Implementación de BLAS para Optimización de Álgebra Matricial

Arantza Ivonne Pineda Sandoval, Ana Luisa Masetto Herrera, Alexis Solis Cancino

2019-02-16

1 Aceleración de Algoritmos de Álgebra Matricial en R

```
library(tidyverse)
```

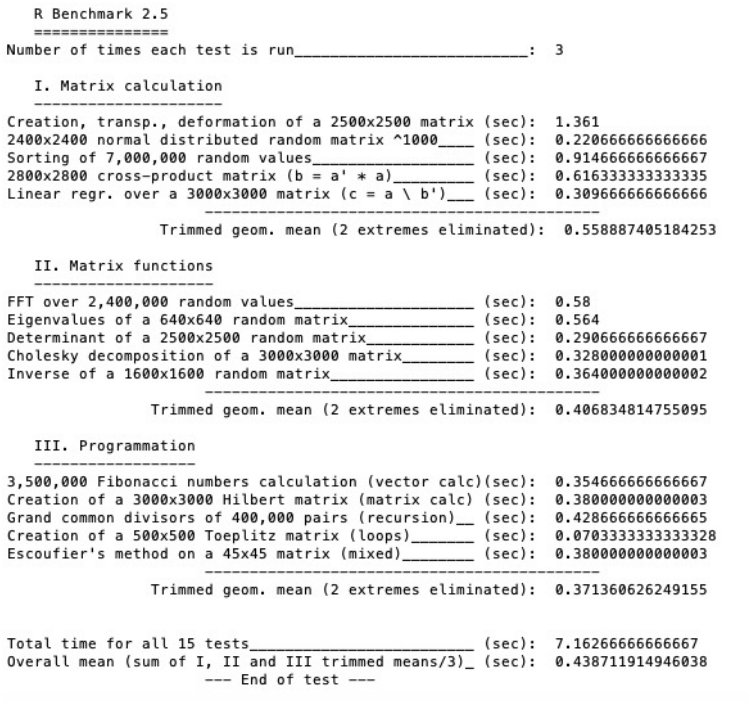
```
## -- Attaching packages ----------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.0       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0

## -- Conflicts -------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
theme_set(theme_classic(base_size = 13))
options(ggplot2.continuous.colour = "viridis",
    ggplot2.continuous.fill = "viridis")
```

La multiplicación de matrices en R se maneja, en un nivel muy bajo, por la biblioteca que implementa las subrutinas de álgebra lineal básica (BLAS, por sus siglas en inglés). Cualquier versión de R que se descarga de CRAN viene con lo que se conoce como una *implementación de referencia* de BLAS. Esta implementación ciertamente funciona, produce lo que todos concuerdan que son las respuestas correctas, pero de ninguna manera está optimizado. Utilizando ciertas operaciones que se consideran como *benchmarks*, lo siguiente es lo que tardamos en hacer operaciones matriciales en la versión no optimizada de R:

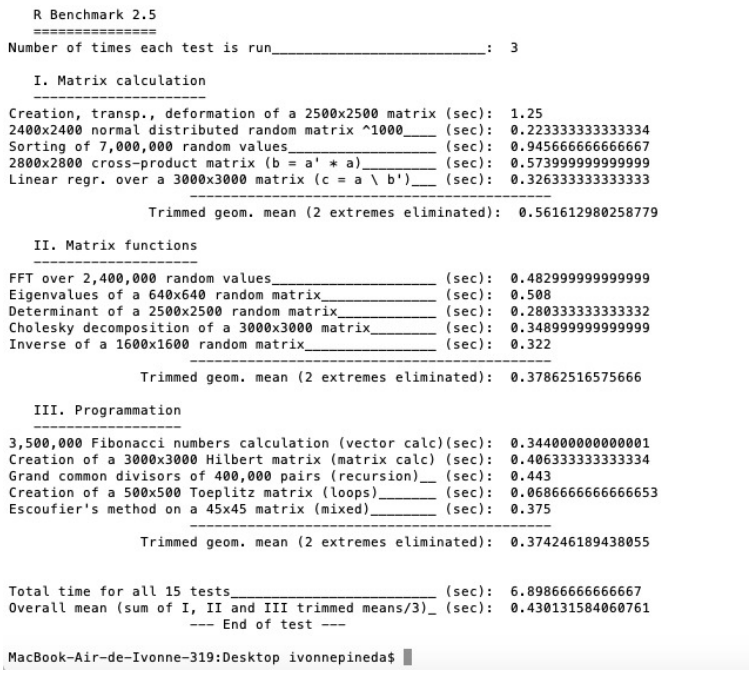
Recientemente, sitios como *RBloggers* sugieren que un gran beneficio de su versión (comercial) de R es que estaba vinculada a una mejor biblioteca de álgebra lineal. El resumen rápido es que realmente sólo hace una diferencia para pruebas de referencia (*benchmarks*) bastante artificiales. Para el trabajo "normal" de día a día es poco probable que se vea una diferencia la mayor parte del tiempo.

En la versión optimizada de R, el tiempo mejora de 7.16 segundos a 6.89 segundos:



```
   R Benchmark 2.5
   ===============
Number of times each test is run_________________________:  3

   I. Matrix calculation
   ---------------------
Creation, transp., deformation of a 2500x2500 matrix (sec):  1.361
2400x2400 normal distributed random matrix ^1000____ (sec):  0.220666666666666
Sorting of 7,000,000 random values__________________ (sec):  0.914666666666667
2800x2800 cross-product matrix (b = a' * a)_________ (sec):  0.616333333333335
Linear regr. over a 3000x3000 matrix (c = a \ b')___ (sec):  0.309666666666666
                    ---------------------------------------------
                 Trimmed geom. mean (2 extremes eliminated):  0.558887405184253

   II. Matrix functions
   --------------------
FFT over 2,400,000 random values____________________ (sec):  0.58
Eigenvalues of a 640x640 random matrix______________ (sec):  0.564
Determinant of a 2500x2500 random matrix____________ (sec):  0.290666666666667
Cholesky decomposition of a 3000x3000 matrix________ (sec):  0.328000000000001
Inverse of a 1600x1600 random matrix________________ (sec):  0.364000000000002
                    ---------------------------------------------
                Trimmed geom. mean (2 extremes eliminated):  0.406834814755095

   III. Programmation
   ------------------
3,500,000 Fibonacci numbers calculation (vector calc)(sec):  0.354666666666667
Creation of a 3000x3000 Hilbert matrix (matrix calc) (sec):  0.380000000000003
Grand common divisors of 400,000 pairs (recursion)__ (sec):  0.428666666666665
Creation of a 500x500 Toeplitz matrix (loops)_______ (sec):  0.0703333333333328
Escoufier's method on a 45x45 matrix (mixed)________ (sec):  0.380000000000003
                    ---------------------------------------------
                Trimmed geom. mean (2 extremes eliminated):  0.371360626249155


Total time for all 15 tests_________________________ (sec):  7.16266666666667
Overall mean (sum of I, II and III trimmed means/3)_ (sec):  0.438711914946038
                      --- End of test ---
```

Figure 1: Benchmark de álgebra matricial versión no optimizada (comercial) de R. El total de 15 pruebas se realiza en un total de 7.16 segundos.



```
   R Benchmark 2.5
   ===============
Number of times each test is run_________________________:  3

   I. Matrix calculation
   ---------------------
Creation, transp., deformation of a 2500x2500 matrix (sec):  1.25
2400x2400 normal distributed random matrix ^1000____ (sec):  0.223333333333334
Sorting of 7,000,000 random values__________________ (sec):  0.945666666666667
2800x2800 cross-product matrix (b = a' * a)_________ (sec):  0.573999999999999
Linear regr. over a 3000x3000 matrix (c = a \ b')___ (sec):  0.326333333333333
                    ---------------------------------------------
                 Trimmed geom. mean (2 extremes eliminated):  0.561612980258779

   II. Matrix functions
   --------------------
FFT over 2,400,000 random values____________________ (sec):  0.482999999999999
Eigenvalues of a 640x640 random matrix______________ (sec):  0.508
Determinant of a 2500x2500 random matrix____________ (sec):  0.280333333333332
Cholesky decomposition of a 3000x3000 matrix________ (sec):  0.348999999999999
Inverse of a 1600x1600 random matrix________________ (sec):  0.322
                    ---------------------------------------------
                Trimmed geom. mean (2 extremes eliminated):  0.37862516575666

   III. Programmation
   ------------------
3,500,000 Fibonacci numbers calculation (vector calc)(sec):  0.344000000000001
Creation of a 3000x3000 Hilbert matrix (matrix calc) (sec):  0.406333333333334
Grand common divisors of 400,000 pairs (recursion)__ (sec):  0.443
Creation of a 500x500 Toeplitz matrix (loops)_______ (sec):  0.0686666666666653
Escoufier's method on a 45x45 matrix (mixed)________ (sec):  0.375
                    ---------------------------------------------
                Trimmed geom. mean (2 extremes eliminated):  0.374246189438055


Total time for all 15 tests_________________________ (sec):  6.89866666666667
Overall mean (sum of I, II and III trimmed means/3)_ (sec):  0.430131584060761
                      --- End of test ---

MacBook-Air-de-Ivonne-319:Desktop ivonnepineda$
```

Figure 2: Benchmark de álgebra matricial versión no optimizada (comercial) de R. El total de 15 pruebas se realiza en un total de 6.89 segundos.

Apple proporciona dos versiones de BLAS, la de referencia y una biblioteca BLAS optimizada (*vecLib*). La instalación de `R` para MacOS descargada de CRAN se envía con las bibliotecas vecLib y BLAS de referencia (la no optimizada se usa por defecto).

Para obtener los resultados de arriba, corrimos las pruebas primero para la versión no optimizada y después con la versión *vecLib*. Para cambiar a la versión optimizada, seguimos los pasos de R: Use faster vecLib[1]. El código que se tiene que poner en la terminal es el siguiente:

[1] https://gist.github.com/nicebread/6920c8287d7bffb03007

```
# Para cambiar a la librería optimizada vecLib
cd /Library/Frameworks/R.framework/Resources/lib
ln -sf  /System/Library/Frameworks/Accelerate.framework/
  Frameworks/vecLib.framework/Versions/Current/libBLAS.dylib libRblas.dylib

# Para regresar a la librería por defecto
cd /Library/Frameworks/R.framework/Resources/lib
ln -sf libRblas.0.dylib libRblas.dylib
```

Además, el `R script` para generar las pruebas se incluye en el archivo `script_blas.R`.

1.1 OpenBLAS y Medición de la Aceleración de Cómputo

Por último, el `R script` para generar las pruebas antes mencionado se utilizó dos veces: la primera vez antes de instalar OpenBlas en la computadora y la segunda vez después de instalarlo. OpenBlas se instaló mediante el comando: `sudo apt-get install libopenblas-dev`. Nuestro `R script` crea un archivo `script_blas.csv` que contiene los tiempos de cada operación algebraica:

```
times <- as_tibble(read_csv(file = "02_files/script_blas.csv"))
```

```
## Parsed with column specification:
## cols(
##   test = col_character(),
##   BLAS = col_double(),
##   no_BLAS = col_double()
## )
```

```
times
```

```
## # A tibble: 9 x 3
##   test                          BLAS no_BLAS
##   <chr>                        <dbl>   <dbl>
## 1 2500x1 vector multiplication   0       0
```

```
## 2 Matrix vector multiplicatio~   0.02    0.01
## 3 Matrix matrix multiplicatio~   9.68   11.2
## 4 Eigenvector of 2500x2500 ma~  74.4    90.1
## 5 Inverse of 2500x2500 matrix   13.4    17.0
## 6 Solving matrix equation sys~   2.39    3.52
## 7 Principal components of 250~  44.4    52.0
## 8 Singular value decompositio~  49.2    54.1
## 9 Total time for all tests     151.    179.
```

```
time_dif <- times %>% mutate(diff = BLAS - no_BLAS)
column_names <- c("Prueba", "BLAS", "no_BLAS",
    "acc_sec")
colnames(time_dif) <- column_names
time_dif
```

```
## # A tibble: 9 x 4
##   Prueba                 BLAS no_BLAS acc_sec
##   <chr>                 <dbl>   <dbl>   <dbl>
## 1 2500x1 vector multi~   0       0       0
## 2 Matrix vector multi~   0.02    0.01    0.01
## 3 Matrix matrix multi~   9.68   11.2    -1.54
## 4 Eigenvector of 2500~  74.4    90.1   -15.8
## 5 Inverse of 2500x250~  13.4    17.0    -3.59
## 6 Solving matrix equa~   2.39    3.52   -1.13
## 7 Principal component~  44.4    52.0    -7.6
## 8 Singular value deco~  49.2    54.1    -4.96
## 9 Total time for all ~ 151.    179.    -28.1
```

Se encontró que la instalación de OpenBLAS llevó a una aceleración de 28 segundos (en total) para todas las pruebas, y la prueba con mayor aceleración resultó ser la de encontrar el eigenvector de una matriz de dimensiones $(2,500 \times 2,500)$. Podemos visualizar la diferencia en segundos para cada prueba:

```
# Ordenamos de mayor a menos, por aceleración:
time_dif$Prueba <- factor(time_dif$Prueba, levels = time_dif$Prueba[order(-time_dif$acc_sec)])

# Graficamos:
time_dif %>% ggplot(mapping = aes(fill = acc_sec,
    x = Prueba)) + geom_col(mapping = aes(y = -acc_sec)) +
    coord_flip() + labs(x = "Aceleración en segundos")
```

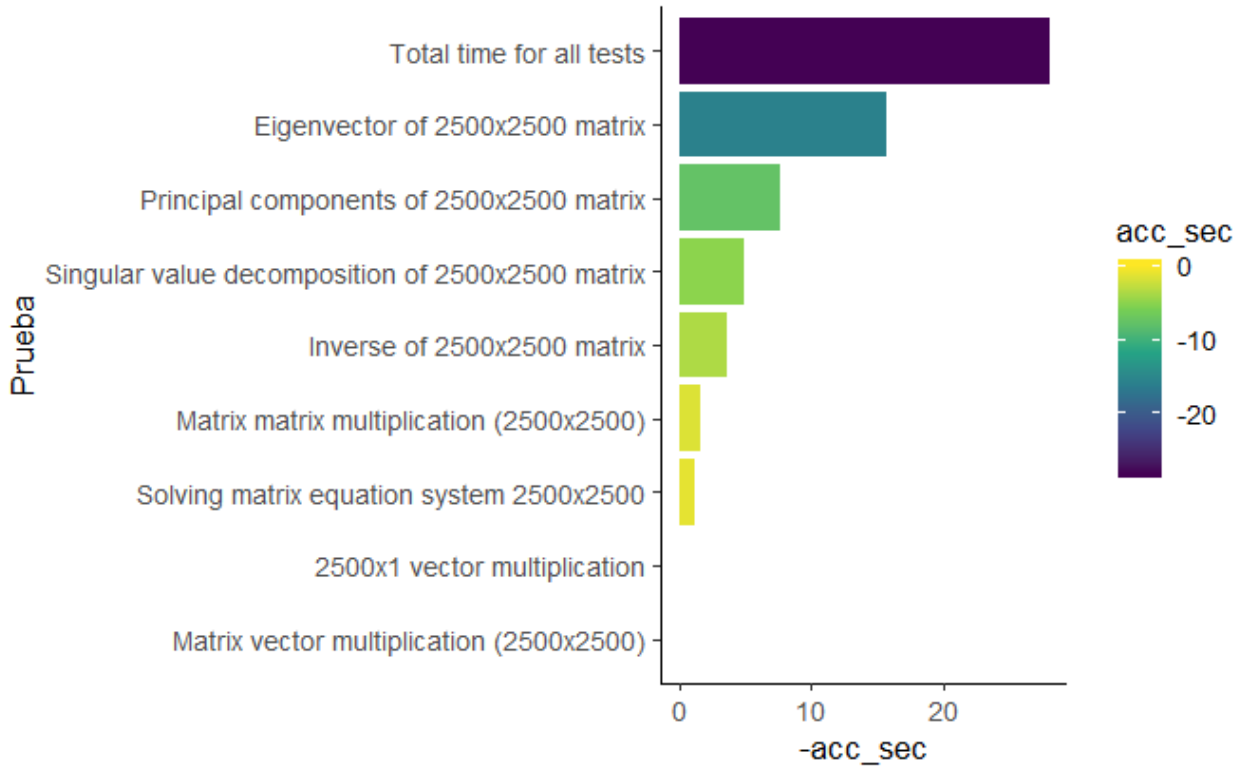


Figure 3: Aceleración (en segundos) por operación matricial.