Universidad de Costa Rica

CA0305 - Herramientas de Ciencia de Datos II

Bitácora 3: Anteproyecto

Alejandro Brenes Calderón - C21319

Santiago Fernández Sáenz- C22943

María Paula Jiménez Torres- C04095

Eyeri Méndez Méndez- C24765

Luis Alberto Juárez Potoy

Escuela de Matemática

I-2024

1 Introducción

El proyecto busca plantear múltiples tareas simples que se realizan en la ciencia de datos y ciencias actuariales. Estas consisten en ejecutar diversos ejercicios con el menor tiempo de ejecución que sea posible, sin importar la cantidad de librerías, líneas de código o complejidad del código, teniendo únicamente en consideración la eficiencia. Se mencionarán algunas de las librerías sobre las que se investigó en los lenguajes de programación de Python y R.

En este sentido, los ejercicios programados en R se realizarán en RStudio y los realizados en Python se ejecutarán en Jupyter Notebook. Junto a cada ejercicio, dentro del código, se añaden las referencias que se utilizaron sobre los paquetes desconocidos por los integrantes.

Al finalizar cada ejercicio, cada uno de los integrantes realizará 10 simulaciones del tiempo de ejecución del ejercicio, con el objetivo de graficar el rendimiento de los ejercicios posteriormente. Para las simulaciones del tiempo se pretende usar 2 librerías, en R la denominada microbenchmark, mientras que en Python correspondería a timeit. Se sabe que hay librerías que grafican directamente el rendimiento de una función (timeit tiene una función para esto, por ejemplo), sin embargo, el objetivo es comparar los tiempos de ejecución entre los distintos integrantes, por lo que se considera más sencillo guardar los tiempos de cada uno y posteriormente compararlos en un único gráfico.

Es necesario mencionar que todos los ejercicios están sujetos a cambios, posteriormente es posible que se añaden comparaciones con las formas más simples de realizar estas operaciones (librerías más conocidas o simplemente paquetes básicos de cada lenguaje), al igual que comparaciones entre librerías (hay varias que ofrecen la opción de paralelización, por ejemplo), y más ejercicios para comparar la eficiencia de ambos lenguajes.

El código completo del proyecto se puede encontrar el repositorio de GitHub, ubicado en la sección de anexos.

2 Ejercicios

A continuación, se presenta una breve explicación de cada uno de los ejercicios, además de las principales librerías que se usaron para realizar estos.

2.1. Ejercicio 1: Operaciones matriciales

Para este ejercicio se pretende realizar una operación sencilla con matrices, la cual es $(matriz^{100} \cdot 10) + 5$. Es importante mencionar que la operación de elevar al exponente 100 la matriz hace referencia a multiplicar la matriz por sí misma 100 veces, pero no de manera matricial, sino elemento por elemento. Para esto se creará una matriz de valores aleatorios entre 0 y 1, de dimensiones $10000 \cdot 10000$, lo cual va fuera del proceso, es decir, no entra para medir el tiempo de ejecución. La idea es mostrar la velocidad de ejecución de la operación, por lo que la creación no es relevante para este contexto. Así, para desarrollar el proceso se hizo lo siguiente:

- En Python: se usa Numpy para la creación de la matriz, mientras que para la optimización de la operación se hizo uso de las librerías NumExpr, Numba y del kernel PyPy.

- En R: para la parte de crear la matriz, al ser esta de un tamaño tan grande, consume mucha memoria y tiempo, por lo que se intentó hacer un objeto bigmatrix, de la librería bigmemory, con el objetivo de utilizar las librerías BLAS y LAPACK con este objeto, para así realizar el ejercicio de manera muy eficiente. Sin embargo, las dos librerías mencionadas anteriormente requieren de una instalación externa y más sofisticada que la instalación de una librería normal para poder usarse de manera óptima, por lo que se decidió no usarlas. Asimismo, se hizo uso de las librerías Matrix, bigalgebra y expm, pero no fueron de gran utilidad ya que dichas librerías se especializan en multiplicación matricial y no en multiplicación elemento por elemento, que es lo que se quiere hacer en este ejercicio. Con todo lo anterior, se tomó la decisión de que la manera más óptima de calcular la operación es simplemente utilizar los operadores $\hat{}$, $*$ y $+$ que vienen incluidos por defecto en R.

2.2. Ejercicio 2: Limpieza de bases de datos

Para este ejercicio se procede a hacer una limpieza de una base de datos: filtrando columnas, corrigiendo ortografía, entre otros. Se descargó una base de datos de muertes en Costa Rica entre los años 2014 y 2021. Además, se quitarán las siguientes columnas: pc, causamuer, des_causa, pcocu, nacmadre, pcregis, gruposcb e instmurio. Lo anterior no entra para medir el tiempo de ejecución, como se ha mencionado anteriormente, el tema de interés es el tiempo de ejecución de el proceso, no de la creación. Ahora, la limpieza de los datos:

- En Python: copy() y drop() se usan para crear una copia del DataFrame original y eliminar columnas específicas. Luego, se aplican filtros utilizando operaciones de comparación en columnas específicas. Las correcciones ortográficas se realizan utilizando el método str.replace(). Además, se utiliza pd.Categorical() para definir categorías y orden en una columna categórica. Así, finalmente, reset_index() se emplea para restablecer los índices del DataFrame resultante.

- En R: Limpieza con data.table: Se convierte el data.frame original en un data.table utilizando la función setDT. Luego, se filtran las filas que cumplan con ciertas condiciones. Se realizan correcciones ortográficas utilizando las funciones chartr y gsub. También se eliminan columnas no deseadas utilizando el operador := para asignar NULL a esas columnas. Además, se realiza un proceso similar de limpieza utilizando data.frames en lugar de data.table.

2.3. Ejercicio 3: Gráficos en paralelo

Para este ejercicio se realizarán múltiples gráficos de una determinada base de datos. Se pretende que se realicen en paralelo. Se usó la misma base del ejercicio anterior. Con las columnas restantes, se realizarán los gráficos, para las numéricas se hará un histograma y para las categóricas un gráfico de barras (un gráfico por columna). Para esto se hizo lo siguiente:

- En Python: se usa la librería joblib para generar los gráficos en paralelo.

- En R: se usa la librería furrr para generar en paralelo los gráficos.

2.4. Ejercicio 4: Imputación de datos por media móvil

Para este ejercicio se programará la media móvil lo más eficiente posible, usando la base de datos de salarios proporcionada en el examen 1.3 del curso CA - 0305 (por el momento, pero está sujeto a cambios para comparar con una mayor cantidad de datos nulos). Se descarga la base de datos ya mencionada, que de igual forma, el cálculo del tiempo para esto no es tomado en cuenta, debido a que es la preparación previa a probar la función. Se permite la utilización de librerías con funciones de media móvil, la comparación de estas permitirá determinar cuál es la más rápida. Para esto se probó:

- En Python: se trabajó con el promedio móvil de librerías ya existentes como pandas, numpy o scipy para ver cual es más eficiente, se buscó paralelizar con joblib o Multiprocessing, además, se investigó sobre la librería modin.
- En R: se probó rollmean del paquete zoo, se pretende paralelizar con Paralell y doParalell, para esto último es imprescindible cerrar los clusters al terminar el proceso. Finalmente se programó sin funciones ya hechas, debido a la cantidad de nulos que devolvían.

2.5. Ejercicio 5: Eliminación de columnas por porcentaje de valores nulos

Para este ejercicio se realizará la eliminación de columnas si el porcentaje de valores nulos que presenta es mayor al indicado. Para realizar esto, al igual que en el ejercicio anterior, se utilizo la base de datos de salarios. Se procedió a hacer lo siguiente:

- En Python: se calcula el número máximo de valores nulos permitidos (max_nulos) con base en el porcentaje dado, luego utiliza dropna() para filtrar las columnas que tienen menos de max_nulos valores nulos, manteniendo así solo las columnas con un número aceptable de valores no nulos. Al final devuelve el DataFrame filtrado.
- En R: se hizo un bucle while para iterar sobre todas las columnas del dataframe y eliminar las deseadas, el cual dio un resultado bastante rápido.

2.6. Ejercicio 6: Imputación de valores nulos por agrupación

Para este ejercicio se imputan valores nulos en la columna Salario base de la base de datos mencionada en ejercicios anteriores. La imputación se realiza por el promedio y agrupando por las columnas Género y Grado de estudio. Al igual que en los ejercicios anteriores, se carga la base de datos sin tomar en cuenta este tiempo. Así:

- En Python: La función imputar_por_agrupacion() se encarga de agrupar los datos por las columnas "Género" "Grado de estudio", y luego usa la función transform() para imputar los valores nulos en la columna "Salario baseçon el promedio del grupo correspondiente. Al final se devuelve el DataFrame con los valores nulos imputados.

- En R: Se dan dos enfoques diferentes para la imputación de valores nulos: data.table y dplyr. En el enfoque data.table, los datos se transforman en un objeto data.table, y luego se imputan los valores nulos con el promedio del grupo utilizando una expresión de asignación por referencia. En cambio, en el enfoque dplyr, se agrupan los datos por "Género" y "Grado de estudio", para después imputar los valores nulos con el promedio del grupo utilizando la función mutate() y ifelse().

3 Conclusiones preeliminares

Tras realizar los ejercicios mencionados anteriormente, se observaron múltiples resultados, a continuación se ponen los más relevantes. Todos estos resultados se pueden ver alterados por descubrimientos de los integrantes del grupo, por lo que no se debe tomar como una conclusión rotunda:

- En el ejercicio de los gráficos, en Python, a uno de los integrantes le presentaba un error la función, lo cual no le ocurrió a los demás participantes del proyecto. Este fallo se solucionó ejecutando el código en una versión posterior de Jupyter Notebook, por lo cual, es importante mencionar que la versión del Jupyter afecta directamente al código, incluso llegando a presentar fallos en el código programado correctamente.

- En R, al paralelizar la función de media móvil, se probó que el tiempo de ejecución aumentaba, por lo que se mantuvo sin esta función.

- A pesar de que hay múltiples funciones disponibles para realizar el promedio móvil, en R se arrojaban más nulos de los que habían originalmente con estas funciones, por lo que se programó sin hacer utilización de las librerías que ofrecen esta función.

- Se hicieron pruebas con el compilador JIT de PyPy, por el momento no se vio un cambio significativo, pero se harán más en el futuro para medir si el costo de instalarlo y hacer que funcione vale la pena.

- Respecto al ejercicio 5, al ser la base de datos no muy grande en el contexto del ejercicio y no contar con columnas con gran cantidad de valores nulos, no se puede apreciar realmente la velocidad del código y, por lo tanto, su eficiencia. Por lo cual no es posible distinguir si realmente la implementación del ejercicio está siendo eficiente o se puede mejorar con alguna librería, herramienta o método.

4 Referencias bibliográficas

Anaconda, Inc. (2020). User Manual. https://numba.readthedocs.io/en/stable/user/index.html

Mersmann, O. Beleites, C. Hurling, R. Friedman, A. Ulrich, J. (2023). Package 'microbenchmark'. https://cran.r-project.org/web/packages/microbenchmark/microbenchmark

pdf

Cooke, D. & Hochberg, T. (2023). NumExpr 2.0 User Guide. https://numexpr.readthedocs.io/en/latest/user_guide.html

García, J. (1992). LA OPTIMIZACIÓN: UNA MEJORA EN LA EJECUCIÓN DE PROGRAMAS. Univesidad de Castilla-La Mancha. https://dialnet.unirioja.es/descarga/articulo/2281727.pdf

GeeksforGeeks. (2024). Timeit in Python with Examples. Sanchhaya Education Private Limited. https://www.geeksforgeeks.org/timeit-python-examples/

Gorelick, M., & Ozsvald, I. (2014). High Performance Python. O'Reilly Media. https://pepa.holla.cz/wp-content/uploads/2016/08/High-Performance-Python.pdf

Joblib developers. (2021). Embarrassingly parallel for loops. https://joblib.readthedocs.io/en/latest/parallel.html

Joblib developers. (2021). joblib.Parallel. https://joblib.readthedocs.io/en/latest/generated/joblib.Parallel.html

NumPy. (2022). Numpy.vectorize. Recuperado el 2 de mayo de 2024 de https://numpy.org/doc/stable/reference/generated/numpy.vectorize.html

Python Software Foundation. (2024). timeit — Measure execution time of small code snippets. https://docs.python.org/es/3/library/timeit.html

S.A. (2024). Introduction to data.table. https://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.html

StackOverflow. (2012). When should I use the := operator in data.table?. https://stackoverflow.com/questions/7029944/when-should-i-use-the-operator-in-data-tabl

5 Anexos

Repositorio de GitHub con el código del proyecto:

`https://github.com/Santifer0803/Proyecto_herramientas_II`