UNIVERSIDAD DE Belgrano
BUENOS AIRES - ARGENTINA

Comparación del EDA en Python y R sobre las vacunas de COVID-19 en Argentina.

ALUMNO: IGNACIO NICOLÁS CASTRONUOVO
CARRERA: INGENIERÍA EN INFORMÁTICA
MATRICULA: 502-11655
MAIL:IGNACIO.CASTRONUOVO@COMUNIDAD.UB.EDU.AR
TUTOR: MG. PABLO MIGUEL ÁNGEL PANDOLFO
DIR. DE CARRERA: ING. SERGIO OMAR AGUILERA
CICLO: 2022

Agradecimiento

En pocas palabras, me gustaría agradecer a las personas que estuvieron presentes y me ayudaron mucho a lo largo de mi carrera en la Universidad de Belgrano, para lograr ser la persona que soy hoy.

Primero a mi familia, que siempre durante toda la carrera me apoyaron y confiaron en mí. A los profesores por haberme aportado conocimientos nuevos y formarme para mi vida profesional. A Pablo Pandolfo por brindarme su ayuda en la elaboración de la tesina y, además, siempre que necesité su ayuda estuvo presente.

Finalmente, a mis compañeros y amigos que gracias a ellos viví una de las etapas más lindas de mi vida y seguro que sin su presencia, no hubiera sido lo mismo.

Índice

Índice de figuras

Índice de tablas

1 Resumen

En el presente proyecto final de carrera, se comparan los lenguajes de programación R y Python a la hora de realizar un análisis exploratorio de datos relacionado a las vacunas de COVID-19 aplicadas en Argentina.

Las métricas que se toman para poder realizar la comparación son: tiempo de ejecución, facilidad de aprendizaje, mejor manipulación de datos, mejor gráfico de datos, mejor en estadísticas de datos, mejor en lectura de datos.

Para comparar ambos lenguajes de programación, la herramienta que se utiliza es la matriz ponderada, permitiendo darle un puntaje a cada métrica.

Para obtener los resultados, se realizan dos pruebas sobre el análisis exploratorio de datos en R y Python para así saber, que ponderación tiene cada métrica en los dos lenguajes seleccionados.

Finalmente, una vez obtenidos los resultados de la comparación de Python y R, se analizan y se comparan ambos lenguajes para generar la conclusión final.

2 Introducción

2.1 Planeamiento y contexto del problema

Hoy por hoy, el mundo está viviendo una situación nunca vista antes, sobre todo con las dimensiones y consecuencias a nivel mundial. El problema que causó todo esto, es el famoso COVID-19.

Ni la peste negra o cualquier otra pandemia del pasado ha tenido efectos tan importantes y globales al mismo tiempo. Economías enteras se han paralizado, los gobiernos han intentado distintas estrategias, algunas mejores otras no tanto, pero a pesar de ello, los números de contagios y muertes fueron importantes en todo el planeta.

Las personas y familias se vieron obligados a suspender actividades, hubo rubros paralizados en nuestro país por muchos meses, llegando en muchos casos a necesitar cerrar sus negocios, pérdida de puestos de trabajo, endeudamiento, entre otras cosas. Sumado a lo anterior, el dolor de muchos que padecieron la enfermedad o perdieron seres queridos por esta pandemia. Cambió la forma de vivir y de relacionarnos.

Para buscar algo positivo dentro de tanta negatividad, se desarrollaron y aplicaron tecnologías de comunicaciones remotas, que permitieron en parte, para algunos sectores, seguir trabajando. Otra consecuencia positiva fue el incremento del e-commerce.

En Argentina y en el mundo cada día se registran más casos de enfermos y muertes por COVID-19, pero a la vez también las vacunaciones contra el virus están ayudando a combatirlo. Al principio de la cuarentena era más difícil ver datos de la gente que se enfermaba o que moría cada determinado tiempo, pero hoy en día eso es más frecuente y accesible hasta incluso con las vacunas

aplicadas. En base a los problemas generados por el virus, se ve la necesidad de elaborar un análisis exploratorio de datos (EDA) sobre un dataset con cada lenguaje de programación, para realizar una comparación entre ambos, que permita identificar cuál de ellos es más eficiente en base a las métricas establecidas para este fin.

2.2 Idea Directriz de la Tesina

La investigación, hace una comparación, entre R y Python, permitiendo así conocer cuál de los lenguajes de programación es más adecuado o eficiente sobre el otro para la realización del análisis exploratorio de datos (más conocido por sus siglas como EDA) aplicado a un caso particular el cual son las dosis de vacunas aplicadas en Argentina contra el COVID-19. Esto se hace utilizando un dataset con personas vacunadas contra el COVID-19 en Argentina y con el establecimiento de métricas para realizar la comparación entre los dos lenguajes.

2.3 Hipótesis de trabajo

Aunque R y Python son dos lenguajes que pueden ser utilizados para realizar un análisis exploratorio de datos en la aplicación de vacunas contra el COVID-19 en Argentina, la hipótesis es que “R es el más adecuado ya que es un lenguaje más orientado al análisis estadístico y datos mientras que Python es un lenguaje de propósito general”.

2.4 Objetivo general y específicos

Determinar entre R y Python cuál de los dos lenguajes aplica mejor al momento de realizar un análisis exploratorio de datos sobre la aplicación de las vacunas de COVID-19 en Argentina en el año 2022.

Para esto se propone un trabajo de investigación que permite analizar un dataset sobre las personas vacunadas contra el COVID-19 en Argentina, mediante la utilización de los lenguajes de programación R y Python para poder realizar una comparación en base a las métricas definidas en la tesina.

2.5 Metodología

El enfoque de la investigación es deductivo y de carácter teórico y experimental ya que se realiza la codificación del análisis exploratorio de datos en los lenguajes de programación.

Se aplica de forma trasversal, dado que se recolectan los datos de un único lugar y momento. Por último, es un trabajo de tipo correlacional, porque se comparan Python y R en el análisis exploratorio de datos.

2.6 Justificación del trabajo

Como ya se mencionó anteriormente en la sección de "Planeamiento y contexto del problema" hoy en día se está atravesando una etapa difícil y complicada en todo el mundo a causa del COVID-19. El hecho de poder ver que lenguaje es mejor para realizar el análisis exploratorio de datos, permite saber a los data scientist o a cualquier persona que quiere tener más información sobre las vacunas, a responder con mayor claridad, facilidad y eficiencia las preguntas que se plantean al momento de conocer mejor los datos. Esto permite que ya se sepa que lenguaje de programación conviene usar.

2.7 Delimitaciones o alcances de la tesina

Este trabajo final de carrera, se enfoca en comparar la performance de ambos lenguajes de programación, por lo tanto, el límite que presenta es el análisis comparativo de las métricas

definidas que permite llegar a comprobar cual lenguaje elegir para este fin. Se ven los resultados obtenidos en R / Python y no el análisis de los datos sobre el COVID-19 propiamente dichos.

Con respecto al objeto sometido a la comparación, son solamente los lenguajes de programación R y Python quedando fuera los demás lenguajes de programación.

El entregable final contiene una matriz ponderada que permite comparar los dos lenguajes con las métricas establecidas y el código del EDA.

3 Marco teórico

3.1 Conceptos fundamentales

3.1.1 Análisis exploratorio de datos.

El análisis exploratorio de datos, más bien conocido como EDA, sirve para analizar e investigar un conjunto de datos resumiendo las características principales que poseen y por lo general se utilizan métodos para la visualización para que esos datos se transformen en información. El EDA ayuda a la hora de saber cuál es la mejor forma de poder manipularlos y así tener la solución o respuesta que se requiere para poder descubrir patrones, detectar anomalías, probar una hipótesis, etc.

El análisis exploratorio de datos se usa generalmente para conocer que aportan los datos sacando de lado el objetivo principal de su utilización, por ejemplo, permite comprender mejor las variables del conjunto de datos y la relación que existe entre ellas. Otro caso en el que puede ayudar es saber si las técnicas estadísticas que se están usando para el análisis de los datos, son correctas o se necesitan otras técnicas. En resumen, el EDA es un proceso de descubrimiento de datos.

Como se mencionó antes, el objetivo del análisis exploratorio de datos es ayudar a estudiar los datos antes de hacer cualquier suposición. Este se puede utilizar para garantizar que los resultados son precisos y se pueden utilizar para lograr los objetivos buscados. Además, permite verificar si las preguntas que se están haciendo son correctas. Responde a preguntas relacionadas con conceptos estadísticos.

Existen cuatro tipos principales de EDA:

1. **No gráfico univariante.** Es la forma más simple donde los datos que se analizan consisten en una sola variable. Al ser una variable única, no interviene en las causas o relaciones que existen. El objetivo del análisis univariante es la descripción de los datos e identificación de patrones que hay entre ellos.

2. **Gráfico univariante.** Los métodos que no son gráficos, no permiten mostrar una imagen completa de los datos. Por lo tanto, se requieren métodos gráficos. Los gráficos comunes de gráficos univariantes son:

   a. Diagramas de tallo y hojas, estos muestran los valores de datos y la distribución que tienen.

   b. Histogramas, diagramas de barras en los que cada barra representa la frecuencia (recuento) o la proporción (recuento/recuento total) de casos para un rango de valores.

   c. Diagramas de caja o boxplot, que representan gráficamente el resumen de cinco números de mínimo, primer cuartil, mediana, tercer cuartil y máximo.

3. **No gráfico multivariante.** Se obtienen los datos de más de una variable. Las técnicas de EDA no gráficas y multivariantes por lo general muestra la relación entre dos o más variables de los datos mediante la tabulación cruzada o utilización de las estadísticas.

4. **Gráfico multivariante.** los datos multivariantes utilizan gráficos para mostrar relaciones entre dos o más conjuntos de datos. El gráfico más utilizado es un diagrama de barras agrupadas o un gráfico de barras donde cada grupo representa un nivel de una de las variables y cada barra dentro de un grupo representa los niveles de la otra variable.

Otros tipos comunes de gráficos son:

1. Diagrama de dispersión o scatter plot, permite graficar y visualizar variables numéricas con un eje x e y. Si posee muchos datos es difícil encontrar un patrón o distinguir densidades. Es útil para ver relaciones entre 2 variables numéricas.

2. Gráfico multivariante, que es una representación gráfica de las relaciones entre los factores y una respuesta.

3. Diagrama de comportamiento, es un gráfico de líneas de datos a lo largo del tiempo.

4. Gráfico de burbujas, que es una visualización de datos que muestra varios círculos (burbujas) en un gráfico de dos dimensiones.

5. Mapa de calor, que es una representación gráfica de datos donde los valores se representan por color.

Actualmente las dos herramientas más comunes para realizar el análisis exploratorio de datos son R y Python.

Para finalizar sobre la explicación del análisis exploratorio de datos, es importante resaltar que no existe una única forma de realizarlo. No hay una regla sobre cómo realizar un EDA, depende de cada uno y de las necesidades o capacidades que tenga para realizarlo a su gusto, ya que la idea principal es poder responder las preguntas que tenemos antes de comenzarlo y en específico, para conocer el conjunto de datos con el que se está trabajando.

3.1.2 Coronavirus y COVID-19.

El coronavirus es un virus que causa enfermedades que pueden generar resfriados comunes o hasta llegar a ser enfermedades más graves como neumonía, síndrome respiratorio de oriente medio, síndrome respiratorio agudo grave, etc. Es importante destacar que el coronavirus que

genero al principio un brote en China, conocido como COVID-19, es una nueva variante de coronavirus no se conocía antes.

Los primeros datos sobre COVID-19 aparecieron en diciembre de 2019, la organización mundial de la salud recibió reportes sobre la presencia de neumonía con un origen desconocido en China. Luego a principios de enero, en ese país pudieron identificar la causa como una nueva cepa de coronavirus. Esta enfermedad se fue expandiendo a diferentes continentes y sus respectivos países, generando una pandemia mundial.

En cuanto al origen específico de esta cepa del virus, algunos expertos concluyeron que inició con el contagio en murciélagos o pangolines y luego se transmitió al ser humano, la verdad es que aún no se ha logrado confirmar el origen del COVID-19.

Algunos síntomas que puede presentar el coronavirus son:

- Síntomas respiratorios (similares a los de un resfriado)
- Fiebre (alta temperatura)
- Tos seca
- Falta de aliento o cansancio
- Dificultades respiratorias

En algunos casos de mayor gravedad, el virus puede generar neumonía o síndrome respiratorio agudo grave que es una neumonía aún más dañina para la salud, insuficiencia renal y hasta incluso la muerte. En otros casos algunas personas no poseen ningún síntoma y se los considera asintomáticos, aunque igualmente pueden contagiar al resto de la población.

Cabe destacarse que a lo largo del tiempo se fueron generando nuevas variantes del virus, como la variante "Delta", o la variante "Ómicron" que en enero de 2022 comenzó a circular comunitariamente. Ómicron es más leve que la Delta (solo comparte algunos síntomas) pero es extremadamente contagiosa.

Según lo informado por la OMS, el coronavirus se transmite por contacto con otra persona u objeto infectado. Por lo tanto, la mejor manera para evitar contraer el virus, es poder seguir las buenas prácticas de higiene y protocolos definidos como se enuncia a continuación:

- Mantenerse alejado de las personas enfermas.
- Utilizar barbijos aprobados que cubran nariz y boca completamente.
- No tocarse la cara (boca, nariz u ojos).
- Mantener una distancia mínima de un metro con el resto de las personas. (Distanciamiento Social).
- Lavarse las manos frecuentemente y a fondo por, al menos 20 segundos, con un desinfectante para manos a base de alcohol o lávalas con agua y jabón. Es importante hacerlo incluso si no hay suciedad visible en las, manos, especialmente luego de toser o estornudar; si está cuidando a alguien; cuando está preparando alimentos, cocinando carnes y/o huevos, luego de comer, ir al baño; si sus manos están sucias, y/o ha estado cerca de una granja o animales salvajes, etc.
- Cuidar la higiene respiratoria cubriéndose la boca y la nariz con el codo o pañuelo doblado cuando toses o estornudas. Desecha inmediatamente el tejido usado.

- Quedarse en casa y practicar el aislamiento social o cuarentena o si no se encuentra bien y tiene alguno de los síntomas característicos de COVID-19 (hisoparse y/o concurrir a una clínica para ser atendido por profesionales).
- Seguir las indicaciones actualizadas de las autoridades sanitarias del país.

En base a las problemáticas sobre la salud que presentaba este nuevo virus, en muchos lugares del mundo se decidió el uso de la cuarentena. Esta consistía en que las personas se queden en sus casas para prevenir el contagio con el virus y el no contagiar al resto en el caso de tener el virus. Dependiendo de cada país, las medidas fueron de una manera más estrictas.

La cuarentena es una medida de salud pública para controlar la transmisión de un virus y generar un efecto domino. Esto les indica a las personas que no se expongan al virus haciendo que no tengan tanto contacto para que no se puede desarrollar y transmitir la infección.

Las medidas que más se recomiendan en el caso de tener el virus, son permanecer solo en una habitación de la casa el mayor tiempo posible sin tener contacto con otra persona, tener un baño propio y utilizar barbijo siempre que sea necesario salir de la habitación. En el caso de vivir con otras personas, también deben llevar mascarillas. Tampoco se debe salir de la casa durante el cumplimiento del aislamiento.

El aislamiento debe durar unos días desde el último contacto con el caso positivo del virus. Esto dura un tiempo ya que la mayoría de las personas desarrollan y transmiten síntomas en los primeros días de ser detectado como un caso positivo. Hay que tener cuidado también si aparecen síntomas durante el aislamiento y en los primeros días después de finalizarla. Es importante cumplir con el aislamiento ya que puede transmitirse desde dos días antes del inicio de síntomas. Las personas

asintomáticas también transmiten el virus. Aunque se realice una prueba y esta de negativa, es necesario continuar con el cumplimiento de esto hasta el último día, ya que pueden aparecer síntomas de la enfermedad de forma posterior.

3.1.3 Vacunas en Argentina.

Las vacunas que se están aplicaron y se están aplicando hoy en día, son las siguientes:

- **Sputnik V.** Su nombre original es GAM-COVID-VAC y fue desarrollada en el Centro Nacional Gamaleya de Epidemiología y Microbiología en Rusia. Es una vacuna autorizada para personas mayores e iguales a 18 años. Se aplican 2 dosis, en la primera las contraindicaciones que posee son: hipersensibilidad a cualquier cosa, antecedente de reacciones alérgicas graves o anafilaxia, enfermedades agudas graves o exacerbación de enfermedades crónicas. Con la segunda dosis las contraindicaciones que posee son: complicaciones graves posvacunación (shock anafiláctico, reacciones alérgicas generalizadas y graves, síndrome convulsivo, fiebre superior a 40° C, etc.) por la inyección del componente 1 de la vacuna.

- **Covishield.** Su nombre original es Vacuna contra covid-19 ChAdOx1 nCoV- 19 Corona Virus Vaccine y fue desarrollada en el Serum Institute en India. Es una vacuna autorizada para personas mayores e iguales a 18 años. Se aplican 2 dosis, en la primera la contraindicación que posee es: hipersensibilidad a cualquier cosa. Con la segunda dosis la contraindicación que posee es: reacción anafiláctica con la primera dosis.

- **Sinopharm.** Su nombre original es SARS COV-2 y fue desarrollada en el Beijing Institute of Biological Products en China. Es una vacuna autorizada para personas mayores e iguales a 3 años. Se aplican 2 dosis, en la primera las contraindicaciones que posee son: hipersensibilidad a

cualquier cosa, antecedente de reacciones alérgicas graves (con compromiso respiratorio que haya requerido asistencia médica), exacerbación de enfermedades crónicas que impliquen compromiso del estado general. Con la segunda dosis la contraindicación que posee es: reacción anafiláctica con la primera dosis.

- **Astrazeneca.** Su nombre original es ChAdOx1 nCoV-19 vaccine y fue desarrollada en el AstraZeneca-Oxford en el Reino Unido. Es una vacuna autorizada para personas mayores e iguales a 18 años. Se aplican 2 dosis, en la primera la contraindicación que posee es: hipersensibilidad a cualquier cosa. Con la segunda dosis la contraindicación que posee es: reacción anafiláctica con la primera dosis.

- **Moderna.** Su nombre original es Vacuna mRAN-1273 COVID-19 y fue desarrollada en el Moderna Switzerland GmbH. Es una vacuna autorizada para personas mayores e iguales a 12 años. Se aplican 2 dosis, en la primera la contraindicación que posee es: Hipersensibilidad a cualquier cosa de una vacuna o a una vacuna que contenga componentes similares. Con la segunda dosis la contraindicación que posee es: anafilaxia o reacción alérgica grave inmediata a la administración de la primera dosis. También incluye contradicciones temporales: enfermedades agudas graves (infecciosas y no infecciosas) o exacerbación de enfermedades crónicas que impliquen compromiso del estado general (ej. asma grave no controlado).

- **Convidencia.** Su nombre original es Ad5-nCoV y fue desarrollada en el Instituto de Biotecnología de Beijing y CanSino Biologics Inc. Es una vacuna autorizada para personas mayores e iguales a 18 años. Se aplica 1 dosis, y las contradicciones que se deben tener son las siguientes: Hipersensibilidad a cualquier cosa de una vacuna o a una vacuna que contenga

componentes similares, antecedente de reacciones alérgicas graves (con compromiso respiratorio que haya requerido asistencia médica), exacerbación de enfermedades crónicas que impliquen compromiso del estado general, epilepsia no controlada y otras enfermedades neurológicas progresivas o historia de síndrome de Guillain-Barré.

- **Comirnaty.** Su nombre original es Pfizer-BionTech COVID-19 Vaccine y fue desarrollada en el Pfizer-BioNTech. Es una vacuna autorizada para personas mayores e iguales a 12 años. Se aplican 2 dosis, en la primera la contraindicación que posee es: Hipersensibilidad a cualquier cosa de una vacuna o a una vacuna que contenga componentes similares. Con la segunda dosis la contraindicación que posee es: anafilaxia o reacción alérgica grave inmediata a la administración de la primera dosis. También incluye contradicciones temporales: enfermedades agudas graves (infecciosas y no infecciosas) o exacerbación de enfermedades crónicas que impliquen compromiso del estado general (ej. asma grave no controlado).

A medida que han ido apareciendo nuevas variantes y que se conocieron mayores y mejores datos sobre el tiempo de efectividad de tener dos dosis de vacunas, se han ido dando refuerzos de estas. Se han hecho estudios para combinar vacunas, haciendo que las personas no necesariamente una vez recibida una determinada vacuna queden dependiendo de la disponibilidad de esta.

3.1.4 Lenguaje de programación “Python”.

Python es un lenguaje de programación multiplataforma, multiparadigma, interpretado y de código abierto que puede utilizarse para cualquier proyecto o software ya que es un lenguaje de propósito general.

Hoy en día es el lenguaje de programación más utilizado y popular del mundo según el índice PYPL (Popularity of Programming Language), esto se debe a la versatilidad y facilidad de aprendizaje que posee (es considerado el más sencillo de aprender) ya que tiene una sintaxis que es única centrada en la legibilidad (los desarrolladores pueden leer y traducir código de Python de forma mucho más fácil que con otros lenguajes).

Python es un lenguaje que se basa en C y C++ y sus raíces vienen del sistema operativo Unix aunque en la actualidad se lo puede ver presente en distintos sistemas operativos. Prácticamente casi todas las aplicaciones más populares de hoy, como lo es Google, Instagram y Netflix, funcionan gracias al mantenimiento que ofrece Python.

El lenguaje de programación Python fue creado por Guido van Rossum. Él cuenta que en su anterior trabajo utilizaba el lenguaje de programación ABC el cual le gustaban algunos aspectos de este, pero no estaba a gusto con el hecho de la dificultad de difusión del lenguaje de programación. Por lo tanto, van Rossum decidió crear su propio lenguaje de programación que no le llevo un poco más de un año para presentar la primera versión en 1991 de USENET.

Por otro lado, van Rossum comenta que le gustaba leer los textos sobre “El circo volador de Monty Python” y entonces para que el lenguaje tenga un nombre corto, único y que sea ligeramente misterioso, paso para llamarse Python.

Algunas ventajas que posee Python son las siguientes:

- Es fácil y sencillo de aprender.
- Tiene buena legibilidad de código.
- Posee libre uso y distribución.

- Permite realizar desarrollos de forma fácil, ágil y rápida.
- Al ser multiplataforma puede utilizarse en diferentes sistemas operativos.
- Al ser tan popular, posee una comunidad grande.
- Es el lenguaje más solicitado en el mercado laboral.

El funcionamiento del lenguaje Python es simple, en lugar de una lista larga de instrucciones que eran estándar para los lenguajes de programación funcional, utiliza módulos de código que son intercambiables.

La implementación estándar de Python es "cpython". No convierte el código en código máquina, sino que lo convierte en código byte el cual no es entendido por la CPU. Por lo tanto, para ejecutar los códigos bytes, se necesita un intérprete llamado "Máquina Virtual Python".

El intérprete de Python para poder ejecutar un programa debe realizar los siguientes pasos:

1. El intérprete lee un código o instrucción de Python. Luego se comprueba que la sintaxis de cada línea sea correcta, pero en caso de arrojar algún error, se detiene la traducción y se muestra el error.

2. Si no existe ningún error, el intérprete procede a generar la traducción a código byte. Si la ejecución fue exitosa, se traduce por completo el código a código byte

3. Finalmente, el código byte es enviado a la Máquina Virtual Python para ser ejecutado nuevamente. Si en el proceso de la ejecución ocurre un error, se detendrá y se mostrará un mensaje con el error.

3.1.5 Lenguaje de programación "R".

R es un lenguaje de programación multiplataforma, interpretado y no compilado, que posee una licencia abierta y totalmente gratuita para la computación estadística y creación de gráficos. R no es uno de los lenguajes más populares como por ejemplo lo puede ser Python, pero igualmente fue creciendo su popularidad en el ambiente de ciencia de datos. Como muchos otros lenguajes de programación, se fundamenta la escritura en línea de comandos, aunque se puede codificar en IDEs para tener mayor practicidad y una mejor estructura.

El código interno de R esta hecho en C y Fortran para cálculos y procesos que sean extensos, aunque muchas funciones de R están escritas en el propio lenguaje, esto facilita al usuario con la comprensión de los algoritmos que las componen y permite modificar de forma más fácil los procesos según las necesidades. También funciona para los sistemas operativo UNIX, Windows, MacOS, FreeBSD y Linux.

Los comienzos de R vienen del lenguaje de programación S que es un lenguaje creado en los laboratorios Bell de Estados Unidos. En estos mismos laboratorios, por ejemplo, se creó el sistema operativo Unix, entre otras cosas.

R también posee una comunidad muy activa para el desarrollo de paquetes, especialmente orientados a la modelización, estadística clásica y visualización de datos que son los puntos más fuertes del lenguaje. Aun así, R posee un repositorio de paquetes muy grande pudiendo encontrar paquetes de cualquier tema relacionado a la ciencia de datos.

En su momento, S y sus estándares al ser de los Laboratorios Bell, restringía su uso para las personas. Por lo tanto, Ross Ihaka y Robert Gentleman, tuvieron la idea de crear una

implementación abierta y gratuita de S. R comenzó con su creación en el año 1992, su primera versión inicial en 1995 y su versión final estable en 2000.

R heredo muchas características provenientes de S, por lo que entonces se puede correr código de este lenguaje utilizando R. Para que se pueda lograr esto, se creó una forma en la cual se puedan realizar tareas con S y otra con R. La proveniente de S el problema que trae es que posee inconsistencias, la sintaxis no es intuitiva y es difícil para las personas que quieren aprender R.

Para poder lograr el objetivo de que R no posea restricciones de uso y sea de código libre y abierto, fue distribuida de manera gratuita a través de la Licencia Publica y General de GNU.

R es un lenguaje en el cual se puede ser de gran ayuda en las siguientes áreas:

- Investigación científica.
- Manipulación de datos.
- Análisis estadístico.
- Inteligencia artificial.
- Aprendizaje automático o Machine Learning.
- Técnicas gráficas.
- Modelado y predicciones.
- Matemáticas financieras.
- Bioinformática.
- Investigación biomédica.

También, algunas de las características que posee y lo benefician, son:

- Lenguaje extensible.

- Orientado a objetos.
- Integrable.
- Gráficos potentes y avanzados.
- Es accesible.
- Bueno para el análisis y calculo estadístico.
- Admite varios tipos de datos.
- Puede procesar grandes conjuntos de datos.
- Gran cantidad de paquetes para la ciencia de datos.

3.1.6 Anaconda.

Anaconda es una distribución libre y open Source para los lenguajes de programación Python y R. Fue desarrollado en Python y lanzado en el año 2012. Es muy común ver a Anaconda en acción cuando se trata de temas sobre ciencia de datos, Machine learning, Ingeniería, análisis predictivos, Big data, etc. Con este se pueden crear desarrollos para Python y R con las librerías que se deseen. Este software también está orientado a la simplificación de despliegues y administración de paquetes.

Anaconda descarga muchas aplicaciones por defecto que son muy utilizadas en las disciplinas mencionadas anteriormente en lugar de tener que ir instalando una por una. Algunos ejemplos de estos son:

- Numpy
- Pandas
- Matplotlib

- Jupyter
- Virtual Studio Code

Anaconda posee una cantidad de paquetes tan grande que no se limita solo al análisis de datos porque también se pueden descargar plugins o librerías que son especializados al desarrollo web, web scrapping, entre otras cosas.

Anaconda también ofrece la posibilidad de ser usado en los sistemas operativos Windows, Linux y MacOS.

En la Figura 1 puede observarse la interfaz del navegador que posee Anaconda con sus distintas aplicaciones y funcionalidades.

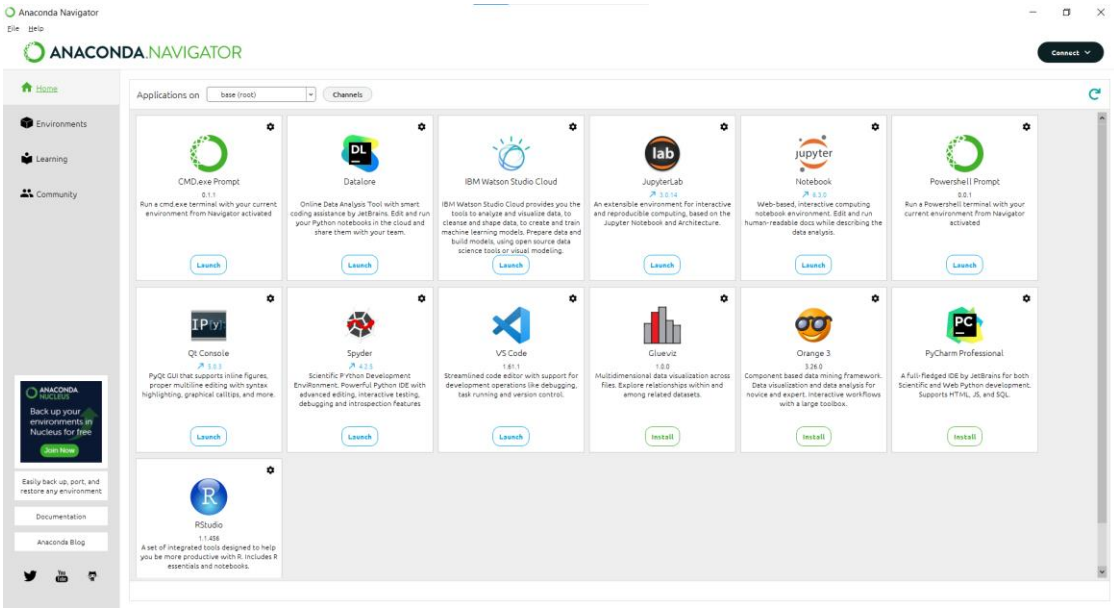


Figura 1. Interfaz de Anaconda.

Solo se deberá oprimir en el botón “Lunch” en el caso de querer ejecutar algún programa. En el caso de querer descargar algún software, se debe oprimir en el botón “Install” y cuando esté instalado, luego aparecerá como “Lunch” para poder correr el programa.

3.1.7 Jupyter notebook.

Jupyter Notebook es una interfaz web de código abierto que permite incluir textos, videos, audios, imágenes y ejecución de código de diferentes lenguajes mediante un navegador web. La ejecución se puede hacer gracias a la comunicación que existe con un núcleo de cálculo, más bien conocido como Kernel. Aunque los desarrolladores de Jupyter Notebook hayan incluido por defecto únicamente el núcleo de Python para codificar, al ser de carácter abierto se permite aumentar los núcleos que se posean disponibles, por ejemplo, se pueden incluir R, Octave, Julia, Haskell, Ruby, C/C++, Java, Fortran, etc. El nombre Jupyter proviene de la unión de los lenguajes de programación Julia, Python, R.

Jupyter Notebook brinda un entorno enfocado en poder satisfacer necesidades concretas y ajustarse al ambiente de la ciencia de datos y simulación numérica. En la interfaz los usuarios pueden escribir, documentar, ejecutar código, visualizar datos, realizar cálculos, ver resultados y muchas cosas más. Al ser cuadernos, esto permite que el código se pueda organizar en celdas independientes, es decir, probar los bloques que se deseen ejecutar del código de forma individual.

Algunos usos que se le dan a las Jupyter Notebooks, son:

- Depuración de datos: Distinguir datos importantes.
- Modelización estadística: Estimación de la probabilidad de distribución de una característica especifica

- Creación y entrenamiento de modelos de aprendizaje automático: Diseño, programación y entrenamiento de modelos basados en aprendizaje automático.
- Visualización de datos: Representación gráfica de los datos.

Jupyter Notebook se creó con el objetivo de ser la evolución Ipython Notebook, teniendo las mismas funcionalidades que ya poseía, pero se le añadió la posibilidad de ejecutar código en diferentes lenguajes de programación. También cabe destacar que la interfaz web de Jupyter Notebook y Ipython Notebook, son de desarrollo en software libre en un "cuaderno computacional" como lo eran en los cuadernos realizados en el programa Mathematica o en Maple y cuyo primer desarrollo fue hecho por William Stein para el programa llamado cálculo Sage (Sage Notebooks). Para poder llevar a cabo este desarrollo, Fernando Pérez y Robert Kern utilizaron en principio el lenguaje de programación Python ya que posee gran versatilidad, es adoptado por la comunidad científica y es un lenguaje libre. Al principio del desarrollo no se centraron en realizar la interfaz web, sino en proporcionar un intérprete de comandos que sea interactivo para Python. Luego en 2010 se comenzó con el desarrollo que generó la aparición de la interfaz de Jupyter Notebook.

Desde ese momento en adelante, los cuadernos computacionales no dejaron de crecer y cada vez aumenta más su uso. Por ejemplo, hoy en día, Github permite que las Notebooks realizadas en Jupyter puedan ser directamente visualizadas desde su página web.

3.1.8 Rstudio.

RStudio es un software que viene dentro de Anaconda que permite el desarrollo con R y otros lenguajes de programación que estén relacionados al trabajo con grandes cantidades de datos,

temas estadísticos, gráficos, análisis de datos, etc. Es un IDE muy completo para el desarrollo que permite también integrarse con algunas herramientas que están enfocadas a la gestión de proyectos. Es un software libre, entonces se puede descargar y utilizar de forma gratuita.

Como se mencionó recién, el IDE puede ser utilizado por cualquiera, pero igualmente las personas que más lo utilizan suelen ser científicos, académicos, analistas financieros, estudiantes. Esto más que nada es por la necesidad de usar el lenguaje de programación R.

RStudio brinda herramientas con las que se espera encontrarse en un IDE moderno, por ejemplo, la sintaxis es coloreada, ayuda a completar código, formatea el código, etc.

El entorno para desarrollar que posee, tiene varias herramientas adicionales dentro del espacio de trabajo, como lo puede ser la documentación del lenguaje R, sistemas de control de versiones (por ejemplo, Git), gestión de proyectos y visualización de datos. También posee un depurador que permite localizar y corregir los errores que haya en el código de forma fácil. Se puede extender mediante paquetes adicionales en base a las necesidades que se posean para realizar el objetivo del proyecto.

Para entrar en la categoría de machine learning, se puede mencionar que brinda un entorno bueno en el cual se puede desarrollar y depurar el código de forma fácil y organizar los documentos que el proyecto posea. Tiene una interfaz que es una plataforma para el análisis y cálculo de proyectos con cantidades de datos grandes o funciones matemáticas complejas.

Para utilizarlo mediante Anaconda, se deberá oprimir el botón de “Install” ya que no viene instalado por defecto. Luego de instalar el RStudio, se podrá utilizar la herramienta.

Algunas ventajas adicionales que presenta este IDE además de lo mencionado anteriormente son las siguientes:

- Muestra los objetos en el lugar de trabajo
- Muestra el historial de comandos
- Integra ayuda
- Posee visores de gráficos y paquetes
- Se pueden abrir varios scripts a la vez
- Se pueden ejecutar pedazos de código
- Se pueden hacer presentaciones con HTML 5

3.1.9 Dataframe.

Los dataframes son estructuras de datos rectangulares que pueden poseer distintos tipos de datos, por lo que entonces, son heterogéneos. Es una de las estructuras de datos más usada para el ambiente del análisis de datos y data science. En breves palabras, un dataframe es una especie de base de datos o tabla la cual posee datos de distintos tipos que luego serán utilizados para un fin en particular.

Los dataframes son una versión un poco distinta a las matrices ya que, en estas, todas las celdas deben tener datos del mismo tipo, pero las líneas de un dataframe admiten datos de cualquier tipo, aunque la columna debe poseer siempre el mismo tipo de dato. En la Figura 2 puede observarse la estructura que posee un dataframe.

Las filas de un dataframe, pueden mostrar casos, personas, observaciones, mientras que las columnas muestran atributos, rasgos o variables. Un ejemplo de esto sería que un dataframe de

vacunas contra COVID-19 aplicadas, las filas muestran la información de las personas. Mientras que las columnas muestran, la vacuna que se aplicó, a que grupo de edad pertenece, sexo, etc.

Es importante remarcar que un dataframe está compuesto a la vez por varios vectores. A los vectores se les puede poner un nombre que luego pasara a ser el nombre de la columna. Si los vectores que se utilizan para generar un dataframe no poseen el mismo largo, los datos no se mostraran y saldrá un error.

Figura 2. Estructura de un dataframe.

3.1.10 Matriz ponderada.

La matriz ponderada, consiste en armar una tabla con filas y columnas que queden enfrentadas permitiendo realizar una elección. En base a la ponderación que se aplique, se podrán comparar las diferentes filas y columnas y así obtener cual es el resultado que mejor aplica a una situación en base a las métricas planteadas y puntuadas.

En síntesis, la matriz ponderada consiste en una serie de criterios ponderados y utilizados para elegir entre un conjunto de varias opciones.

Algunas de las ventajas que presenta la matriz ponderada son:

- **Flexibilidad.** Se adapta al gusto y forma del que va a utilizar la herramienta. Se pueden poner pocas opciones, criterios o muchas opciones y criterios.
- **Fácil.** Es fácil la utilización de la matriz y más para trabajos en equipo. Puede ser vista en una pantalla o tablero.
- **Parametrizable.** Las matrices que posean muchas opciones y criterios, se puede parametrizar en un software y dejar que el programa decida y califique en base a las ordenes que se le envían. Esto igualmente requiere de la ayuda de la programación.
- **Facilidad de consenso.** Cuando hay dificultad en la selección para ponerse de acuerdo, la matriz ponderada puede solucionar el problema.

Existen distintos tipos de matrices ponderadas:

- **Método del criterio analítico completo.** Sirve a la hora de trabajar con equipos que son más chicos y cuando el caso presenta pocas opciones y criterios. Se necesitan hacer las tablas de criterios, priorización y la unión entre criterios y priorización. Para seleccionar cuál de las opciones en base a los criterios se debe seleccionar, es la que en el resultado total obtuvo más puntos. En la Figura 3 puede observarse un ejemplo en el cual se ve el resultado final de la matriz ponderada de este método.

- **Método del consenso de criterios.** Es mejor utilizarla cuando el equipo es más grande ya que se van a considerar más criterios y opciones. No posee tantos pasos como el método del criterio analítico completo.
- **Método de combinación ID.** Es un método basado en causa y efecto en vez de criterios. Es ideal en el caso de que existan muchas interrelaciones entre las opciones.

	Facilidad de uso	Acceso e integración de fuentes de datos	Creación de visualización y paneles de datos	Soporte para R y Python	Precio	Total
Power BI	0,215	0,146	0,003	0,017	0,039	0,42
Tableau	0,215	0,003	0,163	0,049	0,001	0,43
Qlikview	0,014	0,051	0,056	0,001	0,027	0,15

Figura 3. Matriz ponderada.

3.1.11 Librería Pandas – Python.

Pandas es una librería importante y de código abierto de Python. Esta librería es muy utilizada en el área de Data Science y Machine Learning, porque ofrece estructuras que son poderosas y flexibles facilitando la manipulación y tratamiento de datos.

Pandas nació por la necesidad de unir en una sola librería, todo lo necesario para que un analista de datos trabaje con una herramienta en la cual posea todas las funcionalidades que se requieran utilizar en un proyecto como por ejemplo la carga de datos, modelarlos, analizarlos, manipularlos y prepararlos.

En la Figura 4 se muestra un resumen de la librería Pandas

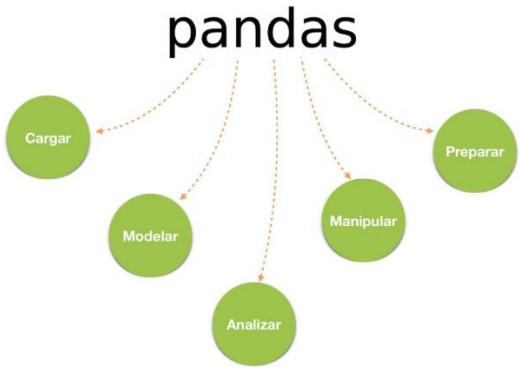


Figura 4. Librería Pandas.

Las dos estructuras de datos principales que posee “Pandas”, son:

- **Series.** Es un estilo de array unidimensional que puede almacenar cualquier tipo de dato
- **Dataframe.** Es una estructura bidimensional con columnas similar a una tabla, que puede almacenar también cualquier tipo de dato. En la Figura 5 puede observarse que, cada columna representa un panda series, por lo tanto, un dataframe está conformado por varios panda series.

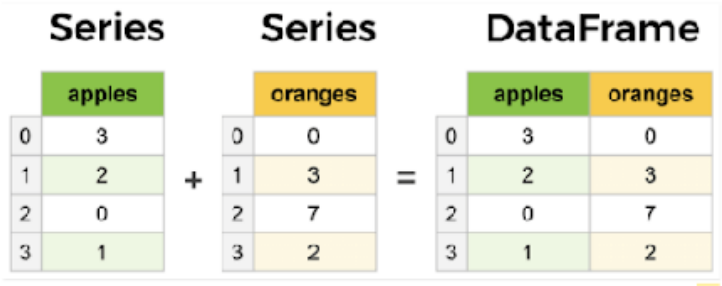


Figura 5. Dataframe.

Los datos por lo general pueden venir de distintas categorías, por eso, Pandas tiene el objetivo de trabajar con el mayor número posible de uniones entre tipos de datos. Pandas ofrece una

facilidad muy grande a la hora de cargar y guardar datos desde distintos tipos de archivos como por ejemplo CSV, JSON, HTML, etc.

En la Figura 6 puede observarse como importar, leer y mostrar los datos mediante el uso de la librería Pandas.

```
import pandas as pd
```

```
[2] df_spotify = pd.read_csv("https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2020/2020-01-21/spotify_songs.csv")
```

```
[3] df_spotify
```

	track_id	track_name	track_artist	track_popularity	track_album_id	track_album_name	track_album_release_date	playlist_name
0	6f807x0ima9a1j3VPbc7VN	I Don't Care (with Justin Bieber) - Loud Luxur...	Ed Sheeran	66	2oCs0DGTsRO98Gh5ZSl2Cx	I Don't Care (with Justin Bieber) [Loud Luxury...	2019-06-14	Pop Remix
1	0r7CVbZTWZgbTCYdfa2P31	Memories - Dillon Francis Remix	Maroon 5	67	63rPSO264uRjW1X5E6cWv6	Memories (Dillon Francis Remix)	2019-12-13	Pop Remix
2	1z1Hg7Vb0AhHDiEmnDE79l	All the Time - Don Diablo Remix	Zara Larsson	70	1HoSmj2eLcsrR0vE9gThr4	All the Time (Don Diablo Remix)	2019-07-05	Pop Remix

Figura 6. Importar, leer y mostrar datos en Pandas.

En un proyecto, también es importante sacar toda la información necesaria para conocer los datos que se tienen, pero la dificultad es que no se puede dar cuenta de esto a simple vista. Por lo tanto, Pandas ofrece diferentes tipos de comandos para poder tener información de los datos con los que se trabajaran. Algunas de las funciones importantes de las que brinda Pandas para poder analizar los datos en profundidad, son:

- **head(n).** Esta función devuelve las primeras n filas del dataframe. Si no se ingresa un valor n, por defecto se mostrarán las primeras 5 filas del dataframe.
- **tail(n).** Esta función devuelve las ultimas n filas del dataframe. Si no se ingresa un valor n, por defecto se mostrarán las ultimas 5 filas del dataframe.

- **describe().** Esta función brinda como resultado la estadística descriptiva de los datos incluyendo la tendencia central, dispersión y distribución de los datos.

Pandas también permite filtrar los datos u obtener columnas de forma muy fácil, rápida e intuitiva. Se pueden hacer exploraciones de datos utilizando distintas funciones como, por ejemplo, las siguientes:

- **df_persona[“Nombre”].** Se mostrarán todos los datos solo de la columna “Nombre”.
- **Iloc.** df_persona.Iloc[1,:]. Se mostrará la fila con índice 1 y todas las columnas. Iloc filtra en base a índices.
- **Loc.** df_persona.Iloc[1,”Nombre”]. Se mostrará la fila con el índice 1 y la columna Nombre. Loc filtra a las filas en base a índices, pero las columnas con sus respectivos nombres.

En la Figura 7 puede observarse, como la librería permite realizar condiciones para filtrar datos que sean iguales, mayores, menores, etc.

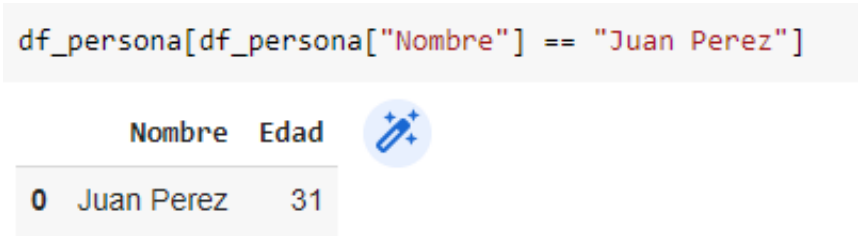


Figura 7. Filtrar datos en Pandas.

Una ventaja que posee es que Matplotlib (librería de gráficos) viene integrada con Pandas por lo que permite de forma cómoda y sencilla graficar directamente desde un Dataframe o Series.

Para poder utilizar esto, es necesario primero indicar que información se quiere mostrar y luego el tipo de grafico que se quiere hacer. Los tipos de gráficos disponibles son:

- **área.** gráficos de áreas.
- **bar.** diagramas de barras verticales.
- **barh.** diagramas de barras horizontales.
- **box.** diagrama de cajas y bigotes.
- **hexbin.** para diagramas hexagonales.
- **hist.** histograma.
- **kde.** gráficos de estimación kernel de la densidad.
- **density.** alias para “kde”.
- **line.** gráficos de líneas.
- **pie.** diagrama de tartas.
- **scatter.** diagrama de dispersión.

En la figura 8 puede observarse un ejemplo en el que se quiere mostrar en un gráfico la edad de las personas del dataframe “df_persona”.

```
df_persona["Edad"].plot(kind="hist")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7feb7633ca90>
```

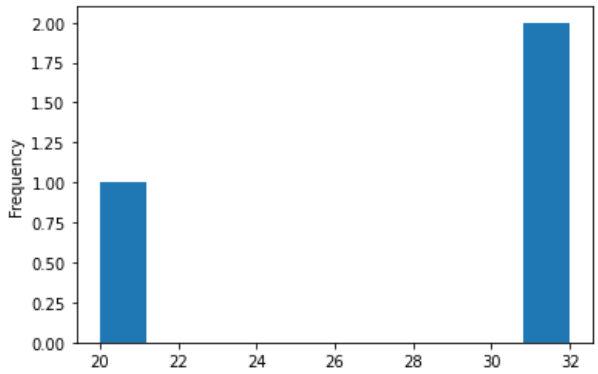


Figura 8. Graficar datos en Pandas.

Pandas es una librería que permite sacar el máximo provecho de los datos que se generan y se tienen que manejar continuamente. Esta librería, hace más fácil todo el ciclo de cualquier dato. Permite realizar operaciones para la gestión y manipulación de cualquier tipo de información sin importar el formato de forma rápida y eficaz.

3.1.12 Librería Matplotlib – Python.

Matplotlib es una librería de Python que permite crear y personalizar visualizaciones estáticas, animadas e interactivas en 2D y 3D. Es una librería muy completa que hacer cualquier tipo de grafico o reporte. Algunos de los gráficos que se pueden crear, son:

- Diagramas de barras.
- Histogramas.
- Boxplots.
- Diagramas de dispersión.
- Diagramas de líneas.
- Diagramas de áreas.
- Diagramas de contorno.
- Mapas de color.
- Etc.

Para poder crear un gráfico en matplotlib, se siguen por lo general los siguientes pasos:

1. Importar la librería

2. Definir la figura en la cual aparecerá el grafico. Esto permite indicar los valores de largo y ancho del eje x e y, pero además la cantidad de subplots dentro del plot principal.
3. Dibujar el grafico. Al existir tanta variedad de gráficos posibles, cada uno posee su propia función.
4. Personalizar el grafico a gusto.
5. Guardar el gráfico. Este paso no es obligatorio para que el grafico funcione.
6. Mostrar el gráfico.

A continuación, se muestra un ejemplo con todos los pasos integrados en el que en la Figura 9 se observa como importar la librería Matplotlib, la Figura 10 el código para graficar los datos y la Figura 11 el gráfico ya finalizado.

```
#importar libreria
import matplotlib.pyplot as plt
```

Figura 9. Importar en Matplotlib.

```
fig, ax = plt.subplots() # se ajusta el plot
ax.hist(df_spotify["acousticness"],color="yellow",ec="black") # se grafica y personaliza el gráfico
plt.savefig("laluca.png") # se guarda el gráfico
plt.show() # se muestra el gráfico
```

Figura 10. Código de Matplotlib.

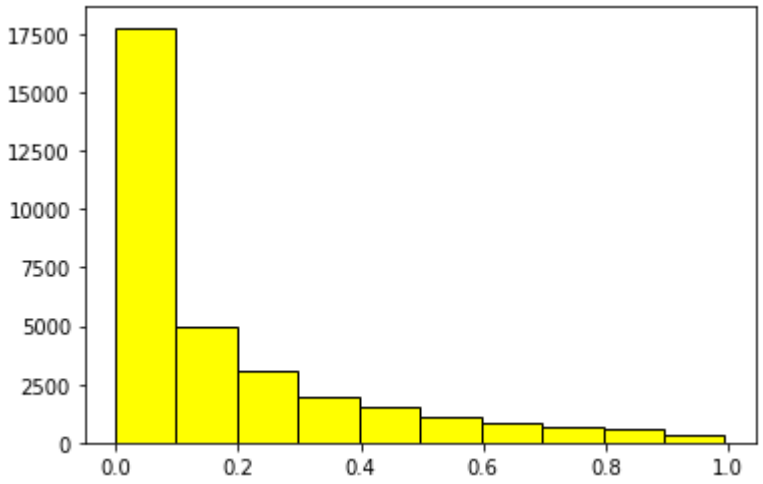


Figura 11. Gráfico en Matplotlib.

3.1.13 Librería Tidyverse – R.

Tidyverse es un conjunto de paquetes de R que están orientados a la ciencia de datos. Esto quiere decir que es de gran ayuda a la hora de importar, transformar, visualizar, modelar y comunicar la información que se utiliza en cualquier proceso de la ciencia de datos.

Las ventajas y desventajas que posee son:

- Una ventaja que tiene es que todos los paquetes comparten un nombre y estructura base. Esto hace que exista una consistencia entre todos esos paquetes para que haya más facilidad de uso. Los paquetes son mantenidos por la comunidad de R y RStudio.
- Otra ventaja es que facilita el análisis y manipulación de datos.
- Dependiendo quien lo use, puede gustarle o no la librería. Por lo tanto, una desventaja es que Tidyverse deja de lado la forma de programar en R más usual y esto puede ser que el cambio sea difícil.

Tidyverse posee 8 paquetes principales:

- **Ggplot2.** Es un paquete de visualización que utiliza la gramática de los gráficos. Con este paquete se pueden crear distintos tipos de gráficos y editarlos a gusto propio.
- **Dplyr.** Es un paquete dedicado a la manipulación de datos que se basa en acciones sobre los datos. Permite mutar o crear variables, seleccionar, filtrar, simarizar, agrupar, acomodar, etc.
- **Tidyr.** Es un paquete que permite transformar los datos para hacerlos más eficientes. O sea, se van a poder transformar filas y columnas de una forma muy eficiente para que los datos cumplan con las condiciones necesarias (cada columna es una variable, cada fila es una observación, cada celda es un valor). Tiene mucho uso e importancia cuando se quieren modelar y graficar los datos.
- **Readr.** Permite la lectura de una forma más amigable y rápida archivos de texto plano como lo puede ser un .CSV
- **Purrr.** Es un paquete que sirve para trabajar con vectores y funciones en un lenguaje consistente. Esta muy relacionado a la programación funcional.
- **Tibble.** Es una transformación del dataframe. Aprovecha lo bueno de los dataframe y mejora algunas opciones que quedaron obsoletas.
- **Stringr.** Es un paquete para trabajar con un análisis de texto y manipulación de strings.
- **Forcats.** Es un paquete especial para trabajar con factores y datos categóricos.

En la Figura 12 puede observarse como está conformado el paquete de Tidyverse.

Figura 12. Paquete Tidyverse.

4 Análisis del problema

4.1 Descripción de la situación actual

La sociedad está pasando por un momento complicado desde hace ya 2 años a casusa del Coronavirus (COVID-19). Cada día hay más personas infectadas por este virus, muertes, problemas económicos y muchas cosas más. Es la primera vez que se registra una pandemia mundial en la cual haya parado tanto al mundo. Cada día también van surgiendo nuevas variantes que generan mayor incertidumbre en la gente ya que al aparecer constantemente nuevas cosas, no se sabrá que podrá pasar el día de mañana.

En base a lo que está pasando, mucha gente necesita de manera eficiente poder analizar, ver y sacar conclusiones en base a preguntas sobre el virus, ya que los datos cada día son más accesibles. Uno de los primeros pasos para poder comenzar con esto es el EDA, hasta el momento no se ha tomado posición sobre cuál de los lenguajes de programación más conocidos y utilizados en este mundo de datos, es más útil o cumple con ciertas métricas imprescindibles para este caso tan importante.

Para esto se ha planeado realizar el EDA en el lenguaje Python y R para, luego de haber establecido las métricas, poder ver cuál es mejor o tiene mayor puntuación que el otro, evaluando mediante una matriz ponderada.

5 Pruebas

5.1 Prueba 1 - EDA hecho en Python

En la Figura 13 puede observarse todo el EDA hecho en Python.

```
# se importan las librerias que se usaran
import pandas as pd
from funpymodeling.exploratory import freq_tbl, profiling_num
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import datetime as dt
```

```
# se crea el dataframe con los datos del archivo .CSV
df_vacunacion = pd.read_csv('covid.csv')
```

```
# se ven los primeros 5 registros
df_vacunacion.head()
```

	sexo	grupo_etario	jurisdiccion_residencia	jurisdiccion_residencia_id	depto_residencia	depto_residencia_id	jurisdiccion_aplicacion	jurisdiccion_aplicacion_id
0	F	12-17	La Rioja	46	Capital	14	La Rioja	46
1	M	50-59	Entre Ríos	30	San Salvador	88	Entre Ríos	30
2	F	30-39	Salta	66	Capital	28	Salta	66
3	M	50-59	Buenos Aires	6	José C. Paz	412	Buenos Aires	6
4	M	40-49	Entre Ríos	30	Federal	35	Entre Ríos	30

```
# se ven los ultimos 5 registros
df_vacunacion.tail()
```

	sexo	grupo_etario	jurisdiccion_residencia	jurisdiccion_residencia_id	depto_residencia	depto_residencia_id	jurisdiccion_aplicacion	jurisdiccion_aplica
1048570	F	60-69	Tucumán	90	Capital	84	Tucumán	
1048571	F	<12	Santa Fe	82	San Cristóbal	91	Santa Fe	
1048572	F	30-39	Jujuy	38	Dr. Manuel Belgrano	21	Jujuy	
1048573	M	70-79	Buenos Aires	6	Olavarría	595	Buenos Aires	
1048574	F	40-49	Mendoza	50	Luján de Cuyo	63	Mendoza	

```python
# se ven los tipos de datos
df_vacunacion.dtypes
```

```
sexo                        object
grupo_etario                object
jurisdiccion_residencia     object
jurisdiccion_residencia_id   int64
depto_residencia            object
depto_residencia_id          int64
jurisdiccion_aplicacion     object
jurisdiccion_aplicacion_id   int64
depto_aplicacion            object
depto_aplicacion_id          int64
fecha_aplicacion            object
vacuna                      object
cod_dosis_generica           int64
nombre_dosis_generica       object
condicion_aplicacion        object
orden_dosis                  int64
lote_vacuna                 object
dtype: object
```

```python
# se ven las filas y columnas
df_vacunacion.shape
```

```
(1048575, 17)
```

```python
# al haber 17 columnas, se eliminaran algunas que se consideren irrelevantes
df_vacunacion.drop(columns=["jurisdiccion_residencia_id","depto_residencia_id",
                            "jurisdiccion_aplicacion_id","depto_aplicacion_id","cod_dosis_generica"], inplace=True)
```

```python
df_vacunacion.shape # se eliminaron 5 columnas
```

```
(1048575, 12)
```

```python
# se renombran algunas columnas
df_vacunacion = df_vacunacion.rename(columns={"grupo_etario":"edad", "jurisdiccion_residencia": "residencia",
                                              "depto_residencia": "localidad",
                                              "jurisdiccion_aplicacion":"residencia_aplicacion",
                                              "depto_aplicacion":"localidad_aplicacion"})
```

```
1 df_vacunacion.dtypes
```

```
sexo                    object
edad                    object
residencia              object
localidad               object
residencia_aplicacion   object
localidad_aplicacion    object
fecha_aplicacion        object
vacuna                  object
nombre_dosis_generica   object
condicion_aplicacion    object
orden_dosis              int64
lote_vacuna             object
dtype: object
```

```
1 # cantidad de registros duplicados
2 df_vacunacion.duplicated().sum()
```

```
422408
```

```
1 # cantidad de nulos
2 df_vacunacion.isnull().sum()
```

```
sexo                    0
edad                    0
residencia              0
localidad               0
residencia_aplicacion   0
localidad_aplicacion    0
fecha_aplicacion        0
vacuna                  0
nombre_dosis_generica   0
condicion_aplicacion    0
orden_dosis             0
lote_vacuna             0
dtype: int64
```

```
1 # analisis univariado var numerica
2 profiling_num(df_vacunacion)
```

	variable	mean	std_dev	variation_coef	p_0.01	p_0.05	p_0.25	p_0.5	p_0.75	p_0.95	p_0.99
0	orden_dosis	1.735165	0.838267	0.483105	1.0	1.0	1.0	2.0	2.0	4.0	4.0

```
1 # analisis univariado var categorica
2 freq_tbl(df_vacunacion)
```

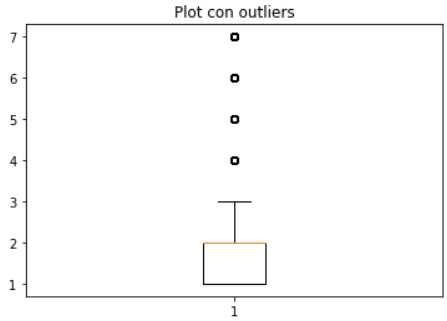
```
   sexo  frequency    percentage  cumulative_perc
0     F     554525  5.288368e-01         0.528837
1     M     491137  4.683852e-01         0.997222
2  S.I.       2912  2.777102e-03         0.999999
3     X          1  9.536752e-07         1.000000

----------------------------------------------------------------

     edad  frequency  percentage  cumulative_perc
0   18-29     195493    0.186437         0.186437
1   30-39     177651    0.169421         0.355858
2   40-49     160823    0.153373         0.509231
3   50-59     122266    0.116602         0.625833
4   60-69     105708    0.100811         0.726644
5     <12      99804    0.095181         0.821825
6   12-17      85405    0.081449         0.903273
7   70-79      69850    0.066614         0.969888
8   80-89      27205    0.025945         0.995832
9   90-99       4293    0.004094         0.999927
```

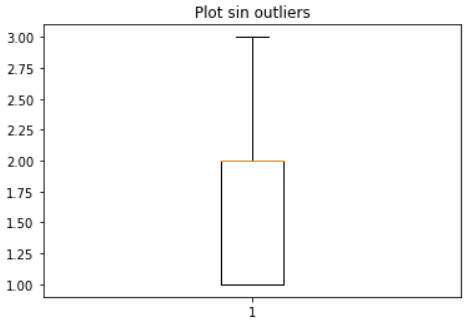
```python
# boxplot con valores que son atipicos
fig1, ax1 = plt.subplots()
ax1.set_title('Plot con outliers')
ax1.boxplot(df_vacunacion["orden_dosis"])
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x25ecfebfc70>,
  <matplotlib.lines.Line2D at 0x25ecfebffd0>],
 'caps': [<matplotlib.lines.Line2D at 0x25ecfed3370>,
  <matplotlib.lines.Line2D at 0x25ecfed36d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x25ecfebf910>],
 'medians': [<matplotlib.lines.Line2D at 0x25ecfed3a30>],
 'fliers': [<matplotlib.lines.Line2D at 0x25ecfed3d90>],
 'means': []}
```

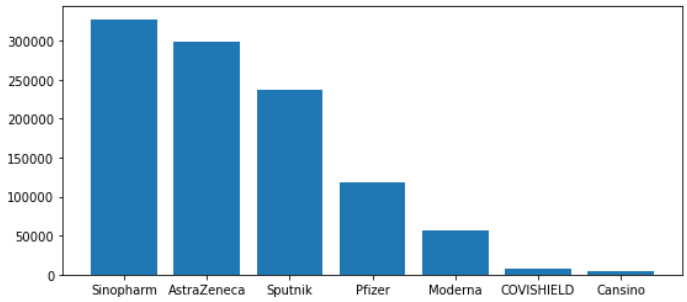



```python
# grafico sin valores atipicos
fig4, ax4 = plt.subplots()
ax4.set_title('Plot sin outliers')
ax4.boxplot(df_vacunacion["orden_dosis"], showfliers=False)
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x25ed0e780a0>,
  <matplotlib.lines.Line2D at 0x25ed0e78400>],
 'caps': [<matplotlib.lines.Line2D at 0x25ed0e78760>,
  <matplotlib.lines.Line2D at 0x25ed0e78ac0>],
 'boxes': [<matplotlib.lines.Line2D at 0x25ed0e6bd00>],
 'medians': [<matplotlib.lines.Line2D at 0x25ed0e78e20>],
 'fliers': [],
 'means': []}
```



```python
# grafico de barras de las vacunas aplicadas
fig, ax = plt.subplots(figsize=(9,4))
ax.bar(df_vacunacion["vacuna"].value_counts().index,df_vacunacion["vacuna"].value_counts().values)
plt.show()
```

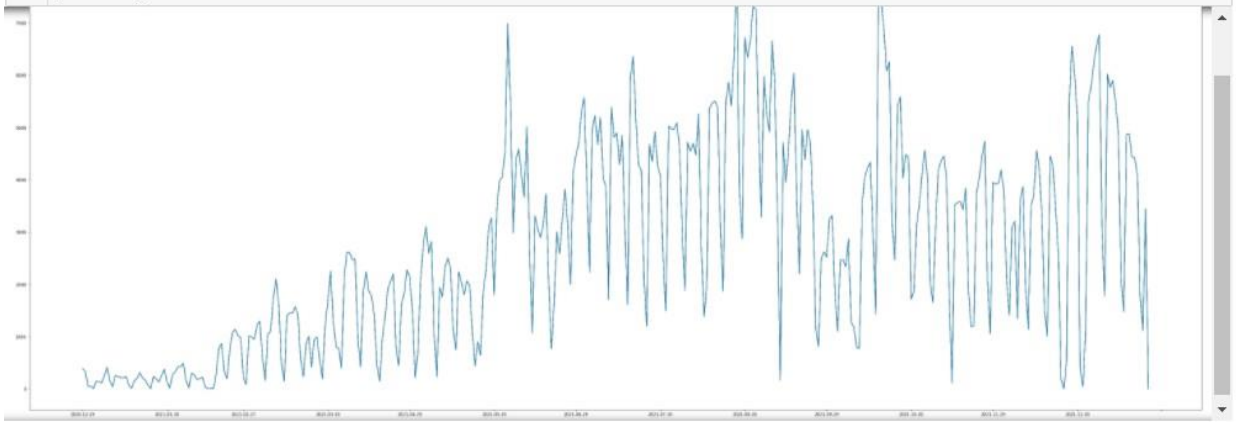



```python
# linea en el tiempo
fig, ax = plt.subplots(figsize=(50,20))

x = df_vacunacion["fecha_aplicacion"].value_counts().index
x =x.sort_values()
y = df_vacunacion.groupby("fecha_aplicacion").count().values

half_year_locator = mdates.MonthLocator(interval=1)
ax.xaxis.set_major_locator(half_year_locator)

ax.plot(x,y)
plt.show()
```



```
# diagrama de torta de sexos
fig, ax = plt.subplots(figsize=(10,10))
ax.pie(df_vacunacion["sexo"].value_counts().values,autopct='%1.1f%%')
plt.legend(df_vacunacion["sexo"].value_counts().index,loc="right")
plt.show()
```

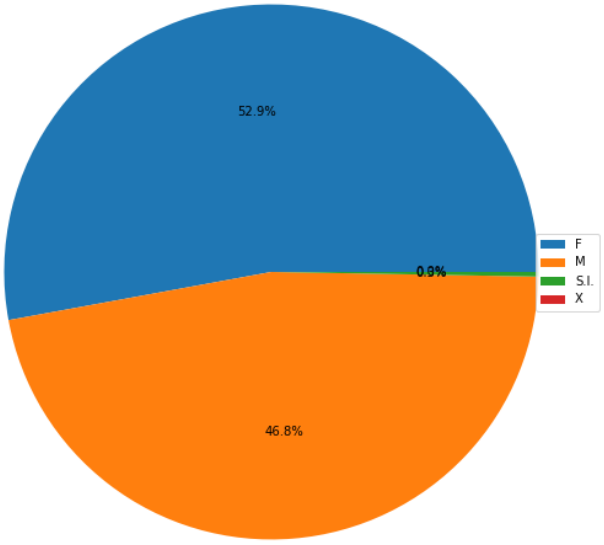


Figura 13. EDA en Python.

5.2 Prueba 2 - EDA hecho en R

En la Figura 14 puede observarse todo el EDA hecho en R.

```
# se importan las librerias que se usaran
library(tidyverse)
library(funModeling)
library(ggplot2)
```

Hide

```
# se crea el dataframe con los datos del archivo .CSV
df_vacunacion = read_delim("covid.csv",delim=",")
```

```
indexing covid.csv [=-------------------------------------------------------------------------] ?, eta:  0s
indexing covid.csv [========================-------------------------------------] 257.72MB/s, eta:  0s
indexing covid.csv [=========================-------------------------------------] 265.42MB/s, eta:  0s
indexing covid.csv [============================-----------------------------------] 252.75MB/s, eta:  0s
indexing covid.csv [==============================---------------------------------] 254.21MB/s, eta:  0s
indexing covid.csv [================================-------------------------------] 241.32MB/s, eta:  0s
indexing covid.csv [====================================-----------------------------] 249.75MB/s, eta:  0s
indexing covid.csv [======================================---------------------------] 237.78MB/s, eta:  0s
indexing covid.csv [==========================================-----------------------] 247.52MB/s, eta:  0s
indexing covid.csv [============================================---------------------] 244.35MB/s, eta:  0s
indexing covid.csv [==============================================-------------------] 237.82MB/s, eta:  0s
indexing covid.csv [================================================-----------------] 239.83MB/s, eta:  0s
indexing covid.csv [==================================================---------------] 238.82MB/s, eta:  0s
indexing covid.csv [====================================================-------------] 240.46MB/s, eta:  0s
indexing covid.csv [======================================================-----------] 237.79MB/s, eta:  0s
indexing covid.csv [========================================================----------] 228.69MB/s, eta:  0s
indexing covid.csv [==========================================================--------] 231.04MB/s, eta:  0s
indexing covid.csv [============================================================------] 232.84MB/s, eta:  0s
indexing covid.csv [==============================================================----] 233.10MB/s, eta:  0s
indexing covid.csv [================================================================--] 232.11MB/s, eta:  0s
indexing covid.csv [==================================================================] 228.64MB/s, eta:  0s


indexing covid.csv [==================================================================] 226.22MB/s, eta:  0s
```

```
⍰[1mRows: ⍰[22m⍰[34m1048575⍰[39m ⍰[1mColumns: ⍰[22m⍰[34m17⍰[39m⍰[36m--⍰[39m ⍰[1mColumn specification⍰[22m ⍰[36m------------
------------------------------------------------------------------------⍰[39m
⍰[1mDelimiter:⍰[22m ","
⍰[31mchr⍰[39m  (14): sexo, grupo_etario, jurisdiccion_residencia, jurisdiccion_residencia_id, depto_residencia,...
⍰[32mdbl⍰[39m   (2): cod_dosis_generica, orden_dosis
⍰[34mdate⍰[39m  (1): fecha_aplicacion
⍰[36mi⍰[39m Use `spec()` to retrieve the full column specification for this data.
⍰[36mi⍰[39m Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# se ven los primeros 6 registros
head(df_vacunacion)
```

sexo <chr>	grupo_etario <chr>	jurisdiccion_residencia <chr>	jurisdiccion_residencia_id <chr>	depto_residencia <chr>
F	12-17	La Rioja	46	Capital
M	50-59	Entre Ríos	30	San Salvador
F	30-39	Salta	66	Capital
M	50-59	Buenos Aires	06	José C. Paz
M	40-49	Entre Ríos	30	Federal
M	30-39	Córdoba	14	Capital

6 rows | 1-5 of 17 columns

Hide

```
# se ven los ultimos 6 registros
tail(df_vacunacion)
```

se... <chr>	grupo_etario <chr>	jurisdiccion_residencia <chr>	jurisdiccion_residencia_id <chr>	depto_residencia <chr>
M	30-39	Buenos Aires	06	Tres de Febrero
F	60-69	Tucumán	90	Capital
F	<12	Santa Fe	82	San Cristóbal
F	30-39	Jujuy	38	Dr. Manuel Belgrano
M	70-79	Buenos Aires	06	Olavarría
F	40-49	Mendoza	50	Luján de Cuyo

6 rows | 1-5 of 17 columns

```
dim(df_vacunacion)
```

```
[1] 1048575      17
```

Hide

```
# al haber 17 columnas, se eliminaran algunas que se consideren irrelevantes
df_vacunacion = subset(df_vacunacion,select=-c(jurisdiccion_residencia_id,depto_residencia_id,jurisdiccion_aplicacion_id,depto_aplicacion_id,cod_dosis_generica))
```

Hide

```
dim(df_vacunacion) # se eliminaron 5 columnas
```

```
[1] 1048575      12
```

```
# cambiar nombre de columnas
df_vacunacion = rename(df_vacunacion,edad=grupo_etario,residencia=jurisdiccion_residencia,localidad=depto_residencia,residen
cia_aplicacion=jurisdiccion_aplicacion,localidad_aplicacion=depto_aplicacion)
```

Hide

```
sapply(df_vacunacion,class)
```

```
               sexo                edad           residencia
        "character"         "character"         "character"
          localidad residencia_aplicacion  localidad_aplicacion
        "character"         "character"         "character"
    fecha_aplicacion              vacuna nombre_dosis_generica
             "Date"         "character"         "character"
 condicion_aplicacion         orden_dosis         lote_vacuna
        "character"           "numeric"         "character"
```

```
# cantidad de registros duplicados
sum(duplicated(df_vacunacion))
```

```
[1] 422408
```

Hide

```
# valores nulos
df_vacunacion[is.na(df_vacunacion)]
```

```
character(0)
```

Hide

```
profiling_num(df_vacunacion)
```

variable <chr>	mean <dbl>	std_dev <dbl>	variation_coef <dbl>	p_01 <dbl>	p_05 <dbl>	p_25 <dbl>	p_50 <dbl>	p_75 <dbl>	p_95 <dbl>
orden_dosis	1.735165	0.8382673	0.4831052	1	1	1	2	2	4

1 row | 1-10 of 16 columns

```
# analisis univariado var categorica
freq(df_vacunacion)
```

```
`guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> = "none")` instead.
```

sexo <chr>	frequency <int>	percentage <dbl>	cumulative_perc <dbl>
F	554525	52.88	52.88
M	491137	46.84	99.72
S.I.	2912	0.28	100.00
X	1	0.00	100.00

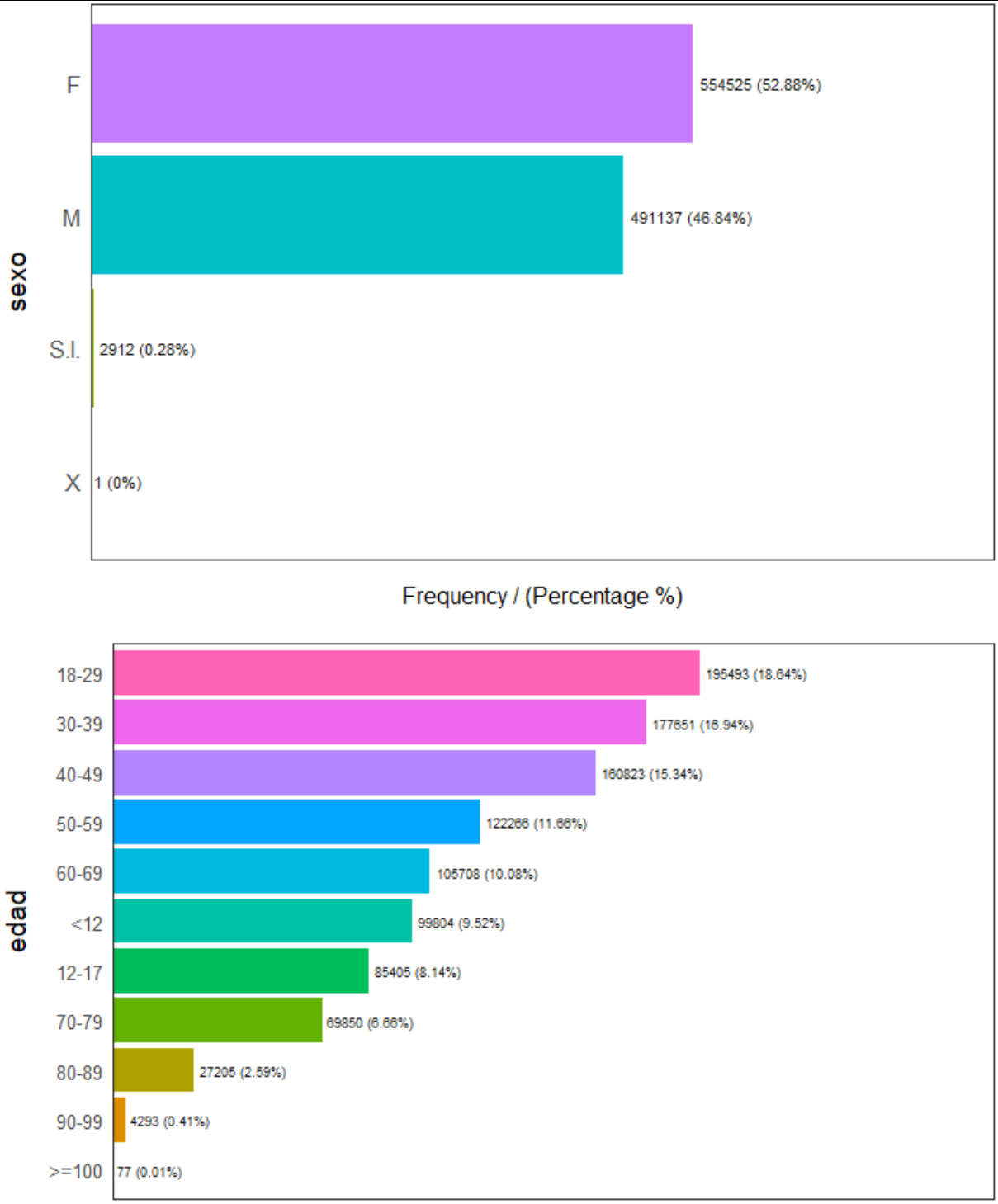
4 rows

```
NA
```

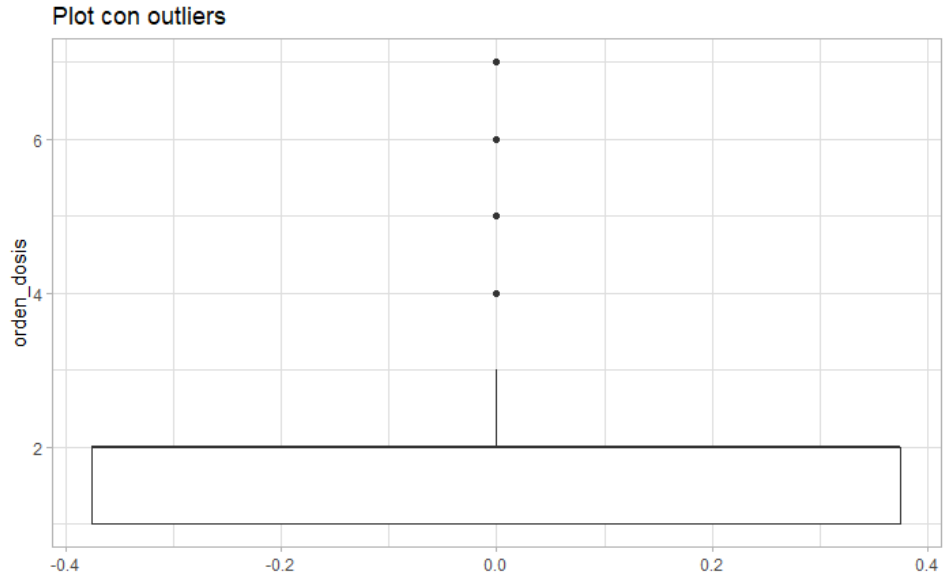
edad <chr>	frequency <int>	percentage <dbl>	cumulative_perc <dbl>
18-29	195493	18.64	18.64
30-39	177651	16.94	35.58
40-49	160823	15.34	50.92
50-59	122266	11.66	62.58
60-69	105708	10.08	72.66
<12	99804	9.52	82.18
12-17	85405	8.14	90.32
70-79	69850	6.66	96.98
80-89	27205	2.59	99.57
90-99	4293	0.41	99.98

1-10 of 11 rows Previous 1 2 Next

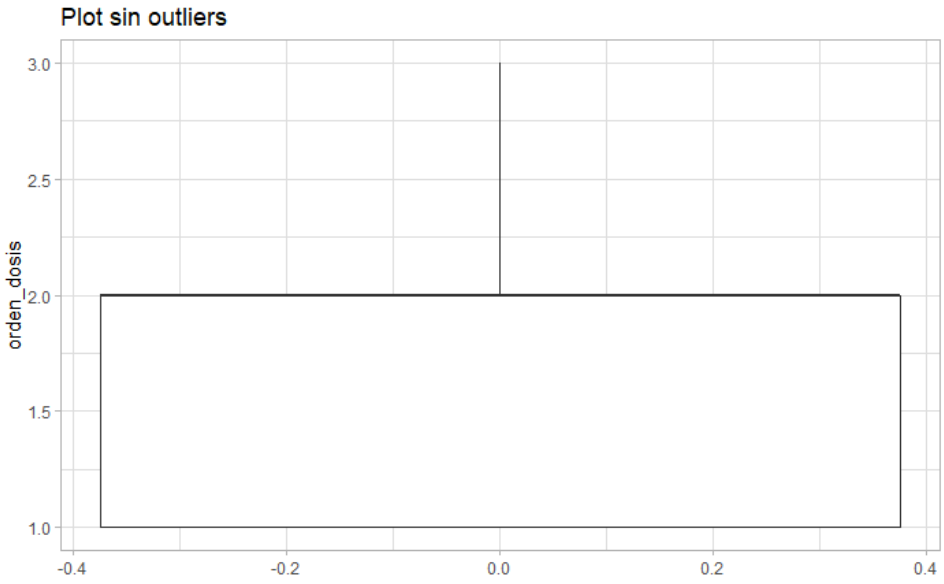
```
NA
```


sexo
F
M
S.I.
X
554525 (52.88%)
491137 (46.84%)
2912 (0.28%)
1 (0%)
Frequency / (Percentage %)
edad
18-29
30-39
40-49
50-59
60-69
<12
12-17
70-79
80-89
90-99
>=100
195493 (18.64%)
177651 (16.94%)
160823 (15.34%)
122266 (11.66%)
105708 (10.08%)
99804 (9.52%)
85405 (8.14%)
69850 (6.66%)
27205 (2.59%)
4293 (0.41%)
77 (0.01%)

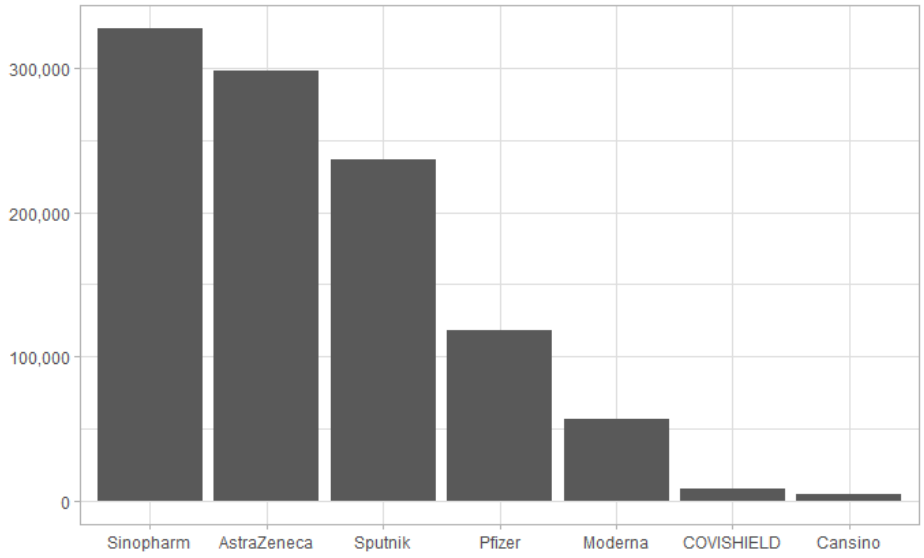
```
ggplot(data=df_vacunacion,aes(y=orden_dosis)) +
  geom_boxplot() +
  theme_light() +
  ggtitle('Plot con outliers')
```



```
ggplot(data=df_vacunacion,aes(y=orden_dosis)) +
  geom_boxplot(outlier.shape = NA) +
  theme_light() +
  scale_y_continuous(limits = quantile(df_vacunacion$orden_dosis, c(0.1, 0.9)))+
  ggtitle('Plot sin outliers')
```

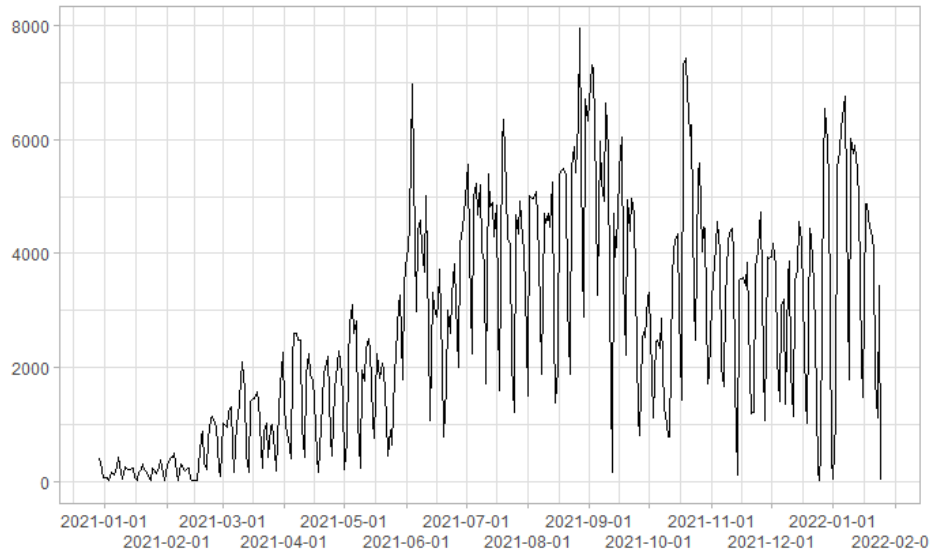


```
require(scales)
ggplot(data=df_vacunacion,aes(x=forcats::fct_infreq(vacuna)))+
  geom_bar()+
  scale_y_continuous(labels = comma)+
  theme_light() +
  xlab("")+
  ylab("")
```



```
datos = df_vacunacion %>% count(fecha_aplicacion)
ggplot(data=datos,aes(x=datos$fecha_aplicacion,y=datos$n))+
  geom_line()+
  scale_x_date(date_breaks='1 month',guide = guide_axis(n.dodge = 2))+

  theme_light() +
  xlab("")+
  ylab("")
```



```r
# nuevo df
porcentaje <- df_vacunacion %>%
                group_by(sexo) %>%
                count() %>%
                ungroup() %>%
                mutate(percentage=`n`/sum(`n`) * 100)
#grafico
ggplot(porcentaje, aes(x=1, y=percentage, fill=sexo)) +
        geom_bar(stat="identity") +
        geom_text(aes(label = paste0(round(percentage,1),"%")),
                  position = position_stack(vjust = 0.5)) +
        coord_polar(theta = "y") +
        theme_void()
```

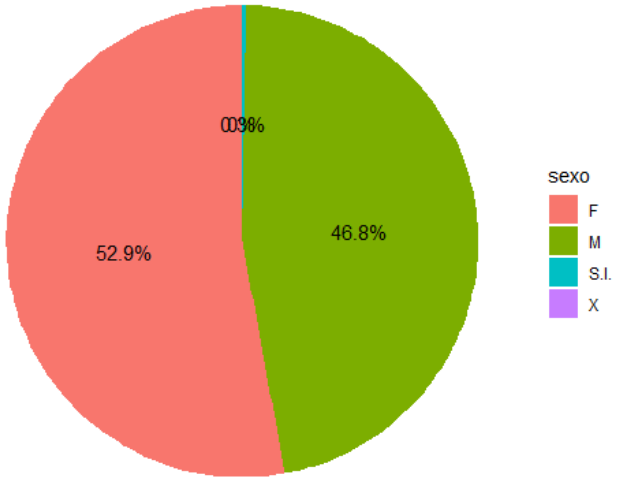


Figura 14. EDA en R

6 Implantación o desarrollo del prototipo

6.1 Métricas

Las métricas establecidas para la realización de la comparación entre R y Python son:

- Tiempo de ejecución
- Facilidad de aprendizaje
- Mejor manipulación de datos
- Mejor gráfico de datos
- Mejor en estadísticas de datos
- Mejor lectura de datos

6.1.1 Tiempo de ejecución.

El tiempo de ejecución es una métrica clave a la hora de trabajar sobre un proyecto de data science. Esto permite saber cuánto tiempo lleva la ejecución de las diferentes partes de un código. Obviamente, dependiendo de para que se lo quiera usar, el tiempo de ejecución es mayor o menor, por ejemplo, en un proyecto sobre redes neuronales es mayor que uno de análisis exploratorio de datos o trabajar sobre un dataset más grande llevara más tiempo de ejecución que uno más chico. Igualmente, no se debe menospreciar el proyecto ya que el tiempo es una variable para tener en cuenta en cualquier trabajo que se realice.

Para poder realizar las pruebas, se utiliza la librería time (Python) y la función proc.time (R).

A continuación, se muestran dos Tablas con los resultados obtenidos en Python y R:

Tabla 1

Tiempo de ejecución en Python

Parte del código	Tiempo aproximado
Importar librerías	4.5 segundos
Crear dataframe	2.2 segundos
Primeros registros	0.0002 segundos
Últimos registros	0.0002 segundos
Tipos de datos	0.0004 segundos
Filas y columnas	0.0001 segundos
Eliminar columnas	0.1 segundos
Renombrar columnas	0.06 segundos
Cantidad de registros duplicados	0.8 segundos
Cantidad de registros nulos	0.3 segundos
Análisis univariado de variable numérica	0.04 segundos
Análisis univariado de variable categórica	2.8 segundos
Boxplot con outliers	0.06 segundos
Boxplot sin outliers	0.06 segundos
Gráfico de barras	0.3 segundos
Gráfico en el tiempo	1.2 segundos
Gráfico de torta	0.3 segundos

Tabla 1. La tabla muestra los resultados del tiempo de ejecución para cada sección del EDA hecho en Python. Autoría propia.

Tabla 2

Tiempo de ejecución en R

Parte del código	Tiempo aproximado
Importar librerías	5.3 segundos
Crear dataframe	3.4 segundos
Primeros registros	0.07 segundos
Últimos registros	0.08 segundos
Tipos de datos	0.1 segundos
Filas y columnas	0.07 segundos
Eliminar columnas	0.5 segundos
Renombrar columnas	0.2 segundos
Cantidad de registros duplicados	15.0 segundos
Cantidad de registros nulos	0.3 segundos
Análisis univariado de variable numérica	1.7 segundos
Análisis univariado de variable categórica	3.9 segundos
Boxplot con outliers	3.2 segundos
Boxplot sin outliers	1.0 segundos
Gráfico de barras	1.0 segundos

Gráfico en el tiempo	0.3 segundos
Gráfico de torta	0.3 segundos

Tabla 2. La tabla muestra los resultados del tiempo de ejecución para cada sección del EDA hecho en R. Autoría propia.

6.1.2 Facilidad de aprendizaje.

Aunque la facilidad a la hora de aprender o utilizar un lenguaje de programación depende de cada persona, el ambiente del cual se venga influye en esta facilitación.

Si es una persona que se dedica constantemente al mundo de la estadística y matemática o del aprendizaje, seguramente R es la opción correcta ya que estos están acostumbrados a trabajar con datos y análisis. R posee una facilidad mucho mayor a la hora de realizar análisis.

Si es una persona que viene del mundo de la programación o quiere aprender a programar, seguramente Python es la opción más fácil ya que conocen sobre el uso de programación orientada a objetos, clases, métodos, etc. Python es un lenguaje de propósito general (orientado a diversos temas) por lo que influye en esta facilidad de aprendizaje.

En mi caso en particular, al provenir del mundo de la informática y programación, el lenguaje de programación que me resultó más fácil de aprender fue Python, aunque R al ya saber sobre programación, me pareció bastante sencillo de entender.

Igualmente, como conclusión, se puede decir que ambos lenguajes son fáciles. Al aprender bien uno, el otro podrá resultar igual o hasta incluso más sencillo que el otro porque la práctica y conocimientos ya están adquiridos.

6.1.3 Mejor manipulación de datos.

Para la manipulación de los datos, gana R ya que con su librería Dplyr de Tidyverse, puede manipular de mejor manera los datos que Python con su librería Pandas. Esto igualmente no quiere decir que las dos librerías no cumplan con su objetivo principal.

Una ventaja que tiene Tidyverse es que la librería es más coherente y grande (funcionalidades de la librería) que Pandas. Además, Pandas fue creado bajo la influencia e inspiración de R. También posee una sintaxis más fácil y rápida para mejorar la implementación.

A la hora de realizar filtraciones de datos, Dplyr mediante la utilización del operador %>% permite encadenar operaciones y hace que los desgloses de datos sean fáciles de escribir y leer, cosa que Pandas no posee. Otra ventaja es que es mucho más fácil aplicar diversos filtros con Dplyr que con Pandas ya que requiere escribir más y produce un código que es más difícil de leer.

Luego sobre la relación que tienen ambos lenguajes con el resumen estadístico, estos son casi idénticos. La única diferencia que se puede encontrar es que con Dplyr, la sintaxis esta más ordenada, pero esto se debe a la mejor filtración que posee ante Pandas. Mencionando otras librerías, se puede decir, por ejemplo, que Funpymodeling y Funmodeling poseen prácticamente la misma sintaxis.

Por último, la creación de columnas derivadas es mucho más sencillo en R ya que a sintaxis de Dplyr es mucho más limpia y fácil de leer.

6.1.4 Mejor gráfico de datos.

Para generar gráficos, nuevamente R se lleva la victoria. Las librerías más comunes y conocidas dentro de R y Python son Ggplot2 y Matplotlib. Con la utilización de Ggplot2, que viene dentro

de Tidyverse, R puede hacer gráficos muy buenos y de una forma muy fácil ya que se necesitan pocas líneas de código.

Estas dos librerías mediante código pueden modificar algunas partes de los gráficos, pero igualmente se sigue notando la superioridad visual que posee Ggplot2 en por ejemplo los contornos de las líneas, labels, leyendas, etc.

Existen también otras librerías que permiten generar gráficos interactivos y en este caso, Python y R pueden compartir la misma librería que se llama Plotly.

Además, hay que recordar que R es un lenguaje de programación dedicado a la estadística, por lo que esto lleva a que los gráficos tengan que ser muy buenos, prolijos y fáciles de realizar.

6.1.5 Mejor en estadísticas de datos.

Los dos lenguajes son buenos para poder trabajar con estadísticas y análisis, pero es conveniente usar R ya que es un lenguaje que está orientado justamente a este tema y a la vez posee una gran cantidad de librerías y las fórmulas generando una gran ventaja.

Python en las etapas iniciales, sus librerías generaban problemas, pero con las versiones actuales, esto mejoró. También hay que tener en cuenta que, al ser un lenguaje de programación de propósito general, está en constante crecimiento en el área de datos y estadística.

6.1.6 Mejor lectura de datos.

La lectura de los datos es el primer paso para poder comenzar a trabajar. Python y R comparten la misma facilidad para recolectar los datos de los archivos y entornos más comunes como por ejemplo un .CSV.

R posee muchas librerías de recolección de datos como por ejemplo para realizar conexiones a bases de datos, realizar web scrapping, conectarse a Apis, etc. pero igualmente Python lo supera ya que tiene algunas librerías que R no porque es un lenguaje de programación de propósito general haciendo que tenga más disponibilidad de archivos y tipos de datos.

Un ejemplo de esto es que, si se quiere obtener datos mediante web scrapping, es mejor Python ya que es más fácil en pequeños y grandes casos mientras que R lo es solo en pequeños casos.

Por lo tanto, si se necesita conectar y extraer información Python es la mejor opción para hacerlo.

6.2 Pruebas y resultados

Este trabajo final de carrera, para hacer la matriz ponderada, utiliza el método del criterio analítico completo en el cual se realizan diferentes tablas. La primera parte incluye la ponderación con las métricas, la segunda parte la ponderación de los dos lenguajes de programación con las métricas una por una y una tabla sola como resultado final, y la tercera parte incluye los valores de la multiplicación de las tablas finales de la primera y segunda parte indicando el lenguaje ganador.

Para la calificación, se usan los siguientes valores:

- **10.** La fila es mucho más importante que la columna. Su valor recíproco es 0.1.
- **5.** La fila es más importante que la columna. Su valor recíproco es 0.2.
- **1.** La fila y la columna son igual de importantes.
- **0.2.** La fila es menos importante que la columna. Su valor recíproco es 5.
- **0.1.** La fila es mucho menos importante que la columna. Su valor recíproco es 10.

Método del criterio analítico completo

En las siguientes Tablas puede observarse la matriz ponderada con sus pasos y resultados aplicando el método del criterio analítico completo.

Tabla 3

Ponderación de las métricas

	Tiempo de ejecución	Facilidad de aprendizaje	Mejor manipulación de datos	Mejor gráfico de datos	Mejor en estadísticas de datos	Mejor lectura de datos	Total	Peso ponderado definido
Tiempo de ejecución	-	0.2	0.1	0.1	0.1	0.2	0.7	0,007
Facilidad de aprendizaje	5	-	0.2	0.2	0.2	5	10.6	0,108
Mejor manipulación de datos	10	5	-	1	1	10	27	0,276
Mejor gráfico de datos	10	5	1	-	1	10	27	0,276
Mejor en estadísticas de datos	10	5	1	1	-	10	27	0,276
Mejor lectura de datos	5	0.2	0.1	0.1	0.1	-	5.5	0,056
Total	-	-	-	-	-	-	97,8	1

Tabla 3. La tabla muestra los resultados de la ponderación de las métricas. Autoría propia.

Tabla 4

Ponderación de los lenguajes con tiempo de ejecución

Tiempo de ejecución	Python	R	Total	Peso ponderado definido
Python	-	5	5	0.961
R	0.2	-	0.2	0.038
Total	-	-	5.2	1

Tabla 4. La tabla muestra la ponderación de los lenguajes con tiempo de ejecución. Autoría propia.

Tabla 5

Ponderación de los lenguajes con facilidad de aprendizaje

Facilidad de aprendizaje	Python	R	Total	Peso ponderado definido
Python	-	1	1	0.500
R	1	-	1	0.500
Total	-	-	2	1

Tabla 5. La tabla muestra la ponderación de los lenguajes con facilidad de aprendizaje.

Autoría propia.

Tabla 6

Ponderación de los lenguajes con mejor manipulación de datos

Mejor manipulación de datos	Python	R	Total	Peso ponderado definido
Python	-	0.2	0.2	0.038
R	5	-	5	0.961
Total	-	-	5.2	1

Tabla 6. La tabla muestra la ponderación de los lenguajes con mejor manipulación de datos.

Autoría propia.

Tabla 7

Ponderación de los lenguajes con mejor gráfico de datos

Mejor gráfico de datos	Python	R	Total	Peso ponderado definido
Python	-	0.2	0.2	0.038
R	5	-	5	0.961
Total	-	-	5.2	1

Tabla 7. La tabla muestra la ponderación de los lenguajes con mejor gráfico de datos. Autoría propia.

Tabla 8

Ponderación de los lenguajes con mejor estadísticas de datos

Mejor en estadísticas de datos	Python	R	Total	Peso ponderado definido
Python	-	0.2	0.2	0.038
R	5	-	5	0.961
Total	-	-	5.2	1

Tabla 8. La tabla muestra la ponderación de los lenguajes con mejor en estadísticas de datos. Autoría propia.

Tabla 9

Ponderación de los lenguajes con mejor lectura de datos

Mejor lectura de datos	Python	R	Total	Peso ponderado definido
Python	-	5	5	0.961
R	0.2	-	0.2	0.038
Total	-	-	5.2	1

Tabla 9. La tabla muestra la ponderación de los lenguajes con mejor lectura de datos. Autoría propia.

Tabla 10

Resumen de la ponderación de los lenguajes con las métricas

-	Tiempo de ejecución	Facilidad de aprendizaje	Mejor manipulación de datos	Mejor gráfico de datos	Mejor en estadísticas de datos	Mejor lectura de datos
Python	0.961	0.500	0.038	0.038	0.038	0.961
R	0.038	0.500	0.961	0.961	0.961	0.038

Tabla 10. La tabla muestra un resumen de la ponderación de los lenguajes con todas las métricas. Autoría propia.

Tabla 11

Resultado final

-	Tiempo de ejecución	Facilidad de aprendizaje	Mejor manipulación de datos	Mejor gráfico de datos	Mejor en estadísticas de datos	Mejor lectura de datos	Total
Python	0.006	0.054	0.104	0.104	0.104	0.002	0.374
R	0.0002	0.054	0.265	0.265	0.265	0.053	0.902

Tabla 11. La tabla muestra el resultado final de la matriz ponderada. Autoría propia.

7 Conclusiones

7.1 Sobre estado del arte

El objetivo de la tesina era determinar entre R y Python cuál de los dos lenguajes aplica mejor a la hora de realizar un análisis exploratorio de datos sobre la aplicación de las vacunas de COVID-19 en Argentina.

Mediante la utilización de la matriz ponderada, el lenguaje de programación R arroja como resultado final 0.902, mientras que el lenguaje de programación Python 0.374.

Esto así demuestra que R, por una amplia diferencia, es el lenguaje más conveniente para realizar un análisis exploratorio de datos sobre las vacunas de COVID-19 aplicadas en Argentina.

7.2 Sobre el proceso de aprendizaje

Presentar esta tesina significa para mí, finalizar mis estudios en la carrera de ingeniería en informática. Este trabajo se realizó en base a conocimientos que adquirí durante la carrera, de forma autodidacta durante el desarrollo del trabajo y mediante la constante aplicación de los conceptos a nivel laboral.

Este proyecto se finalizó de forma exitosa gracias a la constancia, esfuerzo y ganas de aprender, ya que es un tema que a mi particularmente me gusta mucho. Por lo tanto, esto para mí, es un gran paso dado en mi vida.

7.3 Líneas futuras de investigación

Mediante este trabajo, hay muchas posibilidades disponibles sobre futuras investigaciones. Algunas de las que se pueden mencionar, son:

- El análisis de los datos para ver que se podría haber mejorado a la hora de realizar las vacunaciones en Argentina contra el COVID-19.
- La comparación de R y Python incluyendo aún más métricas.
- Utilización de otra herramienta que pueda ser más precisa para la medición de las métricas.
- Realizar un análisis predictivo de cómo serán las vacunaciones en una determinada fecha y luego comparar los resultados, con los datos reales.

8 Bibliografía

Anónimo. (25 de agosto de 2020). *Análisis exploratorio de datos*. Anónimo: IBM. Recuperado de https://www.ibm.com/ar-es/cloud/learn/exploratory-data-analysis

Anónimo. (marzo de 2020). *Coronavirus.* Anónimo: Bupa. Recuperado de https://www.bupasalud.com/salud/coronavirus

Anónimo. (21 de octubre de 2020). *La cuarentena.* España: Ministerio de sanidad. Recuperado de https://www.sanidad.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/documentos/Que_es_la_Cuarentena.pdf

Anónimo. (21 de septiembre de 2021). *¿Cuáles vacunas estamos aplicando en el país?.* Argentina: Argentina.gob.ar. Recuperado de https://www.argentina.gob.ar/coronavirus/vacuna/cuales

Anónimo. (2020). *Python: qué es, para qué sirve y cómo se programa.* Anónimo: Aula21. Recuperado de https://www.cursosaula21.com/que-es-python/

Anónimo. (2021). Qué es Python y por qué empezar a programar con este lenguaje [Mensaje en un blog]. Recuperado de https://platzi.com/blog/python/?gclid=CjwKCAiAxJSPBhAoEiwAeO_fP1IdGyY47gFigzSWtT2DHGvG5cmJvR8i1jLku1_Jjb_knbwb-KmMuxoCdjUQAvD_BwE&gclsrc=aw.ds

Anónimo. (abril de 2022). *PYPL PopularitY of Programming Language.* Anónimo: Pypl. Recuperado de https://pypl.github.io/PYPL.html

Alejandra Milena Machado. (22 de diciembre de 2020). *Por qué aprender Phyton y cuáles son sus ventajas.* Anónimo: Pragma. Recuperado de https://www.pragma.com.co/academia/lecciones/descubre-por-que-aprender-phyton-y-cuales-son-sus-ventajas

Joan Gasull Jolis. (12 de mayo de 2018). *Qué es R y diferencias con otros lenguajes de programación.* Anónimo: Linkedin. Recuperado de https://www.linkedin.com/learning/r-para-big-data-esencial/que-es-r-y-diferencias-con-otros-lenguajes-de-programacion?autoAdvance=true&autoSkip=false&autoplay=true&resume=false

Juan Bosco Mendoza Vega. (2018). *R para principiantes.* Recuperado de https://bookdown.org/jboscomendoza/r-principiantes4/

Anónimo. (18 de octubre de 2021). ¿Qué es R? [Mensaje en un blog]. Recuperado de https://datademia.es/blog/que-es-r

Esmeralda León. (2 de marzo de 2021). *Te contamos qué es el Lenguaje R.* Anónimo: Baoss. Recuperado de https://www.baoss.es/te-contamos-que-es-el-lenguaje-r/

Ignacio Morato. (Anónimo). *Tutorial Anaconda: Qué es, cómo instalarlo y cómo se usa.* Anónimo: Ikkaro. Recuperado de https://www.ikkaro.com/anaconda/

Eduardo Cabrera Granado y Elena Díaz García. (Anónimo). *Manual de uso de jupyter notebook para aplicaciones docentes.* Recuperado de https://eprints.ucm.es/id/eprint/48304/1/ManualJupyter.pdf

Anónimo. (28 de febrero de 2019). *Jupyter Notebook: documentos web para análisis de datos, código en vivo y mucho más.* Anónimo: Ionos. Recuperado de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/jupyter-notebook/

Fernán García de Zúñiga. (02 de octubre de 2020). RStudio, IDE para programar con R. Instalación y primeros pasos [Mensaje en un blog]. Recuperado de https://www.arsys.es/blog/rstudio

Rafael González Gouveia. (22 de marzo de 2019). *¿Qué es R y RStudio?.* Anónimo: Rgg. Recuperado de https://gonzalezgouveia.com/que-es-r-y-rstudio/#Que_es_R_y_RStudio

Fernán García de Zúñiga. (13 de noviembre de 2018). ¿Qué son los datasets y los dataframes en el Big Data? [Mensaje en un blog]. Recuperado de https://www.deustoformacion.com/blog/programacion-tic/que-son-datasets-dataframes-big-data

Betancourt, D. F. (24 de noviembre de 2018). *Cómo hacer una matriz de priorización.* Anónimo: Ingenioempresa. Recuperado de https://www.ingenioempresa.com/matriz-de-priorizacion/

José Luis Chacón. (22 de marzo de 2021). Introducción a Pandas, la librería de Python para trabajar con datos [Mensaje en un blog]. Recuperado de https://profile.es/blog/pandas-python/

Alfredo Sánchez Alberca. (2020). *Manual de Python.* Recuperado de https://aprendeconalf.es/docencia/python/manual/

Rafael González Gouveia. (21 de febrero de 2020). *¿Qué es tidyverse? 8 paquetes para ciencia de datos.* Anónimo: Rgg. Recuperado de https://gonzalezgouveia.com/que-es-tidyverse-8-paquetes-para-ciencia-de-datos/#Otros_paquetes

A. Bell, A. Prescott, H. Lace. (07 de marzo de 2020). Coronavirus: Cuáles fueron las consecuencias económicas globales de otras pandemias en la historia. *BBC.* Recuperado de https://www.bbc.com/mundo/noticias-51750414

Anónimo. (08 de junio de 2020). *La COVID-19 (coronavirus) hunde a la economía mundial en la peor recesión desde la Segunda Guerra Mundial*. Washington, EU: Banco mundial. Recuperado de https://www.bancomundial.org/es/news/press-release/2020/06/08/covid-19-to-plunge-global-economy-into-worst-recession-since-world-war-ii

Anónimo. (06 de agosto de 2020). *Los efectos del COVID-19 en el comercio internacional y la logística.* Anónimo: Cepal. Recuperado de https://repositorio.cepal.org/bitstream/handle/11362/45877/1/S2000497_es.pdf

Anónimo. (29 de octubre de 2019). *Python's Pandas vs R's Tidyverse: Who Wins?*. Anónimo: Analytics India Magazine. Recuperado de https://analyticsindiamag.com/pythons-pandas-vs-rs-tidyverse-who-wins/

9 Anexos

9.1 Código y dataset

En el siguiente enlace, se encuentra el código y archivo .CSV con el que se realiza el EDA:

https://github.com/INC98/EDA