Matías Alejando Castillo Moine and Mónica Balzarini

Procesamiento eficiente de series de tiempo de raster espacio temporales en R

Introducción

El gran volumen de datos en forma de series de tiempo imágenes satelitales (STIS) plantea desafíos técnicos de almacenamiento y procesamiento. Numerosos paquetes de R ofrecen herramientas para su manejo con soporte para out-of-memory files y procesamiento en paralelo, permitiendo el procesamiento masivo de grandes bases de datos de STIS. Sin embargo la mayoría de los desarrollos estadísticos de R tienen base matricial, y el dato debe ser convertido para poder ser procesado; transformación que generalmente ocurre en copias completas de los datos en memoria. El objetivo de este trabajo es comparar la eficiencia de dos estrategias de manejo y procesamiento en R de datos masivos de STIS convertidas a formato matricial en memory-maped files con soporte para procesamiento en paralelo.

Materiales y métodos

Se usaron los paquetes raster (Hijmans 2019), foreach (Calaway et al. 2018), y bigmemory (Kane et al. 2018) para generar métodos de almacenamiento de datos de STIS en formatos alternativos. raster provee clases para el manejo de datos espaciales y extensiones espacio-temporales en formato imagen; bigmemory provee clases y métodos de creación, indexado y almacenamiento de grandes matrices con memoria compartida (shared memory) y archivos locales mapeados en memoria (memory-maped files); foreach provee construcciones lógicas para la ejecución de código en paralelo. En este trabajo, las STIS fueron almacenadas en objetos de la clase RasterStack del paquete raster. La conversión de datos de STIS a matriz asumió series de tiempo de igual longitud para un área definida S. La clase RasterStack de R almacena STIS en formato espacial explícito: cada capa del RasterStack es una imagen raster con idéntico número de píxeles para diferentes fechas de adquisición (Fig. 1), de longitud igual al número de capas del RasterStack. Si bien la clase RasterStack y otras extensiones permiten el indexado de series de tiempo de pixeles de una STIS, su velocidad de proceso limita su aplicación con STIS extensas. Los RasterStack fueron reacomodados en matrices de la clase big.matrix, de dimensión tantas filas como series de tiempo y tantas columnas como capas hay en el RasterStack, y almacenadas en disco (filebacked) para optimizar el uso de memoria. Para procesar en paralelo las series de tiempo de cada píxel se desarrollaron funciones que primero dividen la matriz por partes de manera secuencial, convierten cada parte a la clase matrix (en memoria) y procesa en paralelo las series de tiempo de los píxeles. El formato de almacenamiento depende del resultado,. El índice espacial fue conservado, y una función accesoria permite enmascarar píxeles (filas de la matriz) que no serán considerados en el análisis. En la Figura 2 se muestra el flujo de trabajo.

Resultados y conclusión

Las funciones desarrolladas permitieron leer en forma secuencial por partes la big.matrix y ejecutar funciones en paralelo en cada parte. Almacenar las series de tiempo de los píxeles de una STIS en una big.matrix mejoró el tiempo de procesamiento de las series de tiempo de los píxeles respecto a cuando se usó formato RasterStack. Por ejemplo el tiempo de procesamiento para filtrado de todas las series de tiempo se redujo de un día completo cuando estaban almacenadas como RasterStack a algunas horas cuando estaban almacenadas como big.matrix. Dividir la big.matrix por partes provee un marco de trabajo

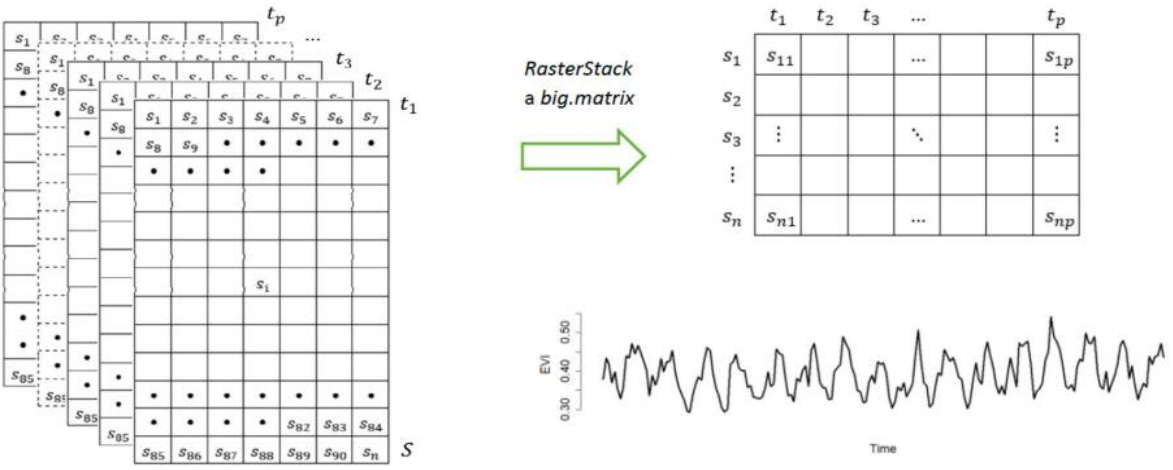

Figura 1. La región S de una imagen satelital queda representada por S={si} con i=1, …,n píxeles. En un RasterStack con imágenes para $t1,t2,…,tp$ hay n series de tiempo de longitud p y pueden ser reacomodadas en una matriz con n filas y p columnas.

```
function ( stack, mascara ) {
    ## Generar índice de píxeles no enmascarados y convertir a bigmatrix
        valid_index <- n - mascara
        matriz <- crear una filebacked big.matrix  n × p con n = número de píxeles y p = número de capas del stack
        results <- crear un objeto de formato adecuado para almacenar los resultados (por ejemplo otra big.matrix)
        for ( i in 1:nlayers ( stack ) ) {
            vectorizar cada capa del RasterStack y almacenar cada vector en columnas de la big.matrix
            }
    ## Crear un índice de partición de la matriz:
        firstRow, lastRow = crear índice ( primeras filas de partición )
    ## Inicializar cluster y procesamiento:
        cl <- abrir y registrar cluster
        for ( i in 1:firstRow ) {
            #Crear un subset de la big.matrix y transformar a clase matrix
            subset <- as.matrix ( sub.big.matrix ( matriz, firstRow, lastRow ) )
            #Procesar en paralelo
            s <- foreach (in subset) %dopar% and combine
            }
            ## exportar los resultados parciales
        }
Compilar resultados y cerrar cluster
}
```

Figura 2. Flujo de trabajo general de una función de procesamiento en paralelo para series de tiempo de imágenes satelitales. En negrita se resalta el bucle for de procesamiento de la big.matrix por partes.

general que permite procesar grandes volúmenes de series de tiempo provenientes de imágenes satelitales usando funciones de R de base matricial.

Bibliografía

- Calaway, Rich, Microsoft Corporation, Steve Weston, y Dan Tenenbaum. 2018. «Foreach Parallel Adaptor for the "parallel" Package». CRAN.
- Hijmans, Robert J. 2019. «raster: Geographic Data Analysis and Modeling». https://cran.r-project.org/package=raster.
- Kane, Michael J., John W. Emerson, Peter Jr. Haverty, y Charles Determan. 2018. «bigmemory: Manage Massive Matrices with Shared Memory and Memory-Mapped Files». https://cran.r-project.org/web/packages/bigmemory/index.html.