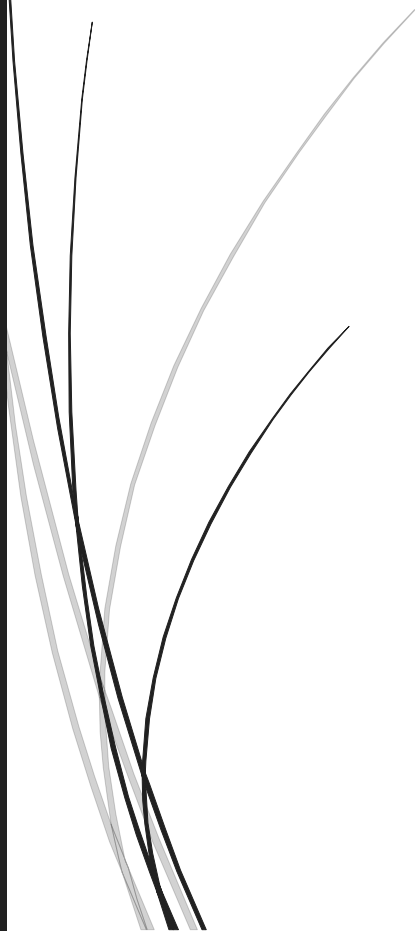
FP-UNA - CONSTRUYENDO EL FUTURO

Estructura de los Lenguajes

Trabajo Practico de Programación

Matias Gabriel Martinez Rebori



Introducción

Este documento describirá mi experiencia en cuanto a las implementaciones realizadas para dar solución al problema dado, los lenguajes de programación utilizados son Python, R, Ruby, Lisp.

Se analizarán las ventajas y desventajas de cada lenguaje utilizado, se hará una comparación entre los lenguajes en base a los criterios de evaluación de lenguajes propuestos por Sebesta. Es mi primera vez utilizando alguno de estos lenguajes.

Desarrollo

Descripción de los lenguajes seleccionados

Los lenguajes elegidos para la resolución del trabajo practico son Python, R, Ruby y Lisp, a continuación, una breve descripción de los lenguajes.

Python es un lenguaje interpretado de alto nivel y de propósito general, su filosofía de diseño es la legibilidad, es fácil de aprender y de usar, multiplataforma y multiparadigma, tipado dinámicamente, donde destaca en los paradigmas imperativo y orientado a objetos. R es un lenguaje de programación para computaciones estadísticas y gráficos, es ampliamente utilizados por matemáticos estadísticos para desarrollar software estadístico y análisis de datos, es interpretado y multiparadigma. Ruby es un lenguaje de programación de alto nivel, interpretado, de propósito general, tipado dinámicamente, soporta varios paradigmas y es multiplataforma, es utilizado principalmente para el desarrollo de aplicaciones web. Lisp es el segundo lenguaje de alto nivel más viejo que aún tiene uso, ha cambiado mucho desde sus inicios, es multiparadigma y tiene tipado dinámico. Elegí Ruby por ser un lenguaje parecido a Python, los otros lenguajes candidatos tenían sus propias desventajas como alta curva de aprendizaje.

Descripción de la experiencia

En Python con librerías fue fácil ya que había mucha documentación y pandas realizaba muy buen trabajo, pero me costó mucho tiempo porque tarde en aprender su framework, implementar las funciones en Python fue fácil ya que pandas provee compatibilidad con numpy y objetos primitivos. La implementación en R fue un poco mas complicada ya que tenia que buscar las funciones para cada caso y ver si eran compatibles para usarse con mi estructura, además algunas cosas tenia que manejarlas manualmente mientras que pandas ya me lo resolvía solo, su método para gráficos es muy potente pero bastante más complicado que en Python. Por último, Ruby, fue la que mas tiempo me costo ya que hay muy poca ayuda de la comunidad en análisis de datos y la documentación es escasa, sin embargo, el framework usado estaba muy bien construido y era parecido a pandas. Utilizar Lisp fue difícil, la documentación es vieja y la ayuda de la comunidad es muy escasa, su sintaxis no es ni de cerca de mi agrado.

Comparación de los lenguajes

La sintaxis de los tres lenguajes del punto uno era bastante cómoda y similar, sin embargo, la que mas me gusto fue la de Ruby que es parecida a la de Python, pero mejor ya que no es necesario paréntesis al llamar a métodos o pasar parámetros, tiene una sentencia para comentar múltiples líneas, agregar un 'end' al final de una sentencia de decisión o iteración me parece que mejora la legibilidad del código. Las sentencias de R son muy parecidas sino iguales a C pero este tipo de notación me es más pesada. La sintaxis de Lisp es antinatural y poco legible.

En los tipos de datos no hay mucho que decir, están bien definidas y trabajadas, todas tienen el tipo Boolean, R sin

embargo ya trae como tipo de dato nativo el DataFrame mientras que en los otros dos lenguajes es necesario utilizar un framework para manejar de una manera más fácil estos datos.

Los tres lenguajes también poseen un alto poder de expresividad, hay varios operadores que permiten hacer mucho con un pequeño programa, la función 'apply' y 'sapply' fue bastante conveniente en R y en una sola línea se puede hacer hacer mucho. Lisp también tiene una interesante expresividad a costa de su legibilidad.

En cuanto al chequeo de tipos ninguno de estos lenguajes son fuertemente tipados por lo tanto no todos los errores son detectados y tienen que hacerlo en run-time debido que su chequeo de tipos es dinámico, resulta en un proceso bastante más lento que lenguajes que lo hacen en tiempo de compilación, sin embargo, los 4 hacen un buen trabajo al detectar errores.

El manejo de excepciones en Python y Ruby son bastante fáciles de usar y poderosos, en R no es tan fácil ya que tiene más opciones y una sintaxis más pesada. Lisp tiene mecanismos para manejar excepciones igual que en otros lenguajes e incluso mas.

Por ultimo el costo, el costo de aprender estos lenguajes es relativamente bajo debido a su simplicidad, excepto por Lisp, sin embargo, en este proyecto, el costo de implementación del ítem 1 del tema 1 fue diferente en los tres lenguajes, empezando por Python, tiene un framework muy bien madurado y una enorme comunidad entorno al desarrollo de programas de análisis de datos o ciencia de datos por lo que es la mejor opción para este campo y donde el costo de implementación de la solución es menor. R es un lenguaje especialmente diseñado para análisis estadístico, gráficos y reportes, no es necesario utilizar un framework debido que tiene una enorme cantidad de funciones nativas para manejar este tipo de problemas, sin embargo, su comunidad es menor que la de Python por lo tanto hay menos recursos

del lenguaje en internet entonces lleva su tiempo. Ruby es un tema aparte, su uso principal es para desarrollo de aplicaciones web por lo que la comunidad y la cantidad de recursos para resolver un problema de análisis de datos es casi nula, el framework que posee para resolver estos problemas aun es joven y no tiene tantas funciones para manejar diferentes problemas, su documentación es bastante simple y casi no contiene ejemplos por lo que es un desafío realizar una solución en Ruby para este tipo de problemas. El costo de implementación en Lisp fue debido a aprender su sintaxis distinta y su estructura, ya que el problema a resolver era bastante fácil y corto.

El costo de escribir programas es Python y Ruby es menor que en R gracias a su alta capacidad de escritura combinada son su simplicidad y sintaxis. La sintaxis de R es un poco mas pesada. El costo de correr un programa en R es menor debido a que su instalación es fácil y tiene una gran cantidad de funciones nativas lo cual hace que no sea necesario utilizar un framework en muchos casos, Python es el siguiente más fácil gracias a su gran comunidad y gran cantidad de frameworks, por último, Ruby debido a su framework joven y con comunidad baja, los tres lenguajes son interpretados por lo cual su tiempo de ejecución es parecida. El costo de mantenimiento en Python es menor gracias a su comunidad y alta legibilidad. Los últimos dos criterios son portabilidad y generalización, Python, Ruby y Lisp están estandarizados mientras que R no, la generalización se refiere a las áreas en las que se puede trabajar con el lenguaje donde el mejor es Python.

Conclusión

Este trabajo tenia una complejidad mas de tiempo que de dificultad, aprender los Python, R y Ruby rápidamente y usarlos fue difícil incluso cuando son lenguajes de alta legibilidad y capacidad de escritura. R es el lenguaje diseñado para este tipo de problemas sin embargo si debiera elegir un solo lenguaje yo elegiría Python ya que tiene una comunidad bastante grande en el área y un framework bastante bueno, cuando necesite gran poder de visualización de dato y gráficos utilizaría R, Ruby no usaría para análisis de datos y Lisp no usaría para nada.

Bibliografía

Pandas getting started

Pandas Dataframe

Pandas Serie

Pandas nlargest

Pandas correlation

Pandas diff

Comparacion Python con R

Drop columns from dataframe in R

Relabel Rows and Columns

Add observation to dataframe

Sort dataframe

Merge in R

Sapply R

Call by reference in R

Daru ruby framework

Daru vector

Daru dataframe

Ruby documentation daru dataframe

Ruby documentation daru vector

Ruby matplotlib

Lisp Tutorial

Pearson correlation wikipedia