rcdo permite analizar datos climáticos con R, y se escribe solo

Elio Campitelli

Palabras clave: CDO - datos climáticos - wrapper - netcdf - paquetes automáticos

CDO es un programa de línea de comandos que provee cientos de subcomandos, llamados "operadores", para realizar operaciones muy comunes en el procesamiento de datos climáticos. Por ejemplo, para calcular una media mensual está el operador `cdo monmean`. Para interpolar entre distintas grillas regulares está el operador `cdo remap` y sus distintas variantes según el método de interpolación (bilineal, bicúbica, vecinos cercanos, etc.). Debido a su velocidad y eficiencia, y a que permite trabajar muy fácilmente con datos más grandes que la RAM, es una herramienta muy popular en la comunidad de ciencias climáticas.

Al ser una herramienta de línea de comandos, para integrarla a un flujo de trabajo usando R es necesario construir los comandos "a mano" y luego ejecutarlos con `system()` o similares. Por ejemplo, para calcular una media mensual climatológica habría que hacer algo como:

```
archivo_entrada <- "datos.nc"
archivo_salida <- "climatología.nc"
system(paste0("cdo ymonmean ", archivo_entrada, " ", archivo_salida))
```

Esto es muy verborrágico, requiere saber de memoria el nombre de los cientos de operadores y no permite usar las sugerencias automáticas del IDE. Es algo que podría automatizarse fácilmente escribiendo una función en R que haga la traducción, pero entonces tendría que crear una función por cada operador. £Es posible crear una función por cada uno de los casi 700 operadores?

Afortunadamente, el diseño de CDO hace que sea posible escribir un wrapper en R de forma automática a partir de su documentación. Así nace el paquete rcdo.

Lo que en realidad desarrollé no es el paquete rcdo en sí, sino una serie de scripts de R que construyen el paquete automáticamente. Estos scripts primero utilizan expresiones regulares para extraer la información de cada operador a partir de la documentación de CDO: nombre del operador, número de archivos de entrada, número de archivos de salida y posibles parámetros. Luego, usando una plantilla que incluye el cuerpo de la función y su documentación, crean una función para cada operador y los archivos de ayuda.

Cada operador de CDO tiene una función rcdo equivalente, que comienza con `cdo_*`. Por ejemplo, el operador para calcular medias mensuales se llama `monmean`, por lo que su correspondiente función es `cdo_monmean()`. La nomenclatura consistente de los operadores y el hecho de que los parámetros de cada operador sean argumentos de su función en rcdo permiten autocompletar con el IDE y descubrir operadores similares. Por ejemplo, para ver todas las estadísticas mensuales posibles basta con escribir `cdo_mon` y el IDE va a listar funciones como `cdo_monmax()`, `cdo_monmin()`, etc. Además, la ayuda de R de cada función muestra la documentación de CDO, por lo que no es necesario abrir el manual en una ventana aparte.

Los operadores de CDO pueden encadenarse, lo cual se traduce muy naturalmente a R usando la pipe para que el input de una operación sea el output de una operación anterior. En concreto, si `datos` es un archivo con datos diarios y queremos calcular el máximo valor de la media mensual para cada mes, se puede hacer:

```
datos <- "archivo.nc"
datos |>
  cdo_monmean() |>
  cdo_ymonmax()
```

```
## CDO command:

##    cdo -L ymonmax [ -monmean [ 'archivo.nc' ] ] {{output}}
```

Las funciones de rcdo devuelven operaciones de CDO sin ejecutar. Para ejecutarlas hay que usar la función `cdo_execute()`. Esto permite concatenar operaciones en vez de ejecutarlas una a una e incluso guardar las operaciones en variables para usarlas en otra parte del código:

```
datos_media_mensual <- datos |>
  cdo_monmean()

record_mensual <- datos_media_mensual |>
  cdo_ymonmax() |>
  cdo_execute()
```

Si no se especifica el nombre del archivo de salida, `cdo_execute()` ejecuta la operación guardando la salida a un archivo temporal. Dado que las operaciones pueden resultar en archivos muy pesados, por defecto estos archivos se eliminan automáticamente cuando dejan de ser accesibles desde el entorno de R. También se puede especificar el comportamiento opuesto: con `cdo_cache_set(carpeta)`, los archivos de salida se guardarán por defecto en `carpeta` y `cdo_execute()` no ejecutará el comando si detecta que no es necesario. Así, se puede elegir priorizar la velocidad de reejecución del código a costa de ocupar más espacio en disco, o ahorrar espacio a costa de tener que reejecutar las operaciones cada vez que se corra el código.

El paquete rcdo está publicado en CRAN y es totalmente funcional. La única funcionalidad todavía no implementada es el soporte para la sintaxis especial de "apply" de CDO que permite aplicar el mismo operador a múltiples archivos de manera eficiente, y la traducción de algunos operadores que no pueden ser creados automáticamente debido al formato de su documentación o su sintaxis. Sin embargo, los operadores faltantes no se usan mucho, y la funcionalidad de "apply" puede suplantarse por `lapply()` o `purrr::map()`, de modo que no son limitaciones cruciales para el uso del paquete.

En esta charla voy a mostrar el funcionamiento del paquete y a discutir las decisiones de diseño que informaron la implementación actual. Además, voy a mostrar cómo es el proceso de generación automática, que puede aplicarse para crear wrappers de otras utilidades en R.