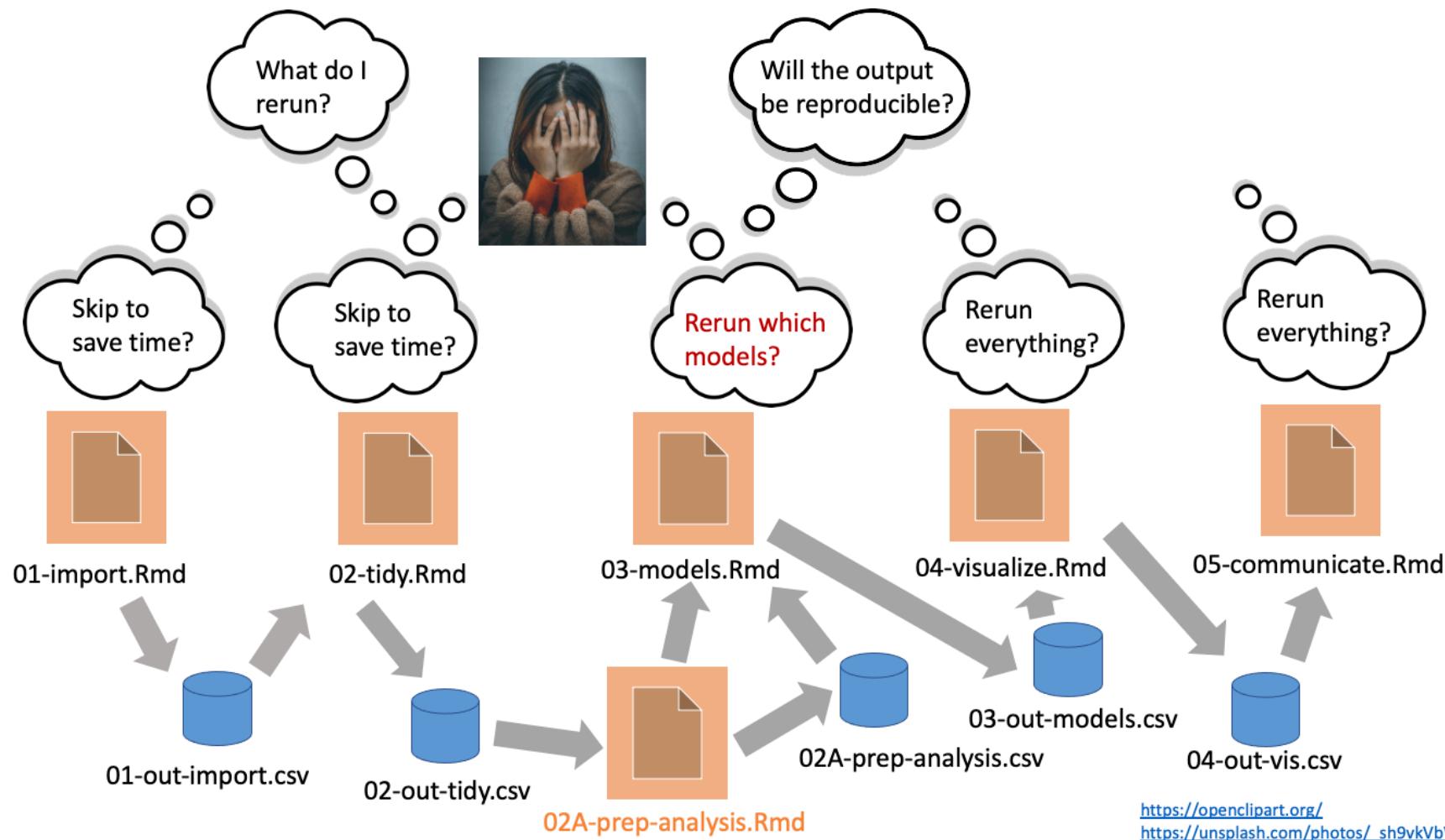
Cada *target* representa una tarea definida por funciones de R, y la estructura de dependencias entre ellos forma un gráfico acíclico dirigido (DAG) que targets utiliza para decidir qué ejecutar y qué saltarse.

Puedes ver toda la documentación aquí:

Manual

Repositorio

¿Por qué targets?



¿Qué debo volver a correr?, ¿puedo saltarme pasos para ahorrar tiempo?, ¿qué modelos específicos se ven afectados?, ¿tengo que rehacer todo para garantizar reproducibilidad?

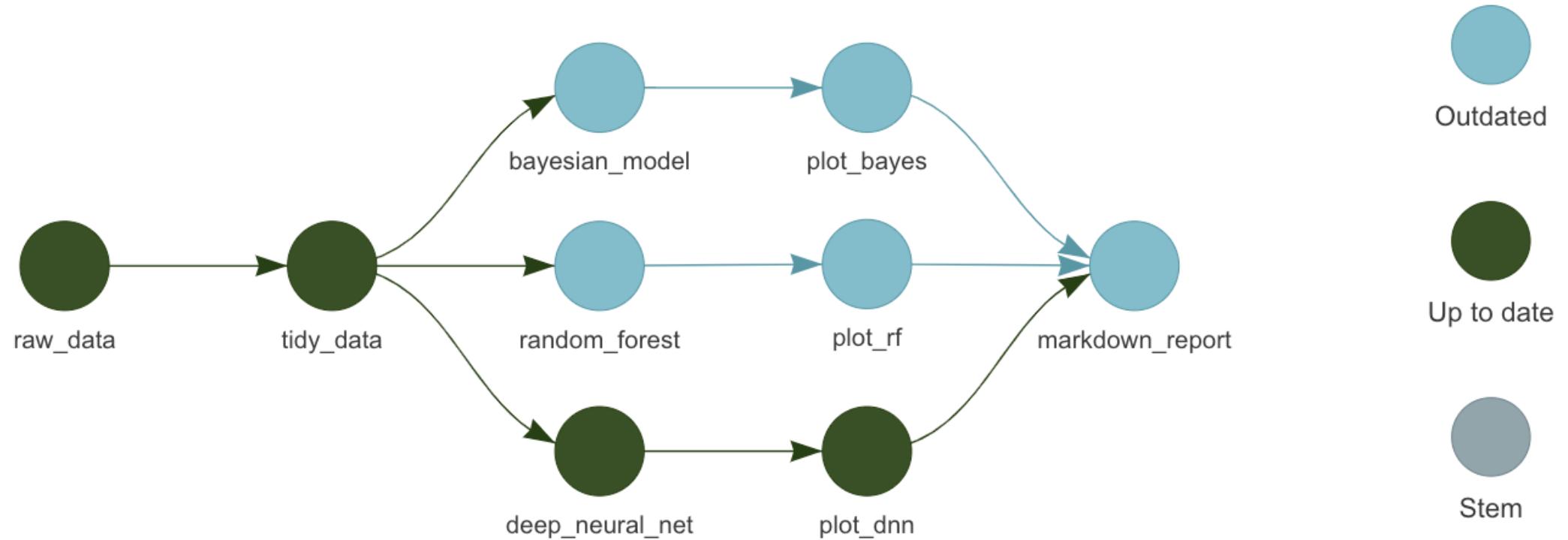
Targets

La filosofía del paquete impulsa un estilo de programación **orientado a funciones dentro de R**, promueve reproducibilidad al abstraer archivos como objetos de R y facilita paralelización implícita, lo que permite construir análisis eficientemente, mantenerlos organizados y asegurar que los resultados sean coherentes con el código y los datos actuales

Algunas alternativas:

- Paquete [drake](#) (antecesor de targets).
- Paquete [workflowr](#) (no gestiona dependencias).

Targets



1. Actualiza solo lo necesario.
2. Cálculos escalables.
3. Gestiona los datos de salida.

Targets

Principales funciones de trabajo:

1. Definición del pipeline:

- `tar_pipeline()` : define el pipeline completo como una colección de targets.
- `tar_target()` : **define un paso del pipeline (un objeto o resultado a producir).**
- `tar_option_set()` : define opciones globales del pipeline (paquetes, formato, recursos).
- `tar_source()` : carga archivos R con funciones o targets auxiliares.

Targets

Principales funciones de trabajo:

2. Ejecución y control:

- `tar_make()` : ejecuta el pipeline, corriendo solo lo que está desactualizado.
- `tar_make_clustermq()` : ejecuta el pipeline en paralelo usando clustermq.
- `tar_make_future()` : ejecuta el pipeline en paralelo usando future.
- `tar_destroy()` : borra el estado del pipeline y la cache.
- `tar_prune()` : elimina targets obsoletos de la cache.

Targets

Principales funciones de trabajo:

3. Inspección y diagnóstico:

- `tar_visnetwork()` : visualiza el DAG del pipeline.
- `tar_glimpse()` : muestra un resumen rápido de los targets.
- `tar_progress()` : muestra el progreso de ejecución.
- `tar_outdated()` : identifica qué targets necesitan re-ejecutarse.
- `tar_meta()` : inspecciona metadatos de los targets.
- `tar_manifest()` : lista todos los targets definidos.

Targets

Principales funciones de trabajo:

4. Branching (reemplaza loops manuales por paralelización declarativa):

- `tar_map()` : define branching dinámico a partir de una lista o vector.
- `tar_combine()` : combina los resultados de múltiples ramas en un solo target.
- `tar_group()` : agrupa ramas para control de ejecución.
- `tar_rep()` : replica un target múltiples veces (útil para simulaciones).

Targets

Principales funciones de trabajo:

5. Reportes y comunicación:

- `tar_render()` : renderiza documentos Quarto/R Markdown como parte del pipeline.
- `tar_quarto()` : renderiza documentos Quarto (.qmd) integrados al pipeline.
- `tar_read()` : recupera el valor de un target ya calculado.
- `tar_load()` : carga targets directamente en el entorno de R.

Recomendaciones iniciales

- Empieza con proyectos pequeños.
- Aunque tome más tiempo, incorpora estas herramientas desde el inicio.
- Tal vez cambie un poco tu forma de trabajar, pero los beneficios valen la pena.
- Consulta la documentación oficial y ejemplos en línea: [!\[\]\(998e30a8dcfb35e9b724d9eb41990449_img.jpg\) Intro a Targets](#)

Universo Targets



🔗 <https://wlandau.github.io/targetopia/packages.html>

Recomendaciones joya

- Estructura de proyectos por Danielle Navarro:  [Slides](#)
- Ciencia Reproducible por Dan Ovando:  [Slides](#)
- Proyectos Buenos por Barbara Vreede:  [Repositorio](#)
- Automatización de workflows en R y Python con targets y snakemake por Diana García Cortés:  [Repositorio](#)
- Introducción a Targets por Will Landau  [Slides](#)
- Protocolos de Análisis Reproducible por Lisa:  [Website](#)

Manos a los datos

Vamos a utilizar una muestra aleatoria de los datos del artículo académico:

 Nota

Blanco, E., Conejeros, J.D., González-Reyes, Á. et al. Heat beyond percentiles: exploring preterm birth risks in Santiago, Chile (1991–2019). *Int Arch Occup Environ Health* **99**, 5 (2026). <https://doi.org/10.1007/s00420-025-02196-x>

Puedes acceder a los códigos de este artículo aquí:  [Repository](#)