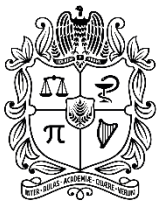

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Construcción de modelos de riesgo con Python

Laura Isaza Echeverri

Universidad Nacional de Colombia
Facultad de Minas, Área Curricular de Sistemas e Informática
Medellín, Colombia
2022

Construcción de modelos de riesgo con Python

Laura Isaza Echeverri

Trabajo Final presentado como requisito parcial para optar al título de:
Magister en Ingeniería Analítica

Director:
Juan David Velásquez Henao, PhD

Línea de Investigación:
Analítica
Grupo de Investigación:
Big Data & Data Analytics

Universidad Nacional de Colombia
Facultad de Minas, Área Curricular de Sistemas e Informática
Medellín, Colombia
2022

Declaración de obra original

Yo declaro lo siguiente:

He leído el Acuerdo 035 de 2003 del Consejo Académico de la Universidad Nacional. «Reglamento sobre propiedad intelectual» y la Normatividad Nacional relacionada al respeto de los derechos de autor. Esta disertación representa mi trabajo original, excepto donde he reconocido las ideas, las palabras, o materiales de otros autores.

Cuando se han presentado ideas o palabras de otros autores en esta disertación, he realizado su respectivo reconocimiento aplicando correctamente los esquemas de citas y referencias bibliográficas en el estilo requerido.

He obtenido el permiso del autor o editor para incluir cualquier material con derechos de autor (por ejemplo, tablas, figuras, instrumentos de encuesta o grandes porciones de texto).

Por último, he sometido esta disertación a la herramienta de integridad académica, definida por la universidad.

LAURA ISAZA ECHEVERRI

_____

Nombre

16/02/2022

Fecha

Agradecimientos

A mi tutor por el tiempo dedicado y los conocimientos brindados. Agradecerle también a toda mi familia por apoyarme durante este proceso. A mi novio y amigos que siempre me acompañaron y a la empresa en la que actualmente laboro por darme el tiempo para seguir formándome.

Resumen

Recientemente se ha venido popularizando el uso de herramientas de código abierto, estas son definidas como softwares desarrollados con una licencia que permite que cualquier persona pueda usarla libremente sin ninguna restricción. El propósito de este articulo presenta una librería para el análisis de riesgo en el lenguaje de programación Python llamada PyRisk, que aprovecha las librerías existentes para computo, empleando procesos de simulación para lograr percibir las capacidades y comportamientos de un escenario en el cual se quiera estudiar o probar algo, sin necesidad de reproducirlo realmente. Como resultado se obtiene una librería para uso científico, logrando demostrar ser competitiva con las funcionalidades que presentan las herramientas comerciales y destacando en características como velocidad de ejecución y procesamiento de grandes volúmenes de datos.

Palabras clave: (Python, análisis de riesgo, código abierto, PyRisk, @Risk).

Abstract

Building risk models with Python

Recently the use of open-source tools has become popular, these are defined as software developed with a license that allows anyone to use it freely without any restriction. The purpose of this article presents a library for risk analysis in the Python programming language called PyRisk, which takes advantage of existing libraries for computing, using simulation processes to perceive the capabilities and behaviors of a scenario in which you want to study or test something, without the need to really reproduce it. As a result, a library is obtained for scientific use, managing to prove to be competitive with the functionalities presented by commercial tools and highlighting features such as speed of execution and processing of large volumes of data.

Keywords: (Python, risk analysis,open source, PyRisk, @Risk).

Contenido

Pág.

Lista de figuras

Pág.

Lista de tablas

Pág.

1. Introducción

Las empresas deben tomar decisiones que sean robustas ante la incertidumbre y la variabilidad de los hechos del futuro; estas acciones conllevan un riesgo inherente sobre el cual se puede tener información y este posteriormente podrá ser analizado para tener un amplio conocimiento sobre su impacto en el negocio. Reconocer el significado del riesgo lleva a evidenciar como las variables pueden influir en el resultado de éste, teniendo en cuenta que estas son cada vez más volátiles; también es importante reconocer que en la vida real no se pueden realizar experimentos extensos para evaluar el riesgo de las situaciones, por lo que este deberá ser estimado bajo la información histórica disponible [1].

La simulación es un medio por el cual los procesos pueden proyectar una condición futura, lo que permite a las organizaciones estudiarlos desde una perspectiva más amplia, procurando una mejor comprensión de la causa y efecto de los eventos que ocurren o que pueden ocurrir [2]; estos escenarios generan información para ser analizada y adquirir un amplio conocimiento sobre el impacto en el negocio. Existen varios métodos de simulación. Por ejemplo:

- Simulación estadística o Monte Carlo: Muestreo sistemático de variables aleatorias.
- Simulación continua.
- Simulación por eventos discretos.

La simulación de Monte Carlo es una de la más utilizadas actualmente para realizar este tipo de procedimientos; este método se desarrolló en los años 40 cuando John Von Neumann aplica la simulación para resolver problemas complejos que no podía resolver de forma analítica. Esta técnica de muestreo analiza las distribuciones de las variables aleatorias replicando el comportamiento real de un sistema; internamente se define un

cómputo en el que los posibles resultados se crean mediante el cálculo repetido de los valores ingresados [2][3].

La utilidad de los modelos de simulación depende de una adecuada selección de los componentes que influyen directamente en la representación del problema; para valorar que los resultados servirán de orientación en la toma de decisiones se realiza una verificación del ajuste a la realidad, además de conocer cuál es el error inherente al proceso aceptable en los resultados; vale la pena aclarar que este error puede ser reducido más no eliminado [4].

La toma de decisiones en ciertas ocasiones es un proceso de prueba y error, por lo que aplicar analítica predictiva y prescriptiva como alternativa para conocer qué tan probable es que ocurra un evento en el futuro, es una de las opciones que mejor se adapta para disminuir la incertidumbre, anticiparse y prevenir el riego del evento que se está evaluando [5].

En el proceso de toma de decisiones las organizaciones enfrentan el riesgo de no conocer exactamente el resultado de sus acciones en el futuro, esto implica que los decisores deben apelar a metodologías que les permitan tomar decisiones robustas; esto es, que las decisiones tomadas sigan siendo válidas antes cambios en las variables que afectan el desempeño de la compañía [5].

Esta propuesta de trabajo final busca diseñar un algoritmo que permita realizar en un lenguaje de programación más avanzado y adaptable como Python; además, usando sus librerías realizar los cálculos del riesgo asociados a una situación mediante la simulación de Monte Carlo y así eliminar gradualmente los problemas asociados a la limitación de información y la velocidad algorítmica.

1.1 Conceptos básicos

Una de las características más importantes de un científico de datos es la habilidad para lograr traducir los datos en conocimiento, usando la realidad del negocio y así lograr un equilibrio que le permita de forma rápida y concisa entender lo que pasa y lo que podría pasar en un futuro; para esto es importante entender los diferentes tipos de analítica de datos y poder recurrir a estas teniendo en cuenta la profundidad del análisis que se desee construir:

1.1.1 Analítica descriptiva

Técnica que responde a la pregunta ¿qué sucedió?, analizando los acontecimientos con datos históricos, resumiéndolos y presentándolos de manera gráfica y/o tabular. Gracias a la analítica descriptiva es posible profundizar en los datos para identificar patrones y así obtener información detallada acerca de un problema en particular [6].

1.1.2 Analítica diagnóstica

Busca principalmente responder el porqué del problema que se presentó en el análisis descriptivo; el análisis diagnóstico es un poco más complejo y requiere que los analistas hagan uso de capacidades analíticas para encontrar fácil y rápido las causas del problema, entendiendo problema como el fenómeno analizado [7].

1.1.3 Analítica predictiva

El análisis predictivo ayuda a determinar lo que probablemente sucederá en un futuro, usando la analítica descriptiva y diagnostica para detectar tendencias, patrones, etc.; es esencial comprender que una buena predicción depende en gran medida de la calidad de los datos por lo que se requiere un tratamiento exhaustivo de la información y una optimización continua. La analítica predictiva hace uso de la matemática avanzada, como la estadística o el aprendizaje automático [8].

1.1.4 Analítica prescriptiva

Es la parte de la analítica que se encarga de coleccionar datos, recomendar acciones y prever que impacto tendrán para facilitar la toma de decisiones, identificando así la mejor decisión entre todas las posibles; además de identificar como impactan esas decisiones que se van a tomar y automáticamente poder generar acciones realistas que ayuden a mitigar las posibles consecuencias sobre el negocio [9].

1.2 Definiciones y conceptos

La simulación es una herramienta de análisis que se ha difundido rápidamente en el entorno empresarial, comprobando claramente su utilidad para apoyar la toma de decisiones relacionada con cualquier tipo de situación que muestre un riesgo [10], ya que mediante esta técnica se puede conocer anticipadamente bajo diferentes condiciones cual va a ser el comportamiento de las variables que describan el problema y posteriormente como actuar cuando realmente se vaya a presentar el escenario que se sometió a prueba. En [11], se indica que *"La simulación es la representación de la realidad mediante el empleo de un modelo u otro mecanismo que reaccionará del mismo modo que la realidad bajo una serie de condiciones dadas.".* Un sistema basado en la simulación tiene la capacidad de considerar tareas o situaciones complejas y proyectarlas mediante la ejecución de muchas combinaciones alternativas, lo que se traduce en un gran número de escenarios posibles los cuales serían difíciles de abarcar y valorar sin la ayuda de un modelo de simulación computarizado [4].

Dentro de la simulación existe un método, conocido como la **simulación de Monte Carlo**, su objetivo principal capturar la representación de realidad mediante la ejecución de experimentos; la simulación de Monte Carlo es una técnica cuantitativa que permite el desarrollo de un modelo lógico-matemático ayudando con la generación de una historia artificial de un sistema [12]. La simulación de Monte Carlo reproduce los valores de una variable cuya distribución es desconocida, basado principalmente en la selección de números aleatorios para las variables independientes que afectan directamente a la situación plasmada, considerando sus distribuciones de probabilidad [13]. Para lograr aplicar este método es necesario contar con la suficiente información, que permita establecer cómo se comportan las variables asociadas al planteamiento y como estas pueden afectar o son afectadas por otras variables dentro del sistema [14].

Riesgo aparece con el reconocimiento de la incertidumbre que tiene una situación, ya que este representa acciones que pueden tener más de un resultado; cuando este es detectado se le puede asignar una probabilidad de ocurrencia para así tomar decisiones respaldadas por un análisis previo que logra identificar cual es el mejor camino por seguir para la situación inicialmente planteada [15].

Distribución de probabilidad es un término matemático que se usa para denominar la frecuencia de posibles valores de un experimento si este se llevase a cabo, es decir, ésta describe la probabilidad de un evento. Dentro de una distribución de probabilidad puede haber variables aleatorias discretas o continuas [16].

1.3 Flujo para la construcción de análisis de riesgo

El análisis de riesgo es un método de análisis cuantitativo diseñado para observar los resultados en forma de distribuciones de probabilidad, esta técnica comprende los siguientes pasos [15]:

1.3.1 Objetivos de la simulación

El propósito de la simulación es lograr percibir las capacidades y comportamiento de un escenario en el cual se quiera estudiar o probar algo, sin necesidad de reproducirlo realmente [15].

1.3.2 Identificación de las variables sobre las cuales se va a medir el riesgo

Para lograr cuantificar el riesgo es necesario identificar las variables sobre las cuales se va a medir el riesgo de la situación en cuestión; cuales de estas variables podrían afectar directamente el aumento o disminución del riesgo que se quiere analizar [15].

1.3.3 Selección de las distribuciones de probabilidad

Dentro de este paso es necesario identificar la función de probabilidad que se ajusta a cada una de las variables afectadas por el riesgo sobre el cual se desea reducir el impacto, teniendo en cuenta que la distribución que se quiere buscar para explicar el

comportamiento de los datos de entrada es desconocida, por lo tanto es necesario ir probando con las diferentes distribuciones que existen e ir variando sus parámetros de entrada hasta encontrar la que realmente se ajusta al problema en cuestión y pueda modelar y dar respuestas acertadas para la toma de decisiones [15].

1.3.4 Simulación computacional

En este paso se ejecutará de forma automática por parte de una herramienta (computador) la cantidad de iteraciones o cálculos definidos por el usuario con el fin de obtener una muestra que sea representativa de la realidad en un menor tiempo, para lograr tomar decisiones que ayuden a mitigar el impacto del riesgo asociado a la situación planteada, conociendo la probabilidad de los diferentes resultados posibles [15][11].

1.3.5 Análisis de riesgo

Finalmente se obtendrá una visión del riesgo que facilite la toma de decisiones en cada momento del ciclo del proyecto o situación planteada al inicio. Se entiende por análisis de riesgo el uso de la información disponible para identificar los peligros existentes y estimar el nivel de riesgo presente en esa situación [5][15].

1.4 Herramientas de usuario final

Los modelos de simulación para evaluar el riesgo de una situación a través de hojas de cálculo de Excel mediante simulaciones de Monte Carlo son muy comunes y actualmente no reflejan mucha diferencia en sus funcionalidades; además, poseen varios problemas asociados a la limitación de información y la velocidad algorítmica.

1.4.1 @Risk

@Risk es un complemento de Microsoft Excel que forma parte de los softwares DecisionTools Suite de *Palisade Corporation* y que permite realizar modelación y análisis de riesgo mediante simulación en diferentes áreas como: finanzas, ciencia, ingeniería

entre otras, usando simulaciones de Monte Carlo para mostrar múltiples resultados probables y sus riesgos asociados.

Dentro de este complemento se incorporan funciones que permiten especificar la distribución de probabilidad que se va a usar en el análisis, además de parametrizar los atributos; los resultados de las distribuciones de salida se pueden presentar en gráficos de resumen que complementan el análisis de riesgo asociado a la situación planteada [17].

1.4.2 SAS (Statistical Analysis System)

SAS en un lenguaje de programación desarrollado por SAS Institute; este lenguaje opera principalmente sobre tablas de datos, para leerlas, transfórmalas, combinarlas, analizarlas y generar reportes sobre los datos de entrada. Este software es licenciado. SAS está pensado como un software de análisis y manejo de grandes datos; la herramienta pretende facilitar el proceso de la toma de decisiones mediante modelos predictivos, descriptivos, de simulación y optimización [18].

Dentro de SAS existe un módulo llamado **Sommelier** que permite realizar simulación Monte Carlo y análisis Bayesiano, donde las distribuciones de probabilidad pueden ser discretas o continuas; a diferencia de otras herramientas comerciales SAS no posee una interfaz gráfica para interactuar con el análisis, por el contrario, hay que realizarlo mediante líneas de código, que si bien ya varias funciones están predefinidas las demás deben ser desarrolladas por el usuario [18].

1.5 Lenguajes de programación y paquetes asociados

Los lenguajes de programación están diseñados especialmente para dar órdenes a una computadora, de manera exacta y no ambigua [19]. Las soluciones creadas bajo un lenguaje de programación toman el nombre de algoritmo, es decir un método de cálculo; para un mismo problema o situación pueden existir algoritmos diferentes, cada uno con un costo distinto en términos de recurso computacionales.

1.5.1 R

Es un lenguaje y entorno de software libre enfocado principalmente en el análisis estadístico, nació como reimplementación del software S y fue desarrollado en 1993 en la Universidad de Auckland; su desarrollo actual esta administrado por R Development Core Team; proporciona una cantidad amplia de herramientas (modelos lineales, modelos no lineales, análisis de series de tiempo, test estadísticos de diferentes clases, etc.) que permiten a cualquier científico de datos realizar análisis de alto nivel debido a la utilidad de los resultados obtenidos [20].

Unas de las características más relevantes de R es que se pude integrar fácilmente con distintas bases de datos, además de que existen librerías que facilitan su uso desde lenguajes de programación interpretados como Python. Antes que nada, R ya posee por defecto herramientas (Librería STATS) que permiten la lectura y manipulación de bases de datos, sin embargo, se debe hacer uso de otras librerías en este lenguaje que permiten la creación de un análisis de riesgo [21].

Dentro de este lenguaje existen todas las funciones que se pueden usar para modelar una distribución de probabilidad, estas son llamadas por su mismo nombre, por ejemplo, `runif (n, min, max), rnorm (n, mean, sd), rbinom (n, size, prob), rpois (n, lambda),` entre otras.

Dentro de R no existe una función exclusiva para realizar simulación de Monte Carlo; pero, teniendo en cuenta las funciones que permiten aproximar los datos a una distribución de probabilidad y haciendo uso de ciclos de programación como el `FOR,` se puede realizar iteraciones del experimento o situación que se desea evaluar [20].

1.5.2 Python

Python es un lenguaje de programación multiparadigma que posee una licencia de *código abierto y gratuito* administrado por *Python Software Foundation*. Este se ha vuelto bastante popular por varias razones, una de ellas es la cantidad de librerías y funciones incorporadas que pueden ser usadas sin necesidad de programar desde cero, otra es la velocidad y sencillez con la que se puede crear un programa, y por último la cantidad de sistemas operativos o plataformas en las que se pude crear contenido [22]. Algunas de las librerías que tiene este lenguaje de programación son:

Pandas (Panel Data): Es una librería de análisis y manipulación de datos de alto rendimiento la cual proporciona estructuras flexibles que facilitan el análisis de información tabular. Usando esta librería se pueden lograr cincos pasos típicos en el procesamiento y análisis de datos, los cuales son: cargar, preparar, manipular, transformar y analizar.

Las estructuras de la librería Pandas son llamadas *data frame*, estas están compuestas por filas y columnas, a los cuales habitualmente se les denomina índices y determinan la forma de acceder a los elementos; además, los datos pueden ser heterogéneos y su tamaño puede ser mutable [23].

Numpy (Numerical Python): Es la librería principal para la informática científica y manipulación matemática, proporciona potentes estructuras de datos, implementando matrices N-dimensionales, garantizando cálculos eficientes en memoria [23].

Scipy (Scientific Python): Las estructuras básicas usadas por este módulo son arreglos multidimensionales proporcionados por *Numpy.* Dentro de este módulo se tiene principalmente procesos de optimización, herramientas de números aleatorios, ajustes lineales y no lineales, funciones de densidad de probabilidad, distribuciones de probabilidad y manipulación de arreglos [23].

Sympy: Es una biblioteca de simulación de eventos discretos [23].

Matplotlib: Es una librería para generar gráficos a partir de los datos contenidos en arreglos de Numpy, tiene módulos predeterminados e incorporados que ayudan en la construcción de gráficos [23].

Dado todo el potencial que tiene este lenguaje de programación, se concluye que este suple las necesidades computacionales, ya que las hojas de cálculo de Excel no están diseñadas para manejar grandes volúmenes de datos y tienen un alto costo computacional [24].

1.6 Requerimientos para herramientas de análisis de riesgo en open Data Science

Se requiere una herramienta de software libre que permita:

- Desarrollar análisis para determinar que influencia tienes las variables de entrada sobre la variable de salida.
- Encontrar la distribución que mejor se ajuste a los datos.
- Ingresar gran cantidad de datos históricos sin que pierda capacidad.
- Interacción directa del usuario para que configure todos los atributos necesarios para el análisis de riesgo.
- Visualización de los resultados, mediante gráficos y tablas con información de valor para la toma de decisiones.

En la Tabla 1 se presentan las principales características de las herramientas comerciales que existen actualmente y que se tendrán en cuenta para el desarrollo de este trabajo final:

Tabla 1 Comparativo de las herramientas comerciales que actualmente están en el mercado.

Herramienta	Herramientas analíticas	Técnicas de optimización	Características	Interfaz de usuario	Open source	Reportes	Tipos de gráficos
@ Risk	**Análisis avanzados:** Permite determinar los efectos de las variables de entrada sobre variables de salida. **BestFit** encuentra la distribución que mejor se ajusta a los datos. **Precision Tree** crea árboles de decisión.	**RISK OPTIMIZER Solver** (optimización lineal y no lineal) y **Evolver** (optimización a través de algoritmos genéticos)	38 FUNCIONES DE PROBABILIDAD	SI	NO	SI	Boxplot, gráficos de tendencia, histogramas, matriz de correlación.
Risk Simulator Software Shop	**ROV BizStats:** Bootstrapping, Segmentación de Grupos, Prueba de Hipótesis, Análisis de Componentes Principales, árbol de decisión**,** Box-Jenkins ARIMA, Auto ARIMA,	Optimización lineal y no lineal. Algoritmo Genético en optimización	45 FUNCIONES DE PROBABILIDAD 6 GENRADORES DE NÚMEROS ALEATORIOS	SI	NO	SI	Gráfico de tornado y araña
Crystal Ball	**Batch Fit:**	**Oracle Crystal**	16	SI	NO	SI	Gráfico

Oracle	series de tiempo Bootstrap Análisis de Sensibilidad. **CB Predictor:** ARIMA, regresión lineal, suavizados exponenciales.	**Ball Decision Optimizer OptQuest:** mejora los modelos de simulación al buscar y encontrar soluciones óptimas de forma automática.	DISTRIBUCIONES DISCRETAS Y CONTINUAS				de tornado, histogramas, gráficos de sensibilidad
Insight.Xla	Árboles de decisión Cadenas de Markov Pronósticos	NA	NA	NA	NA	NA	NA
SimTools.xla	NA	NA	32 FUNCIONES DE PROBABILIDAD	NA	NA	NA	NA
RiskAmp	NA	NA	40 DISTRIBUCIONES DE PROBABILIDAD	NA	NO	NA	NA
SimulAr	NA	NA	22 DISTRIBUCIONES DE PROBABILIDAD	SI	NA	SI	
XLSTAT-Sim Addinsoft	Análisis de componentes principales Análisis de correspondencia Análisis discriminante Modelos de mezclas gaussianas Regresión lineal Regresión logística	NA	30 DISTRIBUCIONES DE PROBABILIDAD	SI	NO	NA	NA
XLSim	NA	NA	NA	NA	NO	NA	NA

1.7 Hipótesis

La hipótesis de este trabajo es la siguiente: Es posible desarrollar herramientas para el análisis de riesgo en el lenguaje Python que aprovechen las librerías existentes para computo científico y traficación, que sean competitivas con las funcionalidades que presentan las herramientas comerciales y de propietario. Los lenguajes de programación como Python y R ya contienen muchas de las funcionalidades requeridas para el análisis de modelos de riesgo y solo es necesario desarrollar unos pocos elementos adicionales.

1.8 Objetivos

1.8.1 Objetivo General

Desarrollar una librería en el lenguaje Python que brinde funcionalidades al usuario final para el desarrollo de modelos de análisis de riesgo que sea competitiva con las funcionalidades brindadas por herramientas comerciales.

1.8.2 Objetivos Específicos

Los objetivos específicos de este trabajo son:

- Desarrollar una App que permita seleccionar que distribución de probabilidad se va a usar para modelar una variable determinada.
- Desarrollar una App que permita el análisis de correlación entre las variables.
- Desarrollar una App que permita la visualización de los resultados.

2. Herramienta desarrollada

La herramienta creada tiene como objetivo principal, demostrar como el lenguaje de programación Python brinda al usuario un desarrollo para el análisis de riesgo, que le permita evaluar el impacto de una situación particular bajo distintitos escenarios y así tener a priori todo el conocimiento que se necesita para definir las posibles soluciones para el fenómeno de estudio.

La librería permite realizar de manera óptima y eficiente las simulaciones correspondientes a un análisis de riesgo, el cual en la actualidad se realiza en hojas de cálculo, las cuales se limitan a cierta cantidad de información y su procesamiento se considera lento.

La solución planteada atiende necesidades de procesamiento y velocidad de cómputo; además, tiene implícitas las funcionalidades del lenguaje de programación Python y sus librerías, teniendo en cuenta, que a hoy la gran mayoría de las distribuciones de probabilidad que están expuesta por ejemplo en @Risk [25][26] se encuentran ya implementadas en Python; en la Tabla 2 se observa el equivalente de las distribuciones , algunas de estas se usaron en la implementación de la librería queriendo recrear ejemplos específicos para mostrar su funcionamiento.

Las distribuciones de probabilidad que se encuentran desarrolladas para el lenguaje de programación Python, están principalmente implementadas en las librerías Numpy y Scipy [27][28].

Tabla 2 Distribuciones de probabilidad Python vs @Risk

Python	@Risk
numpy.random.normal	RiskNormal
scipy.stats.bernoulli	RiskBernoulli
scipy.stats.beta	RiskBeta

scipy.stats.binom	RiskBinomial
scipy.stats.cauchy	RiskCauchy
scipy.stats.chi2	RiskChiSq
scipy.stats.uniform	RiskUniform
scipy.stats.expon	RiskExpon
scipy.stats.gamma	RiskGamma
scipy.stats.geom	RiskGeomet
scipy.stats.hypergeom	RiskHypergeo
scipy.stats.norminvgauss	RiskInvgauss
scipy.stats.laplace.numargs	RiskLaplace
scipy.stats.lognorm	RiskLognorm
scipy.stats.poisson	RiskPoisson
numpy.random.triangular	RiskTriang
scipy.stats.t	RiskStudent
numpy.random.weibull	RiskWeibull
scipy.stats.pareto	RiskPareto
scipy.stats.randint	RiskDUniform
scipy.stats.dlaplace	RiskDiscrete
scipy.stats.burr12	RiskBur12
scipy.stats.erlang	RiskErlang
scipy.stats.fatiguelife	RiskFatigueLife
scipy.stats.logistic	RiskLogistic
scipy.stats.johnsonsb	RiskJohnsonSB
scipy.stats.johnsonsu	RiskJohnsonSU
scipy.stats.levy	RiskLevy
scipy.stats.rayleigh	RiskRayleigh

2.1 Descripción del software

Para la implementación y el desarrollo de la librería se utilizó el lenguaje de programación Python; esta librería permite reunir las funcionalidades principales de un análisis de riesgo realizado por las herramientas comerciales que existen actualmente en el mercado y así poder resolver situaciones de este tipo con mayor facilidad y rapidez.

2.1.1 Arquitectura del software

El paquete se implementó usando Python 3.9 con dependencias de Numpy, Matplotlib, Pandas, Scipy, Random y Seaborn. La arquitectura del paquete (Fig. 1) se desarrolló creando una clase llamada “simulator” que tiene los métodos que permiten realizar el análisis de riesgo.

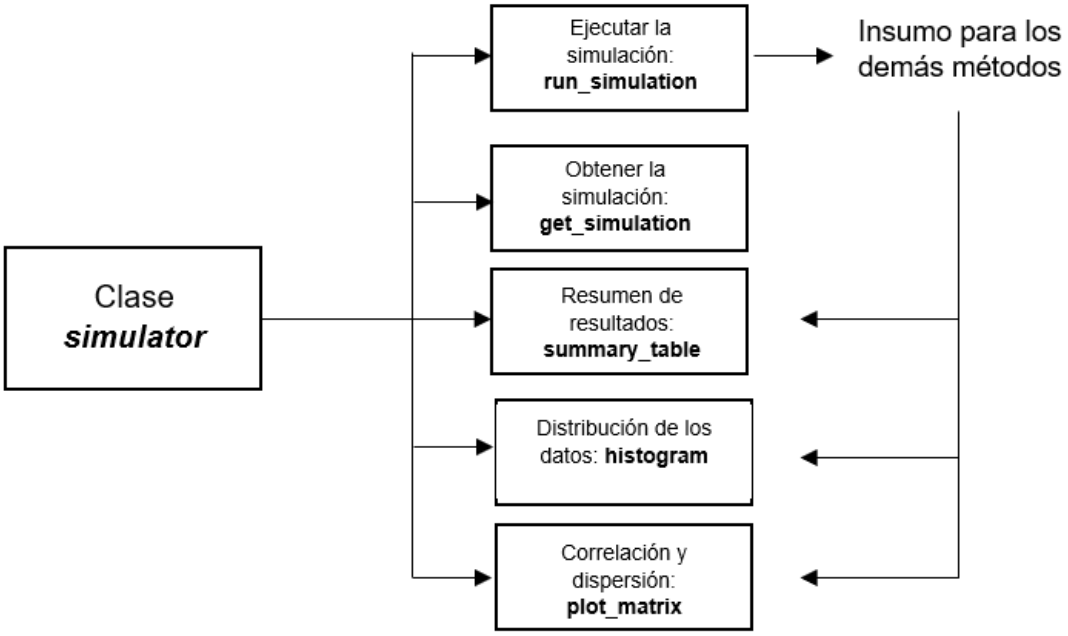

Fig. 1 Arquitectura del paquete.

Al instanciar la clase es necesario definir los siguientes atributos: el número de simulaciones, indica la cantidad de veces que se va a repetir un número n de interacciones, el número de iteraciones define la cantidad de veces que se va a ejecutar el experimento en cada simulación, una función generadora que replique el comportamiento del problema, una función que contenga las variables de entrada y un parámetro alpha que describe la confianza con la cual se van a generar los intervalos de confianza.

Para hacer uso de la librería es necesario seguir un orden de ejecución. El primer método, llamado run_simulation retorna la entrada para los demás, es decir, no se puede ejecutar otro método hasta no haber instanciado la clase con sus atributos y haber corrido el primer paso.

2.1.2 Funcionalidades del software

1. **Ejecución de la simulación (run_simulation):** Método de la clase simulator que usa los atributos asociados a la instanciación de esta, este permite al usuario iniciar la simulación donde se desea observar el comportamiento de la o las variables sobre las cuales se desea realizar inferencia; el usuario no recibirá ningún resultado, ya que estos serán la estructura que se considere como entrada en los demás métodos desarrollados.

2. **Obtener resultados de la simulación (get_simulation):** tiene como prerrequisito haber ejecutado el método **run_simulation**, en este punto el usuario si podrá acceder a los resultados de la simulación realizado bajo los diferentes escenarios y con las diferentes variables que desee observar, se podrá obtener un Pandas Data Frame, con la cual el usuario puede profundizar sobre su análisis, ya que esta permite realizar cálculos adicionales para complementar el ejercicio.

3. **Resumen de los resultados (summary_table):** Método que realiza cálculos de estadísticos descriptivos para cada una de las simulaciones realizadas con el fin de generar un resumen del comportamiento de las variables; la estructura retornada es un Pandas Data Frame.

4. **Distribución de los datos (histogram):** Método que se encarga de generar histogramas para la distribución de los datos de cada una de las variables en cuestión, dentro de este se pueden considerar parámetros de la librería seaborn para cambiar temas visuales del gráfico.

5. **Correlación y dispersión (plot_matrix):** En este método intervienen dos parámetros adicionales, uno de ellos es “plot_type” que describe que tipo de gráfico se quiere retornar, este puedo tomar dos valores posibles, “pairplot” que define el gráfico de dispersión y “corr_plot” que retorna el gráfico de correlación entre variables, este se combina con el otro parámetro necesario llamado “method” en el que se define porque método se desea calcular la correlación (pearson, kendall y spearman).

3. Comparación con otras herramientas

En la actualidad existen varias herramientas con las cuales se puede analizar el riesgo de una situación; estas generalmente están construidas sobre hojas de cálculo [29] (@Risk, Crystal Ball, Risk simulator, etc.); sin embargo, existen limitantes dentro de estas herramientas en cuanto al procesamiento de la información y la velocidad; además, las hojas de cálculo tienen un costo de licencia; por estas razones es importante reconocer que las hojas de cálculo no son la solución para la manipulación de datos; por el contrario, existen herramienta de código abierto que han demostrado su capacidad de procesamiento, almacenamiento y análisis, ya que muchas de estas repuntan en las encuestas que hace actualmente StackOverflow alrededor del mundo como las más usadas y la más querida por las personas que las utilizan [30]; por esto, se decide crear una herramienta que logre suplir estas necesidades y que le permita al usuario generar la solución a su cuestionamiento; además, de ir más allá de este con la forma en que el paquete logra retornar la información, que puede ser aprovechada para profundizar en la toma de decisiones.

La velocidad de ejecución de un modelo es una arista importante dentro del procesamiento de la información, y la simulación de Monte Carlo, que además se considera el motor del paquete es uno de los procesos que más toma tiempo realizar y en el paquete construido, ocupa una pequeña fracción de tiempo de todo el análisis, aproximadamente 37 segundos por simulación, variando la cantidad de iteración.

La herramienta se puede usar de forma interactiva en Jupyter Lab, Jupyter Notebook y Google Collaboratory.

El proceso de simulación de PyRisk , librería desarrollada y de @Risk herramienta comercial, se sometieron a una prueba de tiempos de ejecución (Fig. 2) y (Fig. 3) bajo diferentes escenarios, con el fin de medir la capacidad de ambas herramientas para generar resultados en fracciones cortas de tiempo. Se diseño un experimento tratando de variar la cantidad de simulaciones e iteraciones; la cantidad de simulaciones se definieron así, 1,4,6 y 10 simulaciones y la cantidad de iteraciones variaron entre 1000 y 20.000 iteraciones.

```
from time import time
tiempos=[]
leyenda_simulacion=[]
leyenda_iteracion=[]
simulaciones=[1,4,6,10]
iteraciones=[1000,2000,3000,4000,5000,6000,7000,8000,9000,10000,20000]
for i in simulaciones:
    for j in iteraciones:
        start_time = time()
        s1=Simulator(i,j,infected,varibles_entrada,0.95)
        s1.run_simulation()
        elapsed_time = time() - start_time
        leyenda_simulacion.append(i)
        leyenda_iteracion.append(j)
        tiempos.append(elapsed_time)
df=pd.DataFrame({"Simulacion":leyenda_simulacion,"Iteracion":leyenda_iteracion,"tiempos":tiempos})
print(df)
```

Fig. 2 Ejecución tiempos simulación PyRisk.

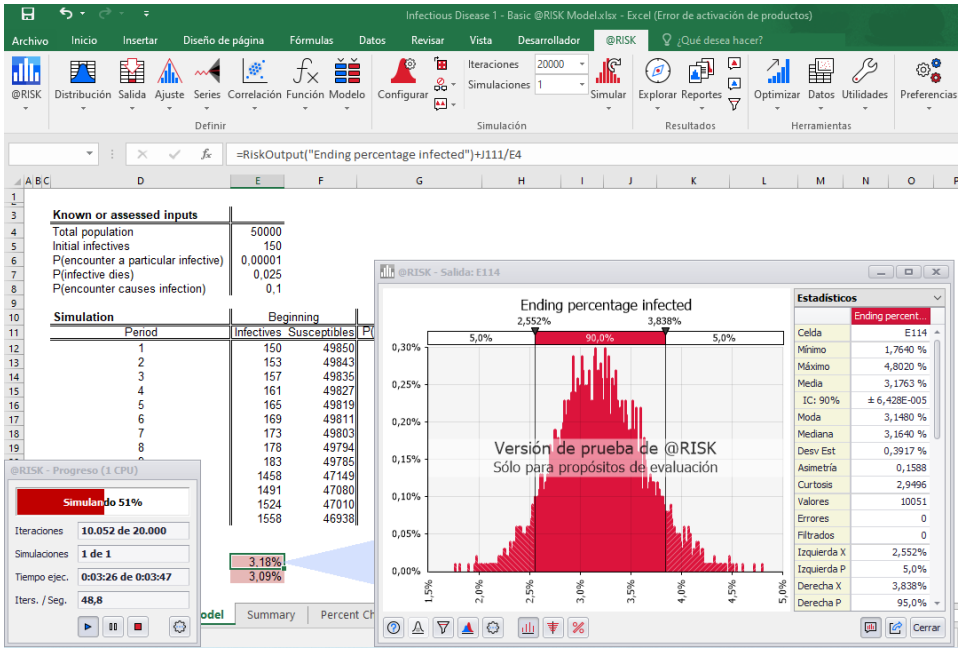

Fig. 3 Ejecución tiempos de simulación @Risk.

Los resultados de la ejecución se pueden ver en las Fig. 4 y Fig. 5; el proceso muestra que, las pruebas realizadas con PyRisk, se comportan mejor que las pruebas realizadas con @Risk; los resultados obtenidos fueron:

En PyRisk librería desarrollada (Fig. 4), como mínimo una simulación puede tardarse aproximadamente 1 segundo y en promedio 8, cuatro simulaciones se tardan como mínimo 5 segundos y en promedio 30, seis simulaciones tardan como mínimo 6 segundos y en promedio 41 y finalmente con diez simulaciones el proceso tarde como mínimo 10 segundos y en promedio 69; todo esto variándolo en diferentes cantidades de iteraciones por cada simulación.

En @Risk herramienta comercial diseñada sobre hojas de cálculo (Fig. 5), como mínimo una simulación puede tardarse aproximadamente 5 segundos y en promedio 41, cuatro simulaciones se tardan como mínimo 28 segundos y en promedio 168, seis simulaciones tardan como mínimo 34 segundos y en promedio 269 y finalmente con diez simulaciones el proceso tarde como mínimo 78 segundos y en promedio 638; todo esto variándolo en diferentes cantidades de iteraciones por cada simulación.

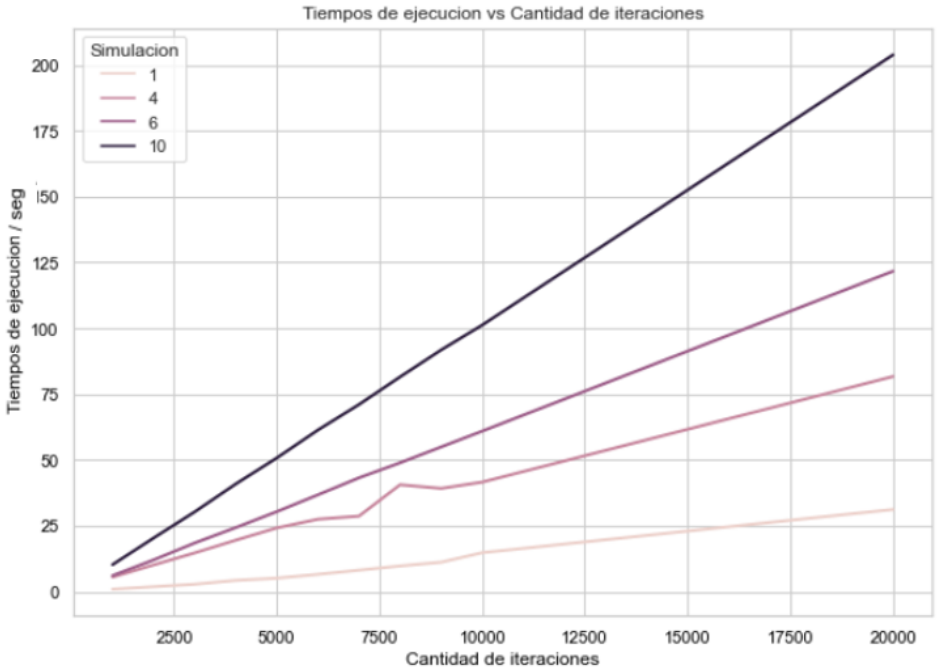

Fig. 4 Resultados tiempos de ejecución PyRisk.

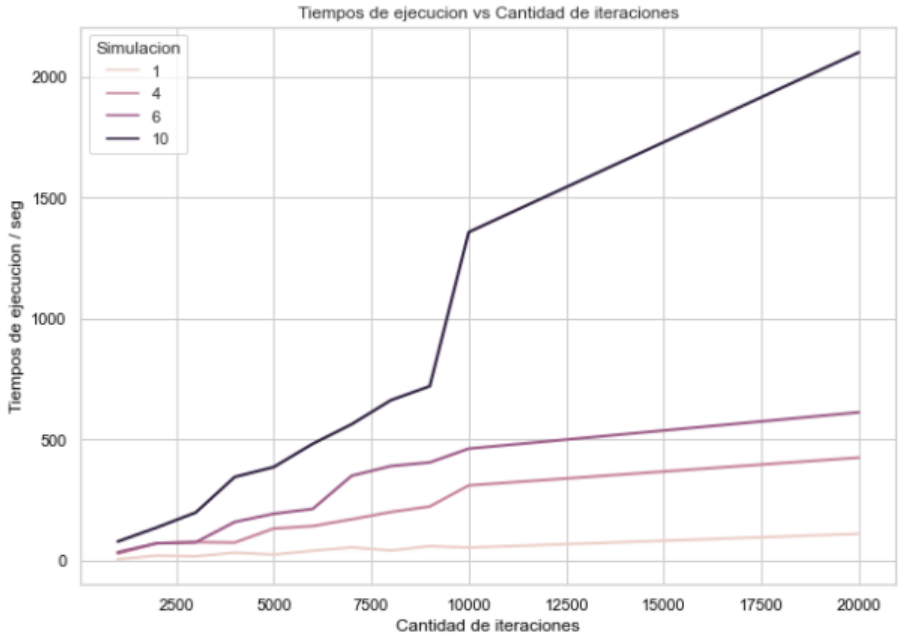

Fig. 5 Resultados tiempos de ejecución @Risk.

4. Ejemplos ilustrativos

4.1 Propagación de una enfermedad

En este capítulo se desarrolla un modelo para simular la propagación de una enfermedad infecciosa y obtener el porcentaje final de individuos infectados y fallecidos, considerando que los contagios y las muertes son cantidades aleatorias. Este ejemplo permite ilustrar diferentes funcionalidades de la librería implementada.

4.1.1 Descripción del problema

Este modelo ilustra una posible forma de simular la propagación de una enfermedad infecciosa. Una población comienza con un número de personas infectadas. Un infectado muere en un periodo determinado, con la probabilidad de 0.025, el resto de la población es susceptible. Un susceptible dado tiene la probabilidad de 0.00001 de encontrarse con un infectado en un periodo determinado, cuando se produce dicho encuentro, el susceptible se infecta con la probabilidad de 0.1 [31].

El modelo simula 100 períodos. La probabilidad clave, es la probabilidad de que un susceptible determinado se infecte en un periodo determinado [31].

Las siguientes ecuaciones son las que se modelan dentro del planteamiento del problema:

1. La cantidad de nuevos contagios se modela mediante una distribución binomial, teniendo en cuenta como parámetros de entrada, la cantidad de población susceptible en cada uno de los periodos en los que se propaga la infección, además, de la probabilidad de que un individuo sea susceptible a dicha infección:

$$contagios \sim B(n,p)$$

$$n = Población\ suceptible\ a\ la\ infección$$

$$p = Probabilidad\ de\ que\ un\ individuo\ sea\ suceptible$$

2. La cantidad de muertes se modela también mediante una distribución binomial, tomando como parámetros de entrada, la cantidad de individuos infectados en cada uno de los periodos, además, de la probabilidad de que un individuo infectado muera:

$$muertes \sim B(n,p)$$

$$n = Individuos\ infectados$$

$$p = Probabilidad\ de\ que\ un\ individuo\ muera$$

4.1.2 Creación de funciones de usuario

Para poder realizar la aplicación de la librería es necesario crear dos funciones, una para el almacenamiento de las variables de entrada que se puede ver en la Fig. 7 y otra que recree los cálculos que se deben hacer para llegar a retornar las variables que se desean simular que se puede ver en la Fig. 8; A continuación, se ilustra cómo deben ser creadas estas funciones en un ejemplo particular, además de como importar la librería para su uso:

4.1.3 Importar librería

En el código de la Fig. 6 se ilustra como debería ser cargada la librería para hacer uso de los módulos programados.

```
1 from Libreria.PyRisk import *
```

Fig. 6 Importar librería desarrollada.

4.1.4 Creaciones variables de entrada

En el siguiente fragmento de código, que se puede ver en la Fig. 7, se define una función, la cual tiene como tarea almacenar las variables de entrada del modelo a simular; esta debe retornar un diccionario de Python, en el cual la clave es el nombre de la variable y el valor almacena sus resultados. La estructura de datos con la cual se retorna la información fue elegida, ya que esta permite almacenar gran cantidad de tipos de datos, además, de identificar cada uno de sus elementos por una clave.

```
def varibles_entrada():

    return {
        "Total_population":50000,
        "Initial_infectives":150,
        "P_encounter_a_particular_infective":0.00001,
        "P_infective_dies":0.025,
        "P_encounter_causes_infection":0.1
    }
```

Fig. 7 Definición de variables de entrada.

4.1.5 Creación de la función que recrea el modelo

En el siguiente fragmento de código se define una función, la cual tiene como tarea modelar el comportamiento del problema que se desea simular, tal y como se muestra en la Fig. 8; es necesario que lo que retorne esta función sea un diccionario de Python en el cual, se considere como clave el nombre de la variable de interés y el valor tome el resultado que se obtuvo para esta, mediante los cómputos realizados. Directamente ya en

el código puede verse para este ejemplo particular, como con una distribución de probabilidad Binomial puede modelarse la cantidad de infectados y muertos.

```
3
4   def infected(
5       Total_population,
6       Initial_infectives,
7       P_encounter_a_particular_infective,
8       P_infective_dies,
9       P_encounter_causes_infection,
10  ):
11
12      Infectives_B = []
13      Susceptibles_B = []
14      P_susceptible_infected = []
15      new_infected = []
16      deaths = []
17      infected_E = []
18      susceptibles_E = []
19      cum_deaths = []
20
21      for i in range(101):
22          if i == 0:
23              Infectives_B.append(Initial_infectives)
24              Susceptibles_B.append(Total_population - Infectives_B[i])
25              P_susceptible_infected.append(
26                  1
27                  - (
28                      1
29                      - P_encounter_a_particular_infective * P_encounter_causes_infection
30                  )
31                  ** Infectives_B[i]
32              )
33              new_infected.append(
34                  int(
35                      np.where(
36                          Susceptibles_B[i] == 0,
37                          0,
38                          np.random.binomial(
39                              Susceptibles_B[i], P_susceptible_infected[i]
40                          ),
41                      )
42                  )
43              )
```

```
            deaths.append(
                int(
                    np.where(
                        Infectives_B[i] == 0,
                        0,
                        np.random.binomial(Infectives_B[i], P_infective_dies),
                    )
                )
            )
            infected_E.append(Infectives_B[i] + new_infected[i] - deaths[i])
            susceptibles_E.append(Susceptibles_B[i] - new_infected[i])
            cum_deaths.append(deaths[i])

        else:
            Infectives_B.append(infected_E[i - 1])
            Susceptibles_B.append(susceptibles_E[i - 1])
            P_susceptible_infected.append(
                1
                - (
                    1
                    - P_encounter_a_particular_infective * P_encounter_causes_infection
                )
                ** Infectives_B[i - 1]
            )
            new_infected.append(
                int(
                    np.where(
                        Susceptibles_B[i - 1] == 0,
                        0,
                        np.random.binomial(
                            Susceptibles_B[i - 1], P_susceptible_infected[i - 1]
                        ),
                    )
                )
            )
            deaths.append(
                int(
                    np.where(
                        Infectives_B[i - 1] == 0,
                        0,
                        np.random.binomial(Infectives_B[i - 1], P_infective_dies),
                    )
                )
            )
            infected_E.append(Infectives_B[i] + new_infected[i] - deaths[i])
            susceptibles_E.append(Susceptibles_B[i] - new_infected[i])
```

```
90              cum_deaths.append(deaths[i] + cum_deaths[i - 1])
91
92      Ending_percentage_infected = infected_E[-1] / Total_population
93      Ending_percentage_dead = deaths[-1] / Total_population
94
95      return {
96          "Ending_percentage_infected": Ending_percentage_infected,
97          "Ending_percentage_dead": Ending_percentage_dead,
98      }
99
```

Fig. 8 Comportamiento de la situación a simular.

4.1.6 Validación

Para probar que realmente la función 2 retorna lo que se necesita y en la estructura de datos correcta, en el código de la Fig. 9, emplea una técnica con la cual se puede hacer esta verificación.

```
1 infected(**varibles_entrada())
```

```
{'Ending_percentage_infected': 0.02424, 'Ending_percentage_dead': 0.00058}
```

Fig. 9 Validación de resultados con respecto a la función 1 y 2.

4.1.7 Uso de la herramienta desarrollada

Para comenzar a usar la librería creada, es necesario realizar una instanciación de la clase llamada “simulator” que almacena todas las funcionalidades o métodos desarrollados para evaluar un análisis de riesgo desde la simulación, y así proceder a la toma de decisiones basada en los resultados del modelo.

4.1.8 Instanciación de la clase

Se instancia la clase, tal y como se ve en la Fig. 10, con los atributos necesarios para que esta funcione; en este punto ya se comienza a hacer uso de la herramienta desarrollada, puntualmente para este caso se realizaron 3 simulaciones, cada una con 2000

iteraciones, además se usó un parámetro Alpha igual al 95%, este se usa para calcular los intervalos de confianza.

```
1  s1=Simulator(3,2000,infected,varibles_entrada,0.95)
```

Fig. 10 Instanciación de la clase generadora de las funcionalidades.

4.1.9 Ejecución de la simulación

La siguiente línea de código inicia el proceso de simulación, haciendo uso del método llamado run_simulation como se puede ver en la Fig. 11, acá solo se realiza la ejecución; sin embargo, si el usuario desea obtener los resultados generados existe otra funcionalidad para hacerlo.

```
s1.run_simulation()
```

Fig. 11 Ejecución de la simulación.

4.1.10 Resultados de la simulación

En la Fig. 12 se muestra el código con el que se ejecuta el método con el cual se pueden obtener los resultados de la simulación ejecutada, en este punto se retorna un Pandas Data Frame por si en algún momento del proceso el usuario desea usar estos resultados con otros fines que aporten al resultado del problema planteado.

```
1 s1.get_simulations()
```

	simulacion	iteracion	Ending_percentage_infected	Ending_percentage_dead
0	0	0	0.02496	0.00064
1	0	1	0.02692	0.00056
2	0	2	0.03054	0.00108
3	0	3	0.02262	0.00040
4	0	4	0.02404	0.00060
...	...	...	...	...
5995	2	1995	0.02856	0.00078
5996	2	1996	0.02586	0.00052
5997	2	1997	0.02988	0.00088
5998	2	1998	0.02434	0.00070
5999	2	1999	0.02952	0.00062

Fig. 12 Resultados de la simulación.

4.1.11 Resumen de la simulación

En la línea de código de la Fig. 13, se usa el método summary_table, que resume el comportamiento de las simulaciones en función de los principales estadísticos descriptivos, que generan información específica acerca de los resultados obtenidos.

Este método retorna una estructura Pandas Data Frame, con esta el usuario también puede hacer uso de los resultados para fines particulares de su análisis.

```
1 s1.summary_table()
```

	simulacion	variable	Mínimo	Máximo	Media	Desviación est	Varianza	Asimetría	Curtosis	Moda	Mediana	IC	Q1	Q3	IQR
0	0	Ending_percentage_dead	0.00018	0.00122	0.000677	0.000141	1.997440e-08	0.298618	0.264257	0.00070	0.00068	(0.0006711222642930635, 0.0006835177357069364)	0.00058	0.000760	0.000180
1	0	Ending_percentage_infected	0.01774	0.04070	0.028152	0.003372	1.137017e-05	0.180213	0.045296	0.02684	0.02812	(0.028004100171280973, 0.028299839828719024)	0.02584	0.030240	0.004400
2	1	Ending_percentage_dead	0.00028	0.00128	0.000675	0.000141	1.992311e-08	0.179597	-0.122741	0.00070	0.00068	(0.0006685902268933671, 0.0006808969773106633)	0.00058	0.000780	0.000200
3	1	Ending_percentage_infected	0.01826	0.04086	0.028094	0.003296	1.086390e-05	0.151012	0.057619	0.02786	0.02796	(0.02794989969814709, 0.028238980301852908)	0.02585	0.030260	0.004410
4	2	Ending_percentage_dead	0.00022	0.00124	0.000672	0.000142	2.002965e-08	0.263510	0.152876	0.00066	0.00066	(0.0006655736999183664, 0.0006779863000816338)	0.00058	0.000760	0.000180
5	2	Ending_percentage_infected	0.01740	0.03956	0.028212	0.003395	1.152939e-05	0.148524	-0.168685	0.02682	0.02814	(0.028063498420688317, 0.02836130157931168)	0.02578	0.030465	0.004685

Fig. 13 Resumen de las simulaciones.

4.1.12 Distribución de los resultados

Método que genera las gráficas, específicamente histogramas, vistos en la Fig. 14, en las cuales se puede ver el comportamiento de las variables simuladas (como es su distribución), esta función retorna la cantidad de gráficos asociados a la cantidad de variables de interés dentro del modelo de análisis.

El usuario puedo hacer uso de parámetros de la librería seaborn que puedan mejorar el aspecto visual del gráfico, como por ejemplo el color, el grosor de la línea etc.

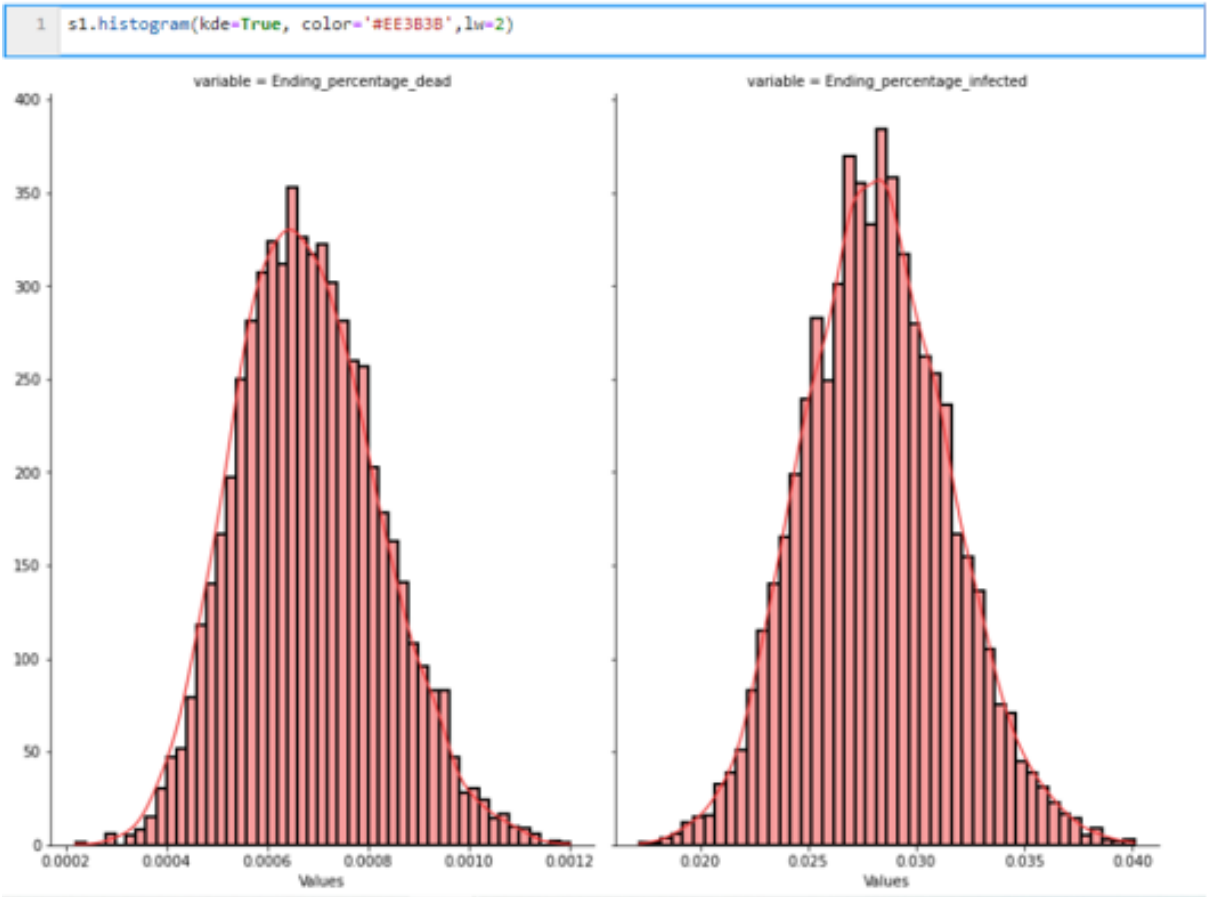

Fig. 14 Distribución de los datos simulados.

4.1.13 Correlación y dispersión

El paquete hace uso del método plot_matrix, como se puede ver en las líneas de código de las Fig. 15 y Fig. 16, la función tiene dos parámetros de interés, uno de ellos es el “plot_type”, en el cual se define que tipo de gráfico se desea obtener, si se usa “pairplot” se retornará la matriz de dispersión de los datos y si se usa “corr_plot”, el usuario recibirá el resultado de la correlación, además, de los valores para medirla; el otro parámetro que recibe función es “method” con el cual se puede variar la forma como se calcula la correlación, es decir, que se puede hacer uso de los métodos de “kendall” , “spearman” o “pearson”.

Fig. 15 Dispersión de las variables.

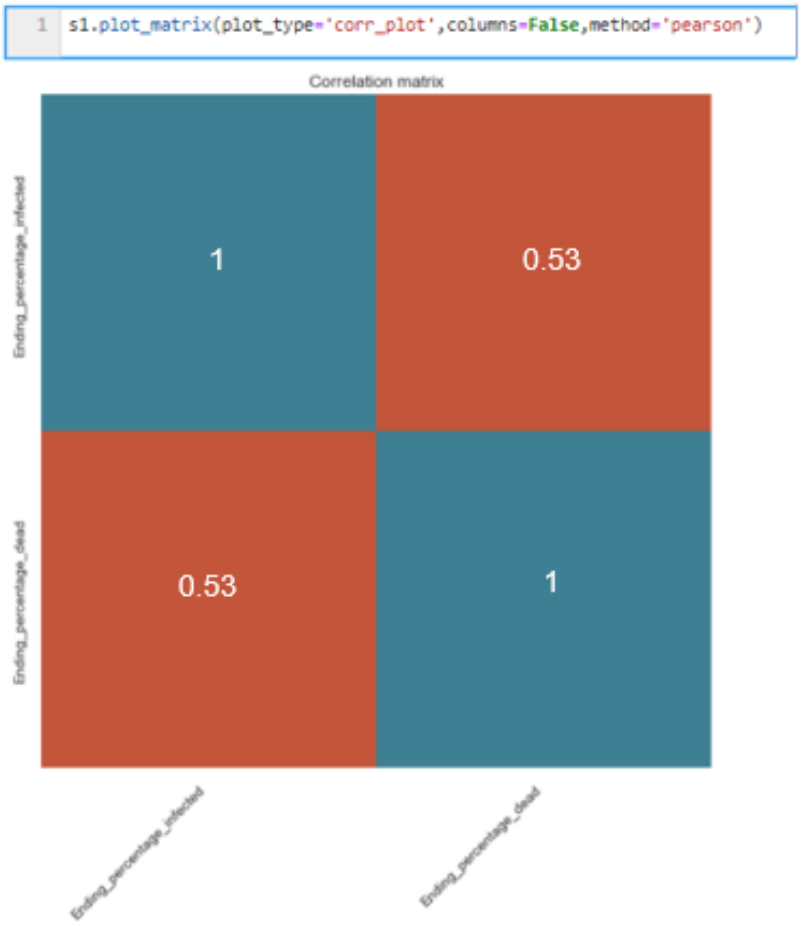

Fig. 16 Correlación entre las variables.

Este es uno de los tantos casos en donde se puede hacer uso de la librería, en el repositorio https://github.com/lisazae/PyRisk, se puede acceder a otros ejemplos en donde se evidencia el uso de la herramienta. Para aplicar directamente la librería en otro caso es necesario la creación de un entorno para posteriormente instalar el paquete.

4.2 Comportamiento de las variables de entrada

En este capítulo se desarrolla la simulación sobre las variables de entrada de un modelo particular, con el fin de identificar su posible comportamiento, tanto en su distribución, como en su correlación y dispersión, para tener conocimiento del posible efecto que estas puedan tener sobre el modelo final que se desea analizar.

El ejercicio ilustra el comportamiento de la sobreventa de tiquetes aéreos considerando diferentes variables de entrada que pudieran afectar o no la utilidad de la aerolínea. Este ejemplo permite abordar otra funcionalidad de la librería implementada.

Las siguientes variables de entrada son las que se tienen en cuenta para hacer uso de la funcionalidad:

1. El precio de la competencia por cada una de las sillas en tarifa plena sigue una distribución normal, con media 180 y desviación estándar 55.

   $$precio\_tarifa \sim N(\mu, \sigma)$$

   $$\mu = 180$$

   $$\sigma = 55$$

2. El porcentaje de ausencias de pasajeros en tarifa plena sigue una distribución normal con media 0.2 y desviación estándar 0.03.

   $$p\_ausencias\_tarifa \sim N(\mu, \sigma)$$

   $$\mu = 0.2$$

   $$\sigma = 0.03$$

3. El precio de la competencia por cada una de las sillas en tarifa descontada sigue una distribución normal con media 80 y desviación estándar 28.

   $$precio\_descontada \sim N(\mu, \sigma)$$

$$\mu = 80$$

$$\sigma = 28$$

4. El porcentaje de ausencias de pasajeros en tarifa descontada sigue una distribución normal, con media 0.1 y desviación estándar 0.01.

$$p\_ausencias\_descontada \sim N(\mu, \sigma)$$

$$\mu = 0.1$$

$$\sigma = 0.01$$

4.2.1 Variables de entrada

Las variables de entrada se definen como se hace en la sección 4.1.4, la Fig. 17 muestra las líneas de código asociadas a este ejemplo particular.

```
def varibles_entrada():

    return {
        "precio_competencia_tarifa":np.random.normal(loc=180,scale=55),
        "porcentaje_ausencias_tarifa":np.random.normal(loc=0.2 ,scale=0.03),
        "precio_competencia_descontada":np.random.normal(loc=80 ,scale=28),
        "porcentaje_ausencias_descontada":np.random.normal(loc=0.1 ,scale=0.01)
    }
```

Fig. 17 Variables de entrada, ejemplo 2.

4.2.2 Ejecución simulación variables de entrada

En la Fig. 18 se muestra el código con el que se ejecuta el método con el cual se pueden obtener los resultados de la simulación para las variables de entrada; además, de otro método que permite obtener dichos resultados, como puede verse en la Fig. 19.

```
s2.run_simulation_entry()
```

Fig. 18 Simulación variables de entrada.

```
1 s2.get_simulations()
```

	simulacion	iteracion	precio_competencia_tarifa	porcentaje_ausencias_tarifa	precio_competencia_descontada	porcentaje_ausencias_descontada
0	0	0	200.913389	0.181506	96.702191	0.110015
1	0	1	201.559132	0.176214	60.162448	0.106849
2	0	2	103.460933	0.207635	112.528910	0.102660
3	0	3	149.596633	0.144265	57.416373	0.085075
4	0	4	157.164448	0.188126	129.156083	0.112080
...	...	...	...	...	...	...
1995	1	995	166.526075	0.172869	90.244925	0.107197
1996	1	996	230.962204	0.154614	108.865315	0.097512
1997	1	997	251.452511	0.161179	3.681594	0.100590
1998	1	998	146.782010	0.163538	103.260909	0.101171
1999	1	999	168.925541	0.202369	72.357344	0.089513

2000 rows × 6 columns

Fig. 19 Obtener resultados variables de entrada

4.2.3 Distribución resultados variables de entrada

Método que genera las gráficas, específicamente histogramas, vistos en la Fig. 20, en las cuales se puede percibir el comportamiento de las variables de entrada (como es su distribución), esta función retorna la cantidad de gráficos asociados a la cantidad de variables de interés dentro del modelo de análisis.

El usuario puedo hacer uso de parámetros de la librería seaborn que puedan mejorar el aspecto visual del gráfico, como por ejemplo el color, el grosor de la línea etc.

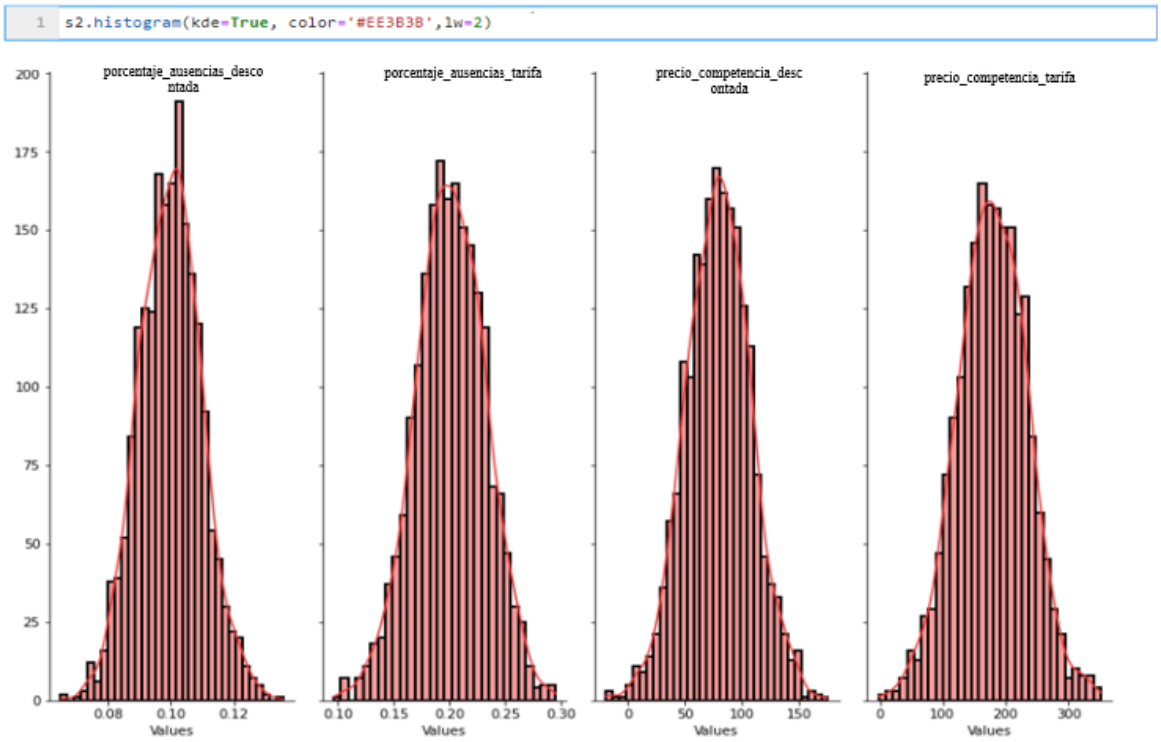

Fig. 20 Distribución variables de entrada.

4.2.4 Correlación y dispersión de las variables de entrada

Se usa igual a la descripción generada en la sección 4.1.13, en la Fig. 21, se pueden ver las líneas de código asociadas y los resultados que retorna el método empleado.

Fig. 21 Correlación variables de entrada.

El método ilustrado en la Fig. 18 se creó con el fin de que el usuario no tuviera que hacer uso de todas las variables de entrada del modelo, sino solamente las de interés, además de no tener que hacer uso de la función mencionada en la sección 4.1.2, es, decir, si bien dentro de la instanciación de la clase queda como atributo, el método no la usa para hacer la simulación, solo tiene en cuenta las variables de entrada y sus distribuciones para generar los valores resultantes.

5. Impacto

El paquete cubre las principales características disponibles en las herramientas comerciales haciendo uso de las funcionalidades de un lenguaje de programación como Python que permite la construcción de algoritmos simples y eficientes [24] en los siguientes aspectos:

1. La estructura de los resultados obtenidos (Pandas Data Frame) permite que el usuario pueda interactuar con estos y evaluar otras posibilidades de análisis que le brinden un panorama más amplio acerca del problema de estudio.
2. Con la evolución de la analítica, la programación es una competencia que todo profesional o científico de datos debe desarrollar, por esta razón es necesario evaluar herramientas especializadas en análisis de datos, esto implica que las hojas de cálculo pasen a ser reemplazadas por lenguajes de programación que generen ganancia en velocidad para cálculos complejo como Python, lenguaje con el cual se desarrolla la librería planteada en todo el despliegue de este documento y que evidencia las necesidades que logra cubrir; desde la simulación, hasta la visualización de los resultados..
3. El uso de un lenguaje de programación avanzado como Python reduce el tiempo de procesamiento específicamente de la simulación a una pequeña fracción de tiempo comparado con las hojas de cálculo; en el capítulo 3 se puede ver como la herramienta desarrollada y una herramienta comercial fueron sometidas a pruebas de estrés y cuáles fueron sus resultados en tiempos de ejecución sobre la simulación.
4. La mayoría de las herramientas comerciales para el análisis de riesgo son licenciadas, por lo que PyRisk representa una ventaja, ya que se encuentra programada en un lenguaje de programación de código abierto y puede ser usada fácilmente sin necesidad gran capacidad computacional.

5. Las funcionalidades de la librería se pueden usar de forma interactiva en Jupyter notebook, Jupyter Lab o Google Collaboratory.
6. La función que recrea el comportamiento del modelo puede ser diseñada haciendo uso de sklearn, tensorflow y demás librerías de Python, y los resultados llevarlos al método que ejecuta la simulación y seguir con el proceso para generar el análisis.

6. Conclusiones

6.1 Cumplimiento de objetivos

6.1.1 Primer objetivo especifico

El primer objetivo se definió así: Desarrollar una función que permita evaluar que distribución de probabilidad se ajusta a los datos simulados de una variable determinada, en la sección 4.1.12 se evidencia el método “histogram()” que hace parte de la clase “simulator”, la función se encarga de graficar por medio de histogramas la distribución de los resultados obtenidos de la simulación; además, de realizar un gráfico por cada variable de interés en el modelo que se desea recrear.

El cumplimiento de este objetivo también va ligado al desarrollo del método expuesto en la sección 4.1.9, ya que, para observar la distribución de los datos, es necesario ejecutar la simulación para obtener los resultados.

6.1.2 Segundo objetivo especifico

El segundo objetivo de definió así: Desarrollar una función que permita el análisis de correlación entre las variables, en la sección 4.1.13 se evidencia como el paquete diseñado es capaz de calcular no solo la correlación entre variables, sino también su dispersión, con el método llamado “plot_matrix”.

6.1.3 Tercer objetivo especifico

El tercer objetivo se definió así: Desarrollar una función que permita la visualización de los resultados, en el capítulo 4 se evidencia como en cada uno de los pasos que se siguen para recrear el modelo planteado en la sección 4.1, muestra visualmente los resultados, por medio de tablas y gráficos que permiten obtener una mayor información acerca del comportamiento del fenómeno analizado.

6.2 Conclusiones y trabajo futuro

En este documento se presenta el paquete PyRisk en el lenguaje de programación Python 3, que permite el análisis de riesgo; el paquete incorpora las principales características de las herramientas comerciales que actualmente existen en el mercado, en una solución de código abierto. Esta herramienta surge de la necesidad de poder generar análisis robusto, que puedan ser desarrollados con alta velocidad y que reemplacen las hojas de cálculo y sus limitaciones.

El paquete puede realizar una cantidad grande de simulaciones, aprovechando las capacidades del lenguaje de programación en el cual fue construido, en una pequeña fracción de tiempo; en el capítulo 3 se puede evidenciar como el método que ejecuta la simulación fue sometido a pruebas bajo diferentes escenarios y en promedio la simulación, variando la cantidad de iteraciones puede tardase aproximadamente 37 segundos y una herramienta comercial, como por ejemplo @Risk en promedio tarda 279 segundos.

La función que recrea el comportamiento del modelo puede ser desarrollada haciendo uso de sklearn y tensorflow; además, de todas las librerías con las que cuenta Python, para posteriormente someter los resultados al proceso de simulación.

El paquete a futuro podría integrase con otras herramientas; además, de hacer uso de otras librerías para la optimización.

7. Bibliografía

[1] D. E. Lehman and H. Groenendaal, *Practical Spreadsheet Modeling Using @Risk*, 1st ed. Boca Raton, 2019.

[2] S. M. Aguilar Mayorca and N. A. Segura Tenjica, “@Risk,” Bogotá, Jun. 2009.

[3] T. Y. Leong, “Monte Carlo Spreadsheet Simulation Using Resampling,” *INFORMS Transactions on Education*, vol. 7, no. 3, pp. 188–200, May 2007.

[4] C. Fullana Belda and E. Urquía Grande, “LOS MODELOS DE SIMULACIÓN: UNA HERRAMIENTA MULTIDISCIPLINAR DE INVESTIGACIÓN,” Madrid, 2009.

[5] F. Funston, S. Wagner, and H. Ristuccia, “Toma de decisiones inteligentes frente al riesgo,” *Deloitte Review*, no. 7, pp. 1–15, 2010.

[6] M. E. Rendón Macías, M. A. Villasís Keever, and M. G. Mirando Novales, “Estadística descriptiva,” *Revista Alergia México-RAM*, vol. 63, no. 4, Oct. 2016.

[7] DataHack, “TIPOS DE ANALÍTICA BIG DATA,” *https://www.datahack.es/tipos-analitica-big-data/*, May 24, 2019. .

[8] C. Espino Timón and X. Martínez Fontes, ““Análisis predictivo: técnicas y modelos utilizados y aplicaciones del mismo - herramientas Open Source que permiten su uso,” 2017.

[9] Informese, “Analítica prescriptiva,” *https://www.informese.co/descargas/informese-brochure-analitica-prescriptiva-mx.pdf*. .

[10] W. Guaita, “Modelos de Simulación de Eventos Discretos y de Procesos Continuos,” *Boletín de Dinámica de Sistemas*, 2016.

[11] J. F. Parra Garcés, “Simulación,” *Revista Colombiana de Estadística*, pp. 31–50, 1981.

[12] C. Azofeifa, “Aplicación de la simulación Monte Carlo en el cálculo del riesgo usando Excel,” *Tecnología en Marcha N˚ 1*, vol. 17, pp. 97–109, 2004.

[13] L. Pablo and R. Gustavo, “Gestión de proyectos : Como dirigir proyectos exitosos coordinar los recursos humanos y administrar los riesgos.,” *Gestión de proyectos*, vol. 1, 2007.

[14] B. C. Richard, R. J. F., and j. , A. Nicholas, *Administración de operaciones: producción y cadena de suministros*. México: McGraw-Hill, 2014.

[15] L. B. Lozano and F. M. Troncoso, “EL ANALISIS DE RIESGO: BASE DE UNA BUENA GESTION EMPRESARIAL,” La Habana, 2001.

[16] I. Flores de la Mota, *Conceptos Básicos de Estadística para Simulación*. México, 2011.

[17] Palisade Corporation, “Programa de complemento para el análisis y simulación de riesgo en Microsoft Excel.” New York, Mar. 2013.

[18] C. J. P. Vargas, O. L. F. Patiño, R. C. C. Lugo, and V. Oquendo, “Software de Análisis Estadístico SAS,” 2013.

[19] Universidad tecnológica Intercontinental, “Algoritmos y programación con lenguaje Python,” Frenando de la mora, 2011.

[20] J. B. Mendoza Vega, *R para principiantes*. 2018.

[21] C. Gil Bellosta, *R para profesionales de los datos: una introducción*. 2018.

[22] A. Visius, “Business & Marketing School,” *https://www.esic.edu*, Oct. 2020. .

[23] P. Bugnion, J. Grout, C. Singh, and B. Telenczuk, “Widgets de Jupyter,” *https://ipywidgets.readthedocs.io/en/latest/user_guide.html*, 2017. .

[24] I. Challenger Pérez, Y. Díaz Ricardo, and R. A. Becerra García, “El lenguaje de programación Python,” *Ciencias Holguín*, vol. 2, pp. 1–13, 2014.

[25] Palisade, “Palisade Help Resources,” *https://help.palisade.com/v8_1/es/@RISK/Function/Function-Reference.htm*. .

[26] P. Corporation, “Guía para el uso de @Risk,” New York, 2006.

[27] The SciPy community, “SciPy User Guide,” *https://docs.scipy.org/doc/scipy/tutorial/index.html*. .

[28] NumPy Developer, “NumPy user guide,” *https://numpy.org/doc/stable/user/index.html#user*. .

[29] D. P. León Sanchez, I. M. Quintero Rodriguez, and W. Zuluaga Muñoz, “Crystal Ball,” Bogotá D.C., 2004.

[30] stackoverflow, “Stackoverflow Developer Survey,” *https://insights.stackoverflow.com/survey/*, Feb. 2020. .

[31] Palisade, “Example Models,” *https://www.palisade.com/models/?language=english&dimension=recent*. .